Politecnico di Torino

Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Meccanica

A.a. 2024/2025

Sessione di laurea Ottobre 2025

Attività di sviluppo di un sistema di controllo in anello chiuso con feedback ottico

Relatori:

Prof. M. Sorli

Ing. A. Bertolino

Ing. A. De Martin

Candidato:

Vincenzo Marra

Sommario

Nata agli inizi degli anni Sessanta la visione artificiale ha subito negli ultimi decenni un rapido sviluppo grazie all'avvento di sensori e dispositivi sempre più performanti. Con l'integrazione di algoritmi di deep learning e intelligenza artificiale è stata ampiamente utilizzata in robotica, applicazioni per guida autonoma, video sorveglianza e tanto altro.

L'introduzione di tale soluzione nell'ambito dei servosistemi risulta interessante per svariati motivi di natura tecnica e applicativa. Anzitutto utilizzare una videocamera come feedback di posizione risulta essere una soluzione non intrusiva e contactless a differenza dei tradizionali trasduttori impiegati nei servosistemi. Questo è particolarmente vantaggioso poiché esclude il contatto diretto tra sensore e attuatore eliminando tutti gli errori sistematici a cui può andare incontro l'elemento di misura. Nel controllo e regolazione l'introduzione di videocamere permette lo sviluppo di strategie di retroazione sofisticate basate sul processamento di immagini che permettono di adattarsi in tempo reale a differenti condizioni ambientali e a disturbi esterni sul sistema. Attraverso le videocamere vi è la possibilità di effettuare misurazioni della posizione lungo diverse direzioni spaziali, contrariamente ai trasduttori mono-assiali. Vi sono diverse soluzioni proposte in letteratura riguardo questo tema, poche si concentrano sull'impostazione e le prestazioni del sistema.

Scopo della presente attività di tesi è quello di sviluppare un sistema di controllo in anello chiuso con feedback ottico, ovvero mediante un sistema di visione in grado di identificare l'oggetto di cui si vuole controllare il movimento. L'attività è condotta su un banco didattico già esistente, un sistema controllo posizione elettrico rispetto al quale si analizza la sostituzione del trasduttore potenziometrico tradizionale con un sistema basato su videocamere. La posizione controllata è quella del carrello di una stampante per computer, movimentato da un motore elettrico in corrente continua che trasmette il moto a una cinghia tramite una ruota dentata. Si definisce la scelta dei sensori (due videocamere o più), degli algoritmi di riconoscimento di immagine e di triangolazione della posizione e si effettua una valutazione preliminare delle prestazioni del set-up così configurato in termini di accuratezza statica, dinamica e risoluzione a fronte della soluzione in anello chiuso con sensore potenziometrico tradizionale. Si effettua, tramite modello, uno studio geometrico relativo al posizionamento delle videocamere scelte, individuando il numero minimo di videocamere per ricostruire la posizione e le configurazioni ottimali che massimizzano la risoluzione combinata dei sensori. Si evidenziano gli errori nella ricostruzione della posizione introdotti da un sistema di

visione artificiale, si determina la caratteristica statica delle videocamere acquistate e si sottolinea la necessità di introdurre un processo di calibrazione che coinvolge quest'ultima durante il posizionamento del sensore. Si analizza l'introduzione delle videocamere nel loop di posizione e come questo si può tradurre in termini di performance dinamiche del controllo posizione e si verifica la banda passante del sistema, confrontandola con quella risultante dalla soluzione con trasduttore potenziometrico. Infine, si implementa un algoritmo di riconoscimento del carrello basato sull'addestramento di una rete YOLO, dimostrando la concreta fattibilità della soluzione proposta.

Ringraziamenti

Desidero ringraziare il Prof. Massimo Sorli per avermi dato l'opportunità di sviluppare questo lavoro di tesi. Un ringraziamento sincero va all'Ing. Antonio Bertolino per la disponibilità, il costante supporto tecnico e l'attenzione dimostrata durante tutte le fasi del lavoro. Un grazie particolare all'Ing. Andrea De Martin, per la preziosa collaborazione, la vicinanza e i suggerimenti che hanno contribuito in maniera decisiva alla realizzazione di questa tesi.

Un pensiero di gratitudine va ora ai miei familiari e ai miei amici, che mi hanno accompagnato e sostenuto lungo questo percorso.
Desidero ringraziare profondamente i miei genitori, che con il loro costante supporto e incoraggiamento mi hanno accompagnato in ogni tappa del cammino.
A mio fratello Francesco, per la stima che mi dimostra e per il suo affetto che ritrovo ogni volta che torno a casa.
Ringrazio nonne, zii e cugini, per l'affetto e la presenza costante che non mi hanno mai fatto mancare. Una menzione particolare va a zio Stefano, per la sua compagnia insostituibile: dai pomeriggi al Torino Underground Festival a vedere i film di 'rottura', alle partite allo stadio, fino alle battute con cui non manca mai di ricordarmi che per lui resterò sempre 'un comunista'.
Un ringraziamento sincero a tutta la Batcaverna, per la quotidiana compagnia, la pazienza (nei miei confronti) e i tanti momenti condivisi che hanno reso più leggero e piacevole questo percorso.
A casa Gambasca, i primi ad accogliermi in questa città, la loro bontà è tale che una torta non basterà mai a sdebitarmi.
Un grazie di cuore anche ai miei amici di *Mesagn(e)* e *Lbaris*, per la vicinanza e il loro supporto in questi anni lontano da casa.
Un sentito grazie a tutti i miei colleghi, che con il tempo sono diventati veri amici, per tutti i momenti passati insieme: dalle giornate interminabili a portare avanti i progetti, passando per le serate a raccontare aneddoti e conoscere gente strana, alle gite fuori porta fino al calcetto serale svolto con grande dedizione ma con evidenti limiti fisici. Con voi ho imparato davvero il significato di *festina lente*: andare lontano non significa correre, ma condividere il passo giusto con le persone giuste.

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

1.1 Contesto

Negli ultimi decenni vi è stato un rapido sviluppo di tecniche di visione artificiale che hanno trovato il loro utilizzo in svariati ambiti ingegneristici come l'automazione e la robotica[1, 2]. Risulta essere sempre più frequente l'utilizzo di robot collaborativi aventi videocamere incorporate nel proprio end effector al fine di automatizzare alcuni processi[3]. I sistemi di visione sono utilizzati anche per remanufacturing, ispezione e controllo qualità[4, 5]. Tale tema può essere sviluppato e approfondito anche nei servosistemi, un ambito dominato da sensori fisici e trasduttori. L'idea di sostituire il tradizionale feedback di un servosistema con un strumento di misura costituito da una o più videocamere in grado di dare le medesime informazioni sulla grandezza fisica che si vuole controllare risulta essere molto interessante per i diversi vantaggi che una soluzione di questo genere può offrire. Questo sistema risulta essere uno strumento non intrusivo capace di sostituire i tradizionali trasduttori eliminando vincoli di natura meccanica e riducendo gli errori sistematici che nascono dall'interazione tra strumento di misura ed attuatore. Attraverso videocamere vi è la possibilità di effettuare misurazioni della posizione lungo diverse direzioni spaziali, contrariamente ai trasduttori mono-assiali. Questa soluzione si colloca in un contesto più ampio di sviluppo di sistemi di misura contactless, già ampiamente adottati in diversi settori. La fattibilità di sistemi di controllo di questo tipo è già dimostrata in letteratura[6, 7]. Bisogna sottolineare che l'utilizzo di videocamere all'interno di un sistema di controllo consente l'integrazione di algoritmi di elaborazione e riconoscimento immagine che permettono di estrarre informazioni sulla posizione ed il movimento dell'oggetto da controllare. Un aspetto interessante è che nel corso dell'ultimo decennio questi algoritmi di object detection sono diventati performanti anche per soluzione real time[8]. Risultano facilmente integrabili con altre tipologie di sensori attraverso il data fusion per sviluppare

sistemi di controllo ibridi[9]. La loro precisione permette la classificazione di oggetti anche di piccole dimensioni e con particolari caratteristiche [10, 11]. In questo lavoro si colloca lo sviluppo di un sistema controllo posizione attraverso feedback ottico realizzato a partire da un banco didattico già esistente. Si analizza la sostituzione del trasduttore potenziometrico con un sistema di visione basato su delle videocamere.

1.2 Descrizione del banco prova

La posizione da controllare è quella del carrello di una comune stampante per computer; il carrello viene movimentato tramite un motore elettrico in corrente continua che ingrana tramite una ruota dentata in una cinghia, uno dei due rami della quale è vincolato al carrello stesso. Il servoattuatore è dotato di un trasduttore di posizione, atto a rilevare in ogni istante la posizione del carrello. Il controllo di posizione è realizzato in modo analogico, e risiede in un rack nel quale sono presenti anche i moduli di generazione funzione, di condizionamento del trasduttore e di regolazione della tensione di armatura del motore elettrico a corrente continua[12].

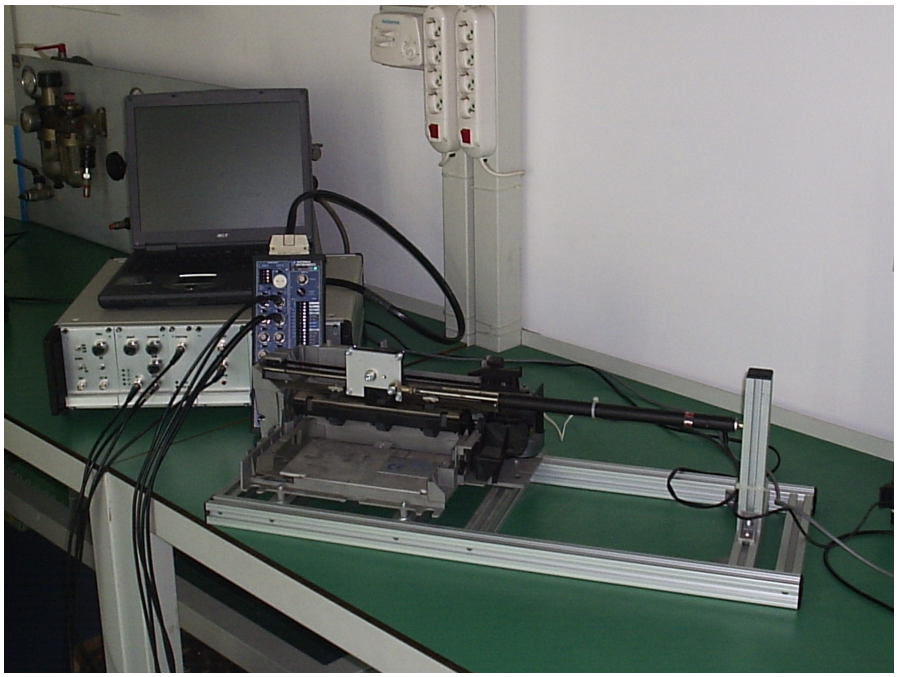

Figura 1.1: Banco prova[13]

Il banco prova su cui si basa l'attività di tesi è costituito dunque da quattro componenti fondamentali:

- motore elettrico a corrente continua;
- slitta traslante con trasmissione a cinghia dentata;
- trasduttore di posizione potenziometrico;
- sistema di acquisizione e controllo.

Il motore presente sul banco è a corrente continua con collettore. Si può infatti osservare in figura 1.1 la presenza di due fili che entrano nel motore e che ne garantiscono l'alimentazione. Il rotore del motore è solidale a un puleggia dentata di diametro 14.5 millimetri che attua il movimento della cinghia di trasmissione. Alla cinghia, che ingrana all'altra estremità con un'altra puleggia folle, è solidale su uno dei suoi rami con il carrello della stampante. Pertanto, una rotazione del rotore del motore viene trasformato, per mezzo della cinghia, in una traslazione del carrello. La slitta traslante risulta avere una corsa di circa 200 millimetri. Il trasduttore è di tipo potenziometrico della Penny & Giles mod. HLP 190/200/8k2. Esso risulta essere a traccia ibrida. Nella seguente tabella sono riportati i principali dati tecnici del trasduttore di posizione:

Trasduttore di posizione potenziometrico	
Marca	Penny & Giles
Modello	HLP 190/200/8k2
Lunghezza totale	310.5 [mm]
Corsa a comportamento lineare	± 100 [mm]
Peso	0.231 [Kg]
Guadagno	0.1 [V/mm] (± 10 [V] F.S. lineare)
Frequenza di taglio (–3 dB)	. . . [Hz]
Alimentazione	± 10 [V]
Resistenza	8 [KΩ]
Campo di temperatura di esercizio	-30 ÷ +100 [°C]
Campo temperatura max esposizione	–65 ÷ 200 [°F] ⇒ –55 ÷ 95 [°C]
Risoluzione	Virtualmente infinita
Isteresi	Minore di 0.01 mm
Massima tensione di alimentazione	74 [V]
Errore di linearità indipendente	0.15 %
Massima velocità di esercizio	10 [m/s]

Tabella 1.1: Dati tecnici del trasduttore di posizione Penny & Giles[12].

Il trasduttore è l'elemento di misura della posizione mediante il quale è possibile effettuare una retroazione analogica per poter effettuare un controllo analogico

ad anello chiuso. Visivamente è costituito esternamente da un cilindro con una estremità fissa e una solidale al carrello della stampante[12]. Si tratta di un dispositivo a tre terminali costituito da un elemento resistivo su cui è possibile far scorrere un contatto strisciante. Due terminali sono collegati agli estremi dell'elemento resistivo di resistenza R mentre il terzo terminale è collegato al contatto strisciante; la resistenza tra quest'ultimo ed uno dei due terminali estremi assume il valore $R_x \leq R$. Di conseguenza, in base alla posizione dell'oggetto, e quindi in base alla posizione del contatto strisciante vi sarà una differente ripartizione di resistenza elettrica e dunque di tensione in uscita dal trasduttore. Ad ogni posizione del carrello corrisponderà dunque una determinata tensione in uscita dal trasduttore potenziometrico. Il trasduttore potenziometrico è proprio l'elemento che deve essere sostituito dalle videocamere nell'anello di posizione del servosistema.

L'apparecchiatura di controllo è adibita alla movimentazione del motore elettrico e al controllo in anello chiuso di retroazione della posizione. Il segnale di comando può essere generato internamente all'apparecchiatura tramite un modulo generatore di funzione, oppure può essere ricevuto da un generatore esterno. Il controllo sull'anello chiuso di posizione è realizzato con scheda analogica, mediante una rete di compensazione PID, di cui è possibile variare il guadagno di ogni singolo componente. L'apparecchiatura dispone inoltre del driver del motore e della scheda di condizionamento del segnale del trasduttore di posizione potenziometrico. L'apparecchiatura di controllo è realizzata in rack standard 19", 3 unità di altezza. I moduli sono realizzati con schede plug-in da 19" tipo Eurocard a 32+32+32 connettori, ciascuno dei quali svolge le seguenti funzioni:

- modulo di alimentazione elettronica;
- modulo generatore di funzione;
- modulo di controllo e compensazione PID;
- modulo condizionatore trasduttore di posizione potenziometrico;
- modulo driver del motore[12].

Nel sistema di acquisizione e controllo per valutare le prestazioni del banco è presente un National Instrument MyRio-1900 ed un PC con interfaccia LabVIEW.

1.3 Obiettivi

Diverse attività di ricerca approfondiscono il tema del machine vision, di cui alcune anche in applicazioni relative a servosistemi. I sistemi vision based sono ampiamente utilizzati in contesti di robotica ed automazione industriale. Ci sono diversi metodi di apprendimento che si basano su reti neurali e deep learning ad esempio per robot manipolatori. Ciò che risulta poco trattato in letteratura è sicuramente come l'introduzione di un dispositivo come una videocamera influenzi l'anello di posizione di un servosistema.

L'attività di tesi si concentra sulla selezione dei componenti necessari per sviluppare a partire dal banco didattico già esistente un servosistema controllo posizione con feedback mediante videocamere e come le specifiche tecniche di tale componentistica influenza in termini di accuratezza statica, dinamica e tempi di risposta il sistema. Lo studio del posizionamento della videocamera e degli errori che essa può introdurre ha una grossa rilevanza nell'attività sviluppata. Si vuole inoltre proporre l'algoritmo di riconoscimento del carrello. Gli aspetti specifici della presente ricerca per affrontare gli obiettivi sopra citati sono:

- Presentare una panoramica delle principali soluzioni valutate per lo sviluppo del sistema sia per le videocamere che per le piattaforme hardware utili all'individuazione della posizione del carrello.
- Scegliere,in base ai vantaggi e svantaggi che ogni opzione presenta da un punto di vista delle specifiche di accuratezza e tempi di risposta che il banco richiede, la soluzione a livello di componentistica tra le varie proposte.
- Analizzare la risoluzione delle telecamere in combinazione con il refresh rate stabilendo se è sufficiente per lo scopo e che genere di accuratezza nella lettura della posizione si può fornire.
- Identificare il numero minimo di videocamere necessarie a stabilire la posizione del carrello e le configurazioni ottimali.
- Studiare gli errori del sistema in base alla posizione delle telecamere. Analizzare come essi influenzano la ricostruzione della posizione e determinare la una caratteristica statica del sensore.
- Studiare a modello come l'introduzione di una videocamera si può tradurre in termini di performance dinamiche del controllo posizione e verificare la banda passante del sistema con il modello, effettuando un confronto con il modello con trasduttore potenziometrico tradizionale.
- Impostare e controllare le videocamere e la piattaforma hardware scelte.

- Sviluppare l'algoritmo di riconoscimento dell'oggetto attraverso rete neurale *YOLO*.

1.4 Struttura della tesi

La struttura della tesi si articola in sei capitoli che sono brevemente descritti di seguito:

CAPITOLO 1 è dedicato all'introduzione della tesi.

CAPITOLO 2 descrive la scelta dei componenti per lo sviluppo del sistema. In particolare, vi è una descrizione dettagliata delle principali soluzioni valutate, con vantaggi e svantaggi da un punto di vista delle specifiche tecniche di ogni componente.

CAPITOLO 3 introduce lo studio geometrico sviluppato per il corretto posizionamento delle videocamere per il feedback della posizione. In particolare, si analizzano gli errori introdotti nella ricostruzione della posizione da parte della videocamera e le configurazioni ottimali dei vari sensori considerati che consentono di massimizzare l'accuratezza nella lettura della posizione dell'oggetto.

CAPITOLO 4 descrive il modello per tener conto del feedback mediante videocamera, l'influenza di quest'ultima sull'anello di posizione e la determinazione di una banda passante del sistema sulla base di una specifica modellazione del sensore.

CAPITOLO 5 tratta l'attività sperimentale condotta al fine di configurare e controllare le videocamere acquistate e sviluppare l'algoritmo di riconoscimento del carrello tramite rete neurale.

CAPITOLO 6 è dedicato alle conclusioni dell'attività di tesi, ad una descrizione dei risultati raggiunti e dei possibili sviluppi futuri.

Capitolo 2

Scelta dei componenti e del layout del banco

In questo capitolo, si analizzano le principali alternative hardware e software per la realizzazione del banco prova con feedback ottico. Esistono innumerevoli piattaforme hardware in grado di implementare un anello chiuso con un segnale di feedback mediante videocamere e ognuno di esse è compatibile con diverse tipologie di sensori. Durante lo svolgimento dell'attività di tesi sono state esaminate diverse soluzioni e confrontate sulla base di diverse specifiche tecniche. Per la piattaforma hardware si valutano le seguenti caratteristiche:

- numero di videocamere compatibili;
- linguaggio di programmazione;
- frequenza di lavoro;
- numero totale di pin;
- GPIO analogici e digitali.

Per la scelta delle videocamere:

- risoluzione;
- frame rate;
- colore dell'immagine;
- Field Of View;
- lunghezza focale;

- tipologia di focus;
- dimensione del sensore.

Ogni piattaforma hardware ha dei vantaggi e svantaggi legati alla frequenza di lavoro e alla capacità di gestire più flussi provenienti da diverse videocamere che rilevano la posizione dell'oggetto. Le videocamere sono inevitabilmente condizionate da parametri intrinseci come la risoluzione ed il frame rate che incidono sull'accuratezza di lettura della posizione e velocità dell'oggetto. Un parametro molto importante da non trascurare per lo sviluppo di questi sistemi è il costo. Con lo sviluppo dell'elettronica e l'incremento d'interesse nei confronti delle tematiche di computer vision, esistono tantissime soluzioni con microcontrollori o altre schede compatibili con videocamere volte all'object tracking. Esistono soluzioni più sofisticate che richiedono un costo maggiore ma riescono efficacemente nell'intento di identificare la posizione dell'oggetto. L'attività di sviluppo, a fini didattici e di ricerca, a valle di un'analisi delle prestazioni di diverse opzioni tiene conto anche dei costi per la scelta della soluzione ottimale.

2.1 Piattaforma hardware

Le diverse piattaforme analizzate rivestono un ruolo importante nell'evoluzione dell'elettronica, del computer vision e dell'intelligenza artificiale. La loro importanza risiede nel fatto che nel corso del tempo sono riusciti a rendere accessibile a tutti la prototipazione di sistemi complessi senza la necessità da parte dell'utente di comprendere a fondo l'elettronica alla base. Hanno reso possibile effettuare collegamenti senza necessità di saldature semplificando nettamente questo aspetto nell'assemblaggio dei sistemi. Alcune piattaforme, come ad esempio Arduino, nascono come semplici microcontrollori programmabili per lo sviluppo di prototipi a livello di automazione o elettronica. Altre soluzioni più sofisticate come ad esempio Raspberry o NVIDIA Jetson, sono dei veri e propri mini computer dotati di una memoria e un sistema operativo in grado di espandere notevolmente le possibili applicazioni. Alcune di queste piattaforme hanno già avuto utilizzi correlati al computer vision per sistemi robotici. In letteratura vi sono diverse applicazioni anche in ambito meccatronico. In [6] è sviluppato un controllo posizione di un attuatore con feedback proveniente da una Raspberry Pi camera module, il processamento dell'immagine per l'identificazione della posizione e la calibrazione è stata affidata ad un Jetson Nano, ed i segnali di controllo inviati al driver del motore tramite un Arduino 2. Sono inoltre sistemi hardware integrabili in applicazioni in cui vengono coinvolti modelli di Machine Learning e data fusion. In [14] vi è l'introduzione di un modello Machine Learning per la stima della posizione di un robot mobile tramite un sistema di visione basato su videocamere Vicon. Il modello riesce ad

acquisire informazioni sulla dinamica e sul posizionamento del robot. All'interno del sistema sviluppato vi è un Raspberry Pi 4 in grado di collezionare informazioni sull'accelerazione del robot e trasferirle ad un computer centrale. Il computer è in grado di ottenere informazioni sull'accelerazione dal Raspberry e sulla posizione dal sistema di videocamere. Tutte queste piattaforme sono compatibili con algoritmi molto sofisticati di object detection come ad esempio YOLO, in grado di garantire performance real time.

2.1.1 Arduino R1 Giga

Arduino nasce come una piattaforma hardware software composta da delle schede elettroniche dotate di un microcontrollore e di un software Arduino IDE che ne permette la programmazione. Nasce come un progetto open source sviluppato all'inizio degli anni 2000 da dei docenti e ricercatori italiani di Ivrea per scopi didattici e professionali. Grazie alla sua incredibile semplicità e rapidità di utilizzo tale scheda è diventata un punto di riferimento nell'elettronica per l'implementazione di sistemi che coinvolgono sensori, attuatori e tanto altro.

Lo stato dell'arte è rappresentato dalla scheda Arduino R1 Giga grazie alla sua potenza e flessibilità. Questa scheda presenta un processore STM32H747XI con dual-core ARM Cortex-M7 a 480 MHz e Cortex-M4 a 240 MHz. Presenta la connessione Wireless e Bluetooth. A differenza delle schede classiche, ad esempio Arduino Uno, presenta molti più pin I/O. Si parla infatti di quasi 80 GPIO tra cui 12 analogici, 12 PWM e 2 DAC. Sono presenti 20 pin per la connessione di videocamere Arducam [15].

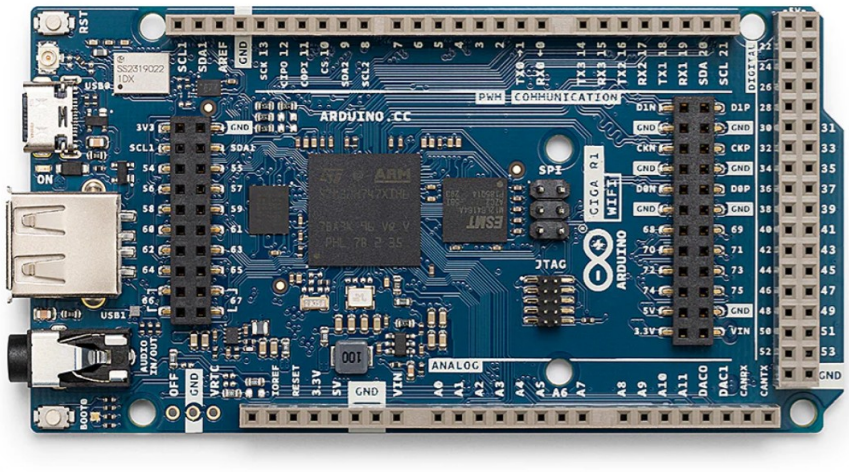

Figura 2.1: Arduino R1 Giga [15]

La scheda Arduino R1 Giga non presenta un'elevata frequenza di lavoro rispetto ad altre soluzioni che verranno descritte successivamente, risulta tuttavia sufficiente per compiti di controllo in tempo reale e gestione di sensori. Tendenzialmente Arduino è programmabile in C++ ma ci sono dei pacchetti di supporto che lo rendono compatibile con MATLAB se è necessario integrare modelli simulati. Un grande vantaggio di tale piattaforma risiede sicuramente nel numero di pin a disposizione, essendo numerosi è facilmente collegabile con più dispositivi divenendo una soluzione con un'ottima scalabilità. Supporta fino a 4 videocamere Arducam.

2.1.2 Raspberry Pi 5

Raspberry Pi 5 rappresenta l'ultimo modello di single board computer in commercio della Raspberry Pi Foundation. Basato sul SoC Broadcom BCM2712, integra una CPU quad-core ARM Cortex-A76 a 2.4 GHz, che offre un incremento notevole a livello di prestazioni rispetto al Pi 4. A differenza di Arduino, tale scheda è un vero e proprio computer dotato di memoria, processore e sistema operativo. Il processore grafico è un VideoCore VII, a 800 MHz, compatibile con OpenGL ES3.1 e Vulkan 1.2, e consente l'uscita Dual 4Kp60 su HDMI con supporto HDR. La memoria disponibile dipende dalla versione acquistata, può arrivare fino a 16 GB. Il nuovo chip, gestisce USB2.0/3.0, Ethernet gigabit, MIPI-CSI/DSI, GPIO e slot microSD, migliorando larghezza di banda e stabilità delle periferiche[16]:

Figura 2.2: Raspberry Pi 5[16]

La frequenza di lavoro di 2.4 GHz risulta notevolmente superiore a quella di Arduino R1 Giga. Risulta essere programmabile in C++, Python, Scratch e Java. Risulta essere una soluzione performante in contesti di computer vision e object tracking per diversi motivi. Dal punto di vista hardware ha delle porte MIPI CSI-2 adibite appositamente per il collegamento con videocamere ad elevata risoluzione e frame rate gestendo con la sola scheda nativa fino a due videocamere contemporaneamente. Tramite Multiplexers HAT è possibile avere fino a quattro videocamere collegate in simultanea. L'interfaccia MIPI, con una larghezza di banda stimata di circa 6 Gbps riesce tranquillamente a gestire il bitrate delle videocamere. Dal punto di vista software vi sono delle librerie con Python ampiamente utilizzate proprio per l'object tracking e la stima della posizione dell'oggetto, come ad esempio *OpenCV*. Non presenta gli stessi pin di una scheda come quella di Arduino, tuttavia Raspberry ha fino a 40 pin che risultano più che sufficienti per il sistema da implementare. Il costo è sicuramente maggiore rispetto ad una soluzione pensata con Arduino soprattutto per la necessità di acquistare ulteriori attrezzature necessarie per il Raspberry. Essendo un computer a tutti gli effetti si necessita di periferiche quali mouse e computer, un sistema di raffreddamento visto il carico computazionale che deve sostenere, un DAC poichè non ha ingressi e uscite analogici ma solo digitali. Rimane una soluzione accessibile adatta per scopi didattici ma anche per prototipazione a livello di ricerca.

2.1.3 NVIDIA Jetson AGX Orin

La NVIDIA Jetson AGX Orin rappresenta una piattaforma edge AI altamente avanzata, studiata per applicazioni come robotica autonoma, visione industriale, veicoli intelligenti e molto altro. Il dispositivo integra una GPU con 2048 CUDA Core e 64 Tensor Core, supportata da 12 core ARM Cortex-A78AE a 2.2 GHz. La memoria è disponibile in varie versioni tipo 32 o 64 GB. Si parla di una soluzione sofisticata che lato hardware ha delle specifiche importanti. Si ha un totale di 699 pin nel package Ball Grid Array del modulo. Di tutti questi pin, 260 sono GPIO con multifunzione. Presenta 6 interfacce MIPI CSI il che significa che senza alcun multiplexer è in grado di gestire lo stream di 6 videocamere.

Essendo un dispositivo utilizzato per applicazioni industriali si utilizza spesso un Developer Kit per lo sviluppo ed il collaudo della soluzione. Il Dev Kit serve a facilitare lo sviluppo e il test: permette di programmare il modulo, testare i sensori, le periferiche e le telecamere senza dover prima progettare un hardware personalizzato. Il solo modulo Jetson non ha connettori utilizzabili direttamente, poichè pensato per essere saldato su una carrier board. Per quanto riguarda i pin del Developer KIt si parla di 2 header a 40 pin e solo 2 porte MIPI CSI sempre espandibili, proprio come Raspberry. Il numero totale dei pin accessibili quindi,

varia in base a se si considera solo il modulo (699 pin) o il Dev Kit (decine di pin tramite connettori e header standardizzati)[17].

Figura 2.3: NVIDIA Jetson AGX Orin Developer Kit [17]

NVIDIA Jetson rappresenta una soluzione efficace e performante per lo sviluppo di un controllo posizione elettrico tramite feedback ottico. Garantisce prototipazione tramite il Developer Kit e può gestire videocamere industriali in grado di restituire con precisione la posizione dell'oggetto. Ha diversi linguaggi di programmazione compatibili ed un numero elevato di pin che lo rende un prodotto con grande scalabilità. Esistono diverse videocamere prodotte appositamente per essere compatibili con NVIDIA Jetson, il modulo inoltre permette di gestire un numero di videocamera più che sufficiente per l'applicazione che si vuole sviluppare. Il costo risulta per queste ragioni molto più alto rispetto a piattaforme come Arduino o Raspberry.

2.1.4 Renesas EK-RA6M4

La Renasas EK-RA6M4 è una scheda con microcontrollore RA6M4 con processore ARM Cortex-M33 a 200 MHz, progettato per prestazioni elevate ed efficienza energetica. É disponibile una memorai flash e 256 KB di SRAM per il codice e i dati. Si può accedere direttamente ai pin del microcontrollore tramite 4 griglie da 40 pin ciascuno. Ha debug integrato supportando fino a tre modalità. Un vantaggio di tale scheda risiede sicuramente nell'efficienza energetica, presenta dei punti di misura di corrente integrati per monitorare con precisione il consumo elettrico

della MCU. Risulta essere una soluzione con grande espansione e compatibilità, include infatti fino a 5 interfacce per il collegamento con altri sistemi, dispositivi o microcontrollori. Presenta connettori Grove, Qwiic ed addirittura uno per il collegamento con Arduino Uno R3. Presenta la porta Ethernet ed ha una memoria esterna da 64 MB[18].

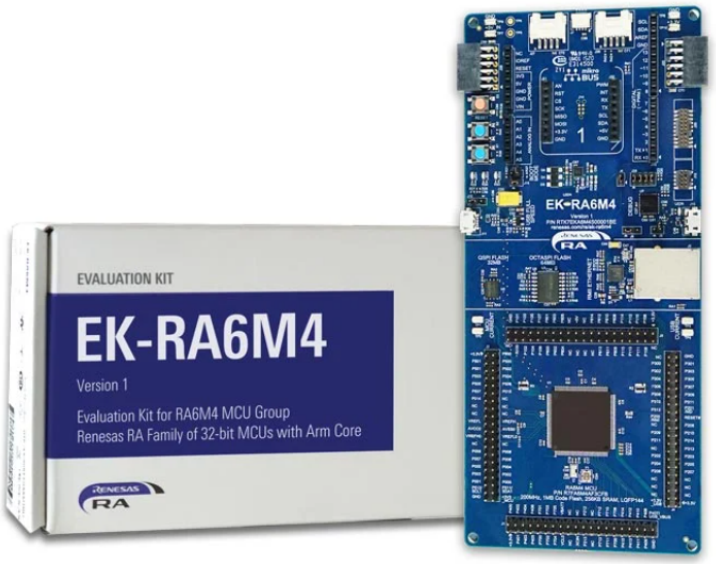

Figura 2.4: Renasas EK-RA6M4[18]

La scheda in questione ha la frequenza di lavoro più bassa tra tutte le soluzioni valutate. Risulta una soluzione costosa, con un'elevata scalabilità grazie alla presenza di oltre 100 GPIO e la possibilità di programmarla in Python, C++ e Rust. Il kit risulta avere un costo accessibile rappresentando una delle soluzioni più economiche.

2.1.5 Scelta della piattaforma

Alla luce di un'attenta valutazione delle prestazioni e dei costi delle soluzioni precedentemente descritte si è scelta come piattaforma hardware per lo sviluppo del sistema il Raspberry Pi 5, in particolare la versione computer module (CM5). Tale scheda presenta una delle frequenze di lavoro più alte tra le soluzioni analizzate ovvero 2.4 GHz. Dal punto di vista software risulta essere una delle soluzioni più vantaggiose poichè programmabile in Python, un linguaggio di programmazione accessibile ed efficace, oltre al fatto che è presente la libreria *OpenCV* con una ricca documentazione proprio per l'object tracking ed il pose estimation. Non presenta un elevato numero di pin, tuttavia risultano essere sufficienti per lo sviluppo del sistema

in questione. Il costo ha sicuramente inciso sulla scelta: Jetson Nano presenta prestazioni maggiori in termine di pin e numero di videocamere collegabili tuttavia risulta essere una soluzione troppo sofisticata con un costo della sola piattaforma pari a 20 volte quello del Raspberry. Il CM5 ha caratteristiche sicuramente superiori rispetto a Renasas ed Arduino, con dei tempi di risposta certamente minori che quindi la fanno preferire a queste due soluzioni. Renasas ed Arduino hanno un elevato numero di pin e diverse videocamere compatibili ma non sono sviluppati appositamente per l'image processing e computer vision. Si riporta una tabella riassuntiva delle varie soluzioni valutate per avere una visione completa dell'analisi svolta:

Piattaforma	**Linguaggio**	**Videocamere**	**Frequenza**	**Pin**
Raspberry Pi 5	Python, Scratch, C, C++, Java	4	2.4 GHz	40
NVIDIA Jetson AGX Orin	Python, C, C++, Scratch, Rust	6	2.2 GHz	699
Renesas EK-RA6M4 (LQFP)	C++, Python, Rust	4	200 MHz	144
Arduino R1 Giga	C, C++	4	480 MHz	103

Tabella 2.1: Confronto tra le piattaforme hardware analizzate

Dal punto di vista del costo, a parte NVIDIA Jetson, sono tutte soluzioni accessibili. Risultano essere tutte compatibili con MATLAB, Simulink nel caso si volessero integrare ulteriori modelli. Sulla scelta ha inciso sicuramente il costo, la volontà di sviluppare un sistema con tempi di risposta brevi e la ricerca di una soluzione già collaudata in ambito object tracking come appunto lo è Raspberry.

2.2 Videocamere

La videocamera rappresenta il dispositivo attraverso il quale è possibile acquisire informazioni sulla posizione dell'oggetto. I suoi parametri, in particolare frame rate e risoluzione del sensore, incidono sulla precisione con cui viene identificata la posizione dell'oggetto. Esistono diverse videocamere compatibili con le piattaforme hardware valutate. Come per la piattaforma, si sono analizzate diverse soluzioni sulla base delle specifiche tecniche delle videocamere e a valle di tutto ciò si è effettuata una scelta.

2.2.1 Autofocus Synchronized Quad-Camera Kit

Arducam Autofocus Synchronized Quad-Camera (Quad-Camera RP5) è un kit progettato per applicazioni di image processing a multi-camera ad alta risoluzione. Le quattro videocamere hanno dei sensori Sony IMX686 con dimensione 1/1.7". Questa soluzione risulta essere molto interessante per il numero di videocamere che mette a disposizione, superando i limiti del solito collegamento con singola o doppia camera che ha Raspberry attraverso le porte MIPI CSI-2. Ha un discreto campo visivo, circa 84 gradi ed una lunghezza focale di circa 5.1 mm[19].

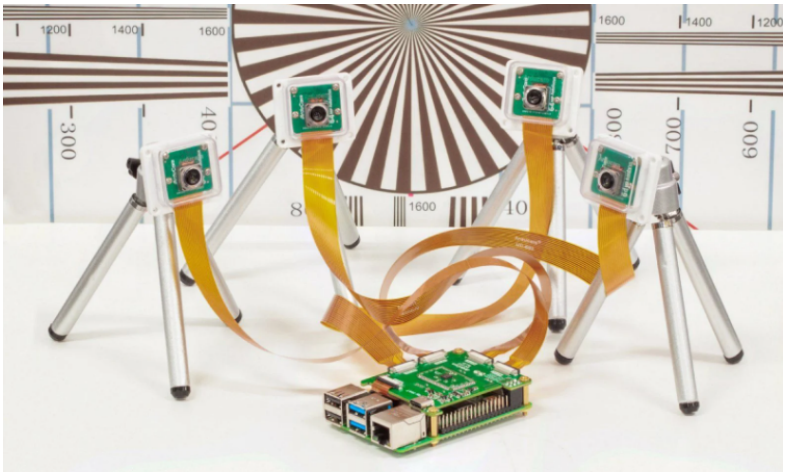

Figura 2.5: Quad camera kit per Raspberry Pi 5[19]

I principali vantaggi sono:

- Elevata risoluzione: il sensore può arrivare fino a 64 MP risultando una delle risoluzioni più elevate per videocamere arducam compatibili con Raspberry.
- Sincronizzazione multicamera: fino a 4 videocamere sincronizzabili attraverso trigger e utilizzabili in simultanea.
- Autofocus motorizzato: il focus è automatico e si adatta alla situazione.
- Ottiche intercambiabili: vi è la possibilità di utilizzare differenti lenti in base al contesto.
- Elevato numero di modalità video: si possono impostare frame rate e risoluzione in base a quanto richiesto dal sistema.

Vi sono tuttavia delle limitazioni da considerare per tale soluzione:

- La risoluzione massima ovvero 64MP la si raggiunge con un numero bassissimo di FPS, incompatibile con il sistema da sviluppare.

- Il carico computazionale potrebbe essere così elevato da limitare la prestazione real-time.
- Il costo è elevato rispetto ad altre soluzioni sempre con medesima piattaforma.
- Installazione e setup a livello software ed hardware sono complessi.

Il kit comprende un Camarray HAT in grado di garantire il collegamento in simultanea di quattro videocamere. La codifica dei colori è come la maggior parte delle camere Arducam a 10 bit. Le videocamere raggiungono anche i 120 FPS con una risoluzione che resta su valori modesti (1280x720), nonostante l'elevato frame rate.

2.2.2 Videocamere MEGA

Esistono diverse videocamere MEGA compatibili con diversi microcontrollori, in particolare con Arduino e Renesas:

- MEGA 3MP Camera Module;
- MEGA 5MP Camera Module;
- MEGA 3 MP M12 Camera Module;
- MEGA 5 MP M12 Camera Module;
- MEGA 3MP NOIR Camera Module;
- MEGA 5MP NOIR Camera Module.

Sono tutte camera module a bassa risoluzione progettate per funzionare con qualsiasi tipo di microcontrollore. Risultano essere una soluzione compatta e robusta pensate per applicazioni in cui si privilegia il basso consumo e la semplicità di utilizzo. Alcune videocamere (MEGA 3MP e MEGA 5MP) hanno l'autofocus mentre i restanti modelli sono tutti con focus manuale. Sono tutte videocamere Rolling Shutter, con un basso FOV che varia tra i 60 ed i 90 gradi. Le lunghezze focali sono tutte attorno ai 3 mm ed i sensori sono di dimensione standard da 1/4". La codifica del colore è solitamente ad 8 bit[20]. I principali vantaggi sono:

- Compatibilità universale: sono progettati per funzionare con una vasta gamma di microcontrollori, molti dei quali in grado di supportare fino a quattro videocamere.
- Consumo energetico basso: il collegamento è attraverso jumper ciò significa che consumano molto meno energia rispetto a moduli connessi tramite interfaccia MIPI CSI 2.

Ci sono diverse criticità riscontrate per queste videocamere:

- Bassa risoluzione: non sono pensate per applicazioni di image processing o object tracking.
- Rolling Shutter: l'acquisizione delle immagine con tale modalità rende più lenta la risposta del sistema.
- Frame rate non specificato: non è riportata la modalità video il che rende impossibile comprendere se ha degli FPS sufficienti per l'applicazione sul sistema.

Figura 2.6: MEGA 3MP e MEGA 5MP Camera Modules[20]

2.2.3 Arducam Rolling Shutter IMX477

La Arducam 12MP IMX477 con autofocus motorizzato è una videocamera per sistema Raspberry Pi che utilizza un sensore IMX477 a 12.3 MP. É utilizzata per applicazioni di computer vision e visione artificiale dove si richiedono elevate prestazioni in termine di risoluzione e frame rate. Rispetto ad altre videocamere per Raspberry aggiunge il focus motorizzato, ed una risoluzione elevata. Possiede una lunghezza focale di 3 mm, un FOV di 78 gradi e diverse modalità video che consentono di impostare risoluzione e frame rate, con quest'ultimo che può raggiungere fino ai 120 FPS. La codifica dei colori è a 10 bit come diverse videocamere Arducam.

Sicuramente il fatto che sia una videocamera Rolling Shutter la penalizza dal punto di vista dell'acquisizione immagini e risposta del sistema, ma la sua elevata precisione la rendono comunque una soluzione valida soprattutto se combinata con altre tipologie di sensori. Non ha ottiche intercambiabili e focus manuale. Date le sue specifiche, il costo del componente risulta maggiore rispetto alle classiche videocamere Pi[21].

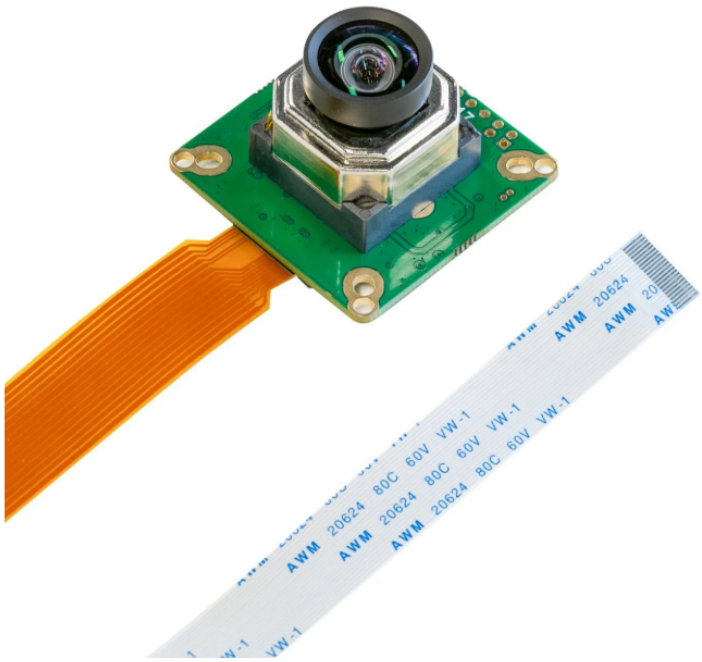

Figura 2.7: Arducam IMX477[21]

2.2.4 Arducam OG02B10 Global Shutter

La videocamera Arducam OG02B10 è un sensore compatibile con Raspberry Pi progettato per applicazioni di machine vision e image processing con particolare attenzione alla rilevazione di movimenti rapidi e riduzione della distorsione grazie al sensore Global Shutter. È compatibile con Pivariety, che consente di sfruttare driver V4L2 personalizzati e maggiore controllo sul sensore, rendendola adatta a scenari di robotica, tracciamento e machine vision in tempo reale. Il sensore in questione, con dimensione di 1/2.9", permette di raggiungere i 60 FPS mantenendo la risoluzione massima, inoltre ha un ampio campo visivo di circa 110 gradi. Queste due caratteristiche lo rendono una soluzione ottima per il sistema da sviluppare. Ha sicuramente una risoluzione limitata. 2MP sono ottimi per visione real-time e applicazioni di robotica, ma modesti nella accuratezza lettura posizione.

La codifica del colore è sempre a 10 bit. Ha diverse modalità video che permettono di variare FPS e risoluzione [22].

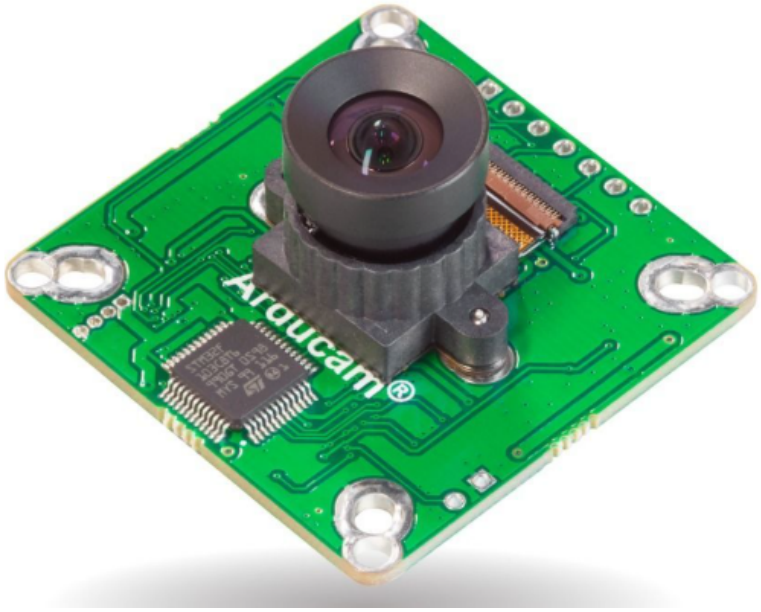

Figura 2.8: Arducam Global Shutter OG02B10[22]

2.2.5 Raspberry Pi Camera Module 3

La Raspberry Pi Camera Module 3 rappresenta l'ultimo modello di Picamera per Raspberry Pi, con significativi miglioramenti rispetto alle versioni precedenti. Il modulo in questione è basato sul sensore IMX708 da 1/2.43", di tipologia Rolling Shutter. Vi sono diverse versioni:

- standard con autofocus;
- standard con fixed focus;
- wide angle con autofocus;
- wide angle con fixed focus.

La versione standard ha un FOV di 75 gradi orizzontali mentre la versione wide angle raggiunge i 120 gradi diagonali. Sono presenti diverse modalità video con piena risoluzione fino a 30 FPS. Il modulo riesce come molte altre videocamere per Raspberry a ridurre la risoluzione per aumentare il frame rate fino a 120 FPS[23].

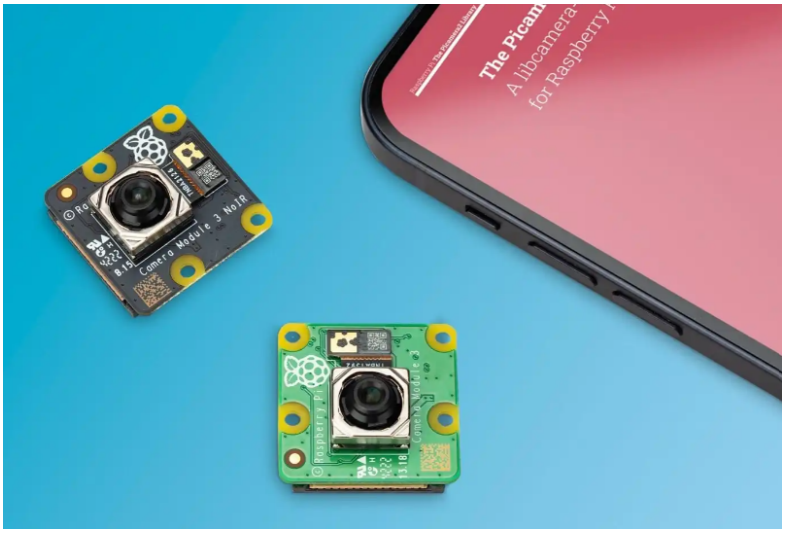

Figura 2.9: Raspberry Pi Camera Module 3[23]

Vi sono diversi vantaggi nell'utilizzare questo modulo per sviluppare il feedback ottico:

- Alta risoluzione: tra i diversi sensori in commercio per Raspberry risulta essere uno di quelli con risoluzione maggiore.
- Ampio campo visivo: grazie alla versione wide angle, si ottiene un FOV di 120 gradi.
- Si possono impostare diverse modalità video.
- Sono disponibili diverse tipologie di messa a fuoco.

Rappresenta una soluzione versatile ed efficace, combinando nelle sue caratteristiche i principali vantaggi del sensore IMX477 per la risoluzione, e del sensore OG02B10 per l'ampio FOV. Le uniche limitazioni sono la tipologia di sensore e l'ottica non intercambiabile. La videocamera è infatti Rolling Shutter il che rallenta l'acquisizione ed il processamento delle immagini. Resta tuttavia un'ottima soluzione per prototipazione in ambito di visione artificiale e robotica.

2.2.6 Multi-Camera Kit per NVIDIA Jetson

La soluzione con Multi-Camera Kit JN Arducam 12MP è un sistema avanzato multi-camera progettato per sfruttare la potenza di elaborazione delle piattaforme Jetson in applicazioni complesse e sofisticate. Risulta essere sicuramente una delle soluzioni più prestanti poichè utilizza tutte videocamere con sensore IMX477 che garantiscono un'elevata risoluzione inoltre è possibile utilizzare in simultanea, tramite

specifica piattaforma Jetson, fino a 6 videocamere. Le videocamere in questione presentano un ampio campo visivo di 120 gradi ed una codifica dei colori a 10 bit[24].

Diversi sono i vantaggi che offre questa soluzione:

- Sistema multi-camera sincronizzato: è possibile attivare lo stream video per sei videocamere in contemporanea.
- Alta risoluzione: le videocamere sono tutte con sensori IMX477 da 12MP.
- Sono disponibili diverse modalità video.
- Integrazione nativa con Jetson Nano: non vi è la necessità di acquistare ulteriori accessori per garantire il collegamento con le videocamere.
- Si può configurare l'ottica in base ad una determinata necessità.

Le uniche limitazioni riguardano il costo che risulta elevato, poichè la soluzione è molto sofisticata, e la tipologia di sensore che risulta essere Rolling Shutter.

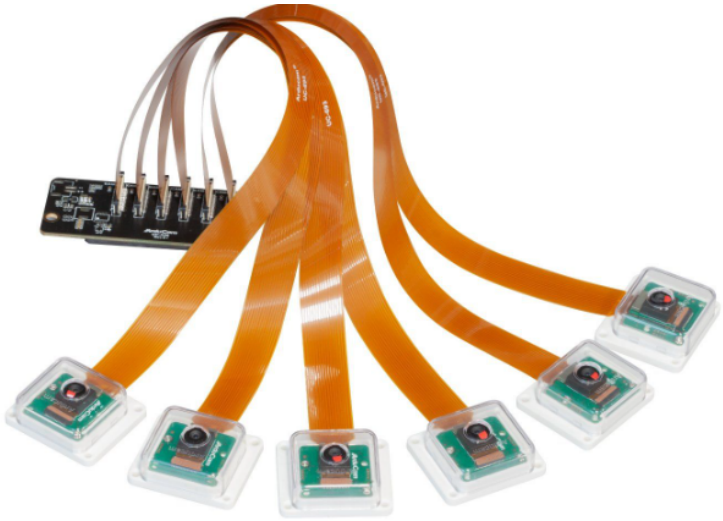

Figura 2.10: Multi-Camera Kit per NVIDIA Jetson

2.2.7 Scelta delle videocamere

In questa sezione si riporta la scelta effettuata tra le varie videocamere considerate per sviluppare il sistema.
Tutte le varie soluzione sono state riportate nella seguente tabella in cui si sono evidenziate le caratteristiche principali utili alla scelta finale. Per numero di videocamere si intende il numero massimo di videocamere collegabili ad una data

piattaforma considerando anche l'utilizzo di eventuali accessori come ad esempio dei multiplexers. Le videocamere, come ampiamente descritto precedentemente, hanno diverse modalità video con risoluzione ed FPS variabili quindi alla voce FPS si riporta il massimo valore di fotogrammi al secondo che una soluzione può fornire in una data modalità. Il FOV riportato è per la maggior parte dei casi quello orizzontale, fatta eccezione per le soluzioni multi-camera in cui è riportato il diagonale. Per le videocamere MEGA, come già accennato, non è presente nelle specifiche tecniche il frame rate.

Soluzione	**FPS**	**Risoluzione**	**FOV[°]**	**Colore**	**Videocamere**
Quad-camera RP5	120	64MP	120	RAW10	4
MEGA	/	5MP	90	RGB	4
IMX477	120	12MP	78	RAW10	4
OG02B10	120	2MP	110	RAW10	4
Pi-camera M3	120	12MP	120	RAW10	4
Multi-camera JN	60	12MP	120	RAW10	6

Tabella 2.2: Confronto tra videocamere analizzate

Si osservi come anche per le videocamere la soluzione più performante dal punto di vista delle specifiche tecniche risulta essere quella che coinvolge NVIDIA Jetson. Le videocamere MEGA non garantiscono prestazioni soddisfacenti dal punto di vista della risoluzione ma soprattutto del frame rate poichè impossibile determinare il numero di FPS delle singole videocamere. Tutte le videocamere risultano essere Rolling Shutter ad eccezione del solo sensore OG02B10, che seppur con una risoluzione più bassa garantisce tempi di risposta brevi. Il sensore IMX477 ha ottime prestazioni con il solo limite di avere un campo visivo modesto rispetto alle alternative valutate. La soluzione con Quad-camera per Raspberry Pi 5 ottiene la risoluzione maggiore anche se bisogna precisare che il valore riportato in tabella 2.2 fa riferimento alla combinazione dei quattro sensori, che hanno 16 MP ciascuno. Si scelgono per lo sviluppo del sistema le seguenti tipologie di videocamere:

- Arducam IMX477;
- Arducam OG02B10;
- Raspberry Pi Camera Module 3.

Le motivazioni che hanno portato a tale selezione sono differenti:

- Utilizzo di un sensore Global Shutter: il sensore OG02B10 è l'unico sensore tra le soluzioni valutate ad essere Global Shutter, è interessante valutare una soluzione con una videocamera che può garantire un'elevata dinamica ed un processo di elaborazione immagine breve.
- Caratteristiche differenti: i sensori OG02B10 e IMX477 sono profondamente diversi dal punto di vista delle specifiche. Il primo ha un ampio campo visivo ed un processamento delle immagini più veloce, il secondo un'elevata risoluzione ed un campo visivo ristretto. Raspberry Pi Camera Module 3, nella versione wide angle, è una combinazione dei due sensori. Risulta interessante valutare la risposta del sistema e le configurazioni geometriche ottimali al variare delle videocamere considerate.
- Limitazioni nell'utilizzo dei multi-camera kit: le soluzioni con diverse videocamere, come i kit per Raspberry e Jetson Nano sono complesse ed hanno specifiche tecniche ottime per il sistema da sviluppare. Tuttavia sono limitanti da un punto di vista dell'analisi, poichè ogni videocamere è identica ed hanno un costo elevato non giustificato dall'attività da sviluppare.

2.3 Componenti del banco

Si riportano, attraverso una tabella riassuntiva, i componenti scelti per sviluppare il feedback ottico per il sistema:

Componente	**Quantità**
Raspberry Pi CM5	1
Raspberry Pi Camera Module 3 IMX708 std.	2
Arducam IMX477	2
Arducam OG02B10	2
Arducam Lightweight Adjustable Mini Tripod Stand	4
Raspberry Pi Active Cooler	1
Alimentatore 27W USB-C	1
Raspberry Pi Mouse	1
Raspberry Pi Keyboard	1
Cavo micro HDMI per Raspberry Pi	1
Scheda SD 512GB	1
ADS1115 Modulo ADC 16 bit 4 canali per Raspberry Pi	1

Tabella 2.3: Lista componenti banco

La differenza principale tra Raspberry Pi 5 e Raspberry Pi Computer Module 5 (CM5) riguarda le caratteristiche hardware e l'utilizzo che si fa di tali piattaforme. Il CM5 necessita di una carrier board per essere utilizzata ed è possibile configurarlo scegliendo porte e funzioni garantendo una grande flessibilità nel suo utilizzo. Per quanto riguarda le videocamere, sono state acquistati due sensori per ogni tipologia scelta in modo tale da poter effettuare un'analisi più ampia e combinarle tra loro sfruttando i vantaggi di ogni singolo dispositivo. Per il sistema da sviluppare sono necessari ulteriori accessori: un sistema di raffreddamento, visto il grosso carico computazionale che la scheda deve sostenere, Mouse, Keyboard e cavo HDMI Raspberry per poter programmare la scheda mediante l'utilizzo di un monitor esterno. È necessario acquistare una scheda SD per il Raspberry Pi Compute Module 5 (CM5) perché, a differenza del Raspberry Pi 5, non include uno slot microSD integrato né un sistema di archiviazione predefinito accessibile. Un modulo

ADC (Analog to Digital Converter) è necessario ogni volta che si devono acquisire segnali analogici usando un sistema digitale come un Raspberry Pi, che non ha ingressi analogici.

Figura 2.11: Raspberry Computer Module 5 kit[25]

Si riportano anche i parametri intrinseci delle videocamere acquistate:

Sensore	**FOV**[°]	**Risoluzione**	**FPS**	**Colore**
IMX477	78	12MP	40/50/120	RAW10
OG02B10	110	2MP	60/80/120	RAW10
IMX708	66	12MP	40/50/120	RAW10

Tabella 2.4: Parametri intrinseci videocamere acquistate

Per il frame rate vengono riportati tutti i valori disponibili in base alle modalità video fornite dalla videocamera. Si precisa che cambiando il frame rate cambia anche la risoluzione in termine di prodotto pixel orizzontali e pixel verticali del sensore. Il FOV riportato risulta essere quello orizzontale.

Capitolo 3

Studio geometrico

In questo capitolo si approfondisce lo studio degli errori del sistema in base alla posizione delle telecamere e l'identificazione del numero di telecamere e configurazione ottimale, in termini di posizioni e orientamento rispetto alla slitta. Sulla base delle specifiche tecniche delle videocamere acquistate si è sviluppato un modello in grado di stimare la risoluzione in termine di rapporto pixel su millimetri al variare della posizione e orientamento della videocamera rispetto al sistema da controllare. É stato possibile effettuare questa analisi al variare del numero di videocamere, con videocamere uguali oppure miste. Successivamente, si è ottenuta la caratteristica statica della videocamera analizzando l'influenza che gli errori sistematici hanno su di essa. Tutto ciò ha richiesto una fase di ricerca per comprendere come modellizzare la videocamera e quali parametri sono rilevanti per la scelta del posizionamento.

3.1 Modello Pinhole

Il modello Pinhole è molto diffuso in ambito computer vision per modellizzare la videocamera. Esso è in grado di descrivere perfettamente la relazione tra i punti dello spazio 3D ed i corrispettivi punti proiettati sul piano immagine del sensore. Tuttavia, non include fenomeni come distorsioni geometriche o sfocatura, ma risulta comunque rilevante perchè in grado di descrivere in maniera efficace il processo di mappatura da coordinate 3D a coordinate 2D che una videocamera compie quando inquadra un determinato oggetto nello spazio. Il modello matematico, incluse le equazioni presentate e le immagini comprese tra la figura 3.1 a 3.3, è basato integralmente su quanto proposto in [26].

Consideriamo la proiezione centrale dei punti nello spazio 3D su un piano. Sia il centro di proiezione l'origine del sistema di riferimento euclideo e si consideri il piano $z = f$ detto *piano immagine* o *piano focale*. Secondo il modello Pinhole, un

punto nello spazio con coordinate $\mathbf{X} = (X, Y, Z)^T$ è mappato sul piano immagine nel punto in cui una linea passante per il centro di proiezione ed il punto stesso incontra il piano immagine.

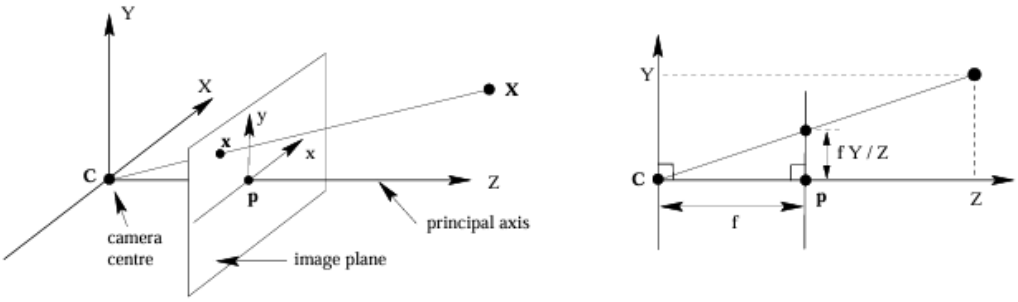

Figura 3.1: Modello Pinhole

C è il centro camera e **p** è il punto principale. Il centro camera è posizionato all'origine del sistema di riferimento. Si osservi che il piano immagine è posizionato di fronte al centro camera ad una distanza f che risulta essere la lunghezza focale. Dalla similitudine tra i due triangoli mostrati in figura 3.1 è possibile scrivere la trasformazione da 3D a 2D sul piano immagine come:

$$(X, Y, Z)^T \rightarrow (fX/Z, fY/Z)^T \tag{3.1}$$

Questa è un'applicazione dallo spazio euclideo tridimensionale $\mathbf{R}^3$ allo spazio euclideo bidimensionale $\mathbf{R}^2$. Il centro di proiezione è chiamato *camera centro* o *centro ottico*. La linea che parte dal centro ottico e risulta perpendicolare al piano immagine è chiamata *asse principale* ed il punto dove tale asse interseca il piano immagine è chiamato punto principale. Adesso introduciamo la notazione $\mathbf{X}$ per i punti nelle coordinate mondo rappresentati dalle coordinate di un vettore omogeneo $(X, Y, Z,1)^T$, $\mathbf{x}$ per il vettore avente tre coordinate del punto sul piano immagine e $\mathbf{P}$ per la matrice 3×4 detta *matrice di proiezione*:

$$\mathbf{x} = P\mathbf{X} \tag{3.2}$$

In componenti esprimibile come:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{3.3}$$

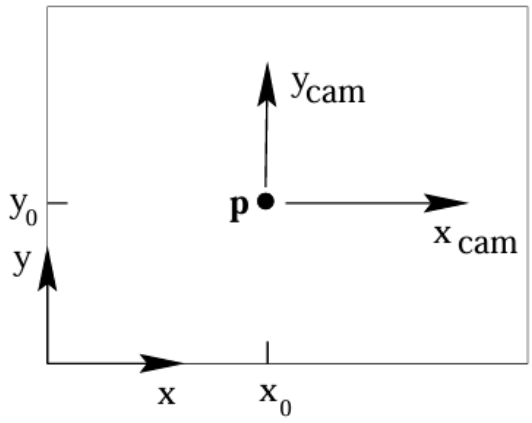

Figura 3.2: Coordinate immagine (x, y) e camera (x_{cam}, y_{cam})

In riferimento all'equazione 3.1 si assume che l'origine del sistema di riferimento è il punto principale. Nelle pratica non è sempre così quindi la mappatura viene modificata con un offset che tiene conto dell'origine del sistema di riferimento sul piano immagine:

$$(X, Y, Z)^T \rightarrow (fX/Z + p_x, fY/Z + p_y)^T \tag{3.4}$$

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{3.5}$$

Riscrivibile tutto nella forma compatta:

$$\mathbf{x} = K[I|0]\mathbf{X_{cam}} \tag{3.6}$$

dove K è detta *matrice di calibrazione*, che risulta essere:

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

Si esprime $(X, Y, Z,1)^T$ come $\mathbf{X_{cam}}$ per sottolineare come tale coordinate sono espresse nel sistema di riferimento camera. Questo sistema prende il nome di *camera coordinate frame*, rispetto al quale si esprimono i punti da proiettare nell'immagine tramite la matrice K, mentre l'asse principale della camera è l'asse z.

Bisogna effettuare delle considerazioni inerenti anche alla rotazione e traslazione di un punto nello spazio 3D rispetto alla videocamera. In generale, i punti nello spazio saranno espressi in termine di coordinate mondo dette *world coordinate frame*, le quali risultano differenti dalle coordinate camera. Questi due sistemi di riferimento sono correlati mediante delle trasformazioni geometriche di rotazione e traslazione. Se $\mathbf{X}$ è un vettore di 3 componenti rappresentante la posizione del punto nello spazio mediante coordinate mondo e $\mathbf{X_{cam}}$ rappresenta la posizione dello stesso punto nelle coordinate camera, allora:

$$\mathbf{X_{cam}} = R(\mathbf{X} - \tilde{\mathbf{C}}) \tag{3.8}$$

dove $\tilde{\mathbf{C}}$ rappresenta le coordinate del centro camera in sistema di riferimento mondo ed $\mathbf{R}$ è una matrice di rotazione 3×3 che tiene conto dell'orientazione del sistema di riferimento camera rispetto al sistema di riferimento mondo.

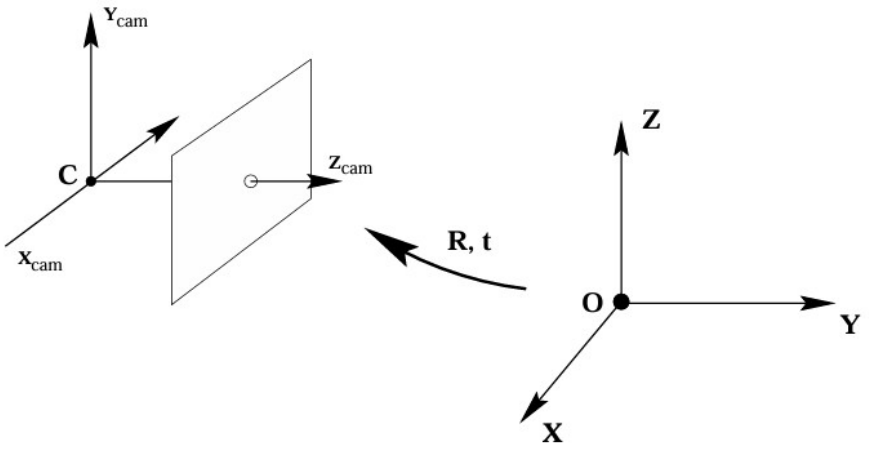

Figura 3.3: Trasformazione euclidea tra sistema di riferimento mondo e sistema di riferimento camera

L'ultima espressione è riscrivibile in coordinate omogenee come:

$$\mathbf{X}_{\text{cam}} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \mathbf{X} \tag{3.9}$$

In forma compatta, considerando anche la trasformazione prospettica sul piano immagine:

$$\mathbf{x} = K[I| - \tilde{\mathbf{C}}]\mathbf{X} \tag{3.10}$$

dove $\mathbf{X}$ ora si trova in un sistema di riferimento del mondo. Questa è la mappatura generale fornita da una camera pinhole. Si osserva che in una generica camera pinhole, $\mathbf{P} = \mathbf{KR}\begin{bmatrix}\mathbf{I} & -\tilde{\mathbf{C}}\end{bmatrix}$ ha 9 gradi di libertà: 3 per $\mathbf{K}$ (gli elementi f, p_x, p_y), 3 per $\mathbf{R}$ e 3 per $\tilde{\mathbf{C}}$. I parametri contenuti in $\mathbf{K}$ sono chiamati *parametri intrinseci* della camera, o *orientamento interno* della camera. I parametri di $\mathbf{R}$ e $\tilde{\mathbf{C}}$, che mettono in relazione l'orientamento e la posizione della camera con un sistema di riferimento del mondo, sono chiamati *parametri estrinseci*. Spesso è conveniente non rendere esplicito il centro camera e rappresentare invece la trasformazione dal mondo all'immagine come:

$$\tilde{\mathbf{X}}_{\text{cam}} = R\tilde{\mathbf{X}} + \mathbf{t} \tag{3.11}$$

In questo caso, la matrice di proiezione risulta essere:

$$P = K[R|t] \tag{3.12}$$

Da tale modello si evidenzia come:

- Vi è una netta distinzione fra parametri intrinseci ed estrinseci della videocamera.
- La matrice di calibrazione K effettua la trasformazione prospettica da punto nello spazio 3D a punto nel piano immagine.

La lunghezza focale f risulta essere un parametro fondamentale nella trasformazione prospettica ed incide anche sul campo visivo a disposizione per una determinata videocamera. La relazione fra lunghezza focale f, larghezza del sensore W Field Of View orizzontale θ_H è espressa dalla seguente formula[27]:

$$tan(\frac{\theta_H}{2}) = \frac{W}{2f} \tag{3.13}$$

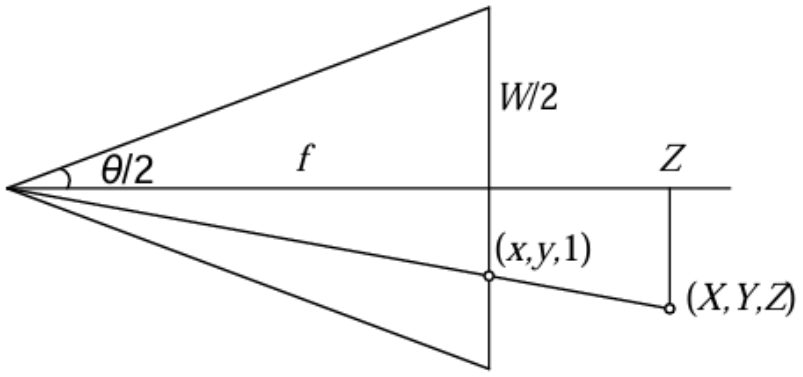

Figura 3.4: Relazione tra lunghezza focale e FOV [27]

Al diminuire della lunghezza focale aumenta il Field Of View della videocamera tuttavia vi è una perdita nella risoluzione dell'oggetto catturando minori dettagli della scena. Analogamente, dunque se alla lunghezza focale viene sostituita la distanza del sensore dall'oggetto d si ottiene, con videocamera frontale alla slitta senza nessun angolo d'inclinazione θ il Field Of View Orizzontale in metri:

$$W_{FOV} = 2\ d\ tan(\frac{\theta_H}{2}) \tag{3.14}$$

Tale formula è valida solo se la videocamera è posta frontalmente rispetto all'oggetto senza alcun angolo d'inclinazione. Se presente un angolo di inclinazione θ allora possono essere utilizzati due approcci:

- Trasformare le coordinate mondo del punto in coordinate nel piano immagine e verificare che non ricadano al di fuori della larghezza W del sensore.
- Approccio vettoriale in cui si considera un cono visivo di apertura θ_H.

3.2 Numero di videocamere

Il modello Pinhole permette di comprendere la motivazione per la quale esiste un numero minimo di videocamere per stimare in maniera accurata la posizione di un oggetto. Considerando $\mathbf{X}$ il vettore coordinate mondo di un punto nello spazio 3D e $\mathbf{x}$ vettore contenente le coordinate dello stesso punto nel piano immagine si è visto come tali vettori sono legati dall'equazione 3.1. Tale relazione può essere riscritta in termine di sistema lineare come segue:

$$A\mathbf{X} = \mathbf{b} \tag{3.15}$$

La soluzione di tale sistema, con un approccio ad i minimi quadrati, risulta essere[28]:

$$\hat{\mathbf{X}} = \left(\mathbf{A}^\top \mathbf{A}\right)^{-1} \mathbf{A}^\top \mathbf{b} \tag{3.16}$$

Vi sono diverse osservazioni che scaturiscono dalla risoluzione di tale sistema:

- Sono necessarie almeno due videocamere per ricostruire un punto 3D. Quando la trasformazione prospettica viene rielaborata per ottenere la forma dell'equazione 3.15, la matrice A diventa una matrice $2m \times 4$, per m videocamere, con ciascuna camera che contribuisce con una riga per ciascuna delle dimensioni x e y. Poiché il vettore $\mathbf{X}$ ha tre componenti è necessario un minimo di due camere per ricostruire la posizione di un punto nello spazio 3D.
- Le videocamere devono essere disposte in modo tale da formare un sistema ben condizionato. Se gli assi ottici sono paralleli il prodotto $\mathbf{A^T A}$ non risulta invertibile. Se gli assi ottici hanno un basso angolo di convergenza il sistema risulterà mal condizionato.

- Aumentando il numero di videocamere il sistema diventa sempre più sovra-condizionato aumentando la precisione nella ricostruzione della posizione dell'oggetto nel mondo 3D.

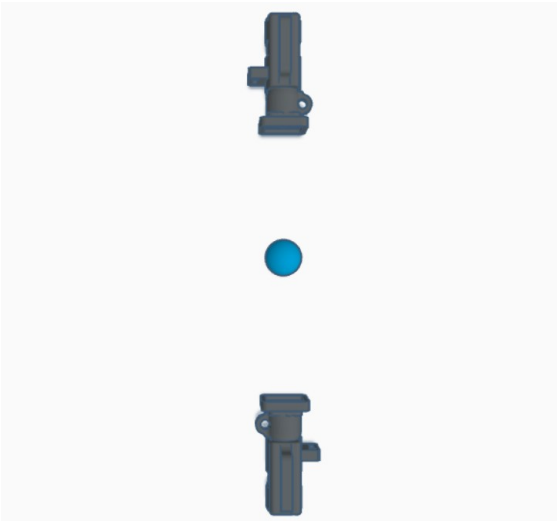

Figura 3.5: Videocamere collineari rispetto ad un punto nello spazio 3D

Figura 3.6: Videocamere con un basso angolo di convergenza rispetto ad un punto nello spazio 3D

Nella configurazione di figura 3.5 si può osservare come non vi è nessun angolo d'inclinazione tra gli assi ottici delle due videocamere, la posizione del punto non

può essere ricostruita. La configurazione di figura 3.6 ha un basso angolo di convergenza fra gli assi ottici delle videocamere, generando una ricostruzione della posizione errata e triangolazioni imprecise.

Un altro tema importante per valutare il numero ottimale di videocamere per lo sviluppo del sistema è come la qualità dei sensori scelti in termini di specifiche tecniche incide su tale numero. Attualmente, vi sono in commercio svariate tipologie di videocamere come webcam economiche, sensori per la prototipazione e videocamere industriali in grado di restituire un'elevata accuratezza. Con tale abbondanza di dispositivi, unito al fatto che spesso la risposta dipende molto dallo spazio di lavoro che si vuole monitorare, non è facile stabilire un iter preciso da seguire. In generale, come si è anche osservato nel paragrafo precedente, aumentare il numero di videocamere permette di ottenere ricostruzioni più precise. In ambienti di lavoro dove sono presenti diverse occlusioni statiche e dinamiche aumentare il numero di videocamera permette di coprire più facilmente l'intero volume di lavoro evitando zone in cui l'oggetto d'interesse non è visibile. *Katz,Aghajan e Haymaker* [28] hanno analizzato l'incertezza di misura della posizione per una rete di telecamere al variare di risoluzione e numero di videocamere in un ambiente simulato mediante metodo Montecarlo. Imposti questi due parametri, sono state valutate mille configurazioni casuali di videocamere nell'ambiente di lavoro ed i risultati sono stati mediati in termine di errore di localizzazione 3D e percentuale di volume di lavoro coperto. L'incertezza nella stima della posizione è stata definita da un errore quadratico medio tra la posizione dell'oggetto $\mathbf{X}$ e quella ricostruita attraverso le informazioni ottenute dalla videocamera $\hat{\mathbf{X}}$:

$$\mathbf{E}(\mathbf{X}, \hat{\mathbf{X}}) = \|\hat{\mathbf{X}} - \mathbf{X}\|^2 \tag{3.17}$$

I risultati hanno evidenziato che:

- Aumentare il numero di videocamere riduce l'incertezza di misura della posizione.
- L'incremento di risoluzione tende a ridurre l'incertezza di misura.
- La riduzione dell'incertezza di misura satura per valori di risoluzione elevata, non è necessario avere una rete con un numero elevato di videocamere.

Seppur tale studio sia diverso in termini di ordini di grandezza e sensoristica utilizzata suggerisce che in un trade-off tra semplicità del sistema e accuratezza della misura della posizione si potrebbe scegliere di limitare il numero di dispositivi senza avere un'eccessiva perdita di accuratezza. Riducendo il numero di videocamere si incorre tuttavia in una riduzione del volume di lavoro coperto rischiando di generare spazi morti. In questo caso, occorre un'attenta analisi delle specifiche del sensore,

in particolare del Field Of View. Aumentare il numero di videocamere implica ottenere i benefici sovraelencati, tuttavia vi sono altri aspetti da considerare, come l'aumento del carico computazionale o la sincronizzazione di più videocamere.

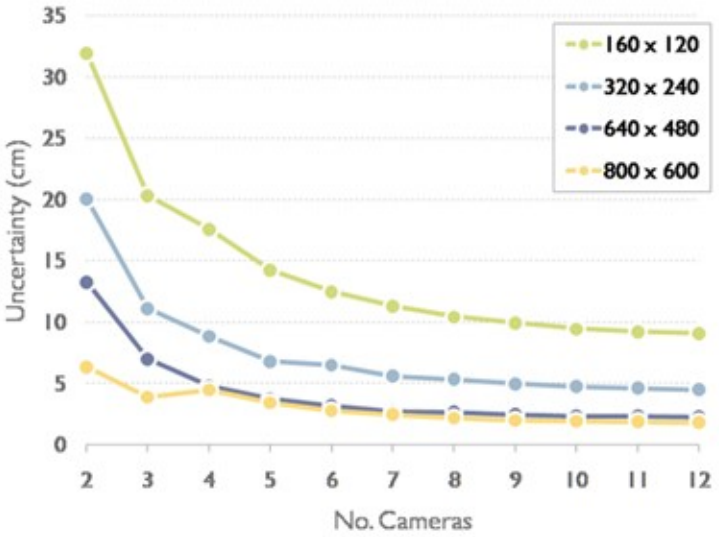

Figura 3.7: Incertezza nella misura della posizione in funzione di numero di videocamere e risoluzione [28]

3.3 Orientamento delle videocamere

Nel posizionamento e orientamento delle videocamere vi sono diversi aspetti critici da considerare:

- Angolo di convergenza tra videocamere: per evitare triangolazioni imprecise è necessario formare un angolo fra gli assi ottici delle videocamere.
- Occlusioni: bisogna evitare che le occlusioni interferiscano sul tracciamento della posizione dell'oggetto.
- Parametri intriseci della videocamera: lunghezza focale, dimensione del singolo pixel e FOV incidono inevitabilmente sull'accuratezza di misura della posizione.

Sanders-Reed [29] hanno studiato la propagazione dell'errore nella stima della posizione di un punto utilizzando due videocamere. Hanno evidenziato come tale errore sia correlato all'angolo di convergenza fra le due videocamere. Un angolo di convergenza di 0 o 180 gradi tra gli assi ottici delle due videocamere massimizza l'errore. L'errore è minimo quando le videocamere hanno assi ottici perpendicolare dunque 90 gradi. Tutto ciò è in accordo con quanto discusso sulla

soluzione del sistema lineare del precedente paragrafo. Se le videocamere hanno assi ottici allineati, anche piccoli errori nella misura portano a grandi incertezze nella posizione 3D. Un intervallo accettabile è compreso fra i 40 ed 140 gradi.

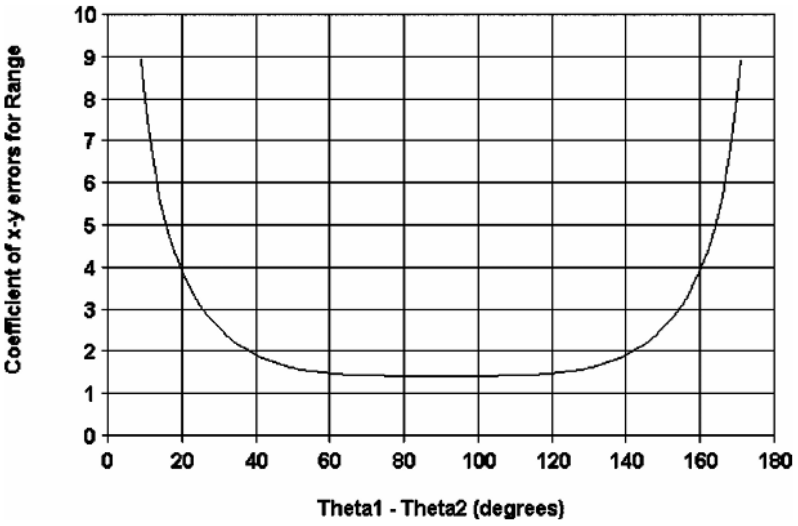

Figura 3.8: Propagazione dell'errore in funzione dell'angolo di convergenza fra due videocamere[29]

Tuttavia, questi studi si basavano su due assunzioni forti:

- Tutte le videocamere erano su un piano alla medesima distanza dal centro del bersaglio.
- Le videocamere erano sempre orientate verso il target che non era mai fuori campo visivo.

Spesso per il posizionamento delle videocamere si sviluppano metodi probabilistici che coinvolgono diversi parametri per valutare la bonta di una configurazione. Definita una griglia di punti si effettuano tutte le combinazioni al variare del numero di videocamere. Ad esempio, in [30] si descrive lo sviluppo di un metodo basato sui vincoli di risoluzione, messa a fuoco e occlusioni. Considerando un oggetto 3D esso viene diviso in diverse superfici visibile, e per ciascuna superficie calcolano la configurazione ottimale che rispetta i vincoli di risoluzione, messa a fuoco e occlusioni. Per ogni parametro identificano un volume 3D in cui il vincolo è soddisfatto per poi calcolare dall'intersezione dei vari volumi le posizioni valide per le videocamere. Il tema dell'occlusione è spesso trattato per ricercare la configurazione ottimale di un sistema di videocamere. *Chen e Davis* [31] hanno sviluppato un modello probabilistico per valutare delle configurazioni di videocamere in presenza

di occlusioni dinamiche. Considerano le occlusioni come dei piani verticali e vicini al target con estensione infinita, dividono lo spazio 3D in due zone di cui solo una in grado di garantire visibilità. *Rahimian e Kearney* [32] hanno migliorato questo modello considerando la degradazione della risoluzione con la distanza, l'effetto dell'angolo di convergenza fra gli assi ottici e tutte le possibili orientazioni degli occludenti. Le occlusioni, sia statiche che dinamiche, sono disturbi spesso presenti in sistemi complessi di questo tipo. Ad esempio, nell'attività sviluppata, vi sono diverse occlusioni che impediscono il posizionamento della videocamera dietro la slitta.

3.4 Analisi videocamere in termine di risoluzione

Il tema del posizionamento delle videocamere deve essere necessariamente trattato considerando i parametri intriseci delle videocamere scelte per sviluppare il sistema. Risulta necessario trovare delle configurazioni che permettono di sfruttare al massimo le caratteristiche dei sensori in modo da rendere il più accurata possibile la misura della posizione. Sulla base del modello Pinhole, è stato sviluppato un codice MATLAB in grado di stimare la risoluzione come rapporto pixel su millimetri. Si considera il piano x-z con origine del sistema di riferimento mondo a metà corsa della slitta in mondo tale che quando il punto si trova alle coordinate $X_W = 0$ $Z_W = 0$ si trovi precisamente in mezzeria. il centro camera ha coordinate mondo generiche:

$$\tilde{\mathbf{C}} = (x_0, z_0) \tag{3.18}$$

se $x_0 = 0$ la videocamera si trova precisamente in mezzeria ad una distanza z_0 dal punto quando esso si trova a metà corsa. Di seguito si riporta la configurazione del modello per maggiore chiarezza:

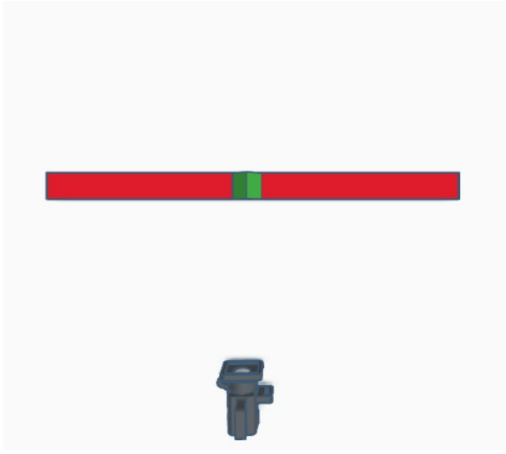

Figura 3.9: Configurazione modello

Per la stima della risoluzione spaziale sono state eseguite le seguenti operazioni:

- passaggio da coordinate mondo a coordinate camera;
- trasformazione prospettica da coordinate camera a coordinate su piano immagine mediante applicazione matrice di calibrazione K;
- derivata della coordinata nel piano immagine u rispetto alla coordinata X_W.

Considerando la sola traslazione rigida l'espressione 3.8 si modifica in:

$$\mathbf{X_{cam}} = (\mathbf{X_W} - \tilde{\mathbf{C}}) \tag{3.19}$$

In componenti:

$$X_{cam} = X_W - x_0 \tag{3.20}$$
$$Z_{cam} = Z_W - z_0 \tag{3.21}$$

Considerando una rotazione di un angolo θ attorno all'asse Y_{cam} la matrice di rotazione diviene:

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.22}$$

Allora le coordinate nel punto nel sistema riferimento camera risultano essere:

$$X_{cam} = (X_W - x_0)cos\theta - z_0 sin\theta \tag{3.23}$$
$$Z_{cam} = -(X_W - x_0)sin\theta - z_0 cos\theta \tag{3.24}$$

Si tenga presente che per il punto appartenente alla slitta $Z_W = 0$ mentre X_W varia in base alla posizione del punto sulla slitta. Quando $X_W = 0$ il punto si trova a metà corsa. Una volta ottenute le coordinate del punto rispetto al sistema di riferimento camera si possono ottenere le coordinate del punto sul piano immagine:

$$u = f\frac{X_{cam}}{Z_{cam}} + p_x \tag{3.25}$$
$$v = f\frac{Y_{cam}}{Z_{cam}} + p_y \tag{3.26}$$

Poichè si ipotizza che il punto si muova solo lungo la coordinata X_W interessa solo la coordinata u per la stima della risoluzione:

$$u = f\frac{(X_W - x_0)cos\theta - z_0 sin\theta}{-(X_W - x_0)sin\theta - z_0 cos\theta} + p_x \tag{3.27}$$

Per stimare la risoluzione si valuta la derivata della coordinata u rispetto alla coordinata X_W:

$$\frac{du}{dX_W} = f\frac{Z_{cam}cos\theta + X_{cam}sin\theta}{Z_{cam}^2} \tag{3.28}$$

Sia N_x il numero di pixel orizzontali a disposizione del sensore ed W la dimensione orizzontale del sensore. Allora la dimensione di un pixel in millimetri sul sensore risulta essere:

$$s_x = \frac{W}{N_x} \tag{3.29}$$

la risoluzione in termine di $[pixel/mm]$ risulterà:

$$Res = \frac{1}{s_x}\frac{du}{dX_W} \tag{3.30}$$

Per tenere conto dell'effetto prospettico ed del fatto che quando il punto si trova in posizioni diverse dalla mezzeria la distanza fra videocamera e oggetto aumenta portando ad una diminuzione della risoluzione si sceglie di porre nell'espressione di Z_{cam}:

$$z_0 \approx \sqrt{z_0^2 + (X_W - x_0)^2} \tag{3.31}$$

Non si sta dunque considerando la coordinata del punto nel sistema riferimento camera, la quale risulterebbe costante e pari a z_0, ma la distanza euclidea del punto dalla videocamera nel piano $x - z$. Il modello sviluppato, seppur trascura alcuni fenomeni come la distorsione ottica o la sfocatura ne ha altri che rendono fedeltà alla videocamera e al suo funzionamento:

- La risoluzione è funzione della lunghezza focale f. Aumentare la lunghezza focale significa diminuire il campo visivo ma osservare con maggior dettaglio gli oggetti in termine di pixel.

- La risoluzione è dipendente dalla distanza dell'oggetto dalla camera e della sua inclinazione.

La risoluzione risulta molto influenzato dall'angolo d'inclinazione θ. Per valori elevati, ad esempio maggiore di 20 gradi, il denominatore diminuisce portando la risoluzione a valori elevati e poco realistici. Nel modello è stato incluso anche il calcolo del Field Of View mediante un approccio vettoriale. In una generica configurazione con angolo d'inclinazione θ l'asse ottico è rappresentato dal seguente versore nel piano $x - z$:

$$\hat{d} = [sin\theta, cos\theta] \tag{3.32}$$

il cono visivo ha apertura θ_H e le estremità del cono sono descritte dai seguenti versori:

$$\hat{s} = [sin(\theta - \frac{\theta_H}{2}), cos(\theta - \frac{\theta_H}{2})] \tag{3.33}$$

$$\hat{d} = [sin(\theta + \frac{\theta_H}{2}), cos(\theta + \frac{\theta_H}{2})] \tag{3.34}$$

Dato il vettore $\vec{v}$ che identifica la posizione del punto sulla slitta rispetto alla videocamera è possibile calcolare l'angolo d'inclinazione ϕ tra $\hat{n}$ e $\vec{v}$:

$$\cos(\phi) = \frac{\vec{v} \cdot \vec{d}}{\|\vec{v}\| \cdot \|\vec{d}\|} \tag{3.35}$$

Un punto si trova al di fuori del Field Of View se è soddisfatta la seguente disequazione:

$$\phi > \frac{\theta_H}{2} \tag{3.36}$$

É interessante sapere, dati x_0 e z_0, qual è l'angolo d'inclinazione θ_1 per il quale il cono visivo interseca con una sua estremità il fine corsa sinistro della slitta. Quando la slitta si trova in quella posizione ha coordinata X_W pari a $-x_{end}$. Bisogno risolvere il seguente sistema proveniente dall'intersezione punto retta sul piano:

$$\begin{cases} x_0 + \lambda sin(\theta_1 - \frac{\theta_H}{2}) = -x_{end} \\ \lambda cos(\theta_1 - \frac{\theta_H}{2}) = z_0 \end{cases} \tag{3.37}$$

É possibile ottenere θ_1 con la seguente relazione:

$$\theta_1 = \arctan(\frac{-x_{end} - x_0}{z_0}) + \frac{\theta_H}{2} \tag{3.38}$$

Analogamente se si volesse l'angolo θ_2 per cui si ha l'intersezione dell'estremità destra del cono visivo con il fine corsa destro:

$$\theta_2 = \arctan(\frac{x_{end} - x_0}{z_0}) - \frac{\theta_H}{2} \tag{3.39}$$

Per una singola videocamera, attraverso questo modello, è possibile stimare la risoluzione al variare della posizione rispetto alla slitta, dell'orientamento e della modalità video. Per il setup sperimentale ricordiamo che sono disponibili tre tipologie di sensori:

- Arducam 12MP IMX477 Autofocus High Quality Camera;
- Arducam 2MP Global Shutter OG02B10;

- Raspberry 12MP Pi Camera Module 3 IMX708.

Inizialmente, si è analizzata la configurazione con una sola videocamera, dove la posizione ottimale risulta essere in mezzeria, frontale rispetto alla slitta. Entrambi i sensori hanno diverse modalità video che permettono di variare il frame rate e la risoluzione in termine di numero di pixel orizzontale e verticale:

Sensore	**Video mode 1**	**Video mode 2**
IMX477	2028×1080p50	1332×990p120
OG02B10	1600×1300p60	1280×720p120
IMX708	1920×1080p50	640×480p120

Tabella 3.1: Alcune Modalità video sensori disponibili

Si analizzano i risultati ottenuti al variare della modalità video per le diverse tipologie di sensori. Fissata una distanza comune dalla slitta pari ad 80 millimetri si ottengono le seguenti curve di risoluzione:

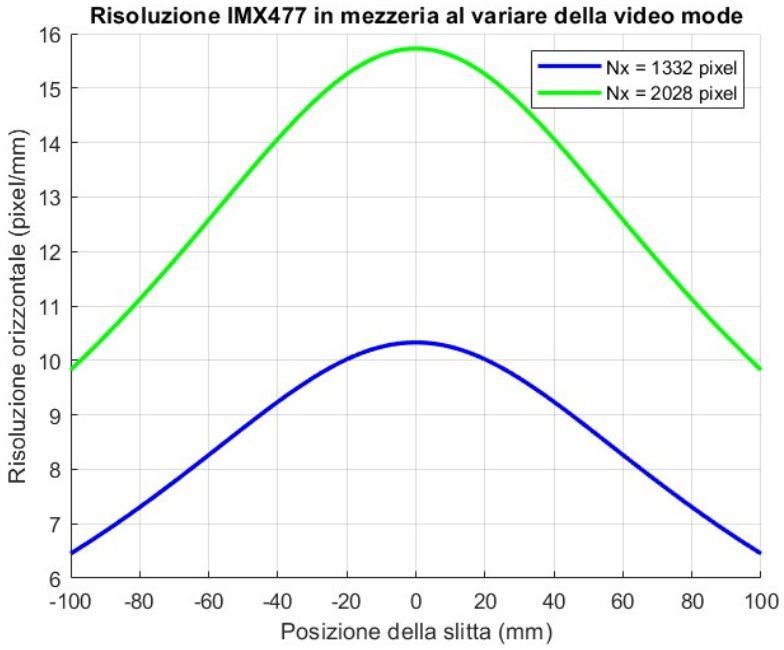

Figura 3.10: Risoluzione IMX477 al variare della video mode

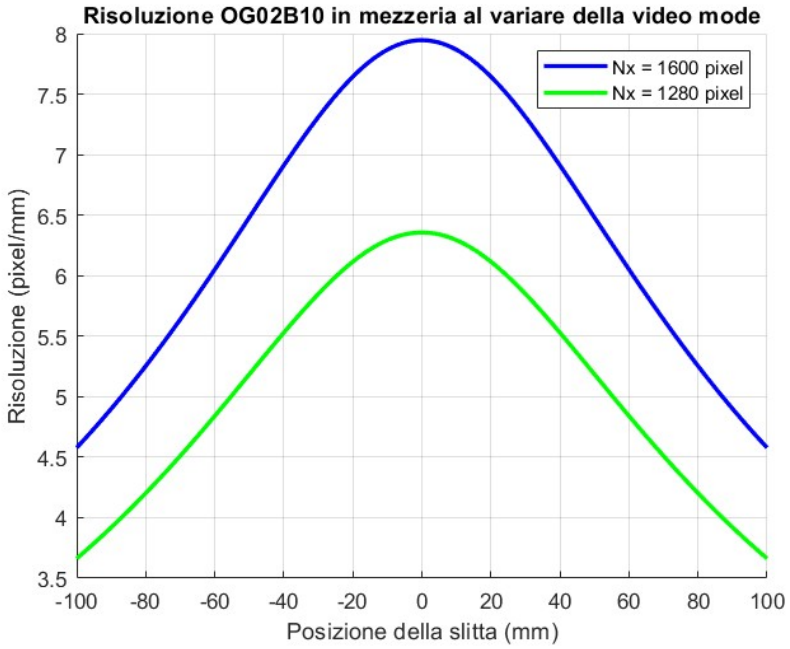

Figura 3.11: Risoluzione OG02B10 al variare della video mode

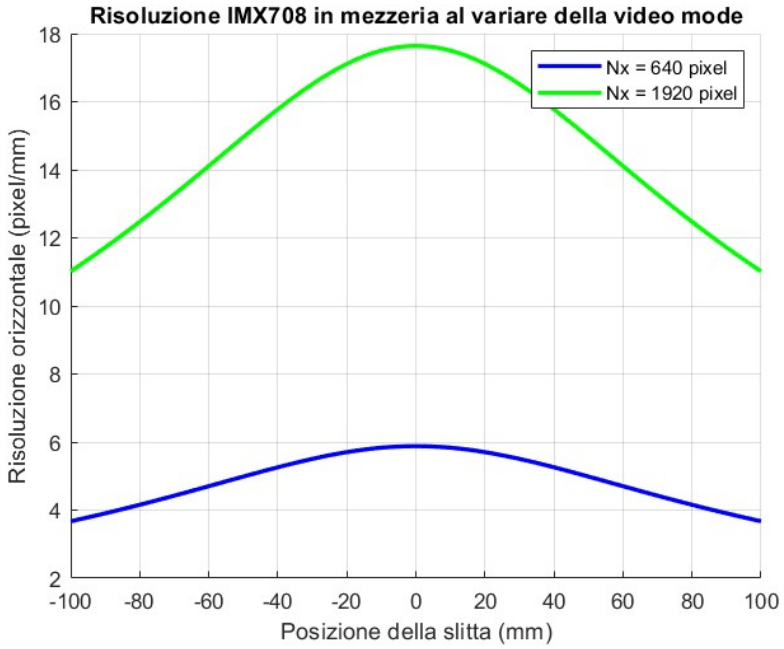

Figura 3.12: Risoluzione IMX708 al variare della video mode

Il modello conferma come la risoluzione dell'IMX477 risulti molto più elevata rispetto alla videocamera Global Shutter, a parità di distanza dall'oggetto da monitorare. Da questo punto di vista si può pensare di utilizzare la videocamera con sensore IMX477 per l'inquadratura frontale poichè in grado di restituire maggiori dettagli. La visione frontale è di particolare importanza nel processo di determinazione della posizione del cursore e quindi avere una videocamera molto più prestante sotto questo punto di vista risulta essere decisivo in termini di accuratezza della lettura di posizione per il banco prova.

Per quanto riguarda la videocamera con sensore IMX708 i risultati evidenziano come la risoluzione sia molto bassa per la modalità video a massimo frame rate il che potrebbe limitare l'utilizzo di tale videocamera se si vuole un'elevata dinamica per il sistema. La videocamera, in termine di risoluzione massima, risulta essere la migliore tra le tipologie scelte, il basso Field Of View potrebbe però limitarne ancora una volta l'utilizzo. Attraverso il modello sviluppato è possibile anche valutare il campo visivo delle videocamere e valutare la distanza limite al di sotto della quale il cursore della stampante non risulta rientrare nel campo visivo del sensore. Si osserva che a parità di distanza, il Field Of View di OG02B10 risulta maggiore rispetto a quello di IMX477 e di IMX708 in accordo con le specifiche tecniche ritrovate sul sito Arducam.

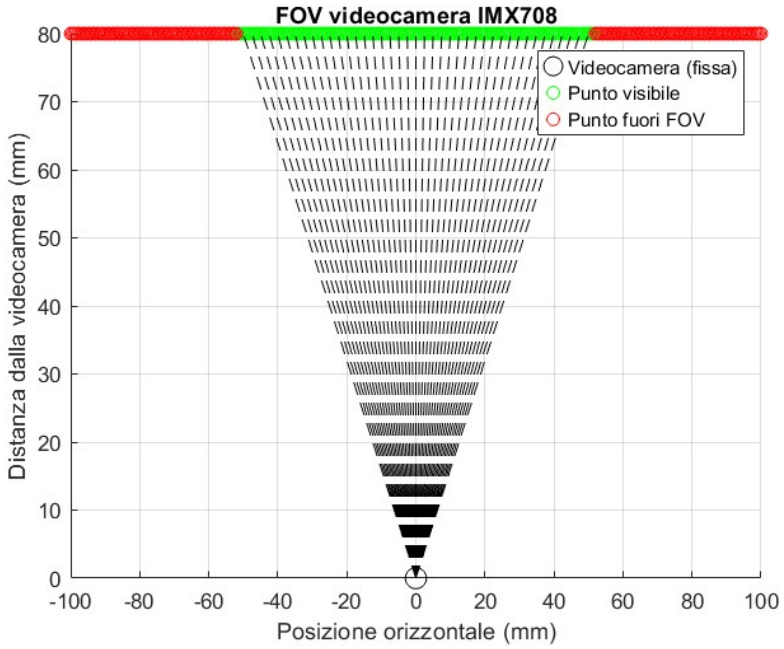

Figura 3.13: FOV videocamera IMX708 in mezzeria ad una distanza di 80 mm dalla slitta

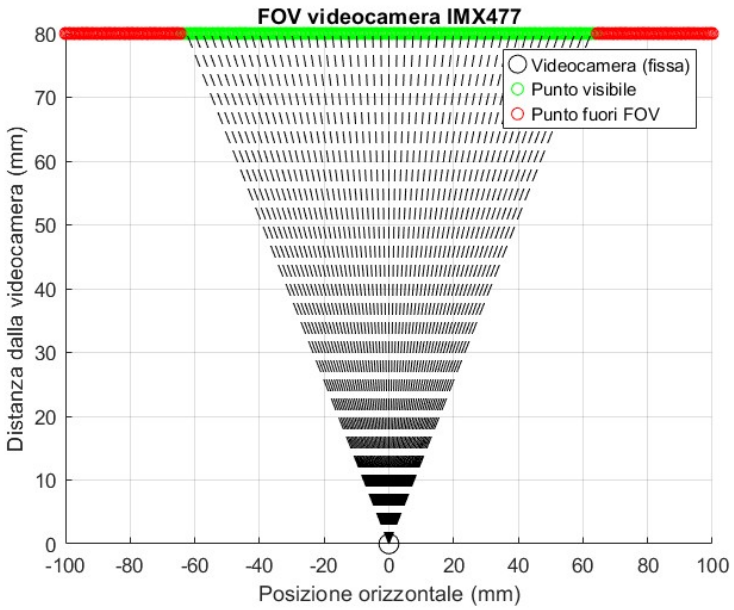

Figura 3.14: FOV videocamera IMX477 in mezzeria ad una distanza di 80 mm dalla slitta

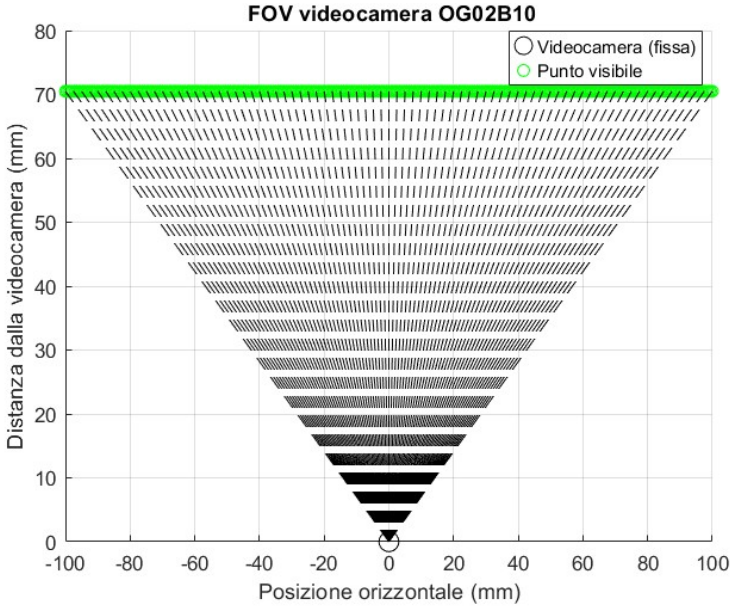

Figura 3.15: FOV videocamera OG02B10 in mezzeria ad una distanza di 70 mm dalla slitta

Per riuscire ad avere un campo visivo che riesca a monitorare la posizione del cursore per tutte le sue posizioni di funzionamento con una delle tre videocamere poste in mezzeria bisogna rispettare una distanza minima di:

- 125 millimetri per IMX477;
- 70 millimetri per OG02B10;
- 155 millimetri per IMX708.

È naturale pensare che aumentare la distanza permette di aumentare il campo visivo a disposizione della videocamera tuttavia vi è un'inevitabile perdita di risoluzione e quindi di accuratezza nella lettura posizione. La stessa IMX477 all'aumentare della distanza, seppur con una modalità video che garantisce la massima risoluzione, osserva a modello una perdita nella risoluzione con performance che diventano paragonabili a quelle della videocamera Global Shutter.

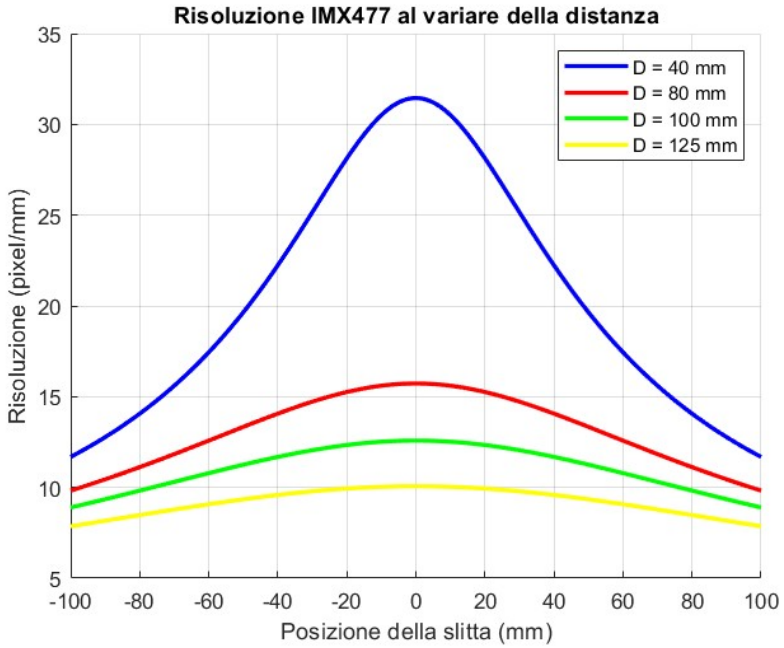

Figura 3.16: Risoluzione videocamera IMX477 posta in mezzeria al variare della distanza dalla slitta, modalità 2028 × 1080p50

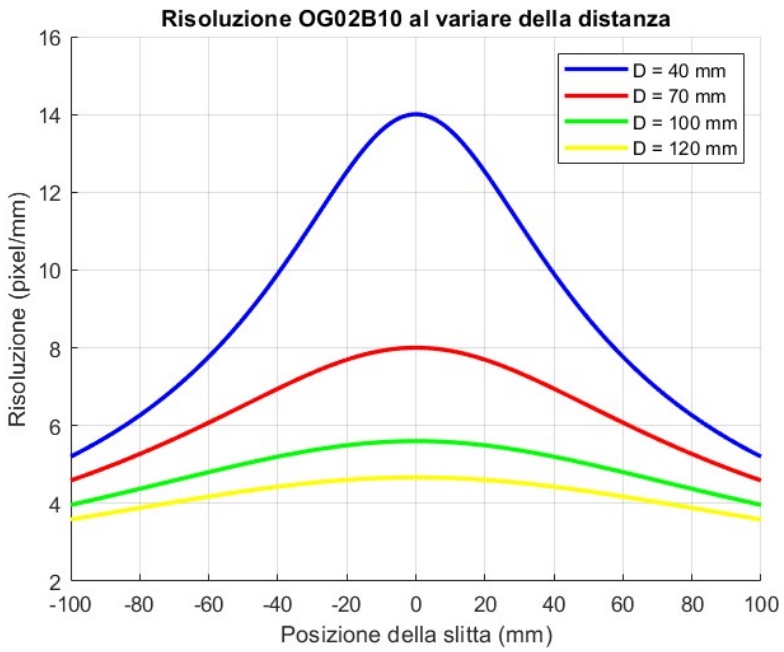

Figura 3.17: Risoluzione videocamera OG02B10 posta in mezzeria al variare della distanza dalla slitta, modalità 1600×1300p60

Per il sensore IMX477 aumentare la distanza dalla slitta in modo da ottenere un campo visivo che riesca ad inquadrare in qualsiasi istante il cursore risulta sconveniente in termini di risoluzione perdendo tutti i vantaggi che questo sensore ha rispetto al Global Shutter nell'accuratezza della lettura di posizione. Oltre alla distanza dalla slitta, è interessante osservare come l'angolo d'inclinazione della videocamera rispetto alla slitta influenza la risoluzione. Considerando positiva una rotazione oraria attorno all'asse y della videocamera, si può facilmente intuire come il picco di risoluzione non si trovi più in mezzeria, poichè la videocamera non inquadra più fontalmente il cursore della stampante. A causa dell'effetto prospettico, il contributo di Z_{cam} diminuisce all'aumentare dell'angolo d'inclinazione con un conseguente incremento della risoluzione massima. Tale effetto, come già accennato in precedenza, porta a risultati che non risultano validi. In una successiva analisi per valutare la configurazione ottimale utilizzando due videocamere l'angolo d'inclinazione verrà limitato ad un valore massimo di 20 gradi. E' interessante confrontare le risoluzioni delle due videocamere se posizionate alle estremità della slitta ed inclinate di un angolo θ.

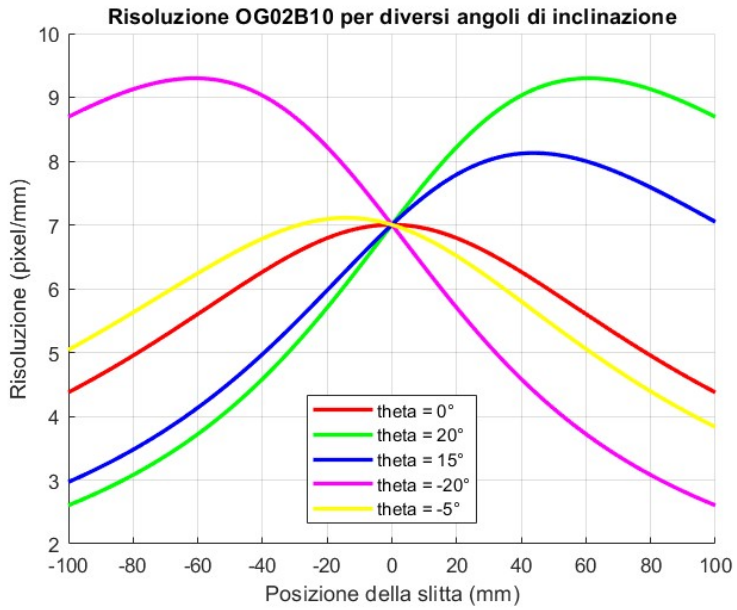

Figura 3.18: Risoluzione per videocamera OG02B10 posta in mezzeria per diversi angoli d'inclinazione

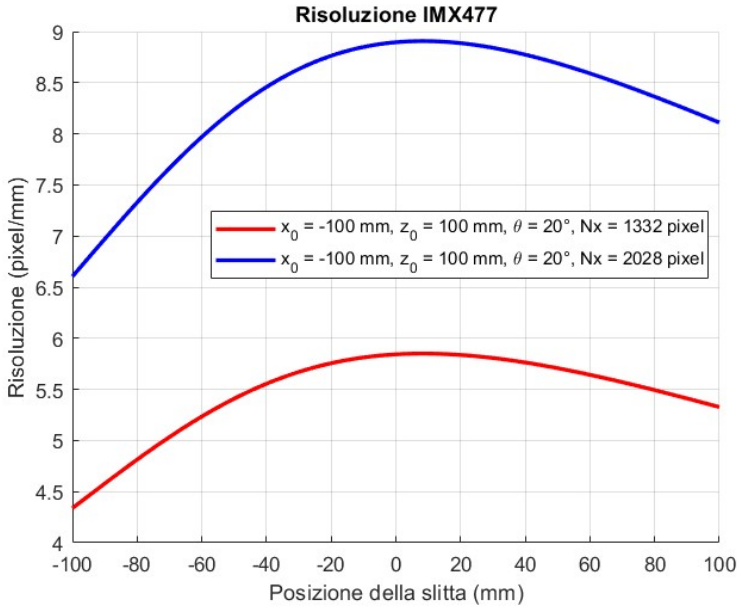

Figura 3.19: Risoluzione IMX477 videocamera posizionata in corrispondenza dell'estremo sinistro della corsa della slitta

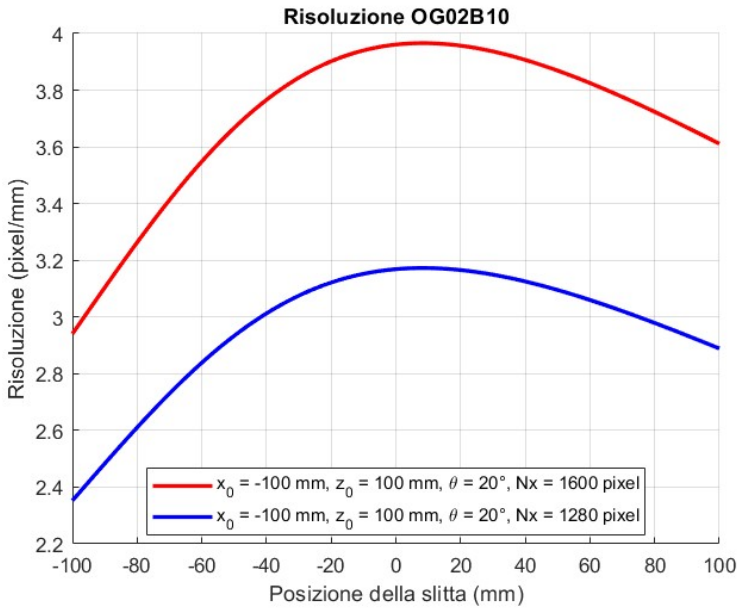

Figura 3.20: Risoluzione OG02B10 videocamera posizionata in corrispondenza dell'estremo sinistro della corsa della slitta

Nonostante la videocamera IMX477 conservi una risoluzione migliore in entrambe le modalità video si evince come la differenza rispetto alla OG02B10 in questi termini sia meno marcata rispetto al caso in mezzeria. In generale, per una singola videocamera la soluzione ottimale risulta essere in mezzeria in modo tale da ottenere una risoluzione abbastanza costante in tutto il campo di funzionamento del sistema. Spostando la videocamera in altre posizioni, con altri angoli d'inclinazioni, si potrebbero ottenere configurazioni a risoluzione media anche più elevata rispetto al caso in mezzeria, tuttavia non si avrebbe una distribuzione di risoluzione omogenea all'interno del campo di funzionamento. Poichè i sensori scelti sono profondamente diversi in termine di Field Of View e risoluzione, dai risultati ottenuti si evince che:

- Se si vuole dare maggiore importanza ai tempi di risposta e ottenere una maggiore larghezza di banda del sistema, si preferisce il sensore OG02B10 poichè ha diverse modalità video a frame rate elevate ed un sistema di acquisizione immagine più dinamico.
- Se si vuole dare maggiore importanza all'accuratezza della lettura posizione i sensori IMX477 e IMX708, con i loro 12 MP sono la scelta ottimale.
- IMX708 possiede una risoluzione leggermente superiore rispetto ad IMX477 se entrambe sono posizionate in mezzeria in modo da massimizzare la corsa.

3.5 Analisi configurazione a due videocamere

Si vuole stabilire la configurazione ottimale nel caso si volessero utilizzare due videocamere per la misura della posizione del cursore della stampante. Per fare ciò, attraverso un modello MATLAB, è possibile valutare tutte le combinazioni possibili di videocamere uguali oppure miste, per diversi angoli d'inclinazione rispetto alla stampante. Si assume che:

- Si dispongono le videocamere solo dal lato frontale della stampante poichè lateralmente oppure dietro le occlusioni presenti renderebbero impossibile il tracciamento dell'oggetto.
- L'analisi viene effettuata su una griglia di 99 punti sul piano $x - z$.

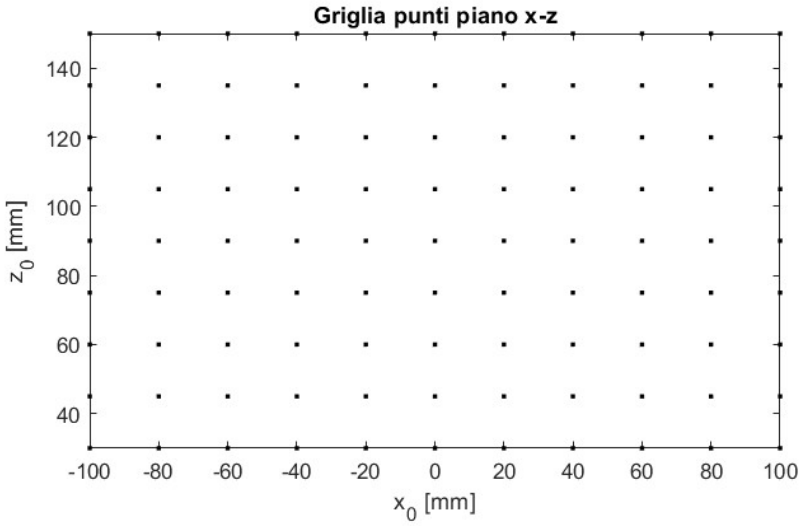

Figura 3.21: Griglia posizioni analizzate

Come si può osservare, non vengono considerati punti troppo vicini alla stampante. Ogni punto è distanziato dagli adiacenti di un passo costante in direzione x ed uno in direzione z. I punti analizzati sono stati scelti in modo tale che fosse presente la posizione in mezzeria dove la singola videocamera è in grado senza alcun tipo di angolo di inclinazione di ricoprire tutti i punti di funzionamento della slitta. La distanza massima considerata è stata 150 millimetri.

Per ogni combinazione è possibile calcolare alcuni parametri d'interesse come:

- risoluzione media R_{avg};
- risoluzione massima R_{max};
- risoluzione minima R_{min};
- risoluzione RMS R_{RMS};
- risoluzione al centro slitta R_c;
- percentuale di corsa coperta.

La stima della risoluzione, è stata opportunamente modificata in modo tale da considerare in tutti i punti di funzionamento il campo visivo effettivo della videocamera. Per ogni configurazione, se in un range di funzionamento le videocamere inquadrano contemporaneamente l'oggetto si considera la risoluzione maggiore, se invece nessuna delle due è in grado di rilevare il cursore si considera risoluzione nulla. Bisogna anche effettuare delle considerazioni sull'angolo d'inclinazione. Si è scelto di considerare per ogni posizione tre angoli d'inclinazione rispetto alla slitta:

- angolo d'inclinazione nullo con videocamera frontale rispetto alla slitta;
- angolo d'inclinazione in modo tale che il bordo sinistro del cono di visione intersechi il finecorsa sinistro della slitta;
- angolo d'inclinazione in modo tale che il bordo destro del cono di visione intersechi il finecorsa destro della slitta.

Non era infatti pratico pensare di poter analizzare tutte le configurazioni possibili al variare di tutti gli angoli d'inclinazione. Bisogna inoltre tenere presente che per soddisfare questi vincoli sugli angoli, in alcune posizioni gli angoli d'inclinazione raggiungo valori molto elevati influenzando inevitabilmente i risultati in termine di risoluzione complessiva. Ad esempio, se una generica videocamera si trova posizionata in corrispondenza dell'estremità sinistra della slitta, con asse ottico perpendicolare alla slitta, per intersecare il bordo sinistro del cono con l'estremità della slitta dovrà effettuare una rotazione attorno all'asse y in senso orario di $\frac{\theta_H}{2}$. É necessario dunque, escludere le configurazioni che implicano un angolo d'inclinazione maggiore di 20 gradi poichè irrealizzabili nella pratica.

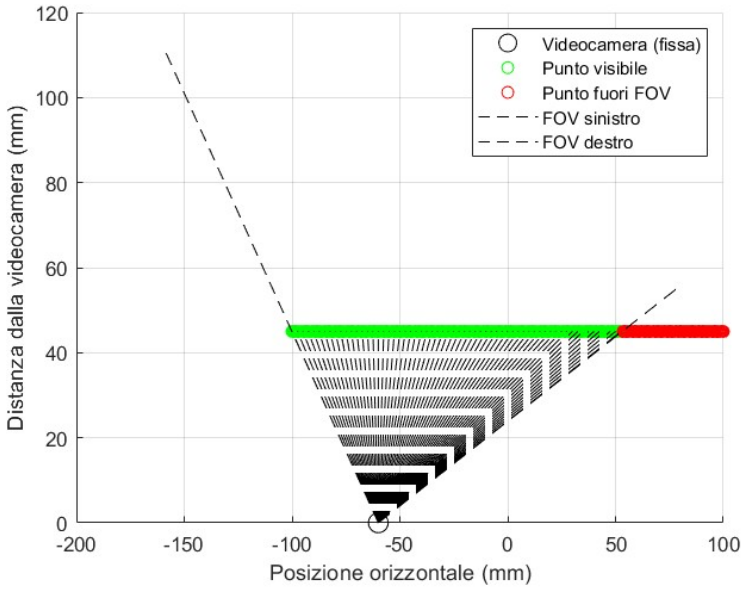

Figura 3.22: Esempio generico cono visivo videocamera OG02B10 che interseca con il bordo sinistro il finecorsa della slitta

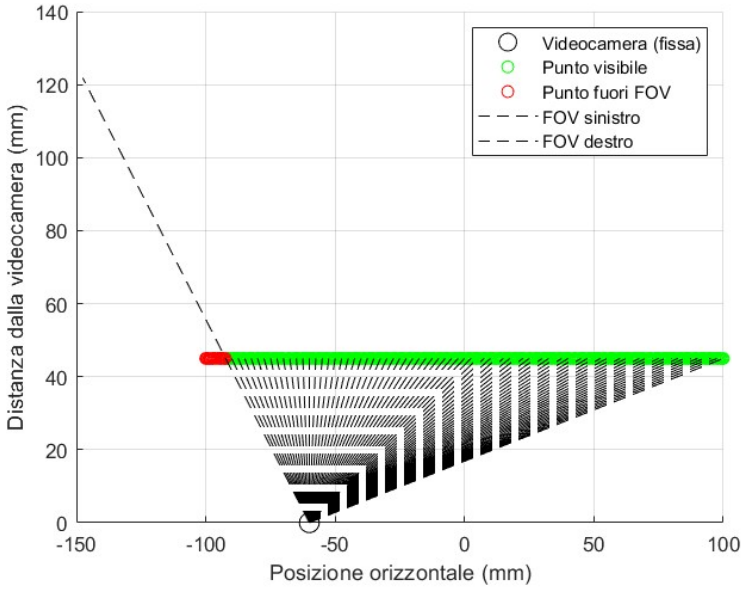

Figura 3.23: Esempio generico cono visivo videocamera OG02B10 che interseca con il bordo destro il finecorsa della slitta

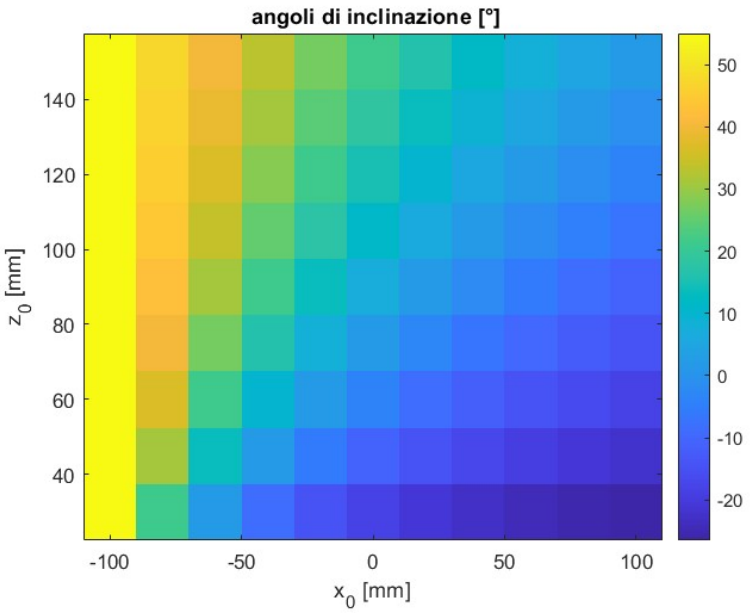

Figura 3.24: Heatmap posizione - angoli assunti dalla videocamera OG02B10 per intersecare l'estremità della slitta con il bordo sinistro del cono visivo

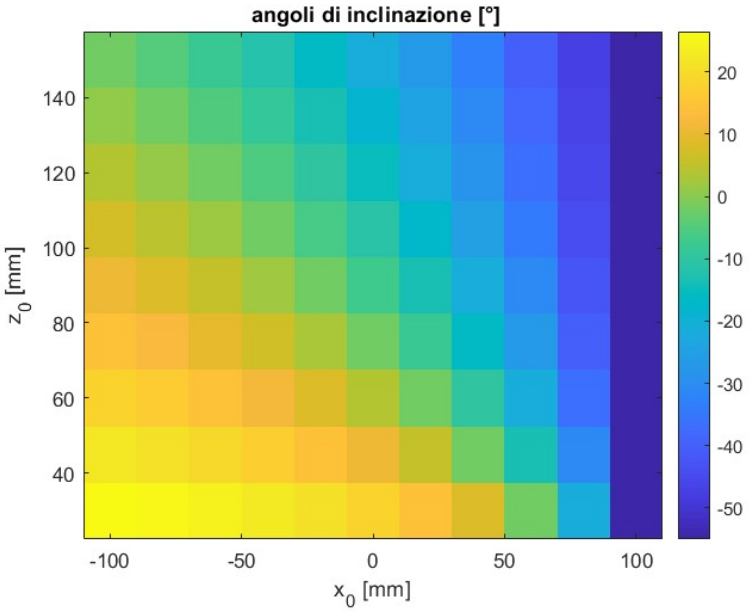

Figura 3.25: Heatmap posizione - angoli assunti dalla videocamera OG02B10 per intersecare l'estremità della slitta con il bordo destro del cono visivo

Tra tutte le combinazioni analizzate si scelgono quelle che massimizzano i parametri elencati precedentemente. La miglior configurazione è quella in grado di massimizzare contemporaneamente corsa coperta e risoluzione media. Si considera come modalità video quella in grado di massimizzare la risoluzione.

3.5.1 Configurazione videocamere IMX477

Per la videocamera Rolling Shutter con sensore IMX477 la miglior configurazione in termine di risoluzione media e corsa coperta risulta essere la seguente:

θ_1 [°]	x_1 [**mm**]	z_1 [**mm**]	θ_2 [°]	x_2 [**mm**]	z_2 [**mm**]
-14.26	0	75	14.26	40	45

Tabella 3.2: Parametri geometrici configurazione ottimale videocamere IMX477

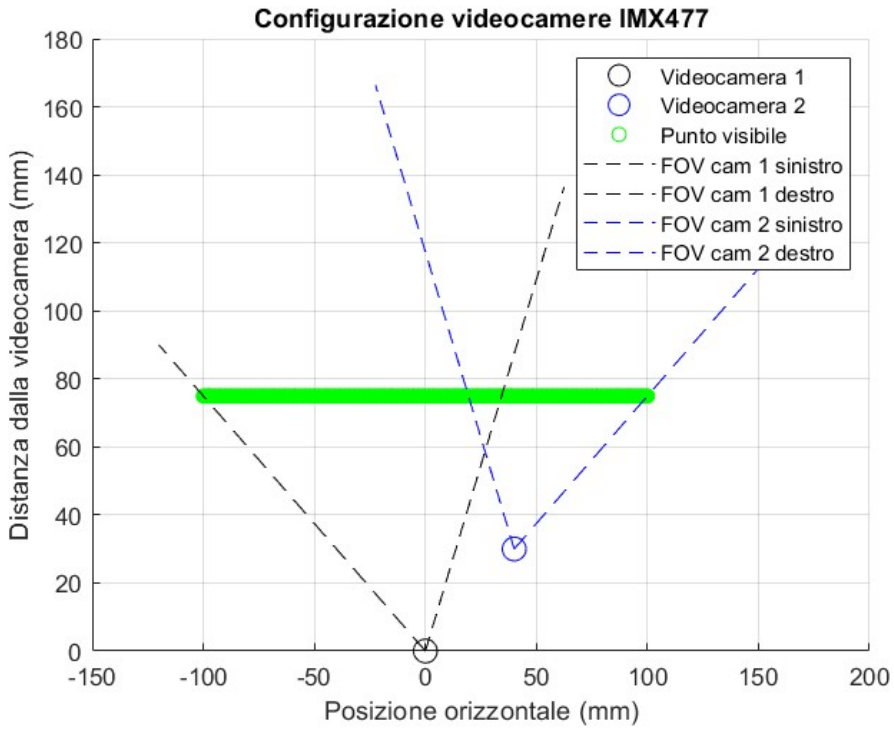

Figura 3.26: Configurazione ottimale videocamere IMX477

I parametri d'interesse sono:

R_{avg}[**pixel/mm**]	R_{RMS}[**pixel/mm**]	R_{min}[**pixel/mm**]	R_c[**pixel/mm**]
20.83	21.53	13.50	15.69

Tabella 3.3: Parametri risoluzione configurazione ottimale videocamere IMX477

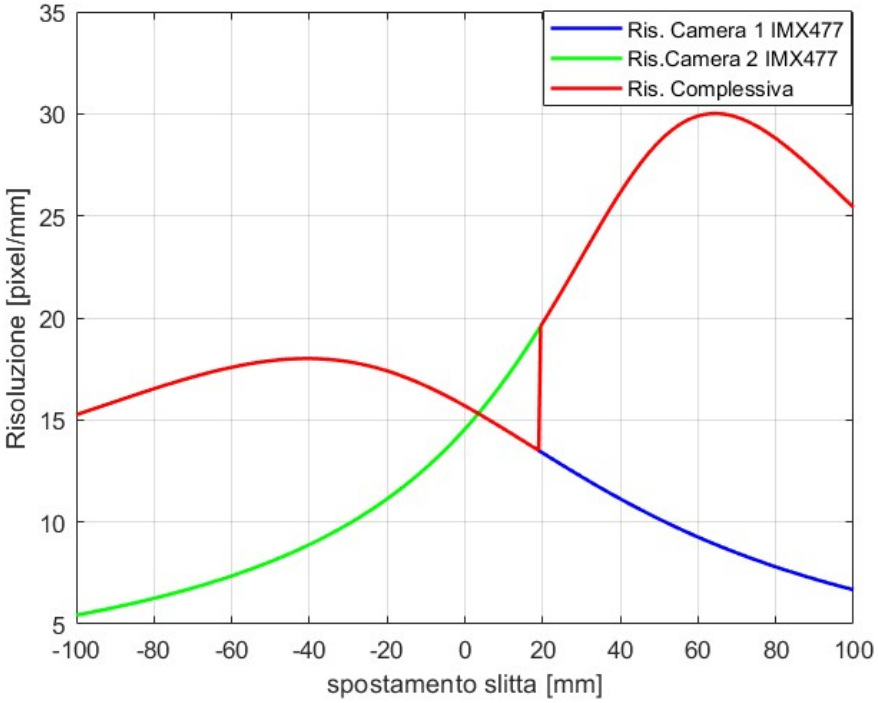

Figura 3.27: Risoluzione combinata configurazione ottimale videocamere IMX477

É interessante osservare che:

- La configurazione ottimale coinvolge due videocamere inclinate una verso l'estremo sinistro del finecorsa l'altra verso l'estremo opposto.
- La configurazione non risulta simmetrica rispetto al centro slitta.
- Una videocamera è posizionata proprio in mezzeria alla distanza minima dalla slitta.

3.5.2 Configurazione videocamere OG02B10

Per la videocamera Global Shutter con sensore OG02B10 la miglior configurazione in termine di risoluzione media e corsa coperta risulta essere la seguente:

θ_1 [°]	x_1 [mm]	z_1 [mm]	θ_2 [°]	x_2 [mm]	z_2 [mm]
-18.3	0	30	14.44	20	30

Tabella 3.4: Parametri geometrici configurazione ottimale videocamere OG02B10

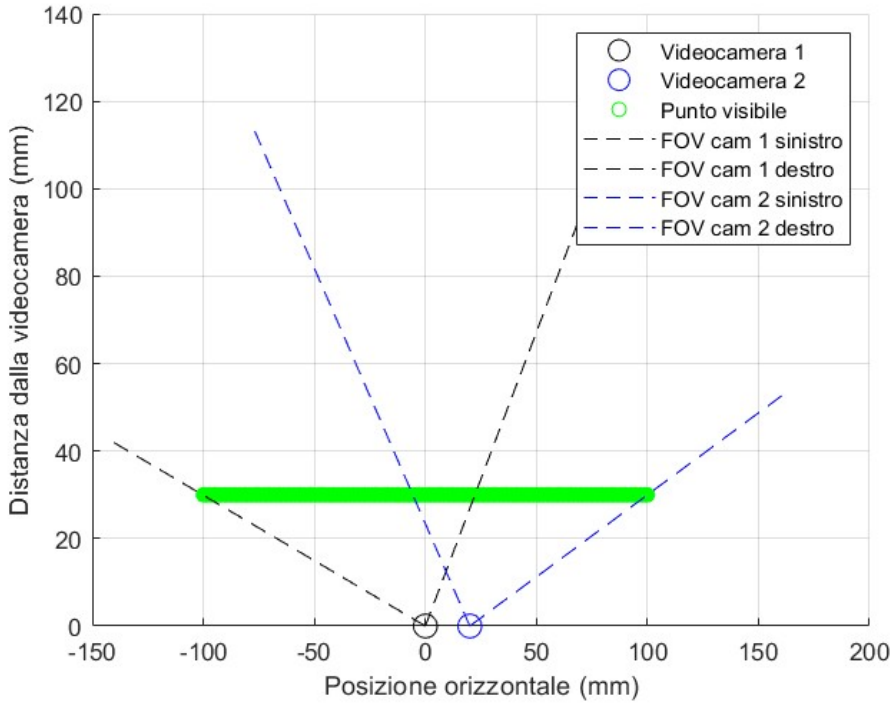

Figura 3.28: Configurazione ottimale videocamere OG02B10

Si osservi come anche in questo caso le configurazioni non sono simmetriche e vi è sempre una videocamera posizionata in mezzeria. A differenza della configurazione con le videocamere Rolling Shutter, l'ampio Field Of View del sensore OG02B10 permette di posizionare entrambe le videocamere alla distanza minima dalla slitta.

I parametri d'interesse sono:

R_{avg}[**pixel/mm**]	R_{RMS}[**pixel/mm**]	R_{min}[**pixel/mm**]	R_c[**pixel/mm**]
16.54	11.30	16.64	16.54

Tabella 3.5: Parametri risoluzione configurazione ottimale videocamere OG02B10

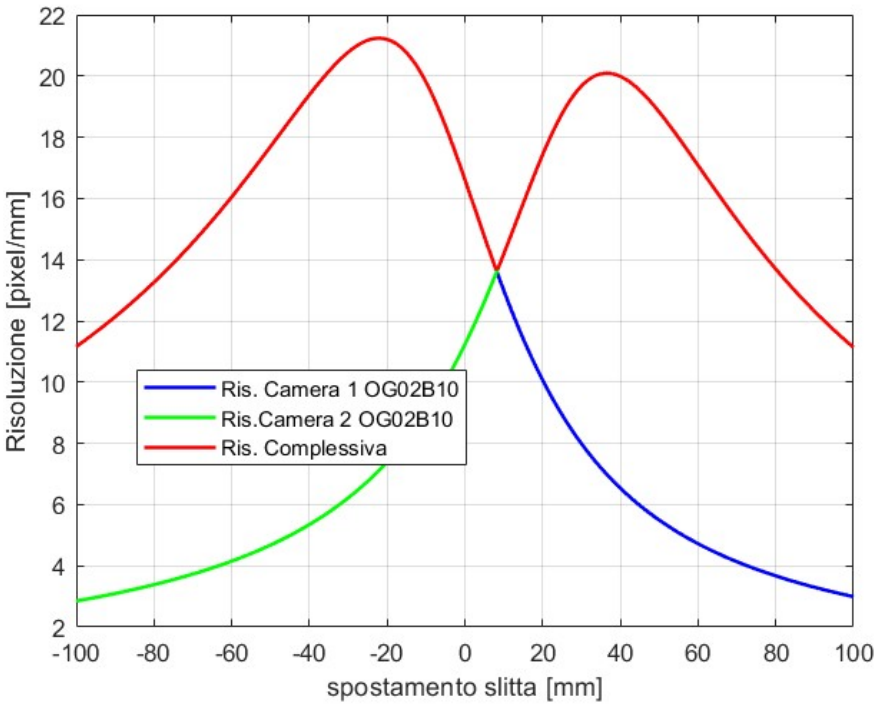

Figura 3.29: Risoluzione combinata configurazione ottimale videocamere OG02B10

La risoluzione media risulta inferiore rispetto alla configurazione con videocamere IMX477, in accordo con le specifiche tecniche dei sensori.

3.5.3 Configurazione videocamere IMX708

Per la videocamera con sensore IMX708 la miglior configurazione in termine di risoluzione media e corsa coperta risulta essere la seguente:

θ_1 [°]	x_1 [mm]	z_1 [mm]	θ_2 [°]	x_2 [mm]	z_2 [mm]
-13.85	-20	75	13.85	20	75

Tabella 3.6: Parametri geometrici configurazione ottimale videocamere IMX708

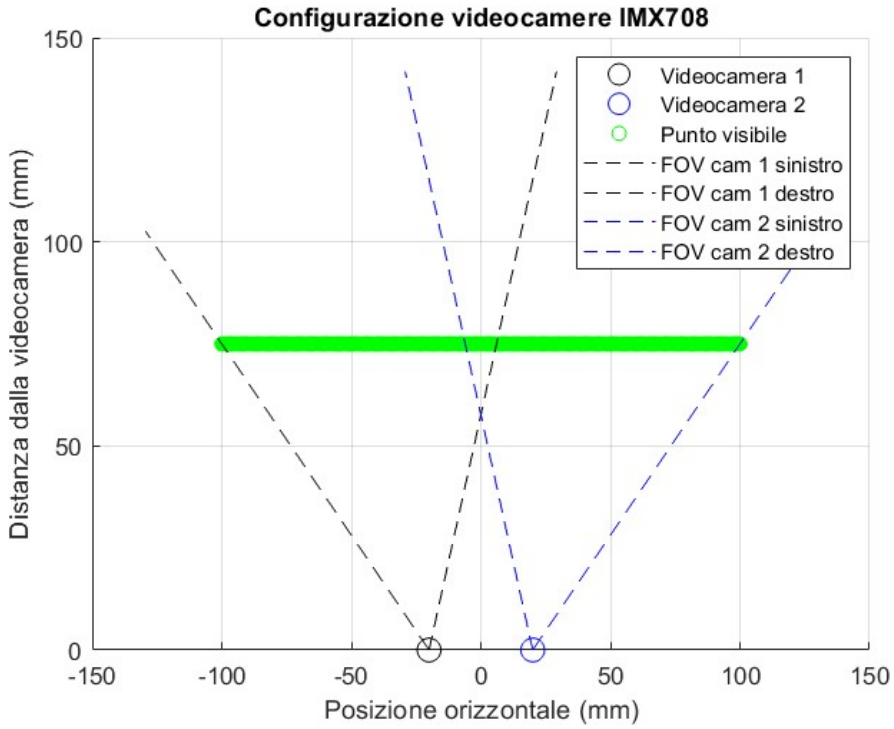

Figura 3.30: Configurazione ottimale videocamere IMX708

Si osservi come in questo caso le configurazioni siano simmetriche e non vi è una videocamera posizionata in mezzeria.

I parametri d'interesse sono:

R_{avg}[**pixel/mm**]	R_{RMS}[**pixel/mm**]	R_{min}[**pixel/mm**]	R_c[**pixel/mm**]
18.78	18.82	15.12	15.12

Tabella 3.7: Parametri risoluzione configurazione ottimale videocamere IMX708

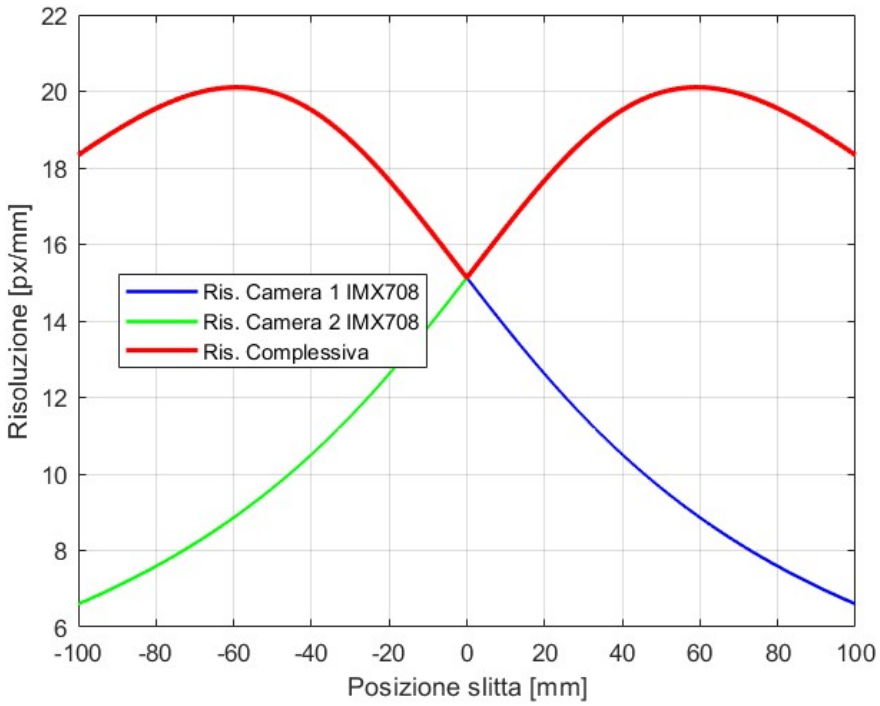

Figura 3.31: Risoluzione combinata configurazione ottimale videocamere IMX708

La risoluzione media risulta inferiore rispetto alla configurazione con videocamere IMX477, in accordo con le specifiche tecniche dei sensori.

3.5.4 Configurazione videocamere miste

Si ricerca la configurazione ottimale con due videocamere diverse. Una videocamera sarà IMX477 mentre la seconda sarà la OG02B10. Di seguito si riportano le posizioni e le inclinazioni delle videocamere:

Videocamera	θ [°]	x **[mm]**	z **[mm]**
IMX477	14.26	40	45
OG02B10	-18.3	0	30

Tabella 3.8: Parametri geometrici configurazione ottimale videocamere miste

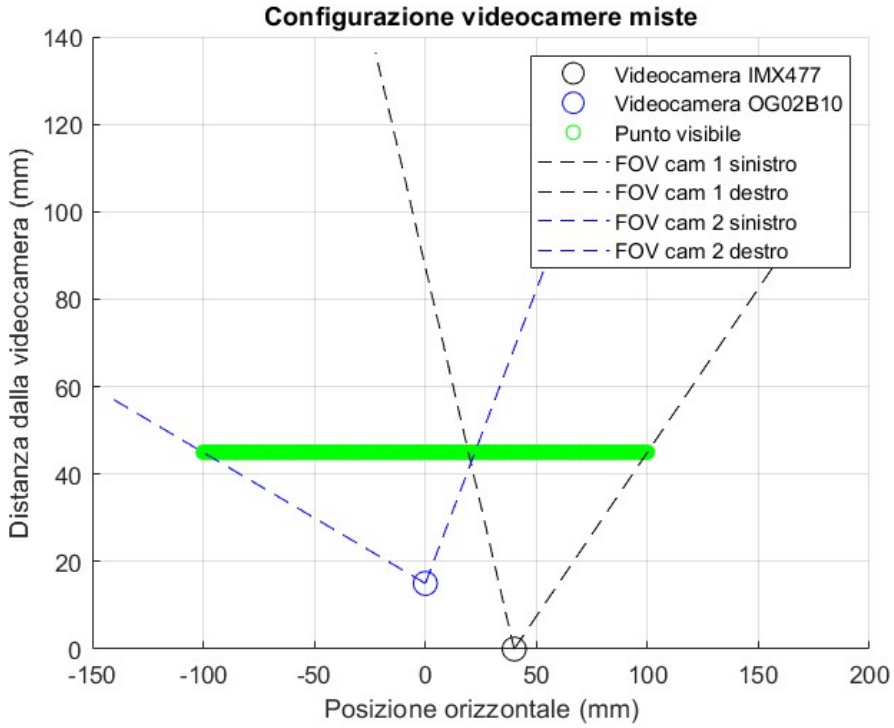

Figura 3.32: Configurazione ottimale videocamere miste

Anche in quest'ultimo caso non vi è una disposizione simmetrica dei sensori, inoltre vi è sempre almeno una videocamera posizionata in mezzeria. Per massimizzare la risoluzione media, il sensore con risoluzione maggiore, ovvero IMX477, è stato posizionato ad una distanza maggiore dalla slitta.

Si riportano i parametri d'interesse di tale configurazione:

R_{avg}[**pixel/mm**]	R_{RMS}[**pixel/mm**]	R_{min}[**pixel/mm**]	R_c[**pixel/mm**]
20.61	21.50	10.33	16.63

Tabella 3.9: Parametri risoluzione configurazione ottimale videocamere miste

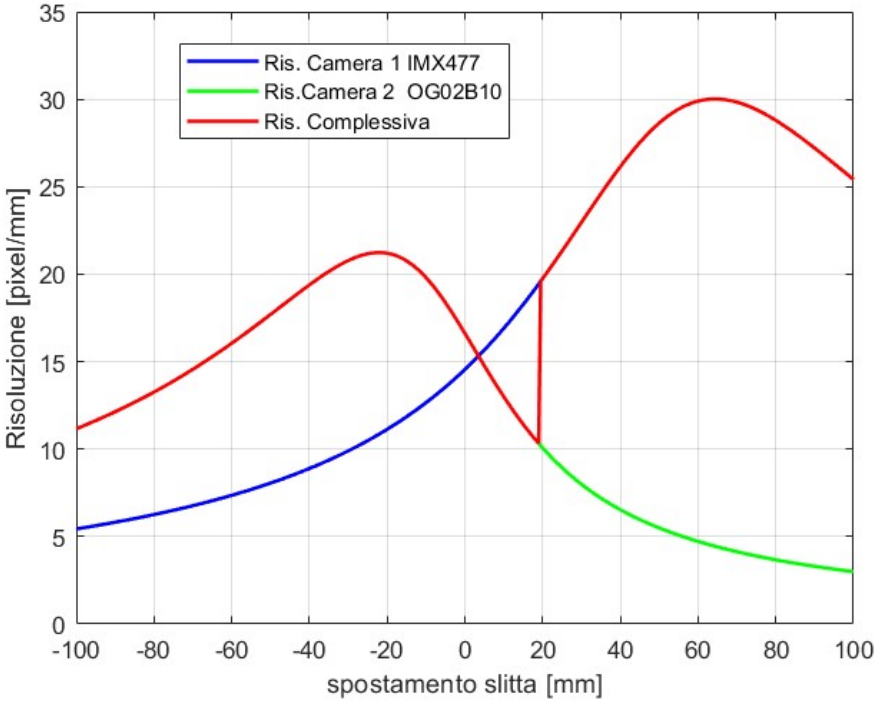

Figura 3.33: Risoluzione combinata configurazione ottimale videocamere miste

Si noti che la risoluzione media non si discosta molta dalla risoluzione media della configurazione ottimale con due sensori IMX477. Si sfrutta l'ampio Field Of View dell'OG02B10 per coprire un'alta percentuale di corsa in modo tale da posizionare la videocamera Rolling Shutter il più vicino possibile alla slitta aumentando la risoluzione media.

Essendo IMX708 e IMX477 sensori con specifiche molto simili non si è riportata la loro configurazione ottimale che risulterebbe comunque con una risoluzione media inferiore rispetto a quella ottenuta con videocamere IMX477. In generale, il FOV più basso dell'IMX708 sconsiglia una configurazione mista con questa videocamera.

3.6 Configurazioni simmetriche

L'analisi svolta evidenzia configurazioni che sono in grado di massimizzare la risoluzione media lungo la corsa della slitta. Questo risultato è raggiunto posizionando le videocamere in modo da creare porzioni di corsa in cui la risoluzione è elevatissima dovuto alla vicinanza del sensore all'oggetto, tuttavia vi sono altre zone in cui la risoluzione diminuisce. Dal punto di vista dell'accuratezza lettura posizione, un risoluzione variabile implica che il minimo spostamento rilevabile cambia a seconda della posizione del cursore sulla slitta. Ciò introduce una non linearità nella catena di misura che bisogna limitare il più possibile. Per questa ragione si può effettuare un'analisi in grado di restituire delle configurazioni con videocamere uguali che massimizzano la risoluzione media ma allo stesso tempo rendono tale parametro il più costante possibile lungo la corsa.

Considerando una generica curva di risoluzione associata ad una videocamera, essa presenta sempre un massimo R_{max} funzione dell'angolo d'inclinazione θ e della posizione della videocamera rispetto alla slitta. Ciò è valido per qualsiasi videocamera, per qualsiasi posizione analizzata. Tra tutte le combinazioni possibili con due videocamere si ricercano le sole che garantiscono:

$$|R_{max,1} - R_{max,2}| < R_{soglia} \tag{3.40}$$

Dove R_{soglia} è stato scelto pari a 3 $pixel/mm$. Di queste configurazioni, si è ricercato la configurazione ottimale che massimizzasse corsa e risoluzione media, al variare sempre dell'angolo d'inclinazione rispetto alla slitta. Tale analisi è stata svolta per configurazioni a due videocamere uguali, sia con videocamere Rolling Shutter che Global Shutter. Non è stata analizzata la configurazione mista, poichè nel caso di due videocamere diverse, con diverse specifiche tecniche l'obiettivo principale resta quello di massimizzare la corsa e la risoluzione media, indipendentemente da ulteriori considerazioni.

La configurazione ottimale con videocamere OG02B10 ottenuta precedentemente rispetta il criterio dell'equazione 3.40, tuttavia non risulta simmetrica da un punto di vista geometrico, per tale videocamera si sceglie tra i risultati ottenuti la seconda configurazione che massimizza la risoluzione media ma che presenta tuttavia un posizionamento simmetrico delle videocamere rispetto alla slitta. Non vi è una sezione dedicata alla Raspberry Pi Camera Module 3 poichè la configurazione ottimale ricavata precedentemente rispetta già questo criterio.

3.6.1 Configurazione con IMX477

Per le videocamere Rolling Shutter la miglior configurazione simmetrica non coincide con quella ottenuta in precedenza con un'analisi completa di tutte le configurazioni.

Videocamera	θ [°]	x **[mm]**	z **[mm]**
IMX477	-14.26	-20	60
IMX477	14.26	20	60

Tabella 3.10: Parametri geometrici configurazione simmetrica IMX477

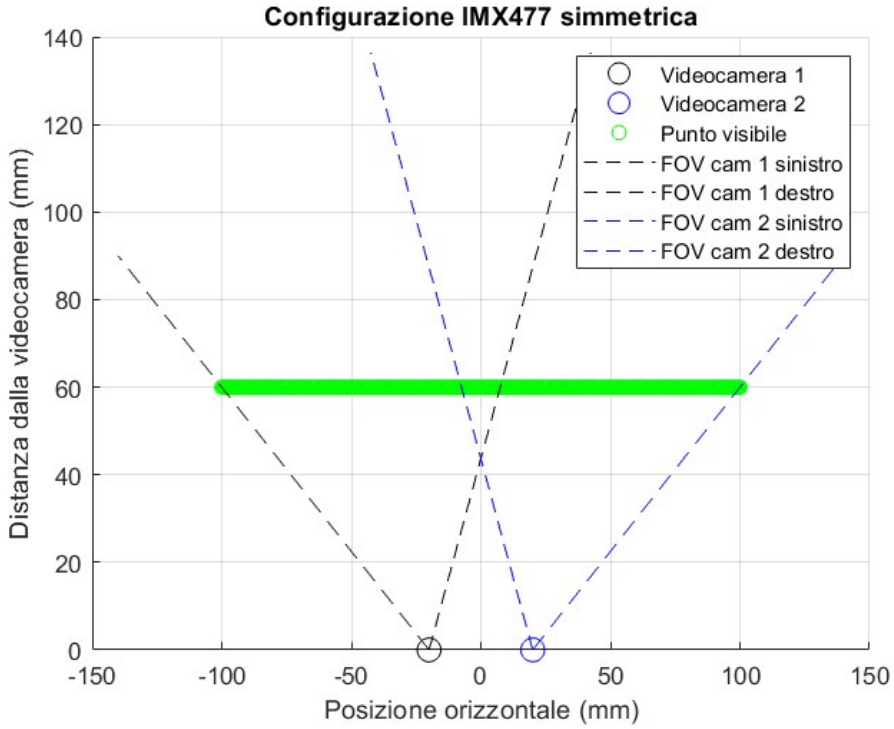

Figura 3.34: Configurazione simmetrica videocamere IMX477

Anche in questo caso, la configurazione ottimale è con una videocamera inclinata verso l'estremo sinistro della slitta e l'altra verso l'estremo destro.

R_{avg}[pixel/mm]	R_{RMS}[pixel/mm]	R_{min}[pixel/mm]	R_c[pixel/mm]
20.63	20.68	15.99	15.99

Tabella 3.11: Parametri risoluzione configurazione simmetrica videocamere IMX477

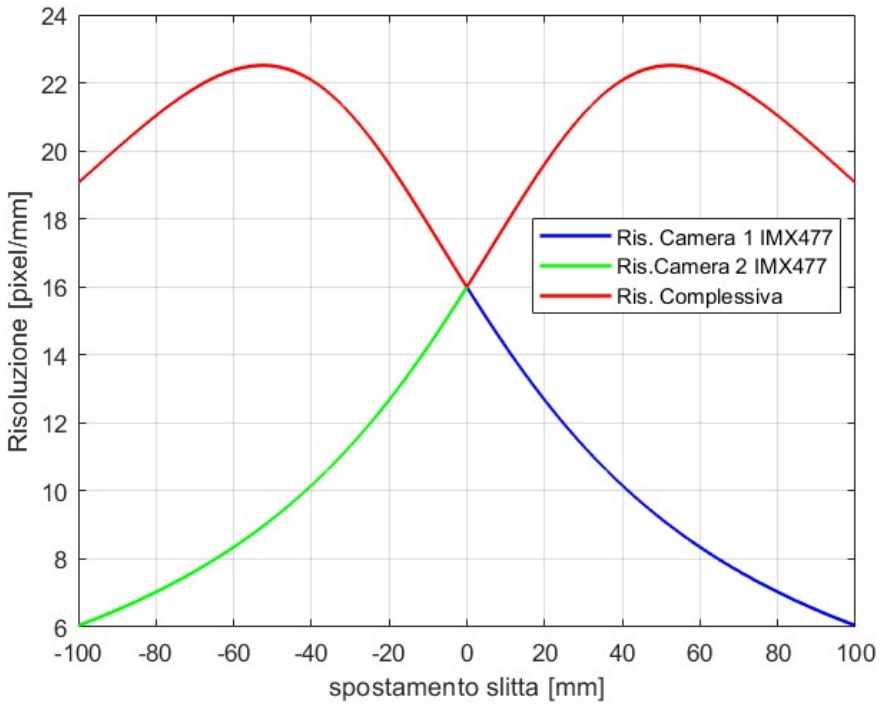

Figura 3.35: Risoluzione combinata configurazione simmetrica IMX477

Dai risultati ottenuti si osserva che:

- La risoluzione media non si discosta molto da quella ottenuta dalla configurazione precedente, risulta leggermente inferiore.
- La risoluzione minima è aumentata grazie alla simmetria della configurazione.
- La risoluzione minima coincide con la risoluzione al centro della slitta.
- La differenza fra i massimi valori di risoluzione è nulla.

3.6.2 Configurazione con OG02B10

Per le videocamere Global Shutter si considera la seguente configurazione:

Videocamera	θ [°]	x **[mm]**	z **[mm]**
OG02B10	-14.5	-20	30
OG02B10	14.5	20	30

Tabella 3.12: Parametri geometrici configurazione simmetrica OG02B10

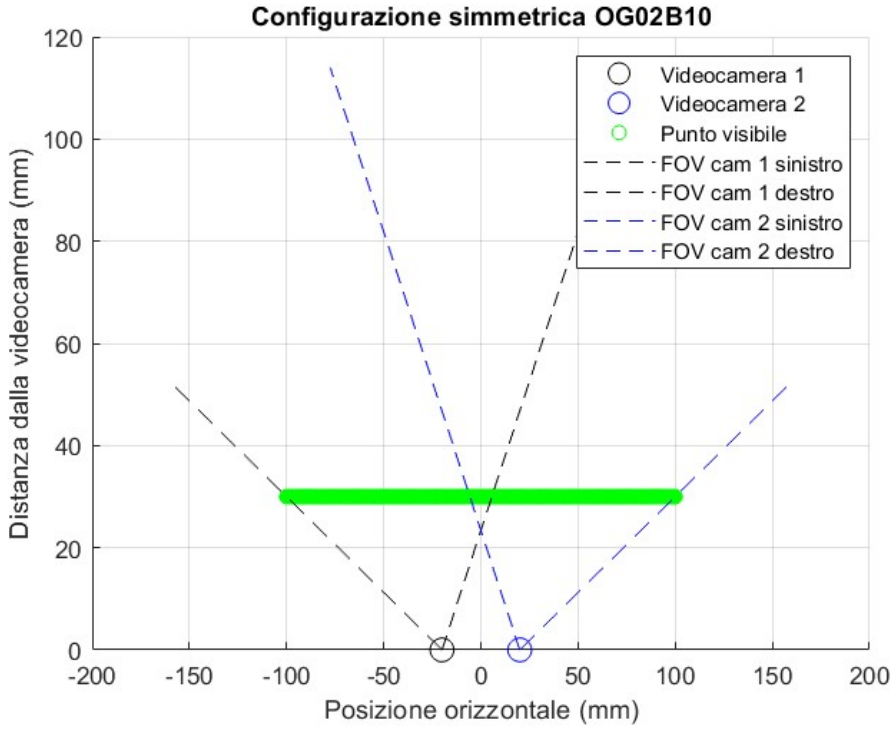

Figura 3.36: Configurazione simmetrica videocamere OG02B10

La configurazione risulta molto simile a quella ottenuta con le videocamere Rolling Shutter con la sola differenza che le videocamere OG02B10 risultano avere un FOV maggiore, di conseguenza possono essere posizionate ad una distanza minore dalla slitta.

R_{avg}[**pixel/mm**]	R_{RMS}[**pixel/mm**]	R_{min}[**pixel/mm**]	R_c[**pixel/mm**]
15.92	16.24	11.12	11.23

Tabella 3.13: Parametri risoluzione configurazione simmetrica OG02B10

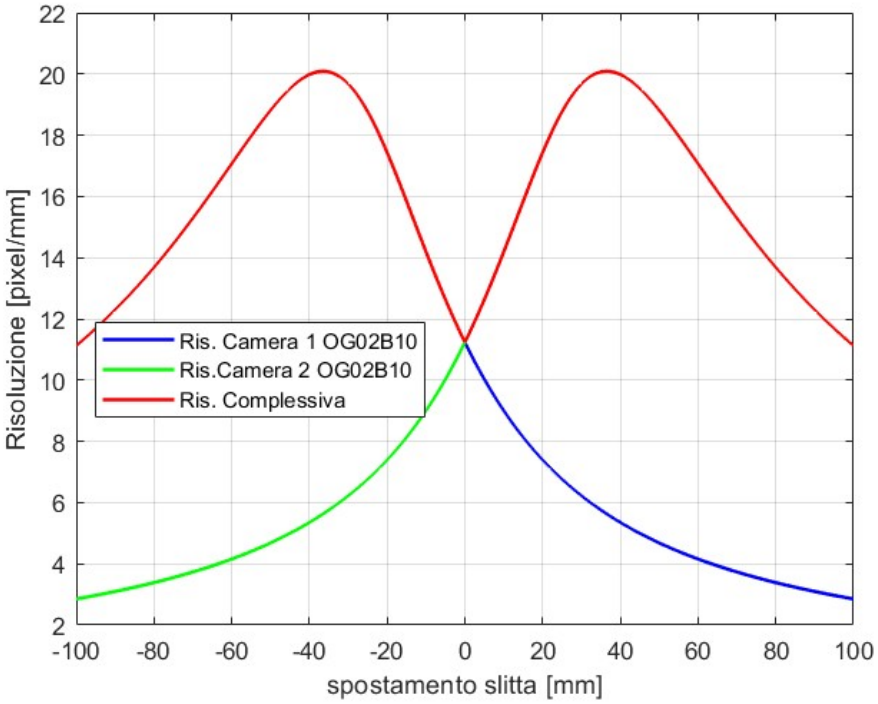

Figura 3.37: Risoluzione combinata configurazione simmetrica OG02B10

Le osservazioni risultano essere analoghe a quelle espresse per la configurazione simmetrica delle videocamere Rolling Shutter. La risoluzione media è inferiore rispetto a quella ottenuta con le videocamere IMX477, in accordo ancora una volta con le specifiche tecniche dei sensori.

3.7 Caratteristica statica della videocamera

In questa sezione si descrive il procedimento che ha portato alla determinazione della caratteristica statica della videocamera. Si analizza l'influenza degli errori sistematici su di essa e come essi comportano la necessità di sviluppare un processo di calibrazione della videocamera per la corretta acquisizione della posizione della slitta per il sistema.

Si vuole determinare una caratteristica che data la posizione effettiva della slitta restituisca la posizione rilevata effettivamente dalla videocamera. Per fare ciò si fa riferimento al modello Pinhole già ampiamente discusso in questo capitolo. Un punto nello spazio 3D viene riportato attraverso una trasformazione prospettica sul piano immagine avente coordinate (X, Y) in accordo con l'espressione 3.1. Tali coordinate sono esprimibili in numero di pixel orizzontali e verticali. Poichè viene effettuata una mappatura da un sistema 3D ad un sistema 2D, ad ogni coppia di coordinate (X, Y) in pixel corrisponderà una determinata posizione dell'oggetto nel sistema 3D. Andando opportunamente a rielaborare l'espressione 3.4 è possibile ottenere la posizione del punto nello spazio 3D a partire dalle sue coordinate nel piano immagine. Tale mappatura dipende anche dalla posizione e orientazione relativa della videocamera rispetto all'oggetto in questione. Un errore nel posizionamento, un'inclinazione della videocamera differente può portare ad errori di ricostruzione che alterano inevitabilmente l'informazione di posizione che si sta cercando di determinare.
Per evidenziare questi effetti si considera il posizionamento più semplice per il sistema in questione. Si ipotizza una sola videocamera IMX477, posizionata in mezzeria che inquadra frontalmente senza alcun angolo di inclinazione la slitta. La videocamera viene posizionata ad una distanza di 125 millimetri lungo z in modo tale che riesca ad inquadrare tutti i punti di funzionamento della slitta. Attraverso un codice MATLAB è stato possibile determinare la coordinata u sul piano immagine per tutti i punti di funzionamento della slitta. Si sottolinea che anche in questo caso si è ipotizzato uno spostamento dell'oggetto lungo la sola direzione x di conseguenza è interessante analizzare la sola coordinata u sul piano immagine poichè è l'unica che ha una variazione durante lo spostamento della slitta. A partire dalla coordinata u è possibile ottenere la posizione del punto nella coordinata mondo X_W attraverso la seguente relazione:

$$X_W = \frac{u - p_x}{f_x} Z_W \tag{3.41}$$

Dove f_x è ottenuto attraverso la seguente relazione:

$$f_x = f \frac{N_x}{W_s} \tag{3.42}$$

con f lunghezza focale del sensore, N_x numero di pixel orizzontali ed W_s larghezza del sensore.

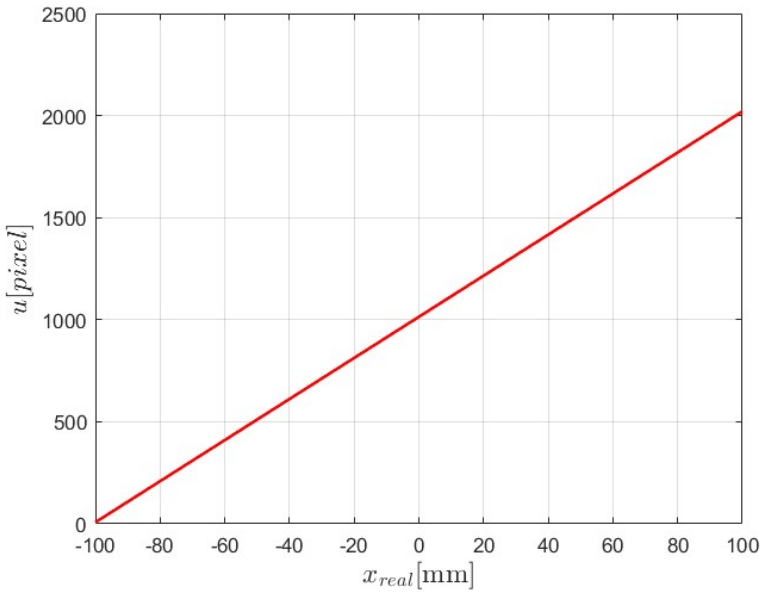

Figura 3.38: Caratteristica statica ideale, x_{real} coordinata effettiva della slitta, u coordinata su piano immagine

Alla luce di tale risultato ottenuto si osserva che:

- La trasformazione prospettica del modello Pinhole da luogo ad una caratteristica statica lineare.
- Quando la slitta si trova in mezzeria le coordinate sul piano immagine coincidono con le coordinate del centro immagine p_x e p_y.
- La caratteristica conferma quanto ottenuto attraverso la determinazione del Field Of View del sensore nella sezione 3.4: quando la slitta si trova al finecorsa di 100 mm, corrispondente a x_{end}, ha coordinata u sul piano immagine pari N_x, mentre quando si trova nella posizione opposta ha coordinata pari a 0 pixel.

Attraverso la relazione 3.42 è possibile ottenere la posizione in millimetri della slitta letta dalla videocamera x_r in assenza di errori di posizionamento di quest'ultima.

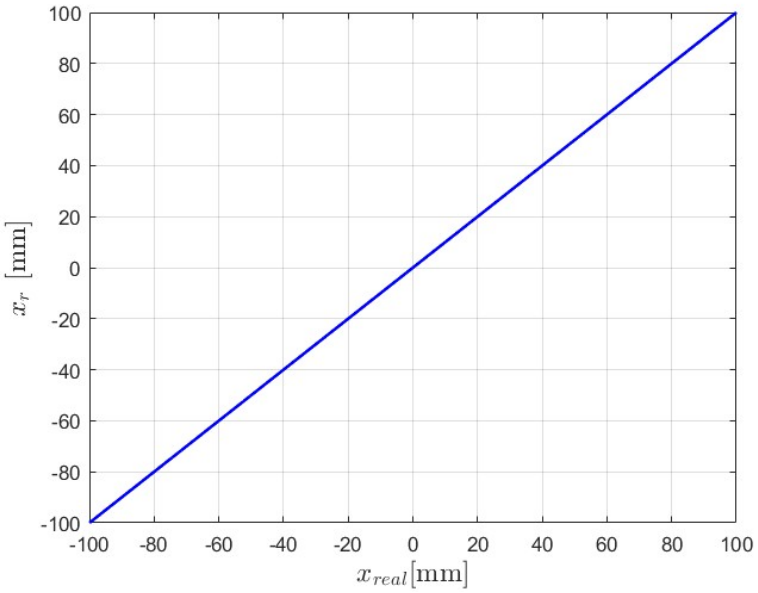

Figura 3.39: Caratteristica statica ideale

Si ottiene dunque una caratteristica lineare con nessun errore di ricostruzione introdotto dal posizionamento della videocamera. Dal punto di vista pratico, è difficile pensare di posizionare la videocamera senza introdurre alcun tipo di errore sistematico. Risulta dunque interessante valutare come cambia tale caratteristica se vengono introdotti i seguenti errori di posizionamento:

- errore di posizionamento lungo direzione parallela alla slitta Δx;
- errore di posizionamento lungo direzione perpendicolare alla slitta Δz;
- errore di orientamento della videocamera rispetto alla slitta $\Delta\theta$.

Per fare ciò si modifica necessariamente la relazione tra la posizione dell'oggetto da identificare e quella della videocamera.

3.7.1 Influenza degli errori sistematici

In questo paragrafo viene valutata l'influenza degli errori sistematici di posizionamento della videocamera, in particolare quello lungo la direzione parallela alla slitta e l'errore dovuto ad un'inclinazione errata della videocamera. Introdurre un posizionamento errato Δx significa modificare il vettore di traslazione $\mathbf{t}$, mentre introdurre un angolo d'inclinazione $\Delta\theta$ implica modificare la matrice di rotazione R. Per quanto riguarda il primo errore sistematico, si valuta un posizionamento errato

di 5 millimetri rispetto alla posizione di mezzeria. In questa fase si è trascurato l'influenza dell'errore di posizionamento lungo la direzione perpendicolare alla slitta.

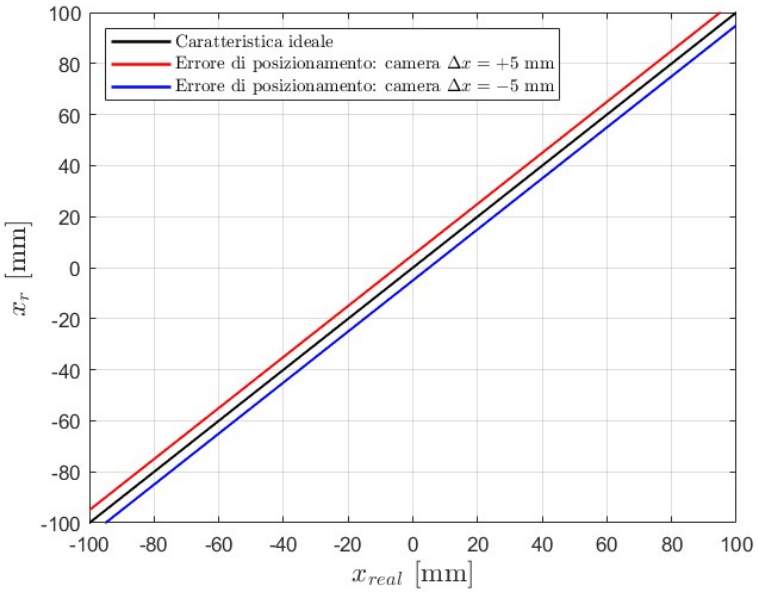

Figura 3.40: Influenza dell'errore sistematico Δx sulla caratteristica statica

Come si può osservare dai risultati ottenuti, introdurre un errore di posizionamento lungo la direzione x comporta traslare verticalmente la caratteristica statica ideale introducendo un errore nella ricostruzione della posizione dell'oggetto pari proprio a tale errore sistematico. La caratteristica resta lineare tuttavia si genera un errore nella lettura della posizione dell'oggetto.

Per quanto riguarda l'errore relativo all'orientazione della videocamera rispetto alla slitta si ipotizza un angolo θ di 5 gradi. La rotazione imposta è rispetto all'asse y uscente dal piano $x - z$. La matrice di rotazione R si modifica secondo quanto descritto dall'equazione 3.22. A differenza di quanto visto per l'errore di posizionamento, in questo caso la caratteristica si modifica e diviene non lineare. In particolare l'errore di ricostruzione risulta maggiore nei punti di funzionamento lontani dalla mezzeria, mentre è minimo in corrispondenza di essa. Per valutare l'errore si effettua la differenza puntuale tra la caratteristica statica ideale e la caratteristica statica influenzata dai diversi errori sistematici e la si rapporta alla coordinata del fondocorsa x_{end} pari a 100 millimetri.

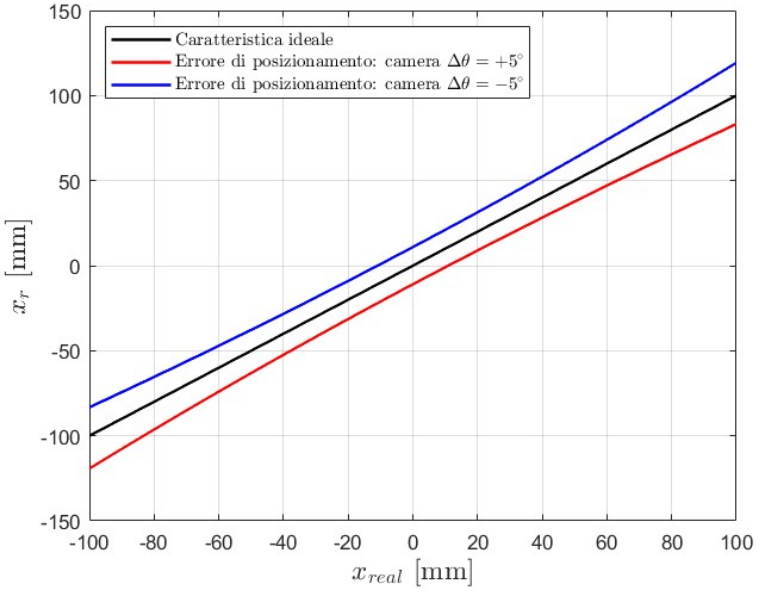

Figura 3.41: Influenza dell'errore sistematico $\Delta\theta$ sulla caratteristica statica

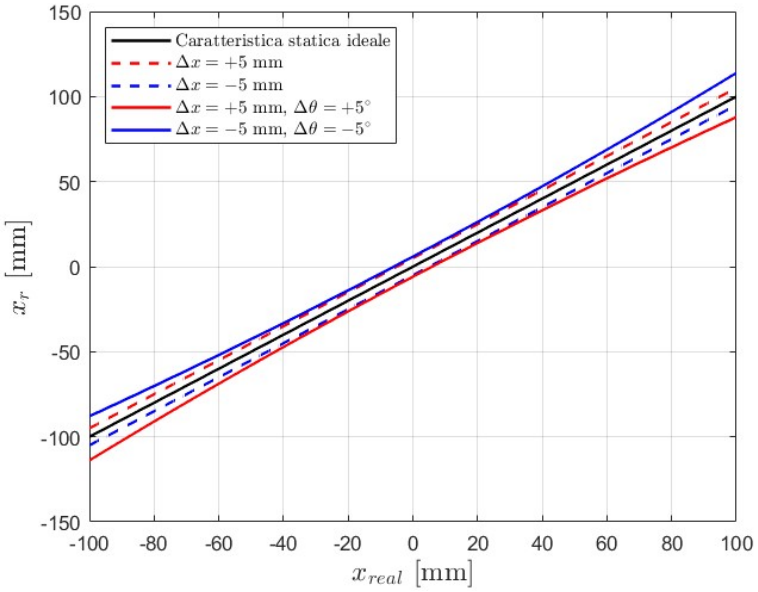

Figura 3.42: Influenza degli errori sistematici Δx e $\Delta\theta$ sulla caratteristica statica

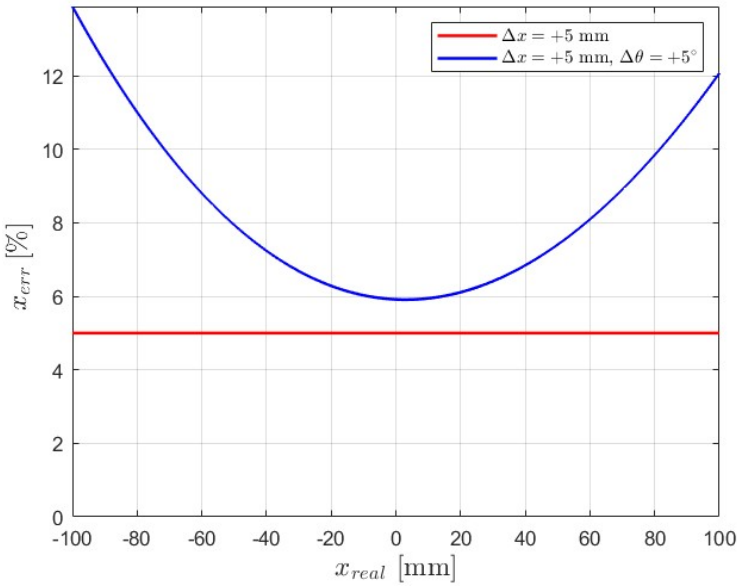

Figura 3.43: Errore di ricostruzione in funzione della posizione dell'oggetto e degli errori sistematici

L'errore introdotto dall'errata inclinazione della videocamera rispetto alla slitta non è simmetrico: in funzione dell'angolo d'inclinazione si può avere un errore maggiore in una sezione di corsa percorsa dalla slitta e minore nella sezione opposta. Con un angolo θ positivo di 5 gradi l'asse ottico punta verso la parte destra della corsa di conseguenza per degli effetti di distorsione e di proiezione l'errore è maggiore nella parte sinistra della corsa.

3.7.2 Influenza della distanza fra videocamera e slitta

In questo paragrafo si analizza l'influenza di un errore nel posizionamento della videocamera nella direzione perpendicolare alla slitta. A differenza di quanto visto per gli errori Δx e $\Delta\theta$, un errore nel posizionamento lungo l'asse z non modifica l'andamento della caratteristica statica la quale resta lineare. Variare la distanza videocamera − slitta modifica la mappatura da coordinate mondo 3D a coordinate nel piano immagine. Se la distanza aumenta, a parità di altri fattori, la medesima posizione reale dell'oggetto avrà una coordinata sul piano immagine differente. Modificare la distanza lungo la direzione z implica cambiare la pendenza della caratteristica $x_{real} - u$. Posizionando la videocamera in mezzeria si è ipotizzato un errore di posizionamento lungo z di 5 millimetri.

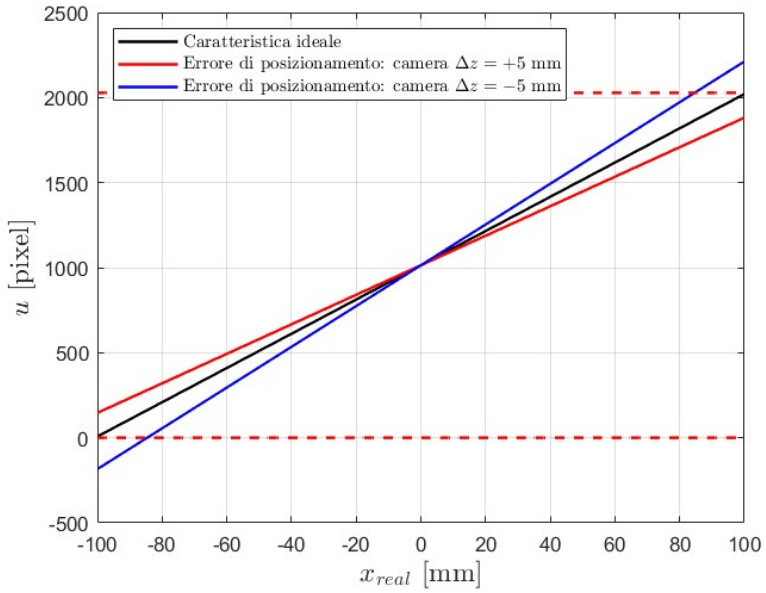

Figura 3.44: Effetto dell'errore Δz sulla caratteristica $x_{real} - u$

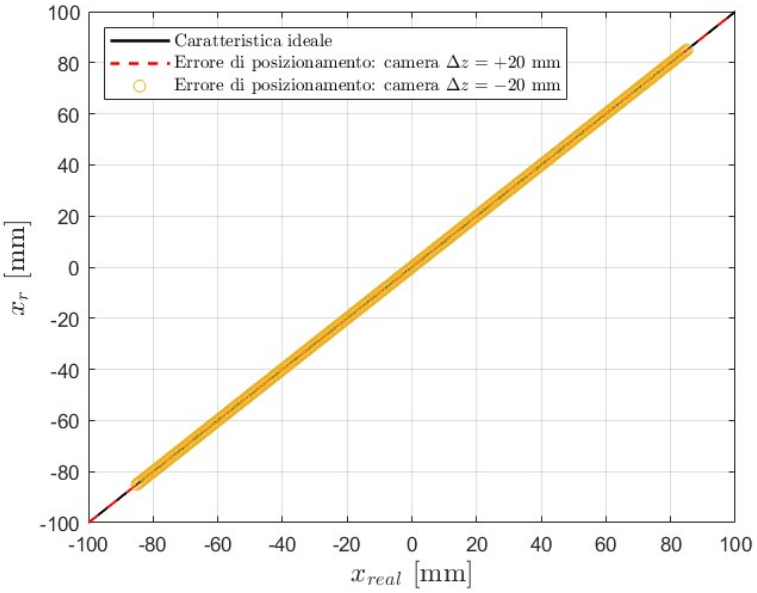

Figura 3.45: Effetto dell'errore Δz sulla caratteristica statica della videocamera

Come si può osservare, se la videocamera è posizionata ad una distanza maggiore rispetto a quella di riferimento aumenta il campo visivo a disposizione e di conseguenza si modifica la mappatura sul piano immagine con le posizioni all'estremità della corsa che non avranno più coordinate sul piano immagine pari a 0 pixel ed N_x. Vi è una criticità nel momento in cui la videocamera risulta essere posizionata ad una distanza minore rispetto a quella di riferimento poiché il campo visivo a disposizione non permette più di rilevare la posizione per tutti i punti di funzionamento. Questo errore di posizionamento non modifica dunque la caratteristica statica ma influenza direttamente il campo visivo della videocamera. Ipotizzando la sola presenza di questo errore la caratteristica statica coincide con quella ideale ma potrebbe non essere valida per l'intero range di funzionamento del sistema. Nella figura 3.45, ipotizzando un errore di posizionamento di 20 millimetri, la caratteristica riesce a rilevare la posizione della slitta fino a 83 millimetri a causa della riduzione del campo visivo. Posizionando la videocamera ad una distanza maggiore la caratteristica statica non si modifica anzi sarebbe in grado di ricostruire la posizione della slitta per corse maggiori.

Questo studio preliminare risulta molto importante poichè sottolinea la necessità di effettuare una calibrazione durante il posizionamento delle videocamere. Utilizzando tali curve ottenute e le informazioni provenienti dal trasduttore potenziometrico si può effettuare una procedura di calibrazione per eliminare l'influenza di tali errori sistematici sulla lettura della posizione della slitta.

Capitolo 4

Modello feedback ottico

In questo capitolo si vuole comprendere come l'introduzione di videocamere come feedback nell'anello chiuso di posizione si può tradurre in termini di performance dinamiche del controllo posizione e verificare la banda passante del sistema. Nel precedente capitolo si è descritto lo studio geometrico effettuato al fine di ricercare il posizionamento ottimale per le videocamere scelte. Tali videocamere rappresentano il feedback del sistema in grado di restituire la posizione della slitta sulla base di un determinato comando di set. Partendo da un modello Simulink già esistente si vuole comprendere come l'introduzione delle videocamere incida sulle prestazioni del sistema. Si prendono come riferimento le configurazioni ottimali ricavate nel Capitolo 3; in particolare, per videocamere uguali si considerano le configurazioni simmetriche poichè si ricerca una risoluzione il più costante possibile nei diversi punti di funzionamento del sistema. Il modello viene modificato per tenere conto dell'introduzione delle videocamere nel feedback della posizione.

4.1 Descrizione del modello

Il modello iniziale e la sua descrizione è tratta da [12] e [33].

Per la descrizione del modello si può dividere l'intero servosistema in tre grandi blocchi funzionali:

- regolazione e controllo;
- motore elettrico;
- attuatore.

Il sistema di controllo, basato sul controllore PID, è pensato per essere utilizzato con il solo guadagno proporzionale attivo.

4.1.1 Motore elettrico

Per quanto riguarda il motore elettrico in corrente continua esso può essere schematizzato come un circuito RL in cui è presente anche una forza controelettromotrice proporzionale alla velocità angolare del rotore. SI effettuano due modellazioni principali:

- Modellazione della dinamica elettrica;
- Modellazione della conversione di potenza, da corrente a coppia meccanica.

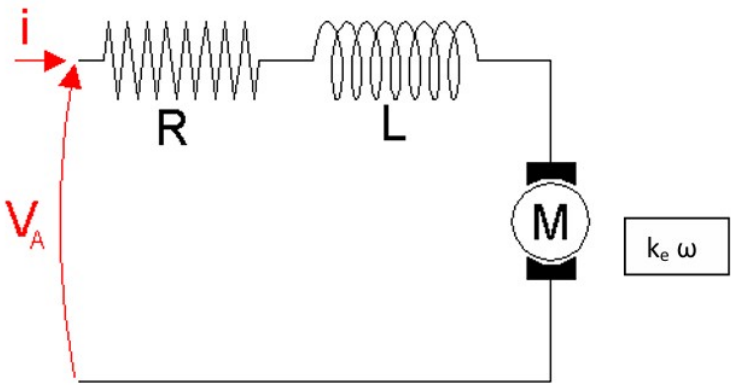

Figura 4.1: Schema del motore elettrico in corrente continua[33]

L'equazione alla maglia di questo circuito equivalente risulta essere la seguente:

$$V_A = R \cdot i + L\frac{di}{dt} + k_e \cdot \omega \tag{4.1}$$

Dove V_A risulta essere la tensione di alimentazione del motore, i la corrente, R ed L resistenza ed induttanza equivalente, k_e la costante elettrica e ω la velocità angolare del motore. Inoltre, in un motore elettrico DC la coppia erogata è proporzionale alla corrente i tramite la costante di coppia k_T che ha circa lo stesso valore della costante elettrica k_e. Dunque, è possibile esprimere la coppia erogata dal motore in funzione della corrente attraverso la seguente relazione:

$$T = k_T \cdot i \tag{4.2}$$

Il motore elettrico è caratterizzato da una velocità di fuga di circa 415 rad/s, una tensione di alimentazione massima di 16 V. È possibile ottenere il valore della costante elettrica k_e dal prodotto di queste ultime due grandezze. Si ricordi che

numericamente tale valore coincide con quello della costante di coppia k_T. Si ottiene una costante di coppia pari a 0.039 Nm. È possibile, a partire da tale equazione differenziale del primo ordine, ottenere un'equazione algebrica in s , operatore di Laplace, utilizzando la trasformata di Laplace. Detta τ_e la costante di tempo elettrica del circuito, la corrente $\bar{i}$ nel dominio di Laplace sarà pari a:

$$\bar{i} = \frac{\overline{V}_A - k_e\overline{\omega}}{R(\tau_e s + 1)} \tag{4.3}$$

Analogamente si riporta l'espressione 4.2 nel dominio di Laplace:

$$\overline{T} = k_T \cdot \bar{i} \tag{4.4}$$

4.1.2 Attuatore

Questo blocco funzionale è rappresentato da una slitta traslante con trasmissione attraverso cinghia dentata. Il sistema è modellizzato attraverso il diagramma di corpo libero della massa traslante m. Partendo dalla coppia meccanica T ottenuta precedentemente, è possibile ricavare la forza F esercitata sulla massa traslante attraverso la seguente espressione:

$$F = \frac{T}{r} \tag{4.5}$$

Dove r risulta essere il raggio della puleggia della trasmissione con cinghia. Di seguito si riporta l'equazione differenziale che si ottiene dall'equilibrio dinamico del sistema:

$$F + F_{ext} = m\ddot{x} + c\dot{x} \tag{4.6}$$

Si considera sia l'attrito viscoso con l'aria tramite il coefficiente di smorzamento c, che le forze esterne che agiscono come disturbo sul controllo posizione F_{ext}. Nel dominio di Laplace si trasforma nella seguente espressione:

$$\frac{\overline{T}}{r} + \overline{F_{ext}} = (ms^2 + cs)\overline{x} \tag{4.7}$$

Nel modello Simulink viene introdotto l'attrito nello schema a blocchi complessivo del servosistema.

4.1.3 Regolazione e controllo

Ottenuta la posizione $\overline{x}$, tramite la funzione di trasferimento che rappresenta l'attuatore del sistema è possibile chiudere l'anello di controllo inviando tale segnale all'algoritmo di controllo. Nel modello iniziale tutto ciò viene effettuato attraverso

un trasduttore potenziometrico modelizzato mediante un semplice guadagno statico. Si ipotizza infatti la larghezza di banda del potenziometrico almeno di un ordine di grandezza maggiore rispetto a quella del sistema. Il sistema di regolazione prende in input l'errore ottenuto dalla differenza fra set e feedback e lo compensa attraverso un controllore PID. All'uscita del controllore vi è l'errore compensato, una tensione che viene inviata al motore elettrico moltiplicata per un determinato guadagno relativo all'interfaccia di potenza.

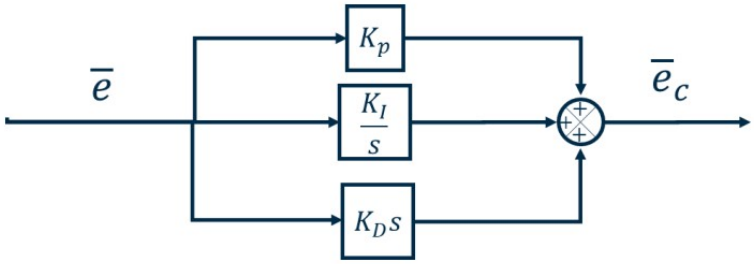

Figura 4.2: Schema a blocchi controllore PID, in ingresso l'errore $\overline{e}$ in uscita l'errore compensato $\overline{e_c}$[33]

Si osservi come il controllore PID abbia tre contributi:

- Proporzionale, attraverso il guadagno K_p;
- Integrativo, attraverso il guadagno K_I;
- Derivativo, attraverso il guadagno K_D.

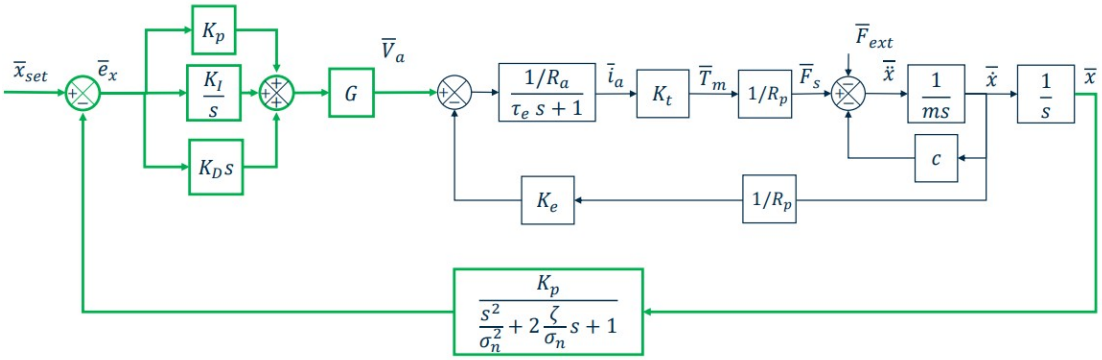

Figura 4.3: Schema a blocchi complessivo servosistema controllo posizione elettrico [33]

Nella figura 4.3 nel ramo di feedback della posizione è stato inserito un secondo ordine nell'ipotesi di considerare la dinamica del trasduttore. Proprio su quel ramo di feedback sono state effettuate delle modifiche a questo modello per considerare la videocamera e le sue caratteristiche per il rilevamento della posizione. Si evidenzia inoltre, che nel modello Simulink, a differenza di quanto è visualizzabile nello schema a blocchi complessivo, è stata inserita una saturazione che limita il valore dell'errore compensato poiché vi sono delle limitazioni fisiche sulla tensione applicabile al motore elettrico in questione.

4.1.4 Videocamera nel ramo di feedback

In questo paragrafo si descrive il procedimento che ha portato all'introduzione della videocamera nel ramo di feedback del sistema controllo posizione elettrico.

La videocamera si può schematizzare come un trasduttore con le seguenti caratteristiche:

- La risoluzione, in termine di pixel su millimetri, può essere vista come un passo di quantizzazione in ampiezza sul segnale di posizione.
- il frame rate di una videocamera è strettamente legato alla frequenza di campionamento.

Sulla base di queste considerazioni si è pensato di utilizzare le curve di risoluzione ottenute durante lo studio geometrico descritto nel Capitolo 3 per costruire delle Look-Up Tables in grado di restituire il passo di quantizzazione in millimetri su pixel in base alla posizione della slitta. La frequenza di campionamento per una videocamera non è altro che il numero di fotogrammi che essa riesce a riprodurre in un secondo. Non è possibile conoscere la posizione dell'oggetto in istanti di tempo compresi tra i due fotogrammi. Nel modello Simulink, all'ingresso del ramo di feedback è stato introdotto un subsystem nominato *Videocamera* che riceve in ingresso il segnale di posizione e lo restituisce quantizzato in ampiezza sulla base della risoluzione e campionato nel tempo sulla base del framerate impostato. Si sono utilizzati i seguenti blocchi:

- *Zero-Order-Hold*: è un blocco che permette di campionare nel tempo un segnale se si imposta il periodo di campionamento. Come periodo di campionamento si è scelto il reciproco del frame rate.
- *1-D Look-up Table*: sulla base delle curve di risoluzione riferite alla configurazioni simmetriche si può determinare il passo di quantizzazione in ampiezza del segnale di poizione. Attraverso questo blocco è possibile scegliere fra diverse configurazioni e automatizzare il processo.

- *MATLAB Function*: questo blocco consente di scrivere una funzione sulla base dei segnali che riceve in ingresso. Utile per riprodurre il blocco *Quantizer* utilizzato per quantizzare in ampiezza un segnale.

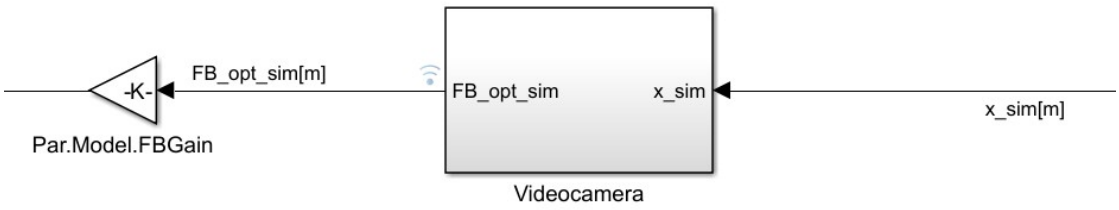

Figura 4.4: Subsystem *Videocamera* sul ramo di feedback

Come si può osservare nella modellazione vengono considerati i soli effetti della risoluzione e del frame rate delle videocamere. Della presenza del Raspberry e dell'algoritmo di riconoscimento e stima della posizione della slitta non se ne tiene conto, nonostante siano due elementi che potrebbero introdurre un ulteriore ritardo nell'anello. Inoltre bisogna considerare che una volta che si è in grado di stimare la posizione dell'oggetto il Raspberry deve inviare degli opportuni segnali di tensione al controllore. Nel modello Simulink si conserva, in mancanza di ulteriori informazioni, il guadagno statico del trasduttore potenziometrico *FBGain*.

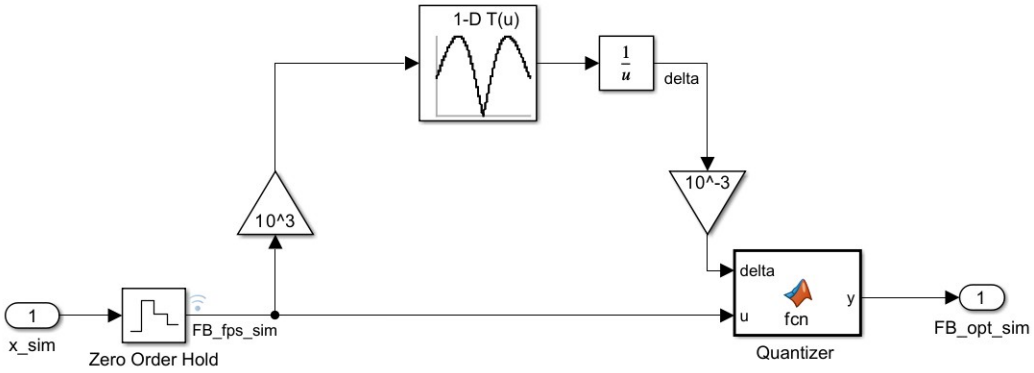

Figura 4.5: Subsystem *Videocamera* nel dettaglio

Sono state inserite le seguenti Look Up Tables:

- configurazione simmetrica videocamere IMX477;
- configurazione simmetrica videocamere OG02B10;

- configurazione simmetrica videocamere Pi Camera Module 3;
- configurazione ottimale videocamere miste.

Le configurazioni inserite hanno risoluzione minima in termine di prodotto numero di pixel verticali N_y ed orizzontali N_x tra tutte le modalità video disponibili. Questo perchè i test effettuati sul modello sono stati svolti con il massimo valore di frame rate ovvero 120 FPS.

4.1.5 Ulteriori modifiche effettuate

Oltre all'introduzione della videocamera nel ramo di feedback, il modello Simulink è stato ulteriormente modificato per tener conto di altri aspetti del sistema da sviluppare e per migliorarne alcuni dettagli. Si elencano le principali modifiche apportate:

- Filtro del primo ordine sul segnale di SET: il modello opera a partire da segnali di riferimento sperimentali, i quali sono caratterizzati da variazioni brusche e tratti discontinui. Per attenuare le componenti ad alta frequenza indesiderate introdotte da questa natura discreta del segnale, si impiega un filtro passa basso del primo ordine, finalizzato a regolarizzare il profilo del segnale di riferimento e garantire una dinamica più compatibile con il comportamento fisico del sistema.
- Ritardo di trasporto sul segnale di FEEDBACK: a valle del subsystem che introduce gli effetti della videocamera nel servosistema viene inserito un ritardo di trasporto per tenere conto della presenza del Raspberry, della sua frequenza di lavoro e dell'algoritmo che stima la posizione dell'oggetto. Si sceglie un ritardo di 2 millisecondi.
- Ritardo di trasporto a valle del controllore PID: introdotto per rendere più realistico il comportamento del sistema. Si sceglie un ritardo di un decimo di millisecondo.

Poichè il modello è stato utilizzato per valutare solo alcuni effetti dell'introduzione della videocamera per poter utilizzare il segnale di feedback in Volt nel controllo e regolazione si è mantenuto il guadagno statico del trasduttore potenziometrico seppur esso non fa parte del sistema da sviluppare.

Nel Simulink sono stati inseriti anche dei blocchi *To Work Space* per esportare i segnali d'interesse su MATLAB, oltre all'introduzione di alcuni blocchi *Gain* per avere i segnali di feedback e di set sia in metri che in Volt.

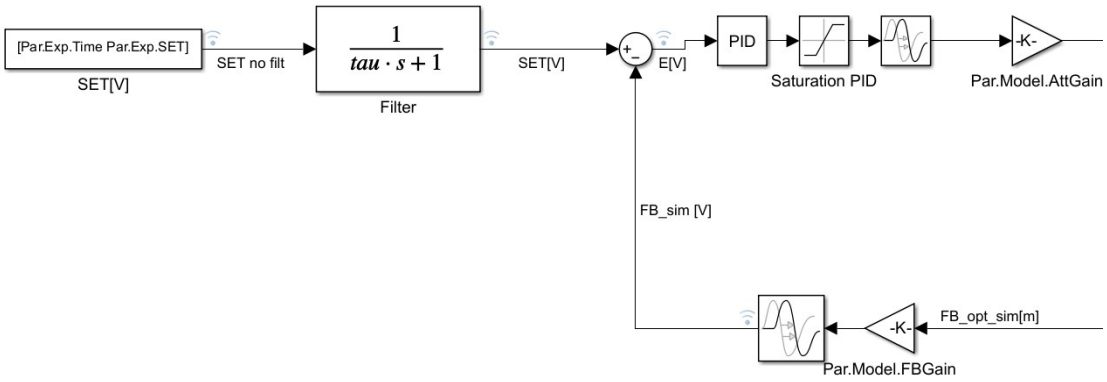

Figura 4.6: Ulteriori modifiche effettuate sul modello Simulink

4.2 Simulazioni

Il modello, a partire da un segnale di riferimento sperimentale permette di effettuare diverse simulazione al variare della frequenza e ampiezza del segnale di SET e del guadagno proporzionale K_p del controllore PID. Poichè il modello iniziale è stato utilizzato per valutare la larghezza di banda del servosistema partendo dalle informazioni note su di esso si cerca di verificare cosa comporta l'introduzione di un dispositivo come la videocamera nel servosistema. Attraverso le simulazioni si vuole:

- Valutare l'errore dinamico fra SET e FEEDBACK per comprendere la risposta del modello all'introduzione della videocamera.
- Confrontare modello con trasduttore potenziometrico e modello con videocamera.
- Stimare la larghezza di banda del sistema con feedback ottico.

Per fare ciò si prende come riferimento la larghezza di banda del sistema con trasduttore potenziometrico ed un guadagno proporzionale costante pari a 5. Essa risulta essere pari a 4 Hz con un'ampiezza del comando di 3 Volt e 2.5 Hz per un'ampiezza di 7 Volt. Si decide dunque di effettuare delle simulazioni con delle frequenze inferiori e maggiori rispetto alla larghezza di banda del sistema iniziale e visualizzare il comportamento dell'errore dinamico attraverso degli istogrammi. In questo modo si può comprendere in maniera molto rapida ed intuitiva come si comporta il sistema al variare della frequenza e dell'ampiezza del segnale di SET ed effettuare un confronto tra i due sistemi. Per tutte le simulazioni la configurazione

selezionata risulta essere quella con videocamere IMX477 con modalità video che garantisce il massimo frame rate disponibile. Si effettuano simulazioni variando anche l'ampiezza del segnale di SET, utilizzando 3 Volt e 7 Volt, in modo tale da evidenziarne l'effetto sul sistema.

4.2.1 Risultati modello trasduttore potenziometrico

A basse frequenze quello che ci si aspetta per entrambi i modelli è una distribuzione dell'errore intorno al valore 0 Volt poichè la regolazione risponde bene ed il feedback è in grado di seguire correttamente il segnale di riferimento.

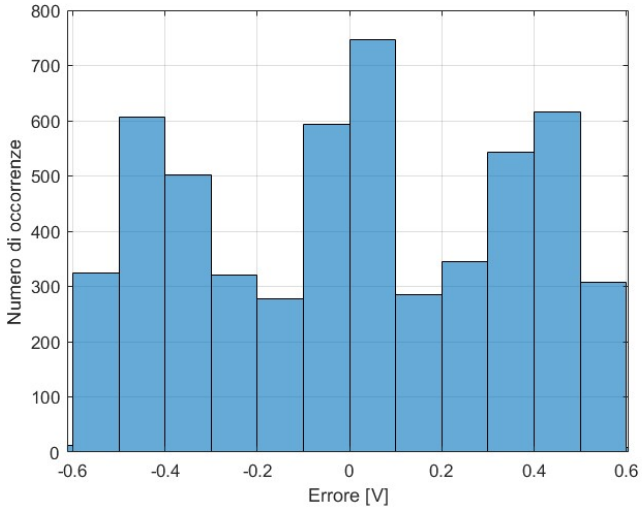

Figura 4.7: istogramma dell'errore per la prova con ampiezza 3 Volt frequenza pari ad 1 Hz modello con trasduttore potenziometrico

Come si può osservare alla basse frequenze l'errore non ha valori superiori a 0.5 Volt e la distribuzione presenta una forma a campana con il numero di occorrenze maggiore in corrispondenza del bin [0-0.1] Volt. All'aumentare della frequenza del segnale di riferimento, si osserva una modifica della distribuzione statistica dell'errore dinamico. In particolare, l'istogramma delle occorrenze dell'errore tende a mostrare un aumento delle occorrenze per valori assoluti di errore maggiori, segno di un peggioramento del controllo. Dal punto di vista statistico, la distribuzione dell'errore si appiattisce nella regione centrale (intorno allo zero) e presenta code più pronunciate, indicando un aumento della varianza e una probabile deviazione da una distribuzione gaussiana centrata. Questo comportamento è compatibile con un sistema che, a fronte di input ad alta frequenza, non riesce a seguire efficacemente le variazioni del SET, accumulando errori dinamici più marcati.

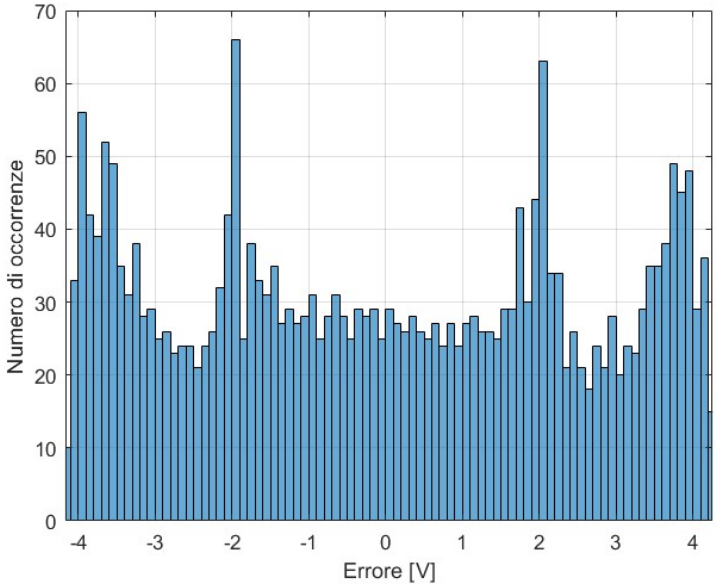

Figura 4.8: istogramma dell'errore per la prova con ampiezza 3 Volt frequenza pari ad 5 Hz modello con trasduttore potenziometrico

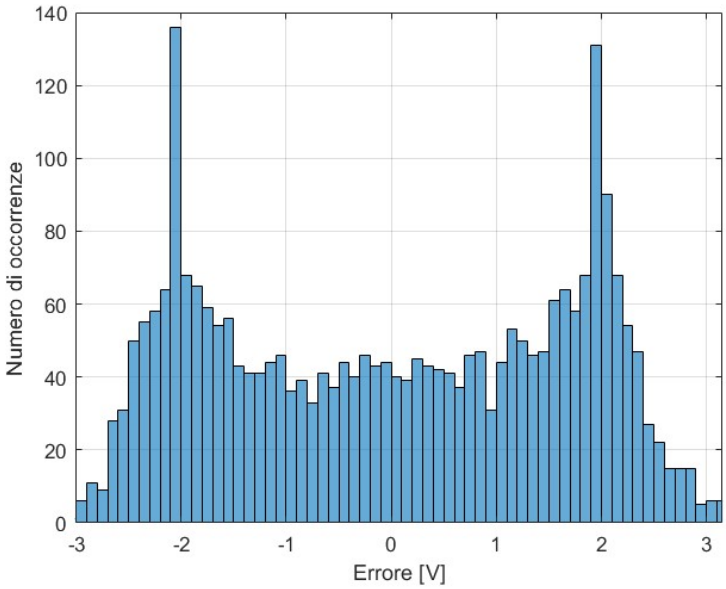

Figura 4.9: istogramma dell'errore per la prova con ampiezza 3 Volt frequenza pari ad 10 Hz modello con trasduttore potenziometrico

All'aumentare dell'ampiezza del segnale di SET, si osserva una variazione significativa nella distribuzione dell'errore, anche in condizioni di bassa frequenza. In particolare, pur mantenendo una forma approssimativamente gaussiana, la distribuzione tende ad allargarsi, ovvero a presentare una maggiore concentrazione di occorrenze in corrispondenza di errori con valore assoluto più elevato. L'incremento dell'ampiezza comporta maggiore tempo per raggiungere il SET. Questo si traduce in errori transitori più pronunciati, che contribuiscono ad aumentare la varianza della distribuzione. La media può restare prossima allo zero se il sistema è simmetrico e ben regolato, ma i valori assoluti degli scostamenti aumentano, generando code più evidenti nella distribuzione.

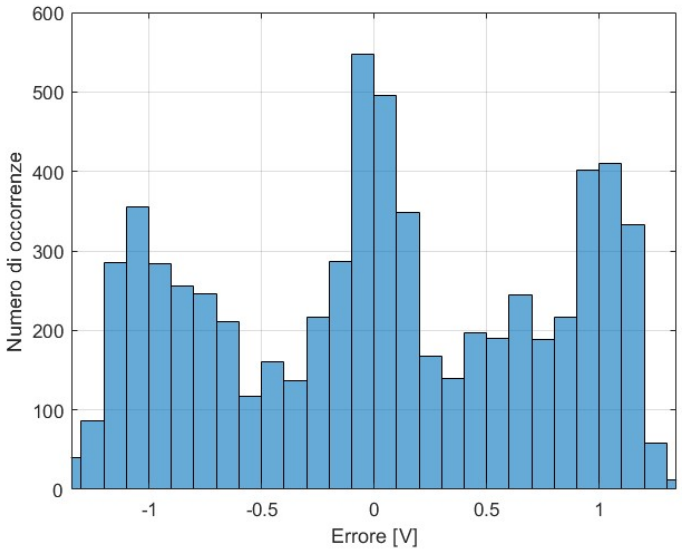

Figura 4.10: istogramma dell'errore per la prova con ampiezza 7 Volt frequenza pari ad 1 Hz modello con trasduttore potenziometrico

Per le prove con frequenza oltre la larghezza di banda del sistema la distribuzione dell'errore è molto simile a quanto visto per le prove ad ampiezza 3 Volt ovviamente tali risultati si raggiungono per frequenze inferiori poichè la larghezza di banda diminuisce all'aumentare dell'ampiezza del segnale di SET.

4.2.2 Confronto tra i modelli

Per il modello con feedback ottico, come già detto in precedenza, si ripropongono le medesime prove effettuate sul modello con trasduttore potenziometrico. Per il confronto fra i due modelli è interessante osservare i segnali di SET e di FEEDBACK per le prove più rilevanti.

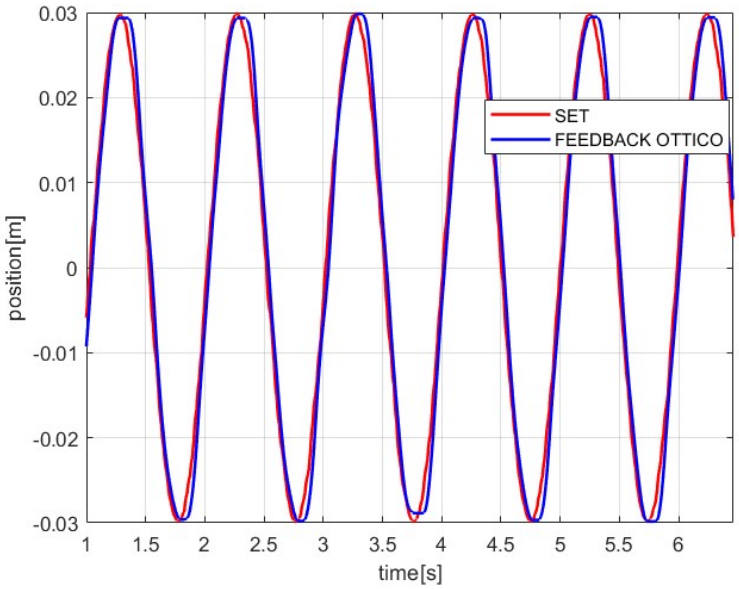

Figura 4.11: SET e FEEDBACK ampiezza 3 Volt frequenza 1 Hz modello con feedback ottico

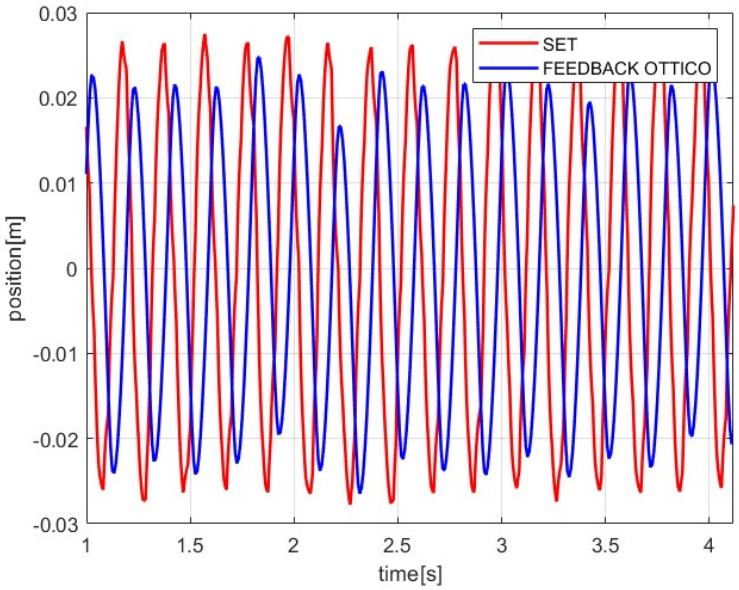

Figura 4.12: SET e FEEDBACK ampiezza 3 Volt frequenza 5 Hz modello con feedback ottico

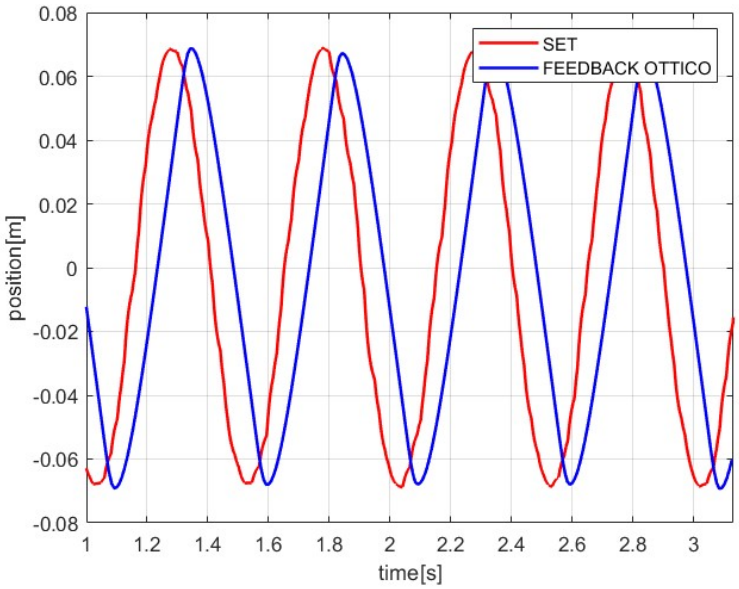

Figura 4.13: SET e FEEDBACK ampiezza 7 Volt frequenza 2 Hz modello con feedback ottico

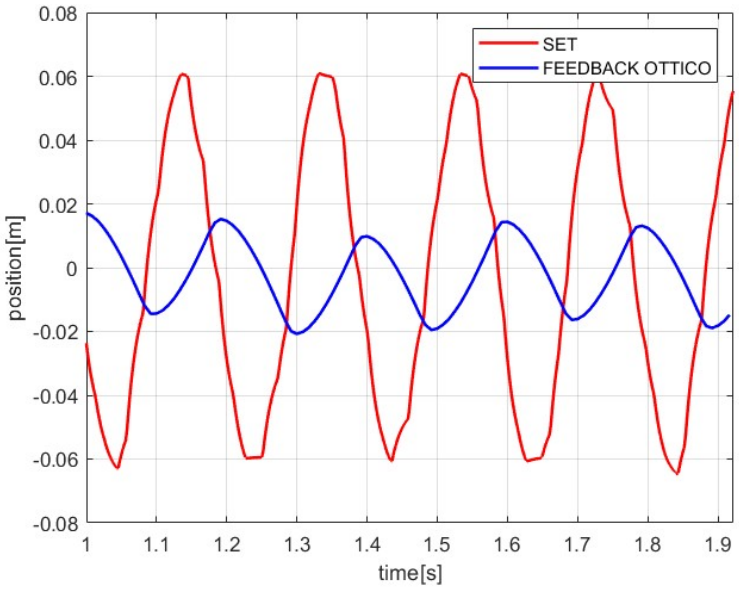

Figura 4.14: SET e FEEDBACK ampiezza 7 Volt frequenza 5 Hz modello con feedback ottico

I risultati ottenuti sono molto simili a quanto ottenuto con il modello con trasduttore potenziometrico, segno che la videocamera, per come è stata modellizzata, non disturba il comportamento del sistema. Si riportano come dimostrazione di ciò i risultati di due prove con modello avente il trasduttore potenziometrico:

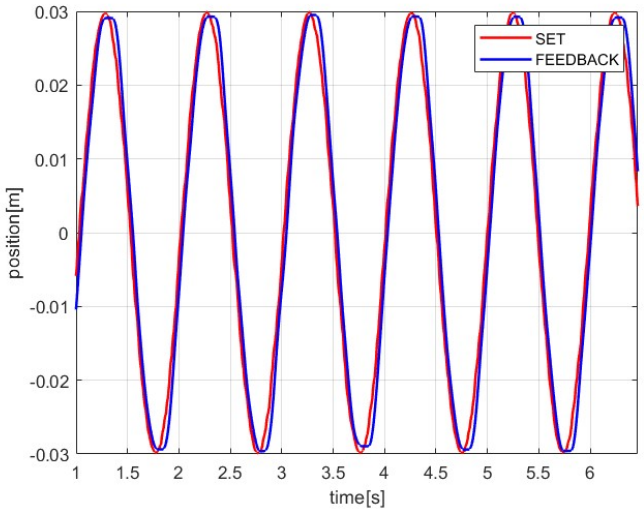

Figura 4.15: SET e FEEDBACK ampiezza 3 Volt frequenza 1 Hz modello con trasduttore potenziometrico

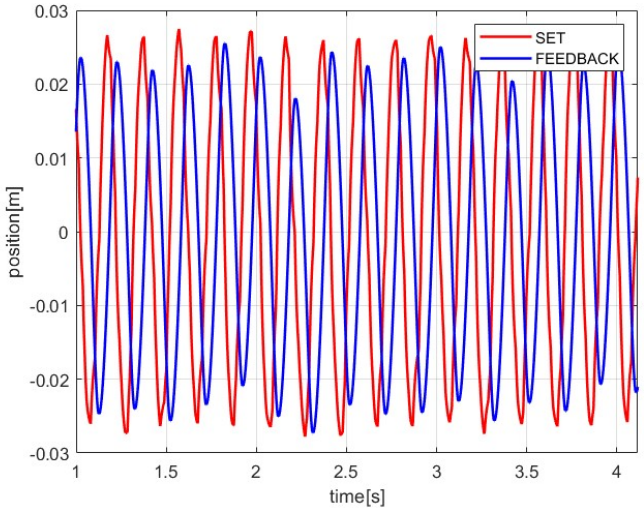

Figura 4.16: SET e FEEDBACK ampiezza 3 Volt frequenza 5 Hz modello con trasduttore potenziometrico

4.2.3 Risultati modello con feedback ottico

In questo paragrafo si riportano i risultati ottenuti nelle simulazioni effettuate con il modello avente il feedback ottico. Le prove effettuate sono le medesime svolte con il modello avente il trasduttore e anche in questo caso i risultati non si discostano molto da quanto precedentemente osservato. L'istogramma dell'errore presenta una distribuzione centrata in zero per prove a bassa frequenza sintomo di una regolazione e controllo efficace sul sistema anche in presenza di videocamere. Le code della distribuzione tendono ad allargarsi all'aumentare della frequenza proprio come evidenziato nel modello con trasduttore.

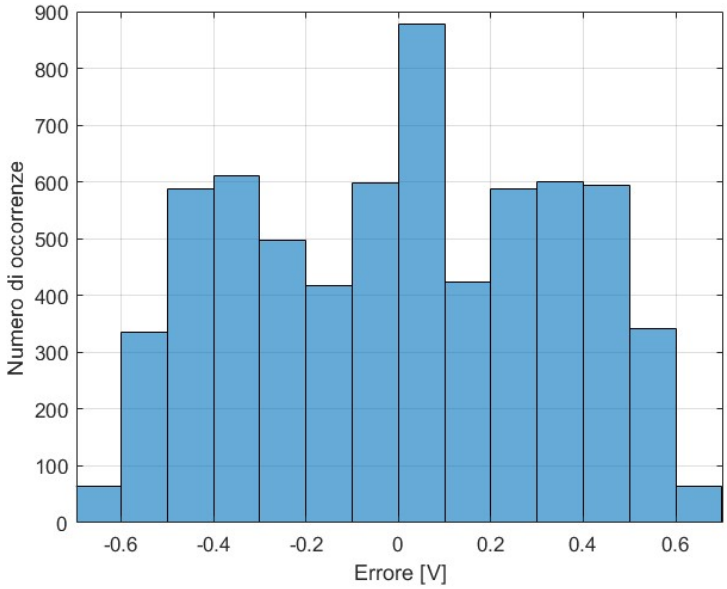

Figura 4.17: istogramma dell'errore per la prova con ampiezza 3 Volt frequenza pari ad 1 Hz modello feedback ottico

Nel caso della figura 4.17 l'errore risulta avere un numero di occorrenze attorno al bin [0-0.1] Volt maggiore rispetto al sistema avente nel ramo di feedback il trasduttore potenziometrico, mentre le code risultano pressochè avere le medesime occorrenze. Alle alte frequenze il comportamento risulta pressochè invariato. Questi risultati sono molto importanti poichè confermano che la videocamera con le sue caratteristiche intrinseche non condiziona il controllo del sistema. Si può affermare che il modello del feedback ottico risulta avere la medesima larghezza di banda del modello con trasduttore potenziometrico, per le prove con bassa ampiezza del segnale di comando.

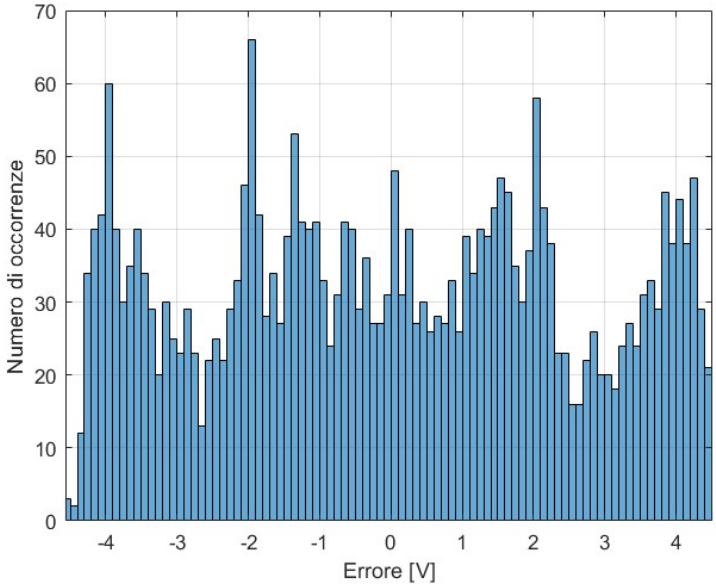

Figura 4.18: istogramma dell'errore per la prova con ampiezza 3 Volt frequenza pari a 5 Hz modello feedback ottico

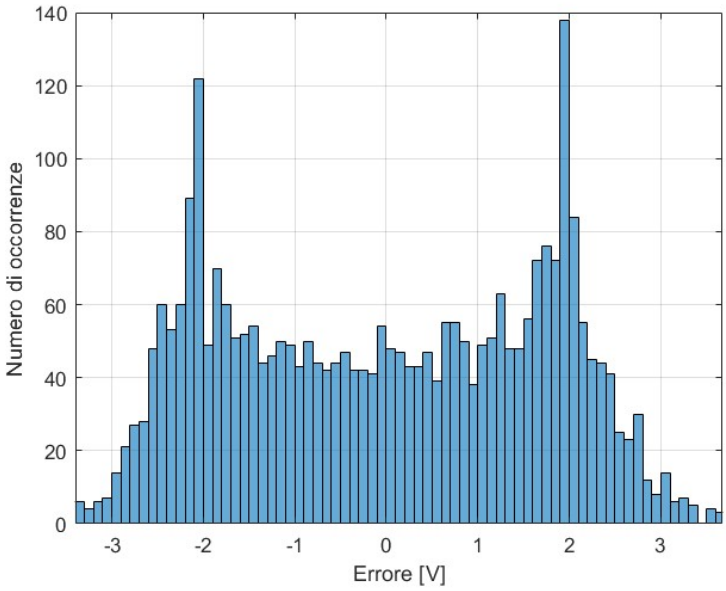

Figura 4.19: istogramma dell'errore per la prova con ampiezza 3 Volt frequenza pari a 10 Hz modello feedback ottico

Risulta interessante evidenziare anche in questo modello l'effetto dell'ampiezza del segnale di riferimento sulla distribuzione del valore di errore durante la prova. All'aumentare dell'ampiezza del segnale di comando si osservano maggiori occorrenze per valori di errore elevato, con l'errore massimo che aumenta in valore assoluto rispetto alle prove con ampiezza pari a 3 Volt.

I due istogrammi riportati nelle figure 4.20 e 4.21 si riferiscono rispettivamente ad una prova a bassa frequenza ed una con una frequenza che supera di molto la larghezza di banda di riferimento del modello iniziale. Anche in questo caso il comportamento è molto simile a quanto osservato nel modello con trasduttore. Si conferma anche per le prove con un'ampiezza di comando maggiore la medesima larghezza di banda del sistema iniziale.

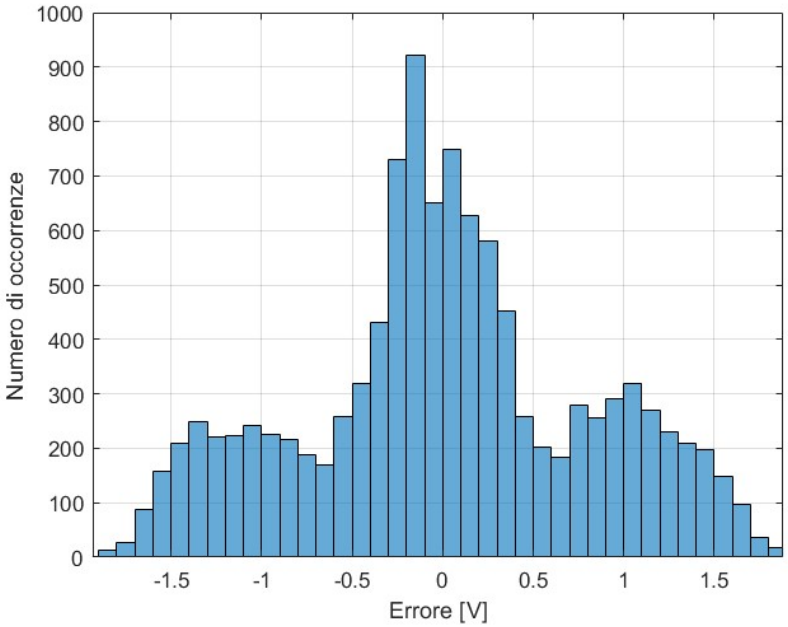

Figura 4.20: istogramma dell'errore per la prova con ampiezza 7 Volt frequenza pari a 2 Hz modello feedback ottico

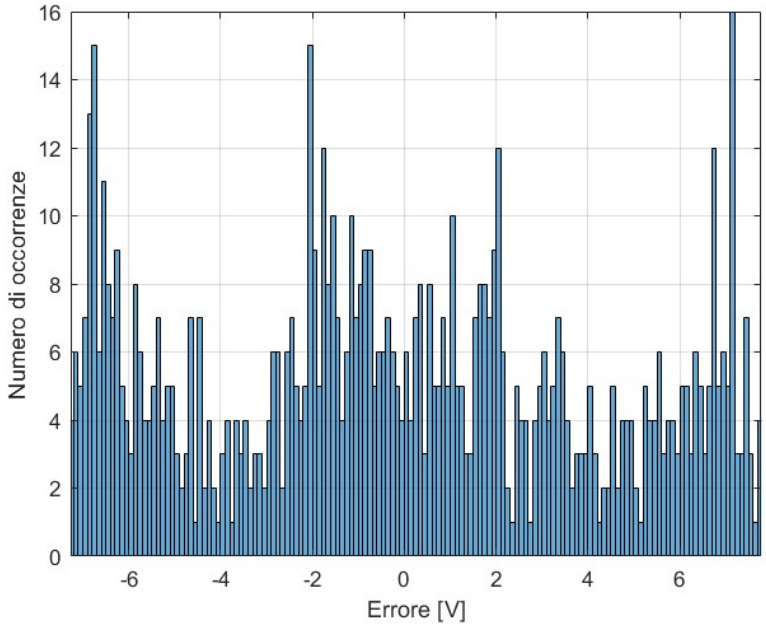

Figura 4.21: istogramma dell'errore per la prova con ampiezza 7 Volt frequenza pari a 5 Hz modello feedback ottico

Si conclude che la videocamera, con le caratteristiche considerate all'interno del modello non introduce una riduzione significativa della larghezza di banda del sistema che resta intorno ai 4 Hz per prove con ampiezza del segnale di comando 3 Volt e circa 2.5 Hz per prove con ampiezza 7 Volt. Ovviamente ciò si può riproporre per ulteriori prove con differenti guadagni proporzionali K_p e con diverse configurazioni di videocamere ma presumibilmente si otterrebbe il medesimo risultato. Da ciò si evince che piuttosto che la videocamera in se ad incidere sui tempi di risposta del sistema, ciò che potrebbe realmente influenzare la dinamica del sistema è la piattaforma hardware e l'algoritmo di riconoscimento e stima della posizione della slitta.

Capitolo 5

Attività sperimentale

In questo capitolo vengono descritte le attività sperimentali svolte che hanno portato allo sviluppo dell'algoritmo di object detection per il sistema in esame. Per l'utilizzo di videocamera come feedback del controllo posizione è necessario innanzitutto che il dispositivo sia in grado di riconoscere l'oggetto d'interesse e successivamente, attraverso image processing, restituire informazioni sulla posizione di quest'ultimo. Durante l'attività di tesi è stato sviluppato un algoritmo di riconoscimento della slitta tramite rete neurale *YOLO* sulla base di un dataset personalizzato. A monte dello sviluppo di tale algoritmo che porta alla rilevazione mediante videocamere del carrello di cui si vuole controllare la posizione sono state svolte delle attività che hanno riguardato l'impostazione del set-up hardware e software per il controllo delle videocamere tramite Raspberry Pi. Per sviluppare il dataset su cui allenare la rete neurale è necessario effettuare delle acquisizioni video del carrello tramite le videocamere posizionate nelle configurazioni ottimali ottenute a valle dello studio geometrico effettuato.

5.1 Configurazione e controllo delle videocamere

Una volta installato il sistema operativo per il Raspberry Pi CM5 ed aver aggiornato i pacchetti necessari per il suo utilizzo, in prima battuta ci si è concentrati sull'impostazione ed il controllo delle videocamere acquistate. La maggior parte delle videocamere Arducam, tra cui quelle d'interesse, possono essere controllate tramite un'apposita libreria detta *libcamera*. Essa permette di controllare la videocamera da terminale di Raspberry tramite appositi comandi permettendo alla videocamera di registrare video, effettuare una preview a schermo di ciò che sta inquadrando, verificarne il corretto funzionamento e modificare i suoi parametri estrinseci. Un'altra libreria di Python installata e utilizzata risulta essere *OpenCV*, una libreria open source specializzata in visione artificiale e elaborazione di immagini tra le più

utilizzate per progetti di Computer Vision.

Per quanto riguarda la connessione hardware, connettere delle videocamere Arducam ad un Raspberry Pi è abbastanza semplice: ogni videocamera è fornita di cavi flessibili 15 pin-22 pin oppure 22 pin-22 pin per la connessione con tutte le tipologie di Raspberry e bisogna collegare un'estremità alla videocamera ed un'altra estremità alla porta MIPI CSI disponibile. Vi sono delle precisazioni da fare a riguardo:

- Le piste conduttive del cavo flessibile devono essere rivolte verso i contatti della porta CSI, ossia verso l'interno del connettore, in modo che i contatti elettrici del cavo entrino in diretto contatto con i pin metallici presenti nella porta CSI della Raspberry Pi.
- Anche per la connessione dal lato del modulo camera tendenzialmente si vogliono le piste conduttive rivolte verso i contatti elettrici del modulo, ma ci sono alcuni moduli in cui la connessione risulta poco intuitiva.

Per ogni videocamera acquistata vi è una guida fornita da Arducam che spiega dettagliatamente gli step della configurazione hardware e software. Vi è una sezione dedicata ad ogni videocamera in cui si descrive nel dettaglio quanto fatto per configurarle e controllarle.

5.1.1 Arducam IMX477

In [34] è presente una guida che permette la configurazione della videocamera IMX477. Per prima cosa tramite terminale viene creata una cartella dal titolo "`imx477_dtb_test`" al cui interno vengono caricati i file necessari per il controllo del driver V4L2 in grado di gestire il modulo in questione. Effettuando l'unzip di quanto scaricato da GitHub è possibile salvare gli overlay del sensore. Gli overlay sono dei file di configurazione che servono per modificare o estendere la descrizione dell'hardware. Nel dettaglio, quando si connette un dispositivo come una videocamera tramite interfaccia CSI, il sistema ha bisogno di sapere come è fatto quel dispositivo e come comunicare con esso. I file overlay permettono di fornire queste informazioni. Attraverso il comando `./build_and_install.sh` è possibile compilare questi file e installare l'overlay sul sistema, così che al riavvio il Raspberry Pi sappia riconoscere e gestire il modulo camera. Un altro step fondamentale per la configurazione di un modulo camera su Raspberry Pi è la modifica del file di configurazione `config.txt` che permette, tra le svariate funzioni, il rilevamento dei moduli camera attraverso il caricamento degli overlay. Per permettere al Raspberry di riconoscere correttamente la videocamera è necessario attraverso il comando `sudo nano /boot/firmware/config.txt` accedere al file di configurazione e modificarlo come segue:

```
1 camera_auto_detect=0
2 dtoverlay=imx477,vcm,cam0
3 dtoverlay=imx477,vcm,cam1
```

La prima riga serve a disattivare il caricamento automatico degli overlay già installati nel Raspberry in modo tale da evitare eventuali conflitti. Le altre due righe sono gli overlay necessari per collegare il modulo IMX477 alla porta *cam/disp0* e *cam/disp1*. Bisogna sottolineare che gli overlay spesso creano problemi nel riconoscimento dei moduli camera generando conflitti che impediscono alla videocamera di essere riconosciuta dal Raspberry. Buona norma è quella di effettuare la modifica del file di configurazione ogni qualvolta si voglia collegare un sensore ad una determinata porta, commentando gli overlay di moduli che non devono essere utilizzati. Per quanto riguarda la configurazione hardware, vale quanto detto in precedenza. L'IMX477 necessita delle piste conduttive rivolte verso i contatti elettrici, sia lato camera che lato Raspberry.

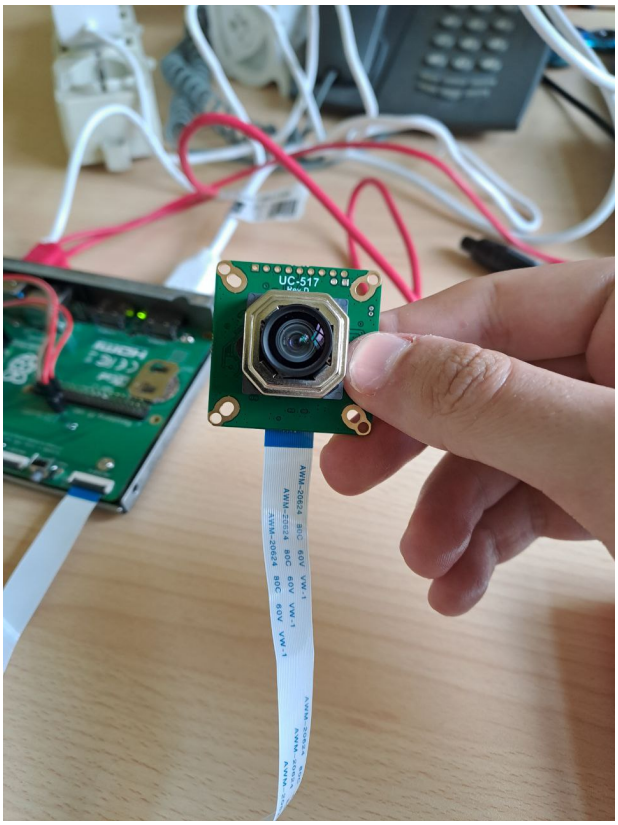

Figura 5.1: Collegamento IMX477 a Raspberry CM5

Dopo aver collegato la videocamera e modificato il file di configurazione, è stato necessario effettuare il reboot del dispositivo con `sudo reboot` e installare *libcamera* scaricando i bash scripts necessari. In [34] sono presenti i links utili ed una descrizione step by step della procedura.

Il modulo IMX477 è diverso dagli altri moduli camera acquistati. Presenta infatti diverse modalità per catturare immagini e video impostabili tramite terminale una volta installato *libcamera*, in particolare:

- Continuos autofocus: una volta selezionata tale modalità l'obiettivo mette a fuoco in maniera automatica ogni qualvolta vi è un cambiamento del soggetto a causa di uno scenario dinamico.
- Single autofocus: utilizzando questa modalità si mette a fuoco solo una volta all'inizio della preview o del video che si vuole registrare con tale modulo.
- Manual focus: il focus è liberamente impostabile dall'utente in base all'occorrenza da tastiera. Premendo il pulsante "f" si genera un autofocus istantaneo, "a" e "d" spostano la posizione della lente.
- Adjust lens position: è simile al manual focus perchè permette di impostare da terminale la posizione della lente.
- Autofocus before capture images: per impostare un autofocus pochi istanti prima che venga scattata una foto attraverso la videocamera.
- Set focus range: serve a definire l'intervallo di messa a fuoco su cui la videocamera deve concentrarsi durante l'autofocus. In pratica, imposta se la camera deve cercare il fuoco su soggetti lontani, vicini o in un intervallo generico. L'opzione *macro* viene utilizzata per soggetti vicini, *normal* per mettere a fuoco una vasta gamma di oggetti ad una distanza standard.

Tutte queste modalità possono essere impostate da terminale per l'acquisizione video o foto attraverso *libcamera*, oppure direttamente impostate in uno script Python che sfrutta la medesima libreria. Come si potrà osservare, l'acquisizione dei video utile per generare il custom dataset su cui allenare la rete verrà effettuata tramite comando da terminale poichè risulta essere una soluzione rapida ed efficace.

5.1.2 Arducam OG02B10

In [35] è presente una guida utile per la configurazione hardware e software del sensore Global Shutter OG02B10. Bisogna sottolineare che tale sensore appartiene ad una famiglia di moduli camera compatibili con Raspberry: Pivariety Camera modules. Di conseguenza la guida non fa riferimento in maniera specifica al sensore

in questione ma a diversi moduli camera. Per quanto riguarda la configurazione del sensore in questione gli step da seguire sono sempre gli stessi: connessione hardware del sensore al Raspberry, installazione di libcamera e modifica del file di configurazione. Per la connessione hardware le piste conduttive devo sempre essere rivolte verso i connettori elettrici del Raspberry:

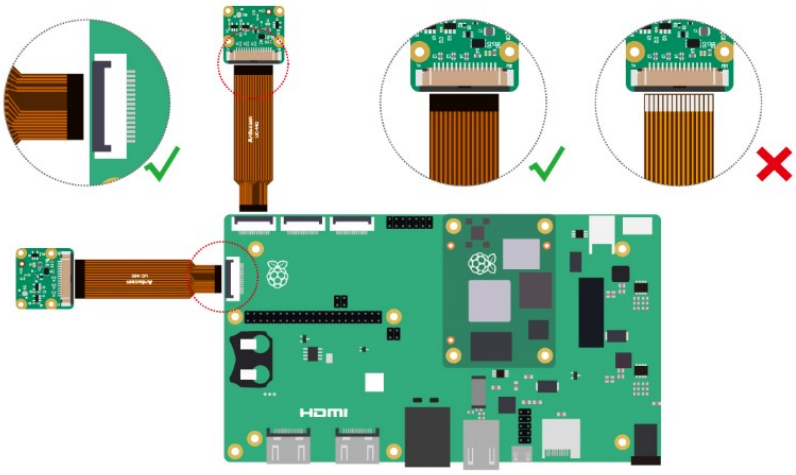

Figura 5.2: Connessione Pivariety camera module a Raspberry Computer Module 4[35]

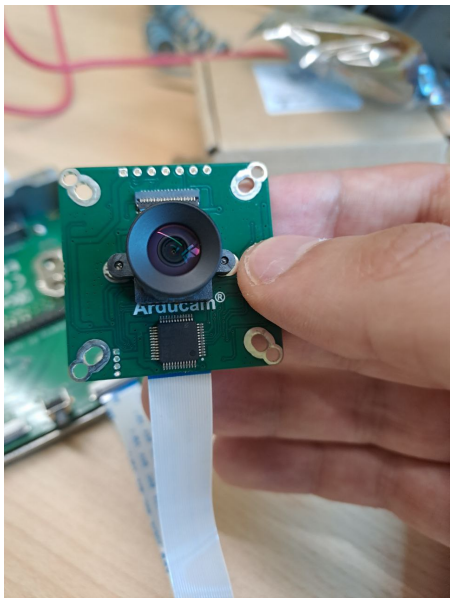

Figura 5.3: Connessione modulo OG02B10 lato camera

Il file di configurazione per il caricamento degli overlay si modifica come segue:

```
camera_auto_detect=0
dtoverlay = arducam-pivariety,cam0 ;
dtoverlay = arducam-pivariety,cam1;
```

Dopo la modifica del file di configurazione è consigliato effettuare il reboot del Raspberry con il comando `sudo reboot`. All'interno della guida è poi presente una descrizione dettagliata su come installare *libcamera* (procedura identica a quella seguita per il sensore IMX477) e dei comandi base per verificare il funzionamento della videocamera attraverso *libcamera* come una preview a schermo di 5 minuti oppure come scattare una foto attraverso appositi comandi da terminali:

```
libcamera-still -t 5000; #preview
libcamera-still -t 5000 -n -o test.jpg; #capture a image
```

Vi è anche una sezione dedicata all'utilizzo dei V4L2 tools per controllare la videocamera.

5.1.3 Raspberry Pi Camera Module 3

Per la configurazione ed il controllo del Raspberry Pi Camera Module 3 si fa riferimento alla documentazione officiale di Raspberry sui Camera Module[36]. Tale documentazione descrive in maniera precisa come utilizzare *libcamera* e modificare i parametri intrinseci della videocamera. Inoltre, in [23] è presente una guida su Picamera2, una libreria Python basata su *libcamera* che permette il controllo della videocamera attraverso apposito script Python e che tornerà molto utile nella fase di object detection. Per quanto riguarda la configurazione del modulo in questione, una volta installato *libcamera*, bisogna modificare il file di configurazione come segue:

```
camera_auto_detect=0
dtoverlay=imx708,cam0
dtoverlay=imx708,cam1
```

Anche in questo caso, una volta effettuata tale modifica è opportuno effettuare il reboot del dispositivo. Per quanto riguarda la connessione hardware, non vi è un riferimento preciso al modulo in questione e al Raspberry Computer Module 5 tuttavia basandosi su immagini presenti nella documentazione che presentavano la connessione lato camera con un cavo flessibile è stato possibile collegare la videocamera al Raspberry. Lato camera le piste conduttive sono sul lato frontale

della videocamera mentre si rivolgono verso i connettori elettrici lato Computer Module.

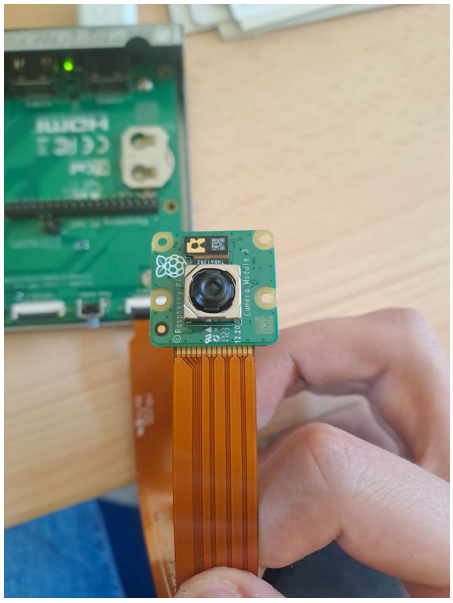

Figura 5.4: Collegamento Picamera Module 3 lato camera

Figura 5.5: Collegamento Picamera Module 3 lato Raspberry

5.1.4 Connessione e controllo di due videocamere

Il Raspberry Pi Computer Module 5 presenta due porte MIPI CSI-2 che consentono il collegamento e lo streaming contemporaneo di due videocamere. Una volta appresso come configurare e controllare i sensori acquistati si è voluto collegare due sensori al Raspberry per verificarne il funzionamento. Per connettere due videocamere sono stati svolti i seguenti step:

- Bisogna utilizzare due jumper nei pin J6 per abilitare la porta *cam/disp1*.
- Bisogna modificare il file di configurazione in modo tale da avere solo gli overlay dei sensori d'interesse.

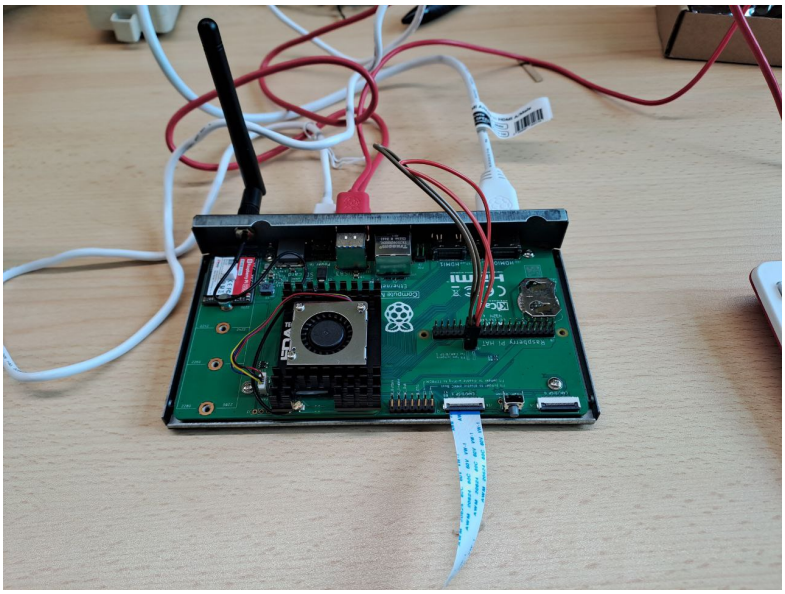

Figura 5.6: Ponticelli su J6 per il collegamento delle videocamere nella porta *cam/disp1*

Le videocamere restano controllabili con le apposite librerie come *OpenCV*, *libcamera* o *Picamera2*. È stato sviluppato uno script in Python che permetteva lo streaming in simultanea di due moduli connessi alle rispettive porte del Raspberry per verificarne il corretto funzionamento. Lo streaming funziona sia con sensori dello stesso tipo che con sensori diversi. Se si volessero collegare più di due videocamere si necessita di un Multiplexer HAT da montare sul Computer Module.

5.2 Acquisizione video

In questa sezione si descrivono le attività di acquisizione video della slitta utili per allenare la rete neurale al riconoscimento dell'oggetto d'interesse. Le acquisizioni sono state svolte utilizzando la libreria *libcamera* utilizzando opportuni comandi da terminale. I risultati sono stati trasferiti dal Raspberry ad un PC per la creazione del dataset ed il traning della rete. Queste ultime due procedure non sono state svolte su Raspberry poichè dispendiose dal punto di vista computazionale.

5.2.1 Libcamera

Per comprendere la procedura di acquisizione video è opportuno conoscere la libreria *libcamera* ed i comandi da terminale utilizzati per tale attività. *Libcamera* è una libreria open source che ha come obiettivo quello di controllare i moduli camera direttamente dal sistema operativo Linux tramite terminale. *Libcamera* fornisce un'API in C++ che configura la videocamera, permettendo poi alle applicazioni di richiedere i fotogrammi. Questi buffer di immagine risiedono nella memoria di sistema e possono essere passati direttamente a codificatori di immagini statiche (come JPEG) o a codificatori video (come H.264). Libcamera non si occupa direttamente della codifica o visualizzazione delle immagini: per queste funzionalità si utilizzano le *rpicam-apps*[36]. Una volta effettuata la configurazione dei moduli come descritto nei paragrafi ad essi dedicati è possibile controllarli attraverso libcamera. Si elencano e descrivono i principali comandi utilizzati per verificare il corretto funzionamento dei moduli camera:

- `rpicam-hello` o `libcamera-hello`: avviano una preview a schermo del modulo camera connesso al Raspberry.
- `rpicam-hello -list-cameras`: permette di visualizzare da terminale un elenco dei moduli camera connessi al Raspberry e tutte le modalità video supportate dal sensore in termine di risoluzione, frame rate e colore.
- `libcamera-hello -t 0`: preview a schermo di durata indefinita.
- `libcamera-vid -t 10000 nome.mp4`: permette di registrare un video della durata di 10 secondi in formato mp4.
- `rpicam-still -output test.jpg`: permette di scattare una foto alla risoluzione massima del sensore.

La struttura del comando da terminale per *libcamera* è sempre la stessa: si richiama la libreria, subito dopo si digita la funzione che si vuole realizzare, ad esempio `hello` oppure `vid`, mentre con `-` è possibile imporre i parametri intrinseci del sensore,

la modalità di focus, la durata dell'acquisizione (in quest'ultimo caso si utilizza il comando `-timeout`). Si riporta una tabella riassuntiva dei principali comandi utilizzati per modificare le acquisizioni video in termine di parametri intriseci del modulo camera:

Comando	**Descrizione**
`--width`	imposta risoluzione in pixel orizzontali
`--height`	imposta risoluzione in pixel verticali
`--framerate`	imposta gli fps
`--shutter`	imposta l'esposizione del sensore
`--autofocus-mode continuos`	imposta la modalità autofocus continua
`--autofocus-mode auto`	imposta la modalità con autofocus singolo
`--autofocus-mode manual`	imposta la modalità focus manuale

Tabella 5.1: Comandi utilizzati per l'acquisizione video con *libcamera*

5.2.2 Comandi per acquisizione video

Sulla base delle modalità video evidenziabili con apposito comando da terminale, ed in accordo con le specifiche tecniche delle videocamera, è stato possibile scrivere una lista di comandi che consente l'acquisizione video della slitta per tutti e tre i sensori acquistati. Per ogni sensore sono stati creati degli appositi comandi da terminale sulla base delle modalità video disponibili.

Si riportano i comandi per il sensore IMX477:

```
1 libcamera-vid --timeout 6000 --width 2028 --height 1520--framerate
      40 Video_IMX_FPS40.mp4
2 libcamera-vid --timeout 6000 --width 2028 --height 1080 --
     framerate 50 Video_IMX_FPS50.mp4
3 libcamera-vid --timeout 6000 --width 1332 --height 990 --framerate
      120 Video_IMX_FPS120.mp4
```

Come si può osservare si seleziona il frame rate e la risoluzione e si impone un tempo di acquisizione in millisecondi, in questo caso si riporta un esempio di una registrazione video della durata di un minuto. Come ultime informazioni nel terminale bisogna inserire nome del video e formato.

Si riportano i comandi per le acquisizioni video relative al sensore OG02B10:

```
1 libcamera-vid --timeout 6000 --width 1600 --height 1300 --
     framerate 60 Video_OG_FPS60.mp4
2 libcamera-vid --timeout 6000 --width 1600 --height 1080 --
     framerate 80 Video_OG_FPS80.mp4
3 libcamera-vid --timeout 6000 --width 1280 --height 720 --framerate
      120 Video_OG_FPS120.mp4
```

Anche per il Picamera module 3 sono stati creati i seguenti comandi per le acquisizione video:

```
1 libcamera-vid --timeout 6000 --width 1920 --height 1080 --
     framerate 50 Video_IMX_FPS50.mp4
2 libcamera-vid --timeout 6000 --width 1280 --height 720 --framerate
      100 Video_Picam_FPS100.mp4
3 libcamera-vid --timeout 6000 --width 640 --height 480 --framerate
     120 Video_Picam_FPS120.mp4
```

5.2.3 Set up acquisizione video

Tale attività risulta fondamentale poichè sulla base delle acquisizione effettuate si andrà a costruire il dataset per allenare la rete al riconoscimento dell'oggetto. È fondamentale effettuare delle acquisizioni che rispecchiano il contesto operativo in cui le videocamere andranno ad identificare la slitta poichè algoritmi di questo tipo danno risultati poco attendibili nel caso la rete non sia allenata in maniera robusta. Per tale motivo, occorre posizionare le videocamere nelle posizioni ottimali ottenute durante lo studio geometrico.

Tra le videocamere disponibili si sceglie la Pi Camera Module 3, la quale viene posizionata in mezzeria alla distanza dalla slitta stimata attraverso il modello sviluppato nel Capitolo 3. In questa fase del lavoro è importante validare il processo che porta al riconoscimento dell'oggetto quindi si sceglie la configurazione geometrica più semplice possibile con un solo sensore posizionato in mezzeria. Per allenare la rete, è necessario che i video acquisiti riproducano in maniera fedele tutte le possibili condizioni operative della slitta la quale può ricevere diversi segnali di riferimento. Per tale motivo, sono state effettuate diverse acquisizioni video per diversi comandi, al variare del frame rate. Per ogni tipologia di comando sono state effettuate due acquisizioni una a framerate minimo, ovvero 60 FPS, e un'altra a framerate massimo cioè 120 FPS. I video sono stati registrati in formato MP4 con una durata di 25 secondi ad acquisizione. Per rendere il training robusto è

utile anche cambiare la posizione della videocamera in modo da inquadrare parte dell'ambiente esterno, in modo tale che sia in grado di distinguere la slitta da altri elementi. Per tale motivo è stata effettuata una registrazione con la videocamera sempre in mezzeria ma ad una distanza maggiore rispetto a quella stabilita dallo studio geometrico.

Nome	**Descrizione**
test-picam-fps60-F1	Comando sinusoidale 1 Hz
test-picam-fps120-F1	Comando sinusoidale 1 Hz
test-picam-fps60-SQ	Comando a gradino
test-picam-fps120-SQ	Comando a gradino
test-picam-fps60-distance	Comando sinusoidale, posizionamento differente

Tabella 5.2: Tabella riassuntiva delle acquisizioni video

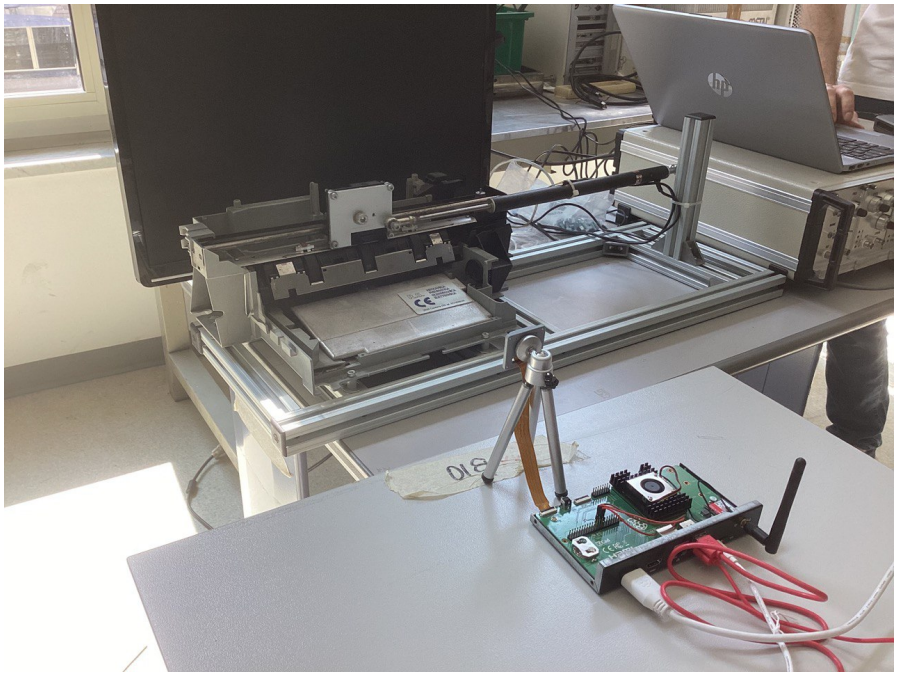

Figura 5.7: Set up Acquisizione video

5.3 Training della rete

In questa sezione si descrive la procedura seguita per effettuare il training della rete neurale volta al riconoscimento dell'oggetto. Per la creazione del dataset personalizzato su cui addestrare la rete ci si è affidato al sito *Roboflow*[37]. *Roboflow* è una piattaforma online utile per chi lavora su progetti di visione artificiale in particolare su modelli di object detection e classificazione di oggetti. Nasce per semplificare tutto il flusso che va dalla raccolta delle immagini fino all'addestramento e al deploy del modello, cioè tutte quelle fasi che normalmente richiederebbero molto tempo e competenze tecniche specifiche. Di seguito si elencano le principali funzionalità che fornisce il sito:

- Caricamento e selezione di immagini e video: si possono caricare immagini e video in diversi formati per la costruzione di dataset personalizzati.
- Labeling: offre un'interfaccia grafica in grado di effetttuare le annotazioni utili al riconoscimento dell'oggetto attraverso opportune bounding boxes. Possiede diversi tools per questa funzione per costruire bounding boxes quadrate, poligonali o personalizzate.
- Preprocessing e augmentation: permette di applicare delle modifiche alle foto o frame video caricati modificando in un range ben definito dall'utente alcuni parametri dell'immagine come ad esempio luminosità, esposizione, saturazione e tanto altro. Ciò permette di aumentare le immagini del dataset rendendo più robusta la rete che è in grado di riconoscere gli oggetti in diverse condizioni non solo nei casi proposti dalle immagini caricate.
- Suddivisione del dataset: tale piattaforma può suddividere le immagini in train, validation e test attraverso delle percentuali impostate automaticamente oppure selezionate dall'utente. Questo garantisce che il modello impari da un sottoinsieme di dati e venga valutato su esempi mai visti.
- Esportazione nei formati standard: il dataset può essere convertito e scaricato nei formati d'interesse. *Roboflow* supporta tutti i modelli *YOLO* fino al più recente e non solo.
- Training e Deploy: la piattaforma offre anche l'addestramento ed il deploy senza effettuarlo su computer in locale tramite Python ad esempio.

Per il caricamento delle immagini si sono utilizzati i frame delle acquisizioni video effettuate in laboratorio. La piattaforma una volta caricato un determinato video permette di selezionare il frame rate per campionare delle immagini da utilizzare nel dataset. Considerando la necessità di utilizzare diverse acquisizioni video ed il limite di 500 immagini per costruire il dataset nella versione gratuita di *Roboflow* è stato necessario limitare il frame rate a pochi FPS.

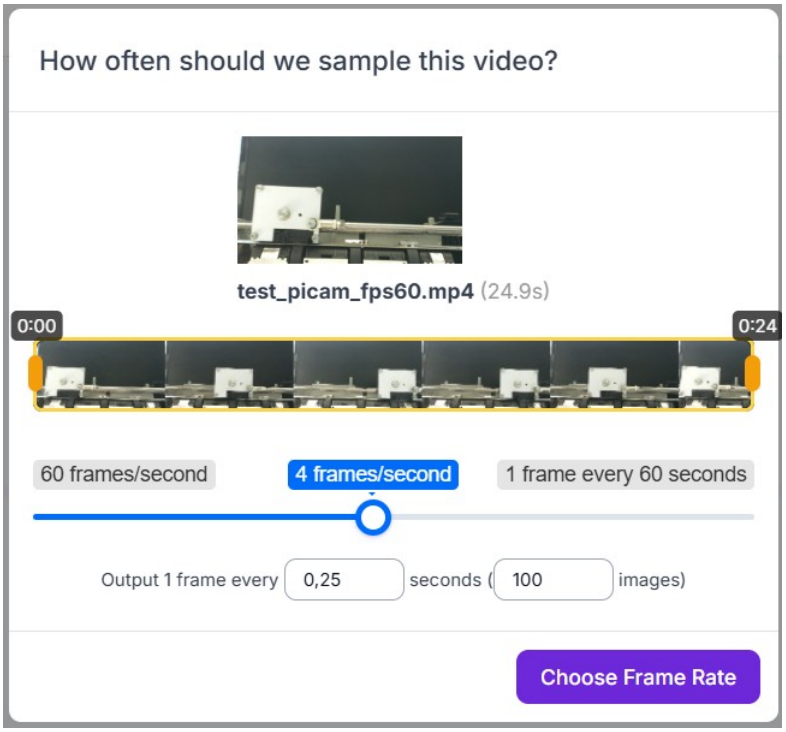

Figura 5.8: Scelta frame rate per cattura immagini su *Roboflow*

Come si può osservare dalla figura 5.8 è possibile selezionare il frame rate tramite apposito cursore oppure il numero di immagini richieste da tastiera. Si può anche tagliare il video per selezionarne una determinata parte d'interesse. I formati supportabili sono *.mp4* o *.mov*. Si sono scelte di base 200 immagini su cui basare il dataset, escludendo le fasi di preprocessing e augmentation che portano il dataset fino a 500 immagini. Una volta effettuato il caricamento delle immagini si è passati alla fase di labeling. Essa risulta essere un passaggio cruciale quando si prepara un dataset personalizzato per il training di una rete neurale. Senza questa fase non si avrebbe nessun riferimento su cui imparare. Per prima cosa si selezionano le classi di oggetti che si vogliono riconoscere. Nell'attività sviluppata tale selezione è abbastanza semplice poichè bisogna identificare un solo oggetto ovvero il carrello di cui si vuole effettuare il riconoscimento ed il controllo della posizione. Tutto il resto risulterà essere *background*, ovvero tutte le aree senza oggetto. Una volta impostate le classi si procede alle annotazioni delle immagini selezionate attraverso l'utilizzo di bounding boxes, ovvero rettangoli che racchiudono l'oggetto d'interesse. Il labeling deve essere il più preciso possibile: un box troppo largo include troppo sfondo, uno troppo stretto taglia parti importanti dell'oggetto da riconoscere. Una volta costruiti i bounding boxes bisogna assegnare ad ognuno di essi la classe corrispondente. Nell'attività sviluppata vi è una sola classe etichettata come "*sled*",

mentre tutto il resto della scena non riceve annotazioni.

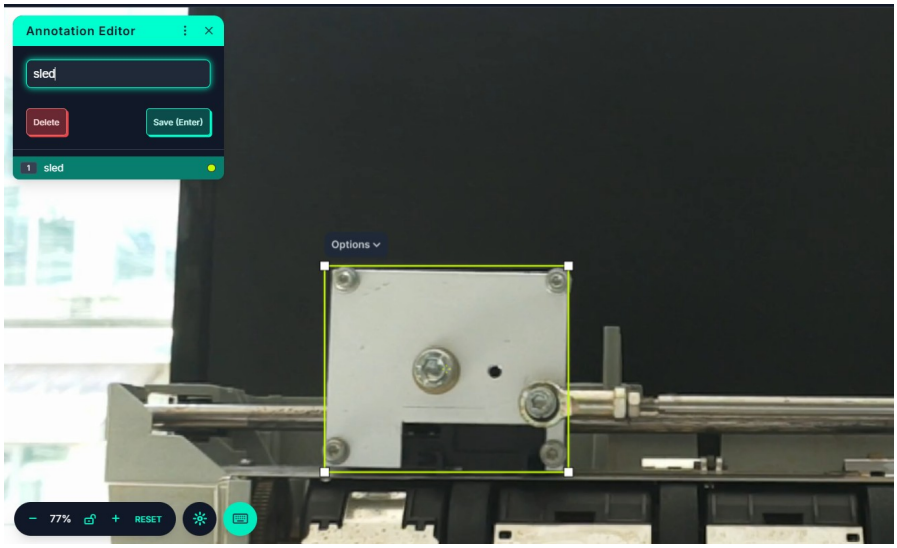

Figura 5.9: Esempio di labeling per un'immagine del dataset

Per effettuare il labeling *Roboflow* fornisce diversi tools in grado di rendere il processo preciso e automatizzato il più possibile. Permette di creare bounding boxes con forme geometriche come rettangoli o quadrati ma anche forme irregolari in base alla geometria dell'oggetto da riconoscere. Una volta concluso il labeling è possibile avere una visione d'insieme di tutto il dataset per controllarlo e verificare che non ci siano eventuali errori come etichette sbagliate o bounding boxes imprecisi che potrebbe compromettere la robustezza della rete trasmettendo informazioni ambigue. Successivamente è possibile svolgere preprocessing e augmentation sulle immagini annotate. Il preprocessing serve a trasformare le immagini in un formato coerente, standardizzato e compatibile con la rete. Ad esempio gli algoritmi *YOLO* lavorano con una dimensione dell'immagine di 640x640 quindi le immagini selezionate necessitano di un ridimensionamento. L'augmentation introduce variazioni artificiali nelle immagini per simulare condizioni reali più varie di quelle catturate nel dataset. Serve a prevenire l'overfitting, cioè quando la rete impara troppo bene i pochi esempi presenti e non riesce a generalizzare su nuove immagini. Sono differenti gli effetti applicabili alle immagini per aumentare la varietà del dataset, da trasformazioni geometriche come rotazioni e ritagli dell'immagini fino ad effetti sulla saturazione, la luminosità ed il contrasto di ogni singola immagine. Attraverso la piattaforma è possibile selezionare gli effetti desiderati ed anche un range di valori sui cui costruire nuove immagini a partire dalle originali. Ad esempio, se si vuole modificare la luminosità dell'immagine in modo tale da rendere l'algoritmo più robusto in diverse condizioni operative è possibile selezionare gli estremi di un range di luminosità

su cui costruire nuove immagini, come −15% e +15%, e la piattaforma riuscirà a generare delle nuove immagini da inserire nel dataset con tale caratteristica variabile attorno a quest'intervallo.

Caratteristica	**Intervallo**
Brightness	[−10% ; +10%]
Saturation	[−25% ; +25%]
Exposure	[−15% ; +15%]

Tabella 5.3: Caratteristiche modificate nella fase di augmentation

Una volta scelte le caratteristiche desiderate ed impostati i rispettivi intervalli, *Roboflow* genera automaticamente le immagini previa richiesta all'utente di selezionare il limite di immagini volute nel dataset. La versione gratuita di *Roboflow* permette di utilizzare fino a 500 immagini, si sceglie proprio tale valore in modo tale da addestrare la rete con un dataset il più robusto possibile. Prima di scaricare il dataset nella versione d'interesse è necessario catalogare ciascuna immagine. In sostanza si prende l'insieme di immagini annotate e lo si divide in tre sottoinsiemi distinti, ciascuno con un ruolo preciso:

- Training set: è la parte di dati che utilizza la rete per imparare.
- Validation set: serve a monitorare come il modello si comporta con dati mai visti durante l'addestramento in una specifica epoca. Attraverso questi dati è possibile calcolare alcune metriche per capire la bontà dell'addestramento alla fine di ogni epoca.
- Test set: questa porzione di dati viene tenuta nascosta per tutto l'addestramento. Viene utilizzata alla fine per valutare la bontà dell'intero training.

Per la scelta della percentuale di suddivisione non vi è una regola fissa, ma si utilizzano spesso rapporti standard. Una suddivisione classica e suggerita anche da *Roboflow* è 70% training, 20% validation e 10% test. Se il dataset è piccolo conviene aumentare la percentuale di training in modo da dare alla rete maggiori informazioni da imparare. Nella attività sviluppata si è scelta la suddivisione 90% training, 6% validation e 4% test. Una volta conclusa la fase di suddivisione è possibile scaricare il dataset selezionando l'algoritmo di object detection d'interesse. Sono disponibili tutte le versioni di *YOLO* fino a *YOLOv12* si sceglie tuttavia *YOLOv8* poichè risulta essere la versione ottimizzata per l'utilizzo su piattaforme come Raspberry Pi. L'addestramento della rete non viene effettuato su *Roboflow* ma su computer attraverso uno script Python sfruttando la libreria di *Ultralytics*.

Da *Roboflow* si scarica una cartella contenente il file *data.yaml* utile per il training della rete. Il training è effettuato con le seguenti righe di codice:

```
from ultralytics import YOLO

model = YOLO('yolov8n.pt')
model.train(data='C:/Users/marra/Downloads/emaopt.v2i.yolov8/data.
    yaml', epochs=50, imgsz=640)
```

Grazie alla libreria *Ultralytics* è possibile scegliere il modello *YOLO* da utilizzare (nel caso in esame è stata scelta la versione 8 nano) il numero di epoche su cui fare il training, la dimensione delle immagini. Alla fine del training si ottengono due file in formato *Pytorch* che possono essere utilizzati per effettuare l'inferenza:

- *best.pt*: risulta essere il modello che si salva automaticamente quando la rete migliora le proprie prestazioni sul set di validazione in base ad una determinata metrica. Viene sovrascritto ogni qualvolta viene superato il valore massimo della metrica di riferimento e risulta essere dunque il modello più performante alla fine del training e quello utilizzato per fare inferenza in tempo reale.
- *last.pt*: contiene i pesi del modello alla fine dell'ultima epoca ovvero alla fine dell'intero addestramento. Non è detto che coincida con *best.pt*.

Oltre questi files, alla fine del training, è possibile ottenere delle visualizzazione diagnostiche per capire se l'addestramento è andato a buon fine e cosa ha visto ed imparato *YOLO*. Vengono proposte dei collage di immagini creati da *YOLO* durante il training. Alcuni collage presentano i bounding boxes attorno all'oggetto d'interesse con solo un numero, poichè la rete durante l'addestramento lavora sempre con ID numerici corrispondenti a determinate classi. Vi sono altri collage dove è possibile invece visualizzare la label corretta. Viene fornita anche la matrice di confusione, un altro strumento grafico utile per valutare la bontà dell'addestramento. Tale strumento presenta due assi. L'asse orizzontale rappresenta le ground truth(cioè gli oggetti che effettivamente sono presenti nell'immagine) mentre sull'asse verticale ci sono gli oggetti predetti dal modello (i quali possono essere o *sled* o il *background*). Ogni cella della matrice indica quante volte si è verificata quella combinazione di verità e predizione. Ad esempio, il quadrante in alto a sinistra è dedicato interamente a *sled*, conta i casi in cui vi era effettivamente *sled* nell'immagine e la rete l'ha riconosciuta. In alto a destra vi sono i casi di falso positivo, ovvero quando non vi era presente *sled* ed invece la rete ha predetto la sua presenza. La cella in basso a sinistra è invece dedicata ai casi in cui vi era *sled* ma la rete non l'ha riconosciuta, mentre l'ultima cella è dedicata all'assenza di *sled* dall'immagine e alla corretta predizione da parte della rete.

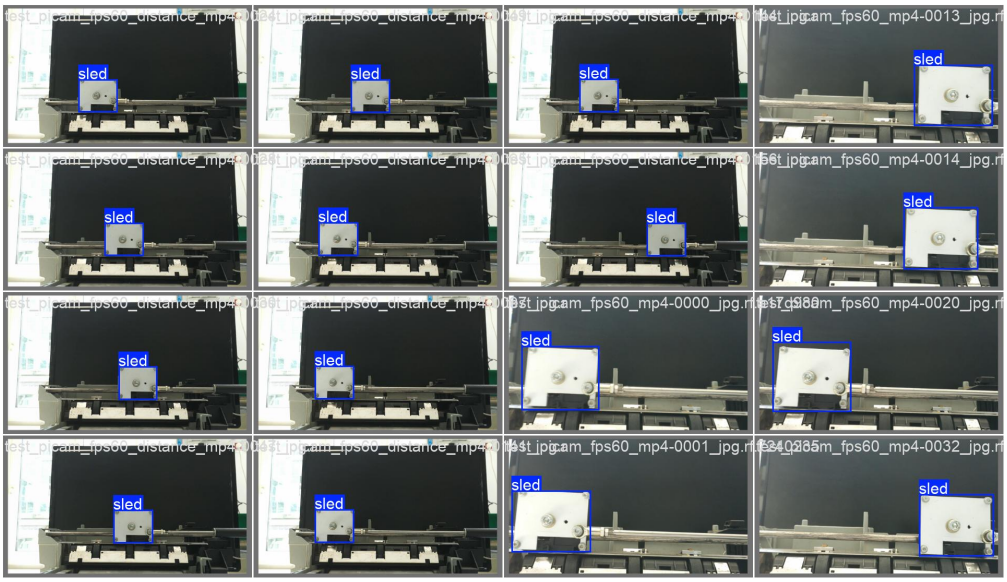

Figura 5.10: Collage batch riconoscimento oggetto

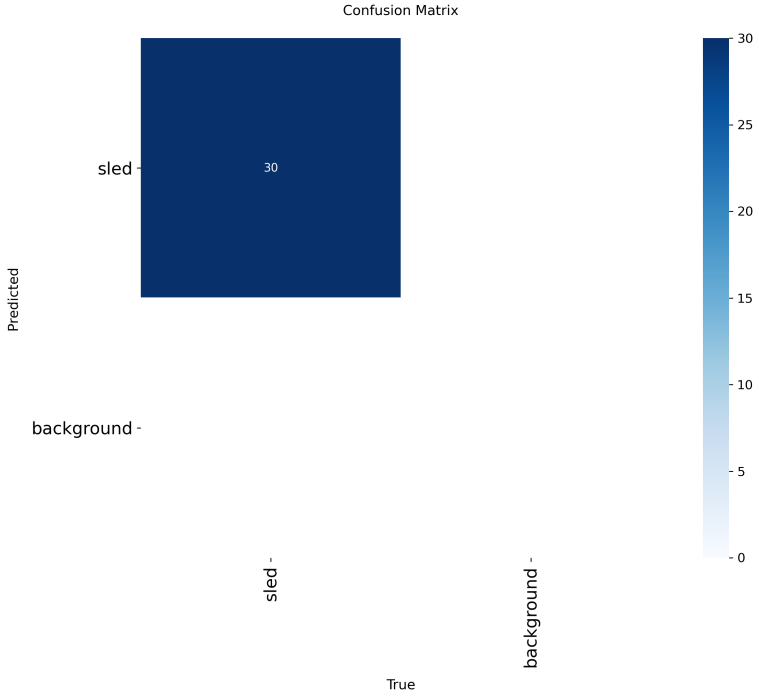

Figura 5.11: Matrice di confusione

Nella figura 5.12 è presente la *Precision Recall Curve*(PR) che evidenzia che il modello ha raggiunto un ottimo risultato tra precisione e richiamo per la classe d'interesse con una precisione media di 0.995 nell'identificazione della slitta.

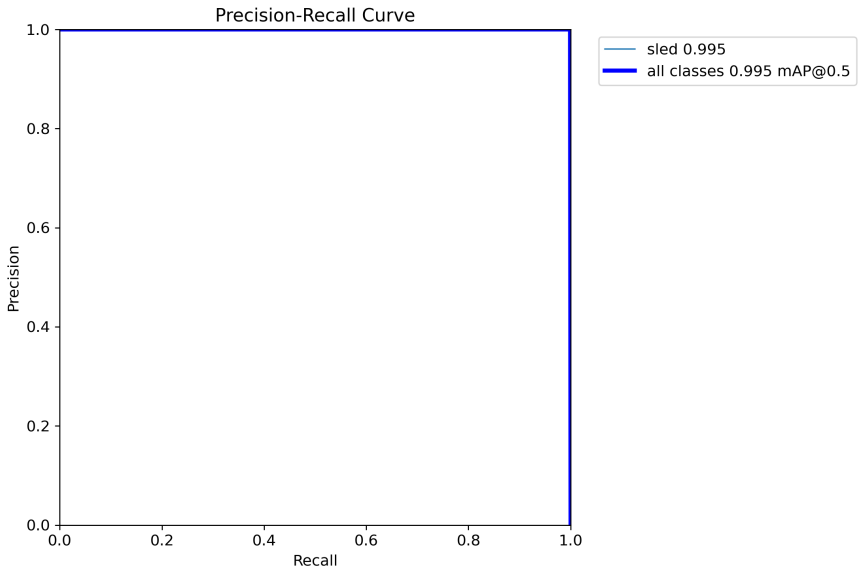

Figura 5.12: Precision-Recall Curve

Si riporta anche la *Precision-Confidence Curve* che presenta un valore di soglia di 0.986. La confidenza è la probabilità che un determinato bounding box assegnato dalla rete contenga davvero un oggetto della classe prevista, mentre la precisione è la percentuale di predizioni corrette effettuate dal modello. Un valore di soglia così elevato è dovuto al fatto che il training della rete è stato svolto con le acquisizione video del carrello sempre nel medesimo scenario dunque il modello è in grado di riconoscere l'oggetto in quell'ambiente specifico. Non è necessariamente un risultato negativo, poichè a differenza di altre applicazioni, in questo caso l'algoritmo di riconoscimento deve funzionare in quella determinata configurazione. La precisione mostra la capacità del modello di minimizzare i casi di falso positivo. La precisione resta ad un valore elevato per tutto il range di confidenza confermando la robustezza della rete nell'effettuare corretti riconoscimenti.

La *Recall-Confidence Curve* mostra la capacità delle rete di minimizzare i falsi negativi. Il recall misura quanti degli oggetti realmente presenti vengono effettivamente trovati dal modello. Come mostrato nella figura 5.14 la curva mantiene un valore di recall prossimo ad 1.00 per un ampio range di valori di confidenza. Sottolinea come il modello è in grado di riconoscere l'oggetto anche nel caso di condizioni non ottimali da un punto di vista ambientale e non genera falsi negativi.

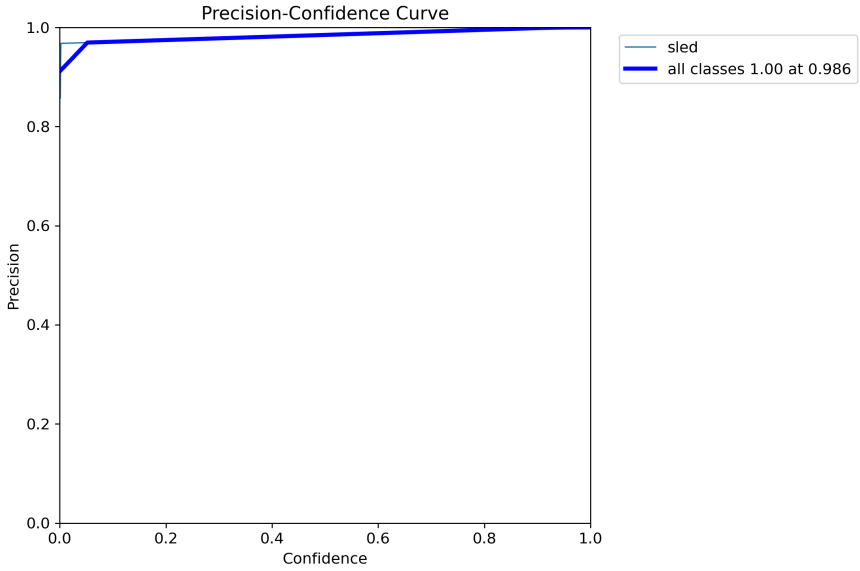

Figura 5.13: Precision-Confidence Curve

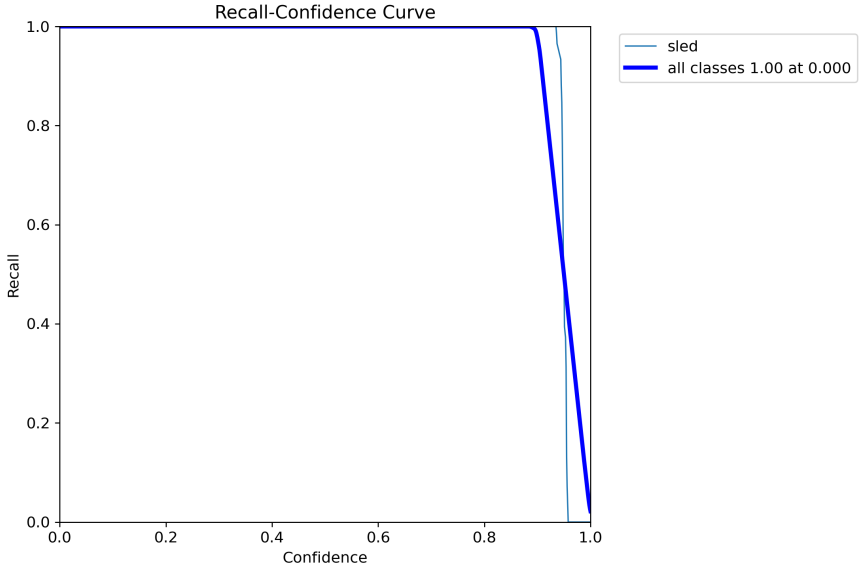

Figura 5.14: Recall-Confidence Curve

Si mostra la *F1 Curve* che mostra l'equilibrio che vi è tra precisione e richiamo nella rete addestrata. La figura 5.15 mostra come la rete offre alta precisione e affidabilità nell'identificazione della slitta per diversi valori di confidenza.

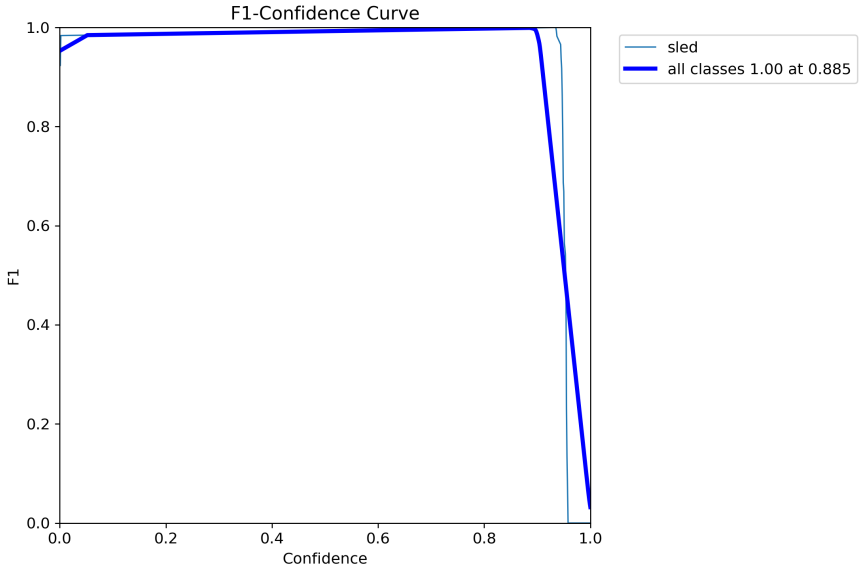

Figura 5.15: F1 Curve

Infine è possibile osservare delle metriche che stimano la robustezza della rete nel corso del training. In figura 5.16 si può osservare come la precisione media *mAP50-95* aumenta nel corso del training per stabilizzarsi ad un valore elevato. A causa del dataset che presenta sempre lo stesso background, la metrica *mAP50* risulta costante con un valore prossimo ad 1 poichè il modello impara molto in fretta a riconoscere gli oggetti in quel determinato scenario. Essendo *mAP50-95* una metrica molto più severa rispetto ad *mAP50* il miglioramento durante il training si evidenzia da quest'ultimo parametro.

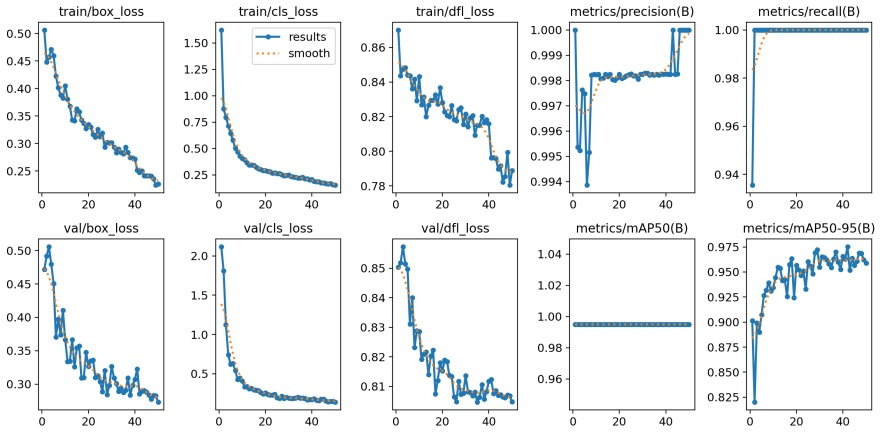

Figura 5.16: Risultati training rete

5.4 Algoritmo di object tracking

In questa sezione si riporta brevemente l'evoluzione degli algoritmi di object tracking fino a *YOLO*. Una maggiore importanza è data all'algoritmo utilizzato per sviluppare l'attività di tesi, descrivendone il funzionamento e riportando i risultati ottenuti.

5.4.1 Origine e storia degli algoritmi di object tracking

Le origini della visione artificiale risalgono all'inizio degli anni Sessanta, quando gli scienziati iniziarono a esplorare come il cervello elabora le informazioni visive. In esperimenti condotti sui gatti, i ricercatori David Hubel e Torsten Wiesel scoprirono che il cervello reagisce a schemi semplici come bordi e linee. Questo costituì la base dell'idea alla base dell'estrazione di caratteristiche: il concetto secondo cui i sistemi visivi rilevano e riconoscono prima le caratteristiche di base nelle immagini, come i bordi, per poi passare a schemi più complessi. Nello stesso periodo emerse una nuova tecnologia capace di trasformare le immagini fisiche in formati digitali, suscitando interesse su come le macchine potessero elaborare le informazioni visive. Nel 1966, il Summer Vision Project del Massachusetts Institute of Technology (MIT) spinse ulteriormente in questa direzione. Sebbene il progetto non abbia avuto pieno successo, mirava a creare un sistema in grado di separare lo sfondo dal primo piano nelle immagini. Per molti membri della comunità scientifica, questo progetto segna l'inizio ufficiale della computer vision come disciplina. Con il passare del tempo le tecniche di object tracking passarono ad approcci più avanzati. o dei metodi più popolari fu l'Haar Cascade, che divenne ampiamente utilizzato per compiti come il riconoscimento facciale. Questo metodo funzionava scansionando le immagini con una *sliding window*, verificando la presenza di caratteristiche specifiche come bordi o texture in ciascuna sezione dell'immagine, e poi combinando queste caratteristiche per rilevare oggetti come i volti. L'Haar Cascade risultò molto più veloce rispetto ai metodi precedenti[38].

Di recente, l'avvento del deep learning e delle reti neurali ha reso i processi più veloci ed efficaci. Le R-CNN (Region-based Convolutional Neural Network) invece di analizzare tutta l'immagine pixel per pixel genera un certo numero di probabili bounding boxes, classificandoli e quindi decidendo quale oggetto è all'interno di quella determinata regione. Dopo che ogni regione è stata classificata si applicano passaggi aggiuntivi come raffinare i bounding boxes o eliminare i duplicati. Il limite di questa strategia è che la pipeline è complessa e lenta: ogni componente deve essere allenato o regolato separatamente. Ogni bounding box deve essere passato al classificatore e analizzato, tutto questo in maniera sequenziale. Questo rende il sistema difficile da ottimizzare e poco adatto a scenari real-time.

5.4.2 YOLO: You Only Look Once

L'algoritmo *YOLO* (*You Only Look Once*) introdotto per la prima volta in [39] risulta essere di fondamentale importanza nella storia della visione artificiale poichè raggiunge prestazioni utili per applicazioni real time. A differenza dei metodi tradizionali descritti in precedenza, tale algoritmo riformula il problema dell'object detection come un unico problema di regressione che a partire dai pixel dell'immagine restituisce direttamente le coordinate dei bounding boxes e le probabilità di appartenere a determinate classi. Dal punto di vista architetturale, YOLO utilizza una singola rete convoluzionale in grado di predire simultaneamente più bounding boxes e le rispettive classi. Il modello viene addestrato su immagini intere e ottimizzato direttamente in funzione della prestazione di detection, senza necessità di pipeline complesse. Questa impostazione ha portato a diversi vantaggi:

- Velocità: *YOLO* nella sua versione base raggiunge una velocità di elaborazione di 45 fps mentre la versione *Fast YOLO* fino a 150 fps, consentendo il suo utilizzo per applicazioni real time.

- Visione globale: poichè il modello osserva l'intera immagine sia durante il training che nella fase di test è in grado di acquisire maggiori informazioni della scena, riducendo notevolmente gli errori di classificazione.

- Generalizzazione: *YOLO* apprende rappresentazioni generali degli oggetti ed è stato testato anche su domini differenti con ottimi risultati.

- Open Source: gli autori hanno reso disponibile il codice di addestramento e di testing oltre a diversi modelli pre-addestrati, favorendone l'adozione e lo sviluppo.

I metodi basati su R-CNN affrontano l'object detection seguendo una pipeline articolata: a partire dall'immagine vengono generati dei bounding boxes candidati che potrebbero contenere oggetti. Ciascuna di queste regioni viene quindi ritagliata, analizzata da una rete convoluzionale per estrarne le caratteristiche e infine passata a un classificatore che ne determina la classe, con un ulteriore passaggio di regressione per affinare le coordinate del box. In altre parole, l'analisi procede in maniera quasi sequenziale e richiede più moduli addestrati separatamente. YOLO, al contrario, propone un approccio radicalmente diverso: l'immagine viene suddivisa in una griglia e una sola rete convoluzionale che predice simultaneamente le coordinate dei bounding box, il grado di confidenza e le probabilità di classe. L'operazione diventa così un unico problema di regressione diretta, privo della complessità delle fasi intermedie. Questa differenza strutturale spiega perché YOLO riesce a garantire prestazioni in tempo reale, mentre le R-CNN risultano più lente pur mantenendo, in alcuni casi, una maggiore precisione nella localizzazione di oggetti molto piccoli.

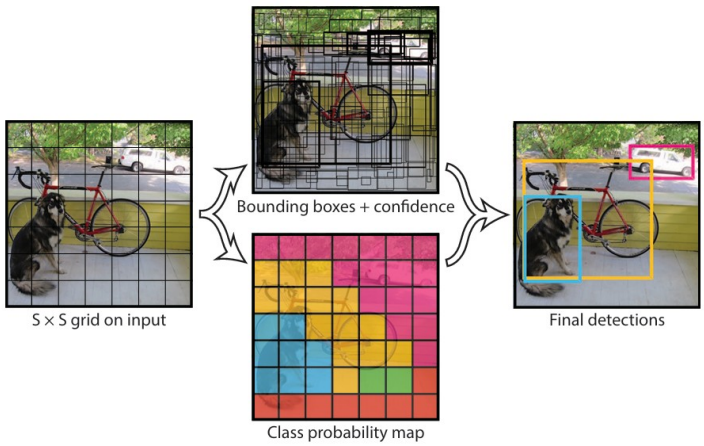

Figura 5.17: Fasi dell'algoritmo *YOLO*: formazione della griglia nell'immagine,creazione bounding boxes e confidenza,mappa di probabilità classi e detection finale[39]

Con l'evoluzione del modello *YOLO*, in particolare a partire da *YOLOv8* e nelle versioni successive come *YOLO11*, è stata introdotta un'architettura scalabile che si adatta a esigenze diverse in termini di velocità, accuratezza e requisiti computazionali. Ogni variante —Nano (n), Small (s), Medium (m), Large (l) e Extra Large (x)— rappresenta un trade-off specifico tra questi fattori. Le versioni nano e small hanno una maggiore velocità di inferenza, sono più leggere da un punto di vista computazionale ma hanno una precisione minore rispetto a modelli più pesanti come Large ed Extra Large. Quest'ultimi seppur molto più precisi hanno tempi d'inferenza troppo grandi non adatti alle prestazioni richieste per il sistema da sviluppare. Si è scelto dunque di optare per la versione *YOLOv8n*. Per l'applicazione in questione la versione nano, seppur più imprecisa, non risulta avere problemi poiché il riconoscimento degli oggetti risulta difficoltoso solo se quest'ultimi sono molto piccoli. In tutti gli altri contesti, questo modello risulta avere una precisione soddisfacente.

5.4.3 Esportazione in formato NCNN

L'implementazione di modelli di computer vision su dispositivi con potenza di calcolo limitata, come sistemi mobili o embedded, può essere complessa. È necessario assicurarsi di utilizzare un formato ottimizzato per le prestazioni richieste. Questo assicura che anche i dispositivi con potenza di elaborazione limitata possano gestire bene attività avanzate di computer vision. La funzionalità di esportazione nel formato NCNN consente di ottimizzare i modelli Ultralytics *YOLO* per applicazioni leggere basate su dispositivi. I modelli NCNN offrono una vasta gamma

di funzionalità chiave che consentono il machine learning on-device, aiutando gli sviluppatori a eseguire i propri modelli su dispositivi mobili, embedded ed edge. Tali modelli sono realizzati per essere efficienti e leggeri, ottimizzati per l'esecuzione su dispositivi mobili ed embedded come Raspberry Pi con risorse limitate. Possono anche ottenere prestazioni elevate con elevata precisione su varie attività basate sulla computer vision[40]. NCNN non è l'unico formato disponibile per effettuare inferenza con Ultralytics *YOLO*. Ci sono svariati formati con una determinata precisione e velocità di inferenza. I benchmark di YOLO11 sono stati eseguiti dal team Ultralytics su dieci diversi formati di modello misurando la velocità e l'accuratezza: *PyTorch, TorchScript, ONNX, OpenVINO, TF SavedModel, TF GraphDef, TF Lite, PaddlePaddle, MNN, NCNN*. I benchmark sono stati eseguiti su un Raspberry Pi 5 a precisione FP32 con dimensione dell'immagine di input predefinita di 640. Sono solo i benchmark per i modelli *YOLO11n* e *YOLO11s* perché le dimensioni degli altri modelli sono troppo grandi per essere eseguiti sui Raspberry Pi e non offrono prestazioni decenti[41].

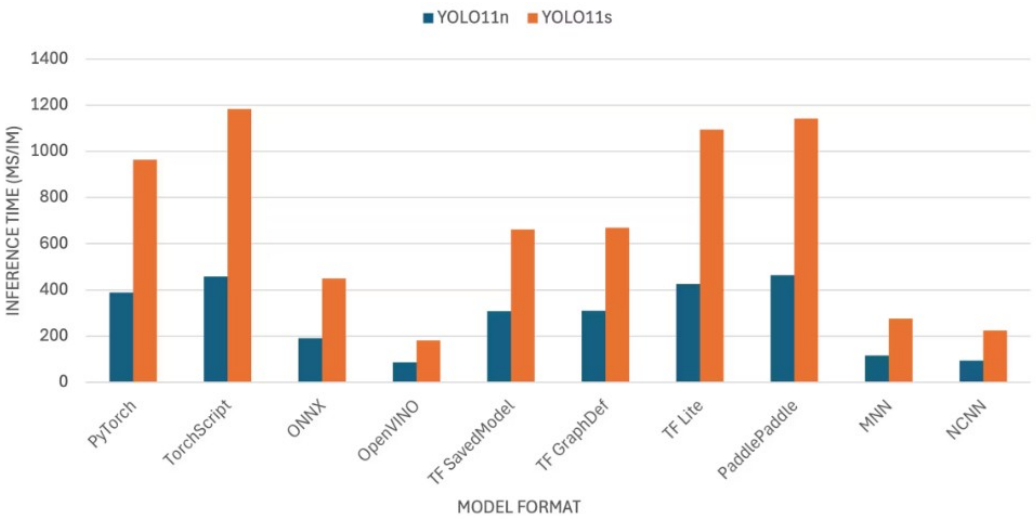

Figura 5.18: Tempo di inferenza per immagine in funzione del modello e del formato di esportazione per una rete *YOLO*[41]

Si può concludere osservando il grafico in figura 5.18 come i formati *NCNN* e *OpenVINO* siano quelli con il tempo di inferenza minore. Il formato *Pytorch* ha un tempo di inferenza troppo elevato non adatto all'applicazione da sviluppare. Si riporta una tabella riassuntiva che presenta i risultati del benchmark precedentemente descritto, fornendo ulteriori informazione come lo stato, le dimensioni su disco, la metrica mAP50-95(B), oltre al tempo di inferenza per ogni combinazione.

Formato	**Dimensione(MB)**	**mAP(B)**	**Inference time(ms/im)**
PyTorch	5.4	0.5101	387.63
TorchScript	10.5	0.5077	457.84
ONNX	10.2	0.5077	191.09
OpenVINO	10.4	0.5058	84.76
TF SavedModel	25.9	0.5077	306.94
TF GraphDef	10.3	0.5077	309.82
TF Lite	10.3	0.5077	425.77
PaddlePaddle	20.5	0.5077	463.93
MNN	10.1	0.5059	114.97
NCNN	10.2	0.5031	94.03

Tabella 5.4: Confronto tra diversi formati di esportazione del modello YOLO: dimensione, accuratezza (mAP50-95) e tempo di inferenza su *YOLO11n*[41]

Come si può osservare la precisione media tra i vari formati è pressochè la stessa, formati come *NCNN* ed *OpenVINO* hanno tempi d'inferenza molto più bassi. La dimensione su disco di un formato *NCNN* è quasi il doppio di un formato *Pytorch*. Alla luce di tutte queste osservazioni, è necessario convertire i files in formato *Pytorch*, ottenuti dopo il training della rete, in formato *NCNN*. Per fare ciò *Ultralytics* fornisce un codice nella sua documentazione[40]:

```
from ultralytics import YOLO

# Load the YOLO11 model
model = YOLO("yolo11n.pt")

# Export the model to NCNN format
model.export(format="ncnn")  # creates '/yolo11n_ncnn_model'

# Load the exported NCNN model
ncnn_model = YOLO("./yolo11n_ncnn_model")
```

5.4.4 Inferenza

In questa sezione si descrive l'inferenza svolta su Raspberry Pi 5 attraverso la videocamera Picamera Module 3. Dopo aver caricato tramite chiavetta USB i files in formato *Pytorch* ottenuti dal training della rete si effettua la conversione di essi nel formato NCNN. Alla fine di ogni conversione si genera una cartella contenente il file di interesse utile per effettuare l'inferenza sul carrello. Prima di commentare il codice sviluppato per l'object detection bisogna effettuare alcune precisazioni:

- È necessario installare da terminale alcuni pacchetti per effettuare l'inferenza: *OpenCV*, *Ultralytics* e *Numpy*. In particolare quest'ultimo pacchetto deve essere alla versione 1.23.5.
- Per evitare conflitti tra i diversi pacchetti installati è necessario creare un ambiente virtuale per i soli pacchetti utili all'algoritmo che si sta sviluppando. Per accedere all'ambiente virtuale si utilizza il comando da terminale `source yolo_object/bin/activate`.
- Il codice è scritto in Python sfruttando le libreria sopraelencate. Una volta entrati in ambiente virtuali, sempre da terminale, è necessario utilizzare il comando `python3 -m idlelib.idle` che permette di lanciare IDLE, l'editor ufficiale di Python. IDLE verrà eseguito usando l'interprete Python dell'ambiente virtuale, visualizzando i pacchetti installati in quell'ambiente.

Una volta lanciato l'ambiente virtuale e l'editor di Python è possibile utilizzare il codice per effettuare l'inferenza. Lo script è diviso in diverse sezioni: richiamo delle librerie utilizzate, impostazione della Picamera, caricamento del modello YOLO, cattura del frame ed inferenza su di esso, calcolo del tempo di inferenza e del frame rate.

Si richiamo le librerie *OpenCV*, *Ultralytics* e *libcamera*. In particolare, si utilizza *Picamera2* per il comando di sensori come il Picamera Module 3.
Si effettua il set up della videocamera. Si può scegliere la risoluzione, il colore, il frame rate e si avvia lo streaming video su Raspberry.
Si carica il modello YOLO d'interesse. Nella funzione `YOLO` va inserito il nome della cartella generata attraverso il file di conversione in formato NCNN.
In un apposito `while` vi è il campionamento del frame della videocamera e l'inferenza su di esso.
Infine, vi è una sezione dedicata alla visualizzazione del frame rate dell'inferenza. Il numero di FPS viene visualizzato in alto a destra nella finestra di streaming video una volta avviato il codice. In questa parte di codice si definisce la posizione, lo stile e le dimensioni del testo per trasmettere informazioni sul frame rate dell'inferenza.

```
import cv2
from picamera2 import Picamera2
from ultralytics import YOLO

# Set up the camera with Picam
picam2 = Picamera2()
picam2.preview_configuration.main.size = (1280, 1280)
picam2.preview_configuration.main.format = "RGB888"
picam2.preview_configuration.align()
picam2.configure("preview")
picam2.start()

# Load YOLOv8
model = YOLO("bestemaopt_ncnn_model")

while True:
    # Capture a frame from the camera
    frame = picam2.capture_array()

    # Run YOLO model on the captured frame and store the results
    results = model(frame)

    # Output the visual detection data, we will draw this on our camera preview window
    annotated_frame = results[0].plot()

    # Get inference time
    inference_time = results[0].speed['inference']
    fps = 1000 / inference_time  # Convert to milliseconds
    text = f'FPS: {fps:.1f}'

    # Define font and position
    font = cv2.FONT_HERSHEY_SIMPLEX
    text_size = cv2.getTextSize(text, font, 1, 2)[0]
    text_x = annotated_frame.shape[1] - text_size[0] - 10  # 10 pixels from the right
    text_y = text_size[1] + 10  # 10 pixels from the top
    # Draw the text on the annotated frame
    cv2.putText(annotated_frame, text, (text_x, text_y), font, 1, (255, 255, 255), 2, cv2.LINE_AA)
    # Display the resulting frame
    cv2.imshow("Camera", annotated_frame)

    # Exit the program if q is pressed
    if cv2.waitKey(1) == ord("q"):
        break
```

5.4.5 Risultati

In questa sezione si riportano e si discutono i risultati ottenuti durante l'inferenza con *Picamera Module 3* e Raspberry Pi 5. La figura 5.19 mostra i risultati ottenuti da parte della rete neurale *YOLOv8* precedentemente addestrata. Come si può osservare, la rete riesce abilmente a riconoscere l'oggetto in questione con un alto grado di confidenza(0.87). Questa evidenzia la robustezza del modello se utilizzato nelle condizioni operative per cui è stato addestrato.

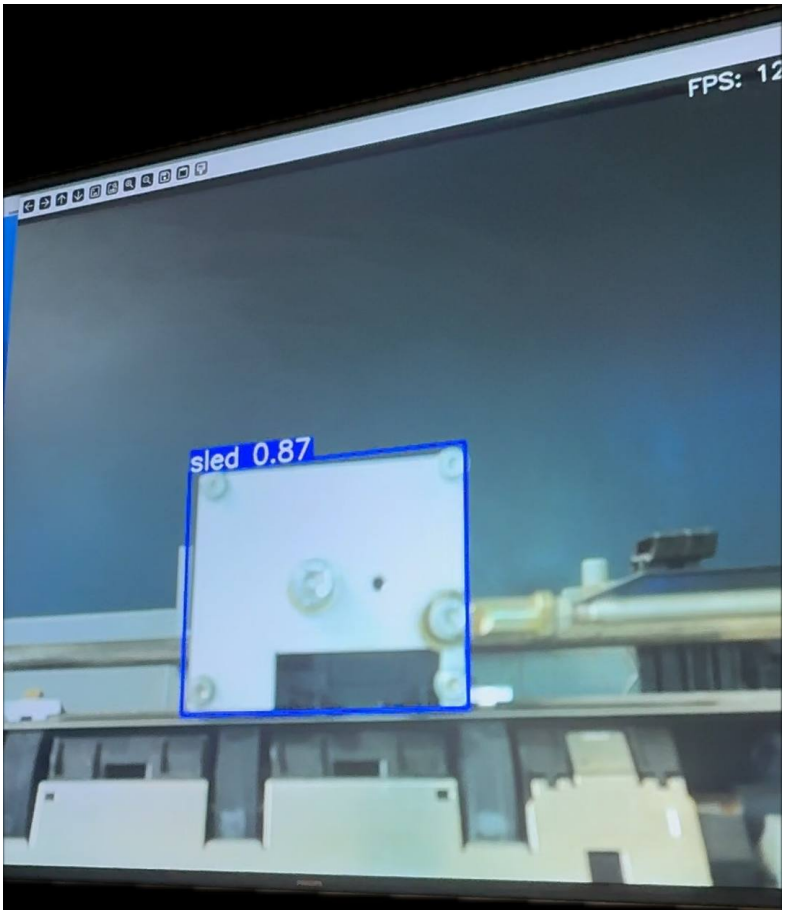

Figura 5.19: YOLOv8 risultati della detection su carrello

Vi sono tuttavia alcune limitazioni che è necessario evidenziare per un analisi completa dei risultati ottenuti. La rete ha un'elevata robustezza se la videocamera con cui si effettua l'inferenza è posizionata in condizioni simili o uguali rispetto a quelle relative alle acquisizioni video con cui si è costruito il dataset. Se la posizione del sensore viene cambiato, oppure il background intorno al banco prova risulta diverso, la rete perde confidenza e in diversi casi non riesce a riconoscere

l'oggetto. Ciò avviene soprattutto se la videocamera viene posizionata ad una maggiore distanza dal carrello, segnalando la necessità di approfondire e migliorare la costruzione del dataset per migliorare la robustezza della rete.
Un altro aspetto non secondario, riguarda sicuramente il frame rate stimato che dipende dal tempo di inferenza della rete. L'utilizzo del formato NCNN permette un tempo di inferenza dell'ordine della decina di millisecondi, molto più breve rispetto al formato *Pytorch*. La figura 5.20 mostra il box plot relativo ai tempi di inferenza ottenuti durante l'attività di verifica dell'algoritmo di object detection per i due formati d'esportazione. Come si può osservare, il 50% dei tempi d'inferenza quando si utilizza il formato NCNN sono compresi tra i 10 ed i 20 millisecondi. Il formato Pytorch fornisce un tempo d'inferenza decisamente maggiore inadatto ad applicazioni real time. Il tempo d'inferenza ottenuto utilizzando NCNN risulta inferiore anche al valore riportato nel benchmark di *Ultralytics* presente nella tabella 5.4. Tutto ciò è dovuto al fatto che il benchmark fa riferimento all'algoritmo *YOLOv11* mentre quello utilizzato per lo sviluppo del sistema risulta essere *YOLOv8* che attualmente è la versione più performante su un sistema come il Raspberry Pi. La presenza di outliers in entrambi i formati non viene giustificata da un cambiamento di scenario o dal movimento del carrello poichè tali dati sono stati acquisiti con l'oggetto fermo e riconosciuto dalla rete, ma possono essere ricondotti a problematiche di natura hardware.

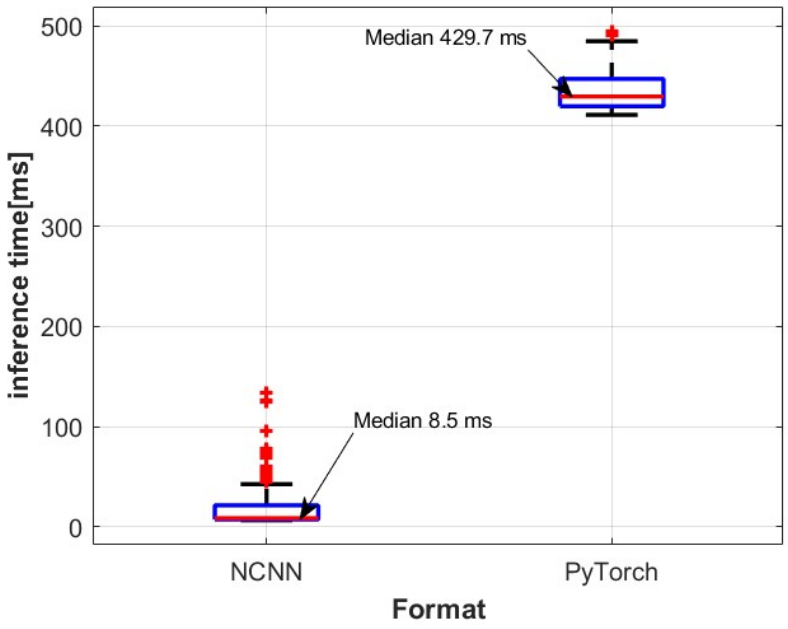

Figura 5.20: Box plot tempi d'inferenza formati NCNN e Pytorch

Nonostante i risultati incoraggianti i tempi d'inferenza oscillano molto su ogni immagine comportando un frame rate molto variabile che è indubbiamente sconsigliato per un'applicazione di questo tipo. Quest'ultimo aspetto, deve essere sicuramente approfondito e ottimizzato per migliorare l'efficacia della soluzione. I risultati ottenuti confermano la fattibilità dell'attività proposta e aprono alla possibilità di effettuare ulteriori step per sviluppare l'intero sistema introducendo la correlazione con la posizione del carrello.

Capitolo 6

Conclusioni

L'obiettivo principale dell'attività di tesi è stato lo sviluppo di un sistema di controllo posizione in anello chiuso con feedback ottico, a partire da un banco didattico già esistente valutando la sostituzione del trasduttore potenziometrico tradizionale con un sistema basato su videocamere.

Tra i principali contributi di questo lavoro vi è sicuramente la scelta della componentistica per lo sviluppo di questo sistema tra diverse soluzioni proposte, valutandone l'efficacia in termini di accuratezza e tempi di risposta del sistema. Lo sviluppo di uno studio geometrico ha permesso di valutare gli errori nel posizionamento delle videocamere e di ottenere il numero minimo di videocamere utili ad identificare la posizione dell'oggetto. Tutto ciò ha portato alla proposta di configurazioni ottimali per sviluppare il sistema sia da un punto di vista della componentistica coinvolta che del posizionamento e collegamento su campo.
Tramite modello, è stato possibile valutare la sostituzione del trasduttore potenziometrico con delle videocamere evidenziando come l'acquisizione video non influenza il sistema e non compromette l'efficacia del controllo. Modellizzando la videocamera come uno strumento di misura in grado di restituire informazioni sulla posizione dell'oggetto si ottiene una banda passante paragonabile a quella del sistema tradizionale con trasduttore potenziometrico. Tale risultato ha evidenziato come il collo di bottiglia del sistema è presente nella fase di processamento dell'immagine da parte della piattaforma hardware scelta.
Proprio sull'algoritmo di riconoscimento dell'oggetto si basa l'attività sperimentale sviluppata. Dopo aver configurato ed impostato il controllo dei sensori acquistati attraverso il training di una rete neurale YOLO è stato possibile sviluppare tale algoritmo, evidenziando la fattibilità dell'operazione.

Sebbene gli esiti siano promettenti, il lavoro presenta delle limitazioni. Il sistema proposto, in termine di componentistica, rappresenta un prototipo a scopo di ricerca

e divulgazione scientifica. Soluzioni più performanti ed efficaci hanno un costo e delle prestazioni differenti. Il modello su cui si basa lo studio geometrico descrive perfettamente il processo di mappatura da coordinate 3D a coordinate 2D che effettua una videocamera quando inquadra un oggetto, tuttavia non tiene conto di ulteriori aspetti come distorsioni geometriche o sfocature.
Grazie all'esportazione in formato NCNN il tempo d'inferenza nell'algoritmo di object tracking risulta adatto a performance real time tuttavia l'inferenza in sé è migliorabile: genera un frame rate variabile che necessita di essere stabilizzato.

Sviluppi futuri potrebbero concentrarsi su migliorare il modello su cui si basa lo studio geometrico proposto, considerando ulteriori effetti che influenzano il processo di mappatura. Si può verificare, applicando la stessa metodologia, l'accuratezza nella misura della velocità, la quale dipende sia dagli FPS che dalla risoluzione.
Il frame rate della videocamera durante l'inferenza necessita di essere stabilizzato, si può lavorare su questa ottimizzazione. La costruzione del dataset merita maggiore attenzione: ci si può documentare maggiormente su questo aspetto per cercare di migliorare la robustezza della rete neurale.
Una tematica importante da approfondire è sicuramente la correlazione con la posizione: una volta riconosciuto l'oggetto è necessario ottenere le sue coordinate durante il processamento dell'immagine per chiudere l'anello. Si può sviluppare uno studio e inserire dei sistemi di elaborazione immagini e data fusion per fondere segnali di più telecamere e informazioni del trasduttore potenziometrico già presente sul banco.

Sebbene tale attività viene sviluppata su un banco prova già esistente, data la versatilità e la flessibilità della visione artificiale i risultati e le metodologie di tale lavoro non restano limitati al tema dei servosistemi, ma possono essere sicuramente estese ad applicazioni inerenti alla robotica e all'automazione industriale. Il lavoro svolto non si limita dunque ad un prototipo accademico, infatti le metodologie utilizzate risultano essere rigorose e possono essere trasferite in diversi ambiti.

Bibliografia

[1] Xingyu Yang, Zhengxue Zhou, Jonas H Sørensen, Christoffer B Christensen, Mikail Ünalan e Xuping Zhang. «Automation of SME production with a Cobot system powered by learning-based vision». In: *Robotics and Computer-Integrated Manufacturing* 83 (2023), p. 102564 (cit. a p. 1).

[2] Luis Pérez, Íñigo Rodríguez, Nuria Rodríguez, Rubén Usamentiaga e Daniel F García. «Robot guidance using machine vision techniques in industrial environments: A comparative review». In: *Sensors* 16.3 (2016), p. 335 (cit. a p. 1).

[3] Piotr Kohut e Kamil Skop. «Vision systems for a UR5 cobot on a quality control robotic station». In: *Applied Sciences* 14.20 (2024), p. 9469 (cit. a p. 1).

[4] Simon Mangold, Christian Steiner, Marco Friedmann e Jürgen Fleischer. «Vision-based screw head detection for automated disassembly for remanufacturing». In: *Procedia CIRP* 105 (2022), pp. 1–6 (cit. a p. 1).

[5] Loukas Prezas, George Michalos, Zoi Arkouli, Antonis Katsikarelis e Sotiris Makris. «AI-enhanced vision system for dispensing process monitoring and quality control in manufacturing of large parts». In: *Procedia CIRP* 107 (2022), pp. 1275–1280 (cit. a p. 1).

[6] Keivan Rahsepas e Ali Sadighi. «Use of Machine Vision for Feedback Computation in Motion Control Systems». In: () (cit. alle pp. 1, 8).

[7] Yapeng Ma, Baoqi Feng, Kaixiang Li e Lei Zhang. «Motion Control of Gallium-Based Liquid Metal Droplets in Abrasive Suspensions Within a Flow Channel». In: *Actuators*. Vol. 14. 9. MDPI. 2025, p. 456 (cit. a p. 1).

[8] Tausif Diwan, G Anirudh e Jitendra V Tembhurne. «Object detection using YOLO: challenges, architectural successors, datasets and applications». In: *multimedia Tools and Applications* 82.6 (2023), pp. 9243–9275 (cit. a p. 1).

[9] Yanyan Dai e Kidong Lee. «Multi-Sensor Fusion for Autonomous Mobile Robot Docking: Integrating LiDAR, YOLO-Based AprilTag Detection, and Depth-Aided Localization». In: *Electronics* 14.14 (2025), p. 2769 (cit. a p. 2).

[10] Hieu D Nguyen, Brandon McHenry, Thanh Nguyen, Harper Zappone, Anthony Thompson, Chau Tran, Anthony Segrest e Luke Tonon. «Accurate Crop Yield Estimation of Blueberries using Deep Learning and Smart Drones». In: *arXiv preprint arXiv:2501.02344* (2025) (cit. a p. 2).

[11] Wei Luo et al. «An efficient visual servo tracker for herd monitoring by UAV». In: *Scientific Reports* 14.1 (2024), p. 10463 (cit. a p. 2).

[12] Giovanni Bracco, Massimo Sorli et al. *Laboratori di meccatronica*. Mc Graw Hill, 2022 (cit. alle pp. 2–4, 73).

[13] Massimo Sorli. *Controllo posizione elettrico*. Materiale del corso di Meccatronica, Politecnico di Torino. 2025 (cit. a p. 2).

[14] Tom Nowak, Alexander Große-Kreul, Marius Boshoff e Bernd Kuhlenkötter. «Enhancing Mobile Robot Position Estimation with Machine Learning Methods Using Camera-Based Tracking». In: *Procedia CIRP* 130 (2024), pp. 964–968 (cit. a p. 8).

[15] Arduino. *Arduino GIGA R1 WiFi*. 2023. URL: https://store.arduino.cc/collections/giga/products/giga-r1-wifi (cit. a p. 9).

[16] Raspberry. *Raspberry Pi 5*. 2024. URL: https://www.raspberrypi.com/products/raspberry-pi-5/ (cit. a p. 10).

[17] NVIDIA. *NVIDIA Jetson AGX Orin Developer Kit*. URL: https://developer.nvidia.com/embedded/learn/jetson-agx-orin-devkit-user-guide/developer_kit_layout.html#developer-kit (cit. a p. 12).

[18] Renasas. *Renasas EK-RA6M4*. URL: https://www.renesas.com/en/products/microcontrollers-microprocessors/ra-cortex-m-mcus/ek-ra6m4-evaluation-kit-ra6m4-mcu-group?srsltid=AfmBOorUhwYky7FxamiahHsNEWh0yehwbr3wEHJWn-BTowXXtYBAmddc (cit. a p. 13).

[19] Arducam. *Autofocus Synchronized Quad Camera Kit*. URL: https://www.arducam.com/64mp-quad-camera-kit.html (cit. a p. 15).

[20] Arducam. *MEGA Camera Module*. URL: https://blog.arducam.com/camera-for-any-microcontroller/ (cit. alle pp. 16, 17).

[21] Arducam. *IMX477 Rolling Shutter 12MP*. URL: https://www.arducam.com/arducam-12mp-imx477-motorized-focus-high-quality-camera-for-raspberry-pi.html (cit. a p. 18).

[22] Arducam. *OG02B10 Global Shutter 2MP*. URL: https://www.arducam.com/arducam-2mp-global-shutter-og02b10-color-camera-modules-pivariety-b0348.html (cit. a p. 19).

[23] Raspberry. *Pi Camera Module 3*. URL: https://www.raspberrypi.com/products/camera-module-3/ (cit. alle pp. 19, 20, 96).

[24] Arducam. *Multi-Camera Kit for NVIDIA Jetson AGX Orin*. URL: https://www.arducam.com/arducam-12mp-multi-camera-kit-for-nvidia-jetson-agx-orin.html (cit. a p. 21).

[25] Raspberry. *Raspberry Computer Module 5 Kit*. URL: https://www.mouser.it/new/raspberry-pi/raspberry-pi-cm5-dev-kit/?srsltid=AfmBOoq9UZOPCNtWwRcWJys7kAUqI4E5tqIzRvmGxdw2yk0bHr40p1xo (cit. a p. 25).

[26] Richard Hartley e Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003 (cit. a p. 26).

[27] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022 (cit. a p. 30).

[28] Itai Katz, Hamid Aghajan e John Haymaker. «A Process for Sensor Configuration in Multi-Camera Networks». In: () (cit. alle pp. 31, 33, 34).

[29] John N Sanders-Reed. «Error propagation in two-sensor three-dimensional position estimation». In: *Optical Engineering* 40.4 (2001), pp. 627–636 (cit. alle pp. 34, 35).

[30] C.K. Cowan e P.D. Kovesi. «Automatic sensor placement from vision task requirements». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10.3 (1988), pp. 407–416. DOI: 10.1109/34.3905 (cit. a p. 35).

[31] Xing Chen e James Davis. «An occlusion metric for selecting robust camera configurations». In: *Machine Vision and Applications* 19 (2008), pp. 217–222 (cit. a p. 35).

[32] Pooya Rahimian e Joseph Kearney. «Optimal camera placement for motion capture systems in the presence of dynamic occlusion». In: nov. 2015, pp. 129–138. DOI: 10.1145/2821592.2821596 (cit. a p. 36).

[33] Andrea De Martin. *Servosistema elettrico in controllo posizione*. Materiale del corso di Meccatronica, Politecnico di Torino. 2025 (cit. alle pp. 73, 74, 76).

[34] Arducam. *Quick Start Guide for 12MP IMX477 Motorized Focus Camera*. URL: https://docs.arducam.com/Raspberry-Pi-Camera/Motorized-Focus-Camera/Quick-Start-Guide/IMX477-Motorized-Focus-Camera/ (cit. alle pp. 92, 94).

[35] Arducam. *Quick Start Guide for Pivariety Camera*. URL: https://docs.arducam.com/Raspberry-Pi-Camera/Pivariety-Camera/Quick-Start-Guide/ (cit. alle pp. 94, 95).

[36] Raspberry. *Camera software, Raspberry Pi Documentation*. URL: https://www.raspberrypi.com/documentation/computers/camera_software.html (cit. alle pp. 96, 99).

[37] Roboflow. *Roboflow*. URL: `https://www.https://roboflow.com` (cit. a p. 103).

[38] Ultralytics. *The evolution of object detection and Ultralytics' YOLO models*. URL: `https://www.ultralytics.com/blog/the-evolution-of-object-detection-and-ultralytics-yolo-models` (cit. a p. 112).

[39] Joseph Redmon, Santosh Divvala, Ross Girshick e Ali Farhadi. «You Only Look Once: Unified, Real-Time Object Detection». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Giu. 2016 (cit. alle pp. 113, 114).

[40] Ultralytics. *Come esportare in NCNN da YOLO11 per un'implementazione fluida*. URL: `https://docs.ultralytics.com/it/integrations/ncnn/` (cit. alle pp. 115, 116).

[41] Ultralytics. *Guida rapida: Raspberry Pi con Ultralytics YOLO11*. URL: `https://docs.ultralytics.com/it/guides/raspberry-pi/` (cit. alle pp. 115, 116).