POLITECNICO DI TORINO

Corso di Laurea Magistrale in

INGEGNERIA MECCANICA

Tesi di Laurea Magistrale

Sistema di visione stereoscopico per il packaging delle mele

Relatore
prof. Giuliana Mattiazzo

Candidato
Michele Radicioni

Anno Accademico 2019/2020

Ringraziamenti

Mi sento in dovere di dedicare questa pagina del presente elaborato alle persone che mi hanno supportato nella redazione dello stesso.

Un sentito grazie al mio relatore Mattiazzo per la sua infinita disponibilità e tempestività ad ogni mia richiesta.

Innanzitutto, ringrazio il mio Tutor Vissio, sempre pronto a darmi le giuste indicazioni in ogni fase della realizzazione dell'elaborato. Grazie a lui ho accresciuto le mie conoscenze e le mie competenze.

Ringrazio tutto lo staff dell'azienda EPF, in cui ho svolto il lavoro complementare alla redazione della tesi, per l'ospitalità e per le skills acquisite sul campo.

Ringrazio mia madre, perché senza di lei non avrei mai potuto intraprendere questo percorso di studi.

1. Introduzione 9
1.1 Produzione globale 10
1.2 Maturazione e raccolta 13
1.3 Tecniche classiche per la valutazione della qualità 14
1.4 Tecniche innovative per il controllo della qualità 17
1.5 Conservazione 19
2. Packaging line 27
2.1 Immissione 29
2.2 Trattamento 30
2.3 Calibrazione 30
2.5 Tracciabilità 33
2.6 Packaging (confezionamento) 35
2.7 Pallettizzazione 41
3. Sistema POM 43
3.2 Impianto 44
3.3 Problemi e possibili sviluppi 53
3.4 Telecamere stereoscopiche 55
4. Sistemi di visione industriale 61
4.1 Mercato 61
4.2 Struttura di un sistema di visione 62
4.3 Applicazioni in ambito industriale 73

4.4 La visione stereoscopica artificiale 85
4.5 Configurazione delle telecamere realsense 105
5. Sviluppo software 114
5.1 Python 116
5.2 Pycharm 119
5.3 Prima fase di sviluppo del programma..... 120
5.4 Prima fase di test..... 122
5.5 PyQt5 125
5.6 Qt Creator 127
5.7 Seconda fase di sviluppo del programma..... 128
5.8 Seconda fase di test..... 132
5.9 Descrizione del programma..... 137
6. Collaudo del software..... 178
6.1 Verifica del calcolo delle coordinate..... 179
6.2 Analisi della disposizione delle telecamere 192
7. Banco prova 210
7.1 Fotocellula 212
7.2 Nastro trasportatore 220
7.3 PLC 229
7.4 Alimentatore 231
7.5 Cattura di oggetti in movimento..... 234
Conclusioni..... 246

Bibliografia 252

CAPITOLO 1

1. Introduzione

Prima di andare a vedere nel dettaglio il prototipo del sistema di visione addetto al packaging delle mele, è corretto spiegarne la necessità. Per questo, di seguito si discuterà dell'importanza commerciale della mela in ambito globale e nazionale e si spenderà qualche parola sull'intero processo di raccolta e conservazione, per poi soffermarci sul packaging nel capitolo successivo.

La mela è presente sulla tavola dei consumatori in tutto il globo. Ciò è dovuto alla bontà e alle proprietà nutritive abbinate a un costo molto accessibile. Infatti, la mela è un frutto di facile ambientamento, conservabile per diversi mesi in ambiente condizionato e quindi disponibile per tutto l'anno. Le condizioni positive o negative investono in maniera trasversale tutti i paesi produttori, per cui, il trend positivo o negativo del settore è "internazionale", riflettendo un quadro globalizzato. La produzione mondiale è ancora in crescita, soprattutto grazie all'introduzione di migliori tecniche colturali nei Paesi in via di sviluppo. Va anche considerato che nei paesi come l'Italia il settore è ben organizzato. La cooperazione tra le regioni produttrici e la regolamentazione comunitaria ha favorito lo sviluppo di pochi consorzi ben strutturati, orientati alla qualità intesa come "bontà" e come "sicurezza" del prodotto. Una politica forte che spinga sull' organizzazione e la qualità sarà determinante per assicurare competitività e futuro ai produttori di mele. Inoltre, diversi indicatori indicano che il mondo della mela rimarrà un settore in grado di competere sul mercato globale.

1.1 Produzione globale

La produzione mondiale per il 2019/2020 è prevista aumentare fino a quasi 5 miloni di tonnellate raggiungendo il totale di 75,7 milioni di tonnellate, in quanto l'incremento in Cina riuscirà a superare abbondantemente il calo dell'Unione Europea. L'aumento delle forniture cinesi dovrebbe anche portare una crescita del mercato globale.

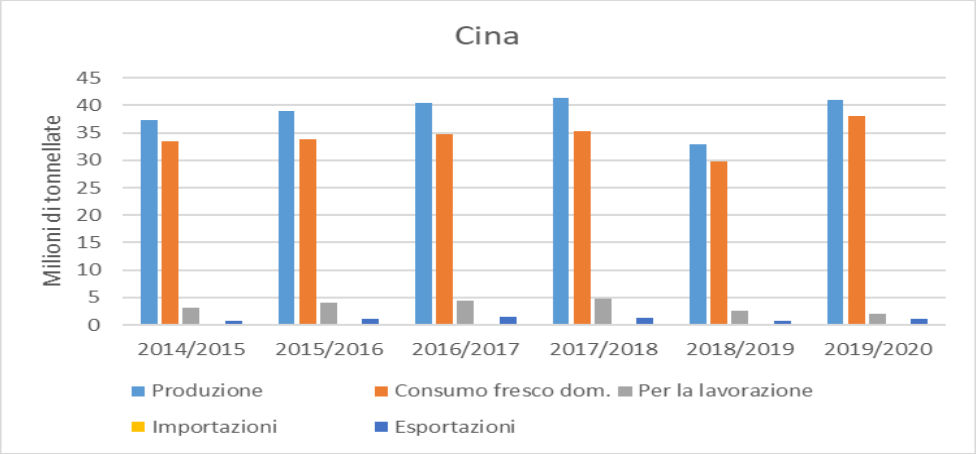


Figura 1.1: Mercato della mela in Cina [1]

1.1.1 Cina

La produzione cinese è prevista salire di 8 milioni di tonnellate, raggiungendo un totale di 41 milioni di tonnellate, riguadagnando tutte le risorse perse lo scorso anno a causa delle cattive condizioni metereologiche. Quasi tutte le maggiori province in crescita hanno usufruito di condizioni di crescita favorevoli durante lo sviluppo del frutto, e si prevede che le forniture di frutta faranno aumentare le esportazioni di 230 mila tonnellate facendole arrivare oltre a 1 milione. Anche se le forniture incrementeranno, la qualità dovrebbe rimanere stabilmente "media", portando ad un aumento dell'importazione per le mele di più alta qualità fino a un record di 100 mila tonnellate.

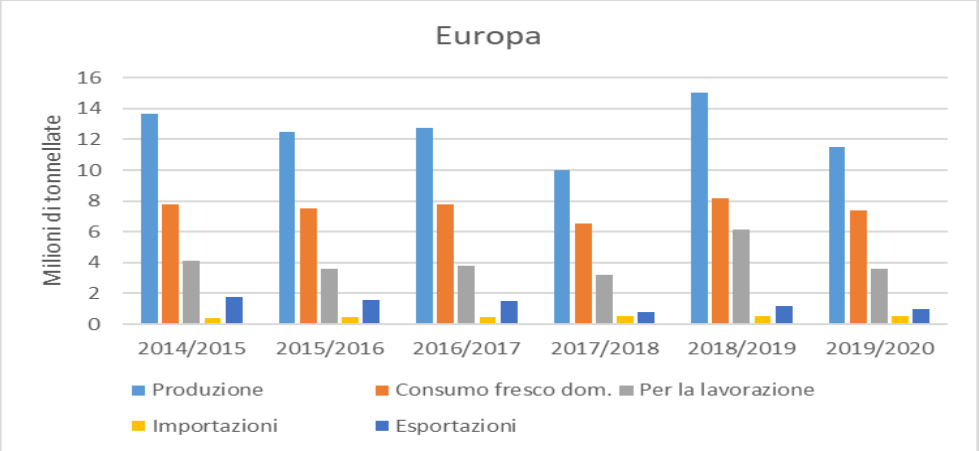


Figura 1.2: Mercato della mela in Europa [1]

1.1.2 Unione Europea

Si prevede che la produzione Europea diminuirà di quasi il 25% arrivando a 11,5 milioni di tonnellate perché la maggior parte degli stati membri, tra cui la Polonia, sono stati colpiti da una serie di gelate, siccità, incendi e grandinate. Ciò ha segnato, per la seconda volta in 3 anni, una riduzione della produzione causata da eventi climatici maggiore del 20%. A causa dell'output ridotto, si prevede un calo delle esportazioni di 200 mila tonnellate arrivando a 975 mila tonnellate il secondo livello peggiore dal 2007/08. Si prevede che le importazioni rimarranno praticamente immutate a 500 mila tonnellate in quanto una parte minore dell'output sarà indirizzato alla lavorazione e la maggior parte, invece, saranno deviate al consumo "fresco".

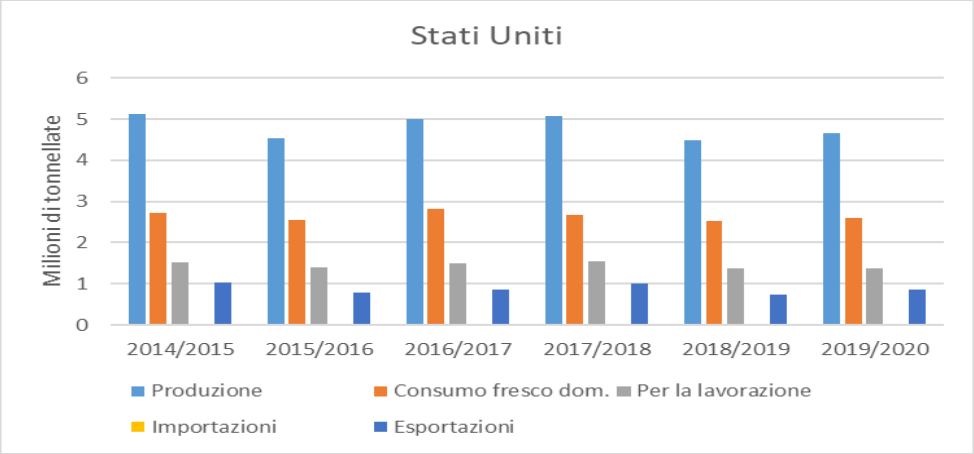


Figura 1.3: Mercato della mela negli Stati Uniti [1]

1.1.2 Stati Uniti

Si prevede che la produzione statunitense crescerà di 179 mila tonnellate raggiungendo 4,7 milioni di tonnellate grazie al favorevole clima estivo. L'aumento delle forniture e la rimozione della tassa di ritorsione del 20% del Messico nel Maggio del 2019 dovrebbero far salire le esportazioni di oltre 100 mila tonnellate, raggiungendo le 860 mila tonnellate. Si prevede che le importazioni rimarranno quasi immutate a 145 mila tonnellate in quanto l'incremento dalla Nuova Zelanda compenserà il calo dal Cile.

1.1.3 Italia

La produzione italiana è stimata a 2 milioni 200 mila tonnellate, più alta dell'anno scorso, ma leggermente inferiore (-6%) rispetto alla media del 2014-2016. I trend sono diversi a seconda delle regioni: sia il Trentino che l'Alto Adige stanno recuperando dall'anno scorso ma rimangono sotto il loro massimo potenziale, mentre i nuovi frutteti di mele che sono stati piantati in Piemonte hanno portato un considerevole aumento della produzione. Le altre regioni sono rimaste ai livelli del 2016.

Tabella 1.1: Produzione della mela in Italia negli ultimi anni [2]

Italia	Cons. 2013	Cons 2014	Cons 2015	Cons 2016	Cons 2017	Prev 2018	% Prev18 / Cons17	% Prev/ Cons 14/ 16
Ton.								
Alto Adige	1.096.184	1.199.224	1.127.227	1.063.676	910.766	999.706	10	- 12
Trentino	460.537	559.608	535.899	535.140	205.026	502.816	145	- 7
Veneto	187.300	224.844	203.279	218.177	176.247	217.647	23	1
Friuli V.G.	49.471	50.400	40.537	40.606	43.660	43.504	- 0	- 1
Lombardia	27.322	37.526	31.632	32.466	26.310	30.181	15	- 11
Piemonte	145.930	175.665	158.048	177.701	141.770	193.462	36	13
Emilia Romagna	149.803	168.948	155.006	169.260	165.504	172.210	4	5
Altri	35.000	40.000	35.000	35.000	35.000	40.000	14	9
TOTALE	2.151.547	2.456.215	2.286.628	2.272.027	1.704.283	2.199.526	29	- 6

Le grandinate hanno influenzato solo aree piccole e hanno causato danni abbastanza lievi. Il tempo degli ultimi mesi farà sì che la quantità di mele destinate alla lavorazione sarà minore rispetto alle stagioni passate, ritornando a livelli normali, ovvero l'11-12% del totale.

Non ci sarà più prodotto disponibile della stagione presente all'inizio della prossima stagione commerciale, il che è positivo. Inoltre, ci sono più volumi di varietà moderne apprezzate dal mercato. L'industria di lavorazione potrebbe rappresentare uno scarico interessante per la frutta di bassa qualità. Infine, considerando le pressioni interne del bacino Europeo e il cambio di tasso non favorevole per gli operatori "terzi", le importazioni dai paesi dell'emisfero sud non dovrebbero aumentare.

In questo contesto le aspettative per la stagione 2018/2019 rimangono positive, soprattutto quando si tratta di mele di alta qualità. Il fatto che il settore delle mele italiano sia ben organizzato è importante per la competitività, per l'innovazione varietale e per i processi di esportazione.

1.2 Maturazione e raccolta

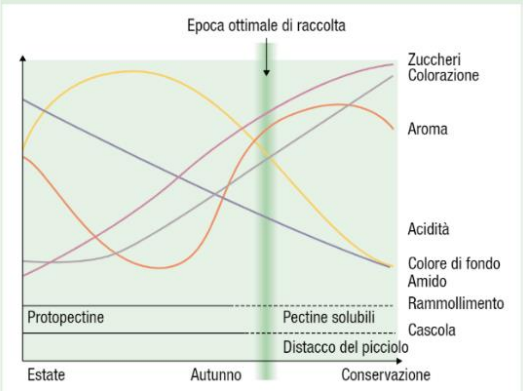


Figura 1.4: Parametri qualitativi pre e post raccolta [3]

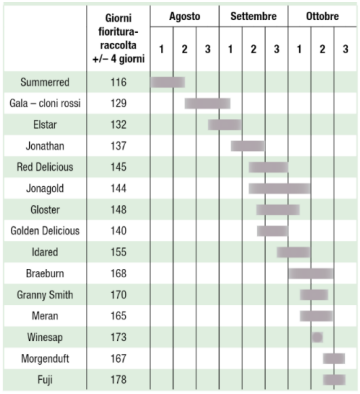


Figura 1.5: Giorni di fioritura e raccolta [4]

La raccolta deve avvenire in determinati intervalli di tempo e modi per garantire l'ottenimento di determinati caratteristiche qualitative intrinseche ed estetiche.

Ogni varietà di mela deve essere raccolta più velocemente possibile e all'interno della propria "finestra annuale di raccolta".

1.3 Tecniche classiche per la valutazione della qualità

Il tempo della raccolta deve corrispondere al conseguimento di determinati requisiti minimi commerciali.

Alcuni di essi sono codificati (norme di qualità CEE, marchi di qualità, DOP-IGP), e servono a valutare le mele da diversi punti di vista: organolettico, estetico, sanitario, ecc. Queste valutazioni possono avvenire impiegando diverse di metodiche di indagine, tra le quali:

- identificazione del colore e del sovraccolore:
- misurazione del calibro, del peso, del volume;

- riconoscimento della forma delle mele e dei difetti presenti sull'epidermide;
- indagini di laboratorio per il controllo del rispetto di requisiti fisici (peso-durezza-colore) e chimici (zuccheri-acidità-degradazione amido);
- calcolo di indici qualitativi seguendo formule matematiche precise (Indice di Thiault, Indice di Perlim, ecc.).

Di seguito si possono vedere le formule che consentono di ricavare alcuni degli indici di qualità più importanti:

$$STREIF = \frac{PO}{IR * AM} \quad \textbf{Equazione 1.1}$$

$$THIAULT = IR + AC * 10 \quad \textbf{Equazione 1.2}$$

$$PERLIM = (0.5 * PO) + (0.67 * IR) + (0.67 * AC) - 10 \quad \textbf{Eq. 1.3}$$

Con:

- PO = penetrometro = [kg/cm^2]
- IR = indice rifrattometrico = [g/l]
- AM = stadio amido = 1-5 moltiplicato per 2
- AC = acidità = [g/l]

Da diversi anni sono presenti sul mercato dei sistemi automatizzati, che possono effettuare questi esami su campioni di decine di mele in pochi minuti. Se necessario, per ogni campione, possono rilevare:

- "peso del singolo frutto, peso medio e totale del campione;
- RSR% in gradi Brix su singolo frutto e medio sul campione;
- durezza della polpa in kg/cm^2, rilevata su un punto per frutto e media sul campione;

- acidità totale sul campione espressa in mEq/L di NaOH e il g/l di acido malico;
- succosità media del campione espressa in % di succo estraibile;
- indice TOP del campione o indice di qualità di Thiault."

Figura 1.6: Rifrattometro per la misurazione del grado zuccherino [5]

Figura 1.7: Penetrometro a puntale per la misurazione della durezza [6]

Figura 1.8: Colorimetro da banco [3]

Figura 1.9: Penetrometro da banco per la misurazione della durezza [7]

Figura 1.10: Titolazione dell'acidità sul succo [6]

Figura 1.11: Analisi automatica in laboratorio [3]

1.4 Tecniche innovative per il controllo della qualità

Tra le tecniche ancora in fase di sviluppo e di tanto in tanto testati anche durante la fase applicativa si trovano:

- N.M.R. = Risonanza Magnetica Nucleare;
- X-Ray Imaging = immagine ai raggi X;
- misura della fluorescenza della clorofilla;
- analisi spettrale della risonanza acustica;
- VIS-NIR;
- naso elettronico;
- PTR-MS.

Queste nuove tecniche sono state escogitate per poter essere applicate in linea su sistemi di selezione automatica, ovvero calibratrici e selezionatrici automatiche, in modo da selezionare la frutta tenendo in considerazione, non solo le caratteristiche estetiche, ma anche le caratteristiche intrinseche-organolettiche.

1.4.1 VIS-NIR

Figura 1.12: Strumentazione del VIS-NIR [3]

Tabella 1.2: Composizione chimica di alcune varietà di mele [8]

Principali determinazioni		Golden Delicious	Renetta Canada	Red Delicious
Umidità	% (g/100 g)	83-86	81-85	85-87
Ceneri		0,21-0,30	0,21	0,23-0,26
Saccarosio		0,55-3,7	0,9-2,9	0,4-0,4
Glucosio		1,8-3,2	1,3-2,4	0,7-3,3
Fruttosio		6,7-9,7	6,0-7,8	6,1-8,4
Acido malico		0,36-0,63	0,47-0,75	0,24-0,38
Cellulosa		0,5-0,8	0,85-1,45	0,6-1,0
Pectina		0,2-0,6	0,6-0,9	0,4-0,7
Vitamina C	mg/100 g di sostanza	1-5	0,6-1,1	0,5
Fosforo		7,0-11,0	9,3-14,7	7,0-11,5
Potassio		73-130	86-140	90-120
Calcio		3,3-5,2	5,0-7,6	4,6-6,9
Magnesio		4,0-5,5	4,2-6,3	3,9-5,4
Ferro		0,06-0,17	0,08-0,17	0,05-0,13

Quando una fonte di luce monocromatica ad alta intensità colpisce una mela, essa interagisce con le molecole chimiche che la compongono. Questo fenomeno avviene soprattutto per la luce con lunghezze d'onda superiori a 700 nm, ovvero quella che si trova nella zona della banda luminosa non visibile del vicino infrarosso (da cui VIS-NIR).

I componenti chimici della mela influenzano l'assorbimento e la riflessione dell'energia luminosa, quindi, analizzando la luce riflessa si possono ricavare informazioni precise sulla natura e la concentrazione dei vari componenti.

Questi sono collegati a molti attributi qualitativi fisici e chimici della mela, tra cui: la durezza, il contenuto di zucchero e l'acidità. L'applicazione del NIR necessita che la strumentazione venga precedentemente "informata" con delle curve di calibrazione in modo da abbinare le lunghezze d'onda riflesse dai frutti, con dati "certi" estratti con tecniche distruttive.

1.4.2 PTR-MS

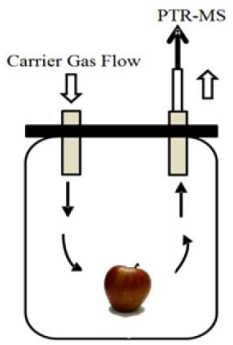


Figura 1.13: Disegno del sistema PTR-MS [9]

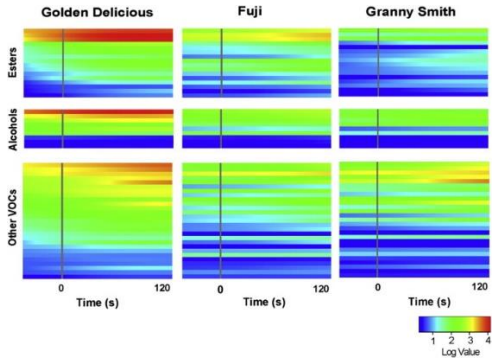


Figura 1.14: VOC fingerprints elaborate dal PTR-MS [10]

Questo sistema è in grado di identificare i composti organici volatili (VOC's), emanati dalle mele. Da quest'analisi si ricavano delle masse atomiche a cui sono abbinati i propri composti aromatici: alcoli-esteri-aldeidi-chetoni. Questa tecnica di solito viene utilizzata per studiare l'andamento nel tempo della presenza dei composti volatili dentro le celle di conservazione, sulle atmosfere controllate o modificate.

1.5 Conservazione

Dopo essere stata raccolta, la mela che non viene consumata immediatamente deve essere destinata alla conservazione, che può essere di breve (1 mese), media (3-5 mesi) o lunga durata (10-12 mesi). In questa fase, di solito, si impiega il freddo come metodo primario che poi può essere eventualmente accompagnato ad altri mezzi per imporre il rallentamento della respirazione delle mele.

1.5.1 Refrigerazione

Costituisce la prima fase per assicurarsi il rallentamento della maturazione della mela. Si effettua ponendo le mele raccolte in appositi

contenitori (casse in legno o plastica), all'interno di celle frigorifere addette allo scopo.

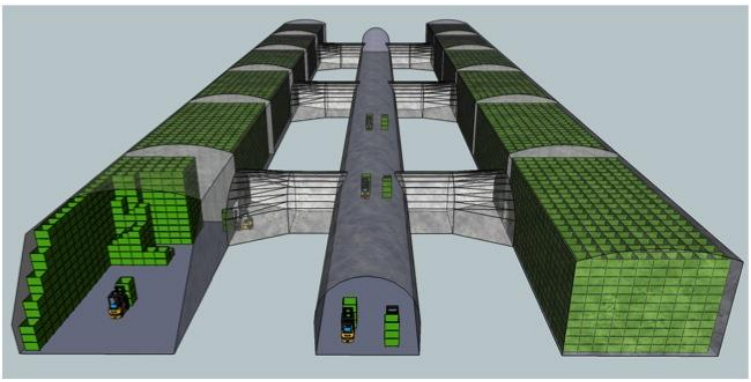


Figura 1.15: Impianto per la conservazione delle mele [11]

L'impianto deve essere in grado di:

- raffreddare le mele, all'intensità corretta, dalla temperatura di campo-raccolta a quella di conservazione;
- mantenere l'U.R.% (umidità relativa) richiesta;
- mantenere la frutta nella migliore condizione possibile, cercando di ridurre al minimo il calo di peso totale.

1.5.2 Regolazione dell'umidità nelle celle

La mela respira e traspira acqua contemporaneamente, in aggiunta, qualsiasi impianto frigorifero, durante lo scambio termico, crea della condensa sotto forma di brina. La diminuzione della massa della mela che si verifica durante la conservazione è influenzata da:

- varietà, dimensione dei frutti, rugginosità della buccia, grado di maturazione, tipo di imballaggio, durata del periodo di conservazione;

- dotazioni tecnologiche della cella: tipo di impianto frigorifero, superficie evaporante in rapporto con la quantità di mele conservate, il coefficiente di riempimento della cella, la ventilazione e la movimentazione dell'aria, la presenza di impianto di umidificazione.

1.5.3 Atmosfere controllate

Qualora la conservazione delle mele avvenisse solamente mediante raffreddamento, si tratta di conservazione in AN, ovvero atmosfera normale; viceversa, quando si effettua anche un'alterazione della composizione dell'aria, essa è detta conservazione in AC, ovvero atmosfera controllata, oppure AM, ovvero atmosfera modificata. Solamente impiegando queste tecnologie la conservazione della mela può raggiungere traguardi temporali massimi fino a 12, mantenendo elevati livelli di qualità.

Impianti AC

La respirazione della frutta all'interno della cella provoca la diminuzione dell'O_2 e l'aumento della CO_2, creando, in seguito ad un lungo lasso di tempo, un equilibrio dei gas paragonabile a quello che si può trovare in una AC. Al contrario, se i valori requisiti devono essere raggiunti in tempi brevi, risulta necessario fornirsi di impianti, congegnati in base a: tipo di formule gassose desiderate, varietali, di gestione, temporali ecc.

I componenti principali che si possono trovare un impianto in grado di realizzare le AC sono:

- macchinari per abbassare le concentrazioni di O_2;
- macchinari per assimilare la CO_2;
- macchine per tenere basse concentrazioni di C_2H_4 (etilene);
- stazione per il controllo delle pressioni gassose (O_2, CO_2, N_2);
- reti di comunicazione celle e macchinari a tenuta stagna;

- quadri elettrici per la gestione diretta delle macchine;
- sistemi hardware e software per gestire ogni parte dell'impianto.

Tabella 1.3: Formule di conservazione delle principali varietà di mele [3]

Varietà	T°	U.R.%	CO_2%	O_2%	Note
Golden Delicious	0,8-1,2	95	2,5-3,0	1,2-1,8	T° maggiori per mele più mature e grosse
Red Delicious	0,8-1,2	95	1,8-2,5	1,2-1,8	Idem c.s.
Renetta Canada	3,0-3,5	+/-85	2,0-2,5 (3,0)*	2,5-3,0	Minimo ventilazione 8h/24h * $O_2 = CO_2$: fiorone
Morgenduft R. Beauty	0,6-1,0	90-95	2,0-2,8	1,2-1,8	
Granny Smith	1,0-1,2	90-95	1,2-2,0	1,2-2,0	$O_2 = CO_2$
Gloster	1,0-1,2	90-93	1,3-2,0	1,2-2,0	
Royal Gala	0,8-1,3	>90	1,5-2,0	1,5-2,0	$O_2 = CO_2$ Max 6-7 mesi
Elstar	1,0-1,5	>90	1,5-2,0	1,2-1,5	6 mesi
Winesap	0,9-1,2	>90	1,2-1,5	1,3-1,6	
Fuji	1,0-1,5	>90	1,5-2,0 (1,3)*	2,0-2,5 (1,5)*	7 mesi – No LO se vitrescente, si LO, se le mele sono buone
Braeburn	1,0-1,5	90-93	</ = 1,2	1,5-2,0	Regimazione ritardata di 20 gg Cons. max 6-7 mesi
Pink Lady	2,0-2,5	90-93	1,0-1,3	1,5-2,0	7 mesi

Gestione delle celle

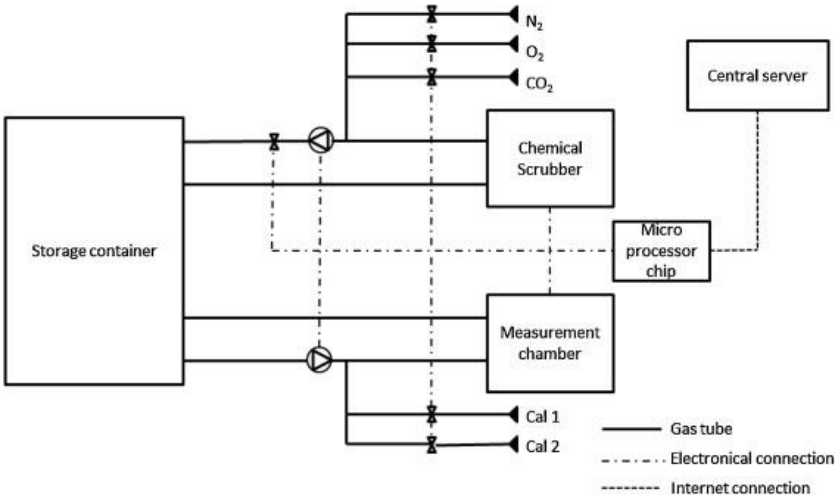


Figura 1.16: Schema di controllo dell'impianto delle celle AC [12]

L‘evoluzione dell’automazione e della computerizzazione dei sistemi e degli impianti, ha portato alla possibilità di realizzare formule e condizioni di conservazione che prima si sarebbero considerate impossibili. Per questo è stata fondamentale l’aggiunta di analizzatori di gas, a elevata sensibilità, di facile utilizzo e taratura.

Le tipologie più comunemente utilizzate sono:

- analizzatore di ossigeno termo-paramagnetico, che approfittando delle proprietà paramagnetiche dell’O_2, tramite un ponte Wheatstone, converte il segnale paramagnetico a T° costante in un segnale elettrico enunciato in % V/V;

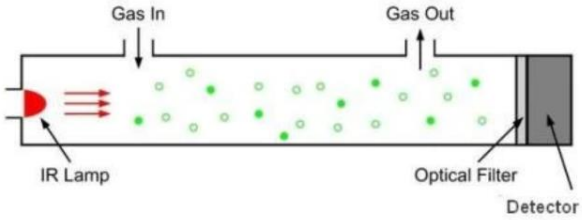


Figura 1.17: Analizzatore di CO_2 a raggi infrarossi [13]

- analizzatore di anidride carbonica a raggi infrarossi, che fonda il suo funzionamento sulla discrepanza di assorbimento da parte dei raggi IR, in base alla concentrazione di CO_2 all’interno di un campione d’aria.

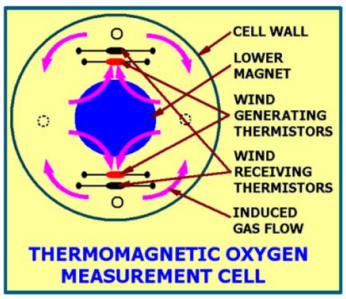


Figura 1.18: Analizzatore di ossigeno termo-magnetico [14]

Tecniche avanzate per le Atmosfere Controllate
Le tecniche d'avanguardia nel campo dell'AC sono:

- Atmosfera Controllata Dinamica (DCA);
- Stress Gassosi Iniziali o Ripetuti a bassi tenori di ossigeno (ILOS e LOS).

Ambedue sono state studiate con lo scopo di gestire il riscaldo comune senza l'utilizzo di composti chimici, permettendo: la conservazione della durezza della polpa, la riduzione della degradazione dell'acidità, la frenata della senescenza ecc.

DCA (Dynamic Controlled Atmosphere)

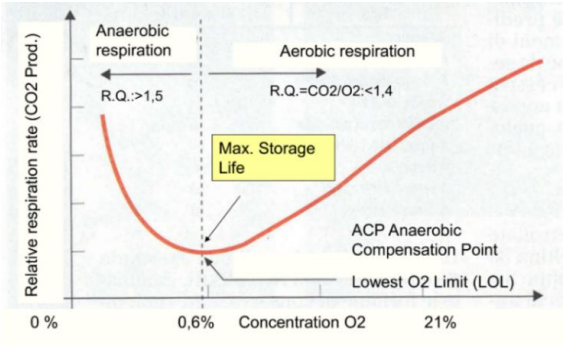


Figura 1.19: Funzionamento della DCA [15]

Il punto debole del sistema, che ne ha fatto ritardare l'applicazione pratica, risiede nella questione che, se si conservano le mele a livelli di ossigeno troppo bassi, si provocano dei fenomeni fermentativi.

L'utilizzo di sensori a fluorescenza e di trasduttori di segnale su sistema integrato a monitor (FIRM Sensor = Fluorescenza Interactive Response Monitor), permette il monitoraggio in tempo reale delle fasi di stress delle mele e conseguentemente permette la modifica delle concentrazioni dei gas in base alle fasi di stress rilevate dai sensori.

La gestione di questi stress permette la conservazione delle mele fino a 6-8 mesi dopo l'avvenimento della raccolta, e permette di gestire il riscaldo comune.

LOS (Low Oxygen Stress)

Questa tecnologia consiste nell'anticipare un processo con stress di ossigeno che bisogna operare, nella fase iniziale (Initial LOS) della conservazione. L'entità dello stress, il decorso e la reiterazione dello stesso, sono determinati dall'accumulo di alcol etilico all'interno della polpa delle mele, che non deve mai valicare i limiti definiti in base alla varietà. Quando si raggiunge il livello massimo, le concentrazioni dei gas, soprattutto quella dell'O_2, devono essere incrementate, per far ripartire l respirazione aerobica delle mele, in modo da metabolizzare l'alcol fino a valori "naturali".

CAPITOLO 2

2. Packaging line

La mela è un frutto che richiede di essere maneggiato delicatamente attraverso tutto il processo di smistamento e confezionamento. Inoltre, per i produttori, è fondamentale offrire ai clienti un frutto sano e di alta qualità.

Ci sono due sistemi principali di packaging:

- commit to pack (o direct pack);
- presizing.

Indipendentemente dal sistema di packaging le operazioni fondamentali rimangono più o meno le stesse.

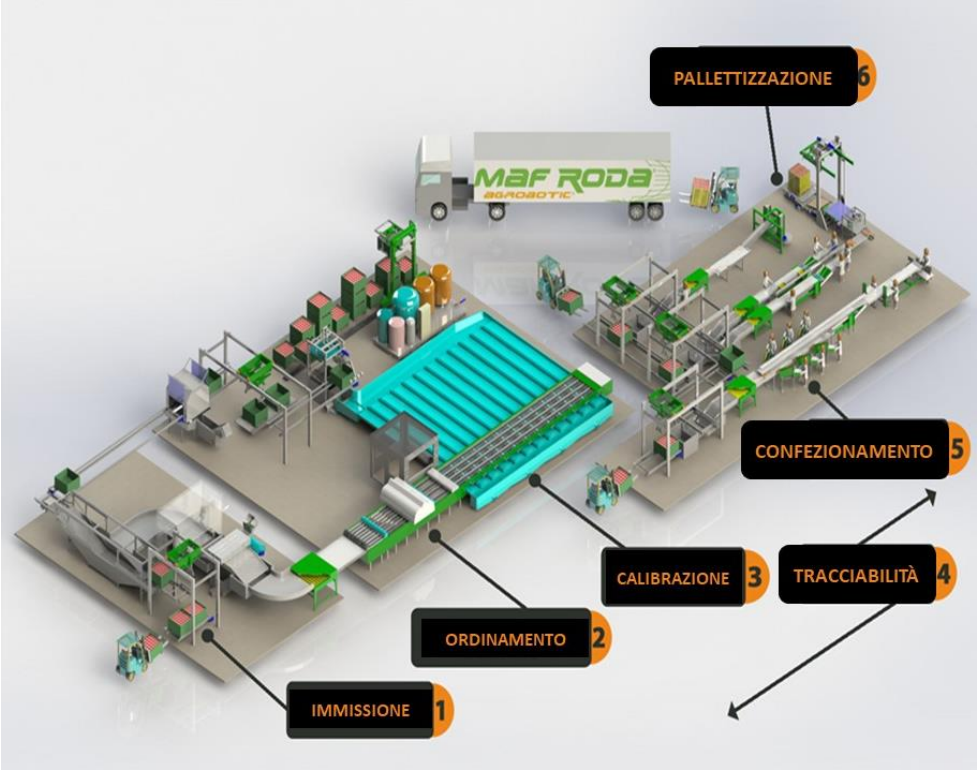


Figura 2.1: Linea di packaging [16]

Per il sistema Commit to Pack, il magazzino si accorda con l'agricoltore in modo da impacchettare la frutta nello stesso momento in cui i primi bidoni vengono svuotati. Uno svantaggio di questo sistema è che se le mele impacchettate non sono vendute in tempo, dovranno essere rimesse in magazzino e dovranno essere spacchettate e vendute per la lavorazione.

Per il sistema Presizing, una volta che la frutta è stato smistata a seconda delle sue dimensioni, viene fatta ritornare nei bidoni che vengono fatti tornare nel magazzino e ci rimangono fino a che non c'è il bisogno di ritirarli fuori. I bidoni di mele "presized" di solito sono composti da lotti di diversi agricoltori messi insieme. Questo può richiedere un'attenzione maggiore ai registri che devono mantenere una sicura

tracciabilità del cibo di ogni agricoltore dentro lo stesso raggruppamento.

2.1 Immissione

Prima di tutto, le mele sono portate fuori dalle atmosfere normali o dalle atmosfere controllate quando è il momento del packaging o del "presize".

Se l'impianto utilizza un sistema di movimentazione idraulico, i bidoni di mele vengono immersi in una vasca d'acqua dove la frutta galleggia fino a raggiungere un nastro trasportatore. Se l'impianto, invece, usa un caricamento "a secco", i bidoni vengono caricati direttamente sul nastro trasportatore.

Figura 2.2: Nastri trasportatori per il trasporto delle casse [17]

Figura 2.3: Svuotamento delle casse per immersione [18]

Le mele vengono poi trasportate fino alle linee di smistamento, dove il personale specializzato rimuove e scarta la frutta danneggiata, difettosa o di dimensioni troppo basse. Tutte le mele che subiscono punture di insetto, degradamento e severe scottature devono essere scartate. Una piccola parte della frutta sarà indirizzata allo smaltimento o alla lavorazione. Le mele poi procedono alla pulitura (immerse in acqua contenente detergenti e disinfettanti commestibili), seguita dal risciacquo ed infine l'asciugatura mediante aria.

Figura 2.4: Spazzolatrice-lavatrice [19]

Figura 2.5: Spazzolatrice a secco con aspiratori [19]

2.2 Trattamento

Le linee di packaging variano nel tipo e nella quantità di composti chimici utilizzati durante il processo. Certe varietà di mele sono spruzzate con cera commestibile in modo da prevenire la perdita di umidità, rallentare la respirazione delle mele in magazzino e per aumentare la brillantezza. Le mele biologiche non vengono spruzzate con la cera e vengono trattate mediante estratti vegetali.

Figura 2.6: Applicatrice di cera biologica [16]

Figura 2.7: Tunnel per l'asciugatura [16]

2.3 Calibrazione

La frutta viene analizzata e misurata a mano o usando tecnologia ottica automatizzata. Esistono standard nazionali e statali per la calibrazione delle mele che consistono nell'apparenza, colore, dimensione, peso e

nella qualità interna. Alcune varietà possono avere più sfumature di colore e dimensioni diverse.

I sistemi di calibrazione elettronica possono selezionare i prodotti in base a peso, diametro e colore. Inoltre, è possibile effettuare una selezione più specifica selezionando una combinazione degli stessi parametri.

Queste tecnologie sono accessibili e di facile utilizzo. Si servono di computer user-friendly dotate di uno schermo touch-screen e un'interfaccia grafica semplice e intuitiva.

Tra le funzionalità più importanti ci sono:

- la tracciabilità dei prodotti e la possibilità di salvare tutti i dati relativi alle calibrazioni eseguite;
- la possibilità di modificare i parametri di lavorazione in tempo reale;
- la possibilità di attivare programmi di lavoro individuali per ogni linea;
- la possibilità di controllare il funzionamento delle macchine a distanza tramite smartphone e tablet.

Nella pagina seguente si analizzerà una linea di selezionamento per capirne meglio lo scopo ed il funzionamento.

SINGULATION SYSTEM
It's the key element of our machines. Fruits enter automatically I the trolleys determining a filling until the 90-95%, synonymous of ample capacity of the line.

PRE-ALIGNING SYSTEM
The pre-aligning system is composed of V-shaped transport channels belt with different speed to best align the fruits before they enter the grading machine.

WASHING, BRUSHING AND DRYING SYSTEM
Fruits arrive automatically in the washing, brushing and drying area.

EMPTY BOXES FEEDING SYSTEM
Empty boxes feeding manual system for all the packaging stations.

FEEDING AND FIRST SELECTION SYSTEM
The product arrives automatically from the land and it's overthrown in the working line where there's a first pre-sorting to eliminate the waste.

DISCHARGE AND PACKAGING SYSTEM
According to the parameters set on the computer, fruits are divided by different sizes and discharged very softly from the trolleys into the packaging tanks.

GENERAL CONTROL COMPUTER
Through the general computer we can control the entire line in various selection parameters: weight, diameter and color.

ELECTRONIC GRADER
Our graders can process multi-fruits. They're compact, tough, modular and gentle with the products machines. We can decide the line number according to the production capacity and also the exits number.

Figura 2.8: schema tipico di una linea di selezionamento [20]

2.5 Tracciabilità

Esistono dei sistemi informatici, che gestiscono ogni grader e forniscono tutti i risultati del grading, i registri e la tracciabilità dei bidoni del prodotto pre-graded.

Figura 2.9: Funzionalità del woopack software [16]

Nei sensi del packaging, si può utilizzare un software suite (ad es. WOOPACK) che può utilizzare i dati provenienti dai sistemi informatici per assicurare la continuità del flusso di dati con il rispetto del flusso fisico del prodotto in modo da coprire la tracciabilità obbligatoria richiesta.

Ogni macchina è controllata da un PLC. Un'interfaccia speciale tra PLC e il software permette lo scambio di informazioni in tempo reale per conoscere il flusso fisico del prodotto, la posizione delle unità di trasporto e lo status dei processi. Questo scambio di informazioni è bidirezionale e permette al software di controllare le macchine per la maggior parte dei processi integrati. Questa relazione speciale tra il PLC e il sistema IT tramite il software rende possibile l'input e il controllo automatico dei dati senza alcun intervento manuale.

Il software, se accoppiato con macchine compatibili può facilitare:

- recupero in tempo reale di tutta l'informazione necessaria ad assicurare la tracciabilità in ogni fase del confezionamento attraverso il tracciamento del flusso;
- informazioni sulla fornitura prelevata per i prodotti introdotti nelle linee tramite lettura del codice a barre;
- quantificazione e identificazione delle discrepanze nell'ordinamento tramite sistemi di pesatura dinamici;
- etichettatura in tempo reale del prodotto nelle unità di confezionamento;
- stabilire la composizione delle unità di packaging nelle unità di trasporto (pallet);
- l'identificazione di pallet tramite il loro SSCC;
- il provvedimento di un sistema di gestione della produzione assistita al computer con tutte le informazioni della tracciabilità e della produzione tramite un database.

Se è accoppiato anche con il sistema di gestione della produzione (PO), il software permette:

- la programmazione degli ordini per ogni linea;
- il controllo della consistenza degli oggetti del prodotto introdotto nelle linee attraverso la lettura del codice a barre;
- informazioni in tempo reale riguardo ciascun ordine e lo stato del progresso degli operatori, attraverso grandi schermi o touch screens a qualsiasi punto strategico della linea;
- gestione in tempo reale del magazzino e dei consumabili;
- etichettatura completamente automatica basata sul PO e i dati del lotto;
- analisi in tempo reale della produttività per tipo di lavoro (PO) e per operatore, se è aggiunto il time-stamping;
- la programmazione automatica delle macchine sulla linea (ceratura, velocità, parametri dei pallettizzatori);

- il provvedimento di un sistema di gestione assistito al computer con tutte le informazioni sui prodotti e gli status dei PO e la tracciabilità mediante un database.

La software suite è una soluzione che può adattarsi a ogni tipo di linea, per il controllo automatico, la supervisione ed il recupero dei dati. L'accoppiamento con i meccanismi automatici delle macchine ompatibili permette la completa interazione che può essere estesa fino a comprendere la gestione dell'intera a linea, dall'ordine di produzione (PO) fino allo scarico dei pallet senza alcun intervento manuale.

2.6 Packaging (confezionamento)

La frutta che deve essere confezionata viene mandata attraverso dei nastri trasportatori verso l'area di packaging. Lungo il percorso, le mele vengono etichettate dalle macchine, poi vengono piazzate su vassoi o borse a mano o mediante un robot che poi li carica nelle scatole. I container pieni vengono etichettati con informazioni riguardanti varietà, dimensioni, classe, lotto dell'agricoltore e impianto, per la tracciabilità e la sicurezza del prodotto. I containers vengono poi pesati e piazzati sui pallet e poi vengono portati all'area di spedizione per il carico, oppure vengono fatti tornare nelle cellule frigorifere se non vengono spediti immediatamente.

2.6.1 Casse

Le mele possono essere riposte all'interno di casse utilizzando dei macchinari automatizzati, come quelli che si possono vedere nelle figure sottostanti.




Figura 2.10: Riempitrice di casse [16][19]

2.6.2 Alveoli di confezionamento

Molto spesso si preferisce mettere le mele all'interno di vassoi che possiedono degli alveoli in cui incastonare le mele. Rispetto alle casse, garantiscono protezione maggiore, ma il processo è molto più lungo e costoso a causa della difficoltà nel posizionamento delle mele.

Figura 2.11: Confezionamento di scatole [16]

Figura 2.12: Disimpilatore di vassoi [16]

Figura 2.13: Alveoli di confezionamento delle casse [16]

2.6.3 Cestini, borse e flowpack

All'interno di questo tipo di contenitori, le mele sono chiuse ermeticamente, per cui, oltre a ricevere protezione dagli urti, possono rimanere negli scaffali per un tempo lievemente maggiore.

Inoltre, il processo è facilmente automatizzabile, quindi risulta veloce e poco costoso.

Il problema maggiore risiede nelle piccole dimensioni di questi confezionamenti che possono ospitare meno di una decina di mele.

Figura 2.14: Riempitrice di cestini [16]

Figura 2.15: Riempitrici di borse [16]

Figura 2.16: Confezionamento di flowpack [16]

2.6.6 Etichettatura

Figura 2.17: Etichettatrici [16]

Questa fase avviene dopo che le mele sono state confezionate all'interno dei contenitori elencati in precedenza. L'etichettatura avviene in modo automatico, senza alcun intervento dell'essere umano.

Ciascun imballaggio deve recare, in caratteri leggibili, indelebili e visibili su uno dei lati dell'imballaggio, mediante stampatura diretta indelebile o mediante etichetta integrata nell'imballaggio o fissata ad esso, le indicazioni che seguono:

- Peso netto;
- Identificazione;
- Natura del prodotto;
- Elenco ingredienti;
- Termine minimo di conservazione;
- Origine del prodotto;

- Caratteristiche commerciali;
- Lotto;
- Prezzo al kg;
- Alcune indicazioni facoltative, tra cui quelle previste dalle norme comunitarie di qualità, il marchio ufficiale di controllo, la denominazione della varietà, il tenore minimo di zucchero, la consistenza massima, etc.

2.7 Pallettizzazione

L'ultimo anello della catena di confezionamento è la pallettizzazione che deve assicurarsi di fornire un prodotto irreprensibile.

L'automazione della pallettizzazione permette di risparmiare in manodopera e nella durata delle operazioni, inoltre questi sistemi possono essere integrati facilmente alle soluzioni di grading e packaging presentate sopra.

I pallettizzatori automatici possono impilare gli imballi su pallet standard e mezzi pallet, in modo tale da pallettizzare velocemente anche piccole quantità. Ciò può servire a risparmiare spazio, ma anche per reagire velocemente alle ordinazioni impreviste.

L'interfaccia grafica per l'utente sviluppata per le operazioni di pallettizzazione permette un intervento rapido e pulito. La gestione della pallettizzazione può essere integrata e eseguita utilizzando meccanismi e risorse tecniche utilizzate per la gestione della tracciabilità.

Figura 2.18: Pallettizzatrice di casse [16]

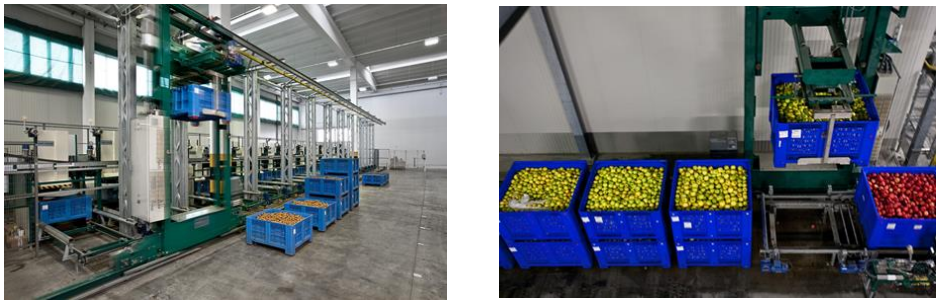


Figura 2.19: Pallettizzazione di bidoni [16]

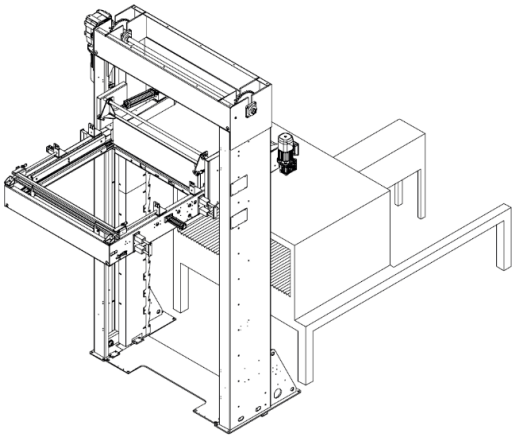


Figura 2.20: Layout del pallettizzatore automatico Icoel [19]

Ideale per l'accatastamento di casse di diversi formati e materiali.

Caratteristiche:

- capacità oraria:
  - pallets 80x120: da 600 a 1000 casse;
  - pallets 100x120: da 730 a 1000 casse;
  - pallets 120x120: da 700 a 800 casse.
- potenza installata: 4 kW.

CAPITOLO 3

3. Sistema POM

Come già detto nel capitolo precedente, La fase di confezionamento è una fase critica, perché le mele devono essere confezionate facendo attenzione che non vengano danneggiate durante il processo.

Il confezionamento è sempre stato fatto manualmente da operai nelle linee di lavoro, ma sono state sviluppate anche delle soluzioni automatiche che implementano dei robot. Una ragione rilevante per l'utilizzo dei robot, è che il confezionamento delle mele è un lavoro molto stancante e monotono per i lavoratori, perché non sono permessi errori, e devono ripetere un'azione all'interno di un intervallo di tempo predefinito. Per questi motivi l'azienda EPF ha sviluppato il sistema: POM – Automatic Smart Packing System.

Nelle pagine seguenti, si analizzeranno i componenti di cui è costituito il POM, andando a descrivere il loro funzionamento all'interno dell'impianto. Questo studio porterà allo sviluppo dell'argomento centrale della tesi, ovvero: come ottimizzare il POM dotandolo di telecamere per la visione stereoscopica.

3.2 Impianto

L’impianto è installato in serie con i lavoratori umani e non c’è pericolo per i lavoratori, sia perché il robot è collaborativo, sia perché si muove all’interno di una gabbia di metallo.

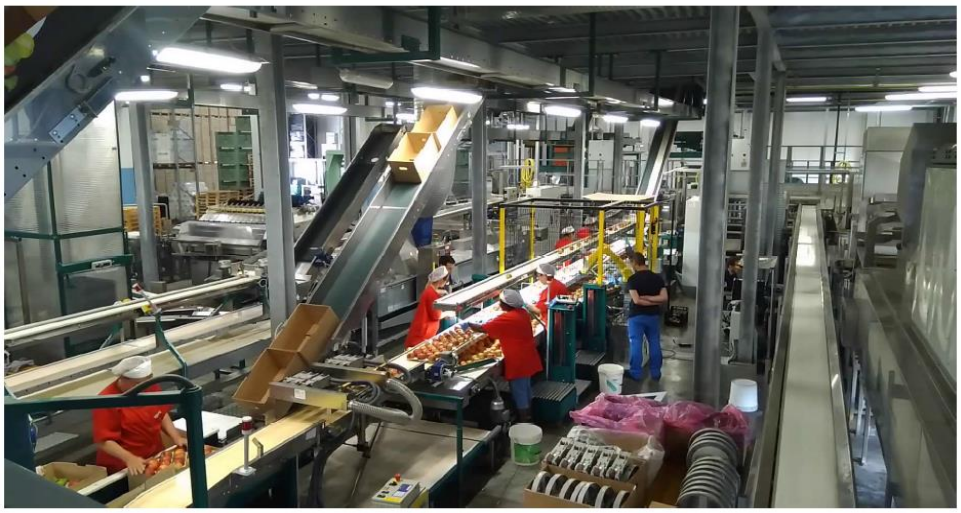


Figura 3.5: Impianto della fabbrica Sanifrutta [22]

Mentre le mele procedono sul nastro trasportatore, sia i lavoratori che il robot fanno il loro lavoro: prendono la mela, controllano l’orientazione e la mettono nel vassoio.

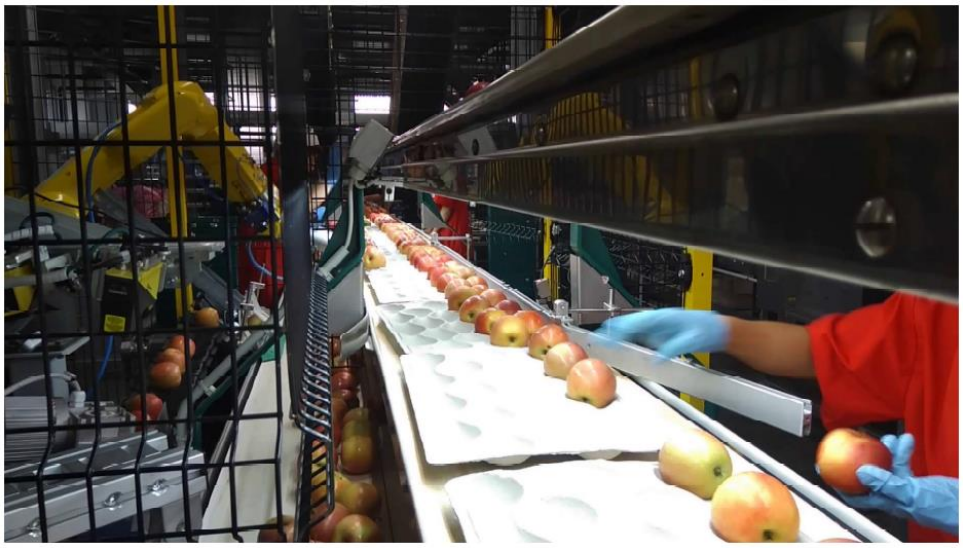


Figura 3.6: Picking effettuato da robot ed operatori umani [22]

Questo impianto consiste di:

- nastri trasportatori;
- convogliatori di mele;
- facchini;
- robot;
- sistema di visione.

3.2.1 Nastri trasportatori

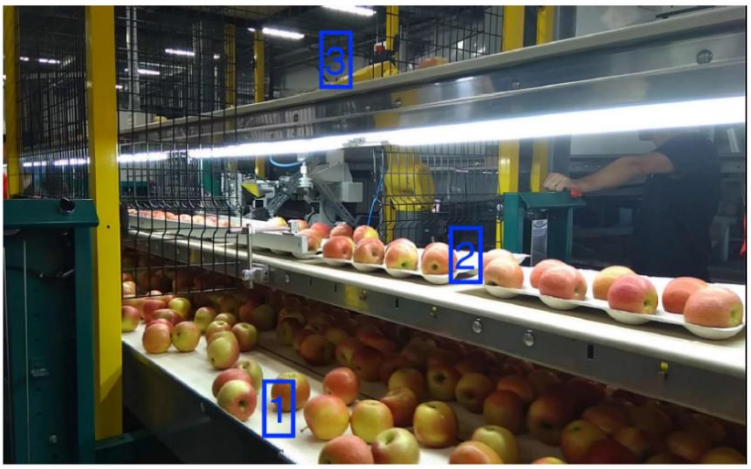


Figura 3.7: Nastri trasportatori [22]

Il nastro trasportatore 1 porta le mele che arrivano dal magazzino che sono state selezionate e messe nei vassoi, dai lavoratori o dal robot.

Il nastro trasportatore 2 è responsabile del trasporto dei vassoi, dove devono essere messe le mele che si spostano sotto.

Il nastro trasportatore 3 è dove sono messe le mele marcie o danneggiate. Queste mele non saranno buttate, ma saranno inviate a fabbriche che producono succhi di frutta.

3.2.2 Convogliatori di mele

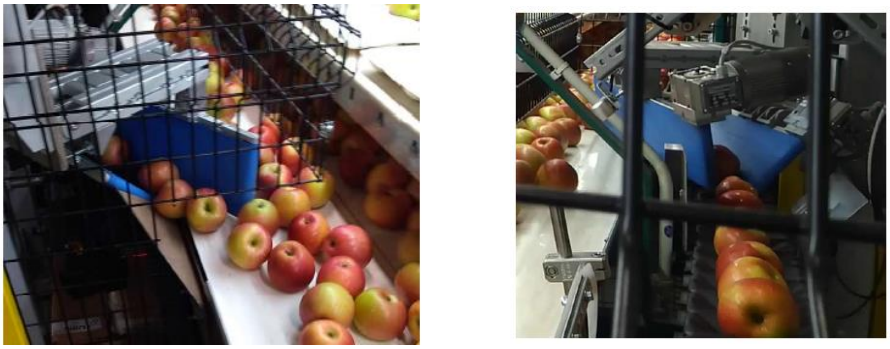


Figura 3.8: Convogliatori di mele [22]

Un facchino può ospitare solo una mela per volta, quindi, non si possono mandare tutte le mele, contemporaneamente, dal nastro trasportatore ai facchini.

Il convogliatore delle mele guida le mele verso i facchini, permettendo solo ad alcune di raggiungerli, lasciando le altre sul nastro trasportatore.

3.2.3 Robot

Tra tutti i robot che si possono utilizzare in un sistema automatico per il confezionamento delle mele, i robot collaborativi sono i più adatti ed i più sicuri, in quanto sono ideati per lavorare vicini ad altre persone.

Il robot utilizzato nell'impianto in questione è il Fanuc CR-7iA, un robot collaborativo a 6 assi, che può sopportare fino a 7 Kg di carico.

Questo robot è stato costruito esattamente per eseguire attività manuali leggere, come la movimentazione di vari tipi di materiali e l'assemblaggio di piccoli componenti. Inoltre, non ha bisogno di barriere: il sensore integrato lo arresta in automatico in caso di collisione con oggetti fissi o umani.

Tabella 3.1: Caratteristiche tecniche del robot

Assi	6
Ripetibilità	±0.01 mm
Peso	53 kg
Momento d'inerzia 4° asse	16,6/0,47 N*m/kg*m^2
Momento d'inerzia 5° asse	16,6/0,47 N*m/kg*m^2
Momento d'inerzia 6° asse	9,4/0,15 N*m/kg*m^2

Tabella 3.2: Caratteristiche di movimento del robot

Asse 1	340°
Asse 2	166°
Asse 3	373°
Asse 4	380°
Asse 5	240°
Asse 6	720°
Velocità massima	1000 °/s

Robot	CR-7*i*A
Robot footprint [mm]	296.5 x 235
Mounting position Floor	●
Mounting position Upside down	●
Mounting position Wall *3	●
Controller	**R-30*i*B Plus**
Open air cabinet	○
Mate cabinet	●
A-cabinet	-
B-cabinet	-
iPendant Touch	●
Electrical connections	
Voltage 50/60Hz 3phase [V]	-
Voltage 50/60Hz 1phase [V]	200-230
Average power consumption [kW]	0.5
Integrated services	
Integrated signals on upper arm In/Out	6/2
Integrated air supply	●
Environment	
Acoustic noise level [dB]	64.7
Ambient temperature [° C]	0-45
Protection	
Body standard/optional	IP67
Wrist & J3 arm standard/optional	IP67

*1) In case of short distance motion, the speed may not reach the maximum value stated.
*2) It is necessary to set a motion speed according to risk assessment of system considering-pinching with the surroundings.
*3) In case of the wall mount, the operation space will be restricted according with the payload.

● standard ○ on request - not available ( ) with hardware and/or software option **Based on ISO9283

Figura 3.1: Caratteristiche tecniche del robot Fanuc CR-7iA [21]

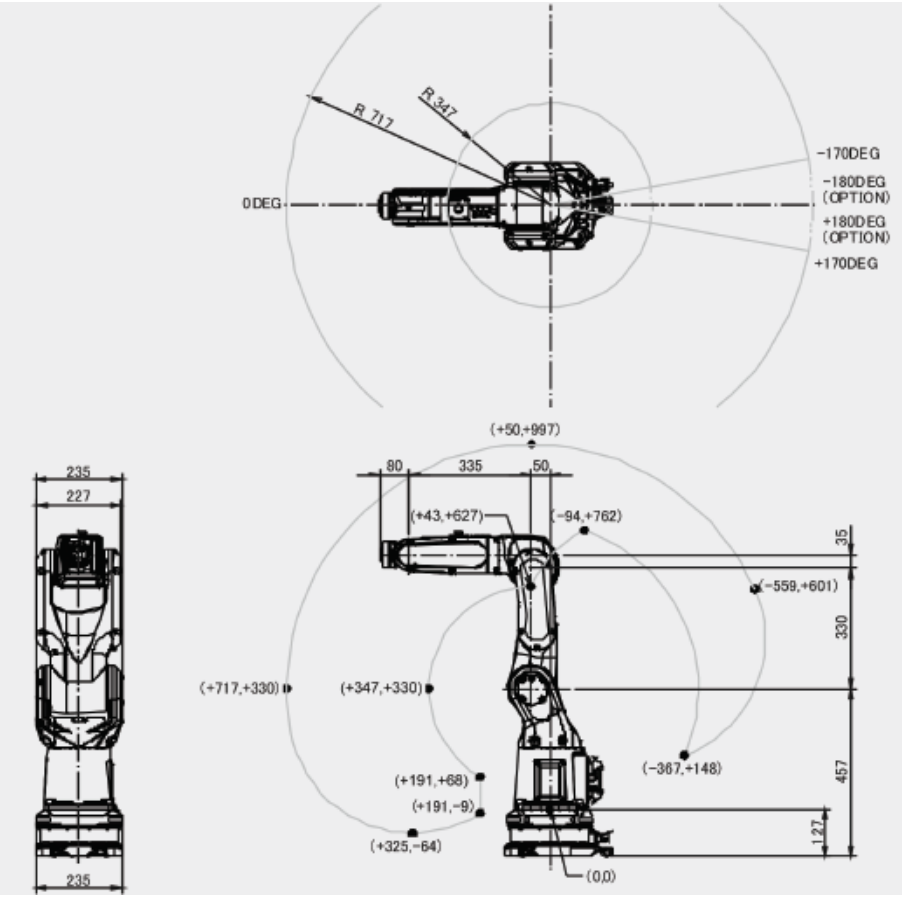


Figura 3.2: Dimensioni ed ingombri del robot Fanuc CR-7iA [21]

Dispositivo di estremità

La frutta e i vegetali sono molto delicati, quindi è necessario un effettore finale che sia in grado di maneggiare tali oggetti senza danneggiarli.

Figura 3.3: Ventosa [22]

La soluzione migliore è utilizzare un effettore finale con una ventosa. La ventosa sfrutta la pressione relativa negativa a causa del vuoto che si crea tra di essa e la mela. Questo è possibile perché a ventosa è fatta di gomma che deformandosi, aderisce quasi perfettamente alla superficie della mela. In questo modo abbiamo la pressione esterna (la pressione atmosferica) che è più alta della pressione interna: la forza risultante sulla superficie della mela è tale che mantiene la mela attaccata alla ventosa.

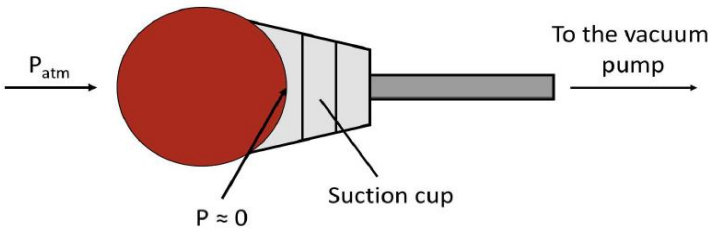
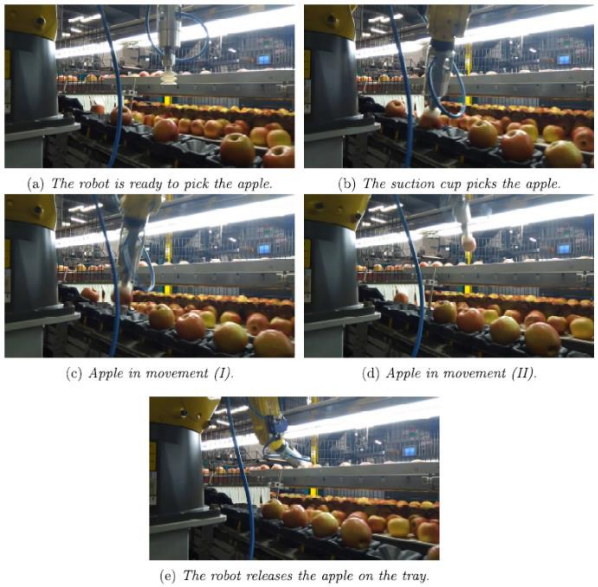


Figura 3.4: Schema di funzionamento della ventosa [22]

Funzionamento nell'impianto

Il robot ha il compito di prendere le mele dal nastro trasportatore e di metterle sui vassoi, con l'orientazione corretta.



(a) *The robot is ready to pick the apple.* (b) *The suction cup picks the apple.*

(c) *Apple in movement (I).* (d) *Apple in movement (II).*

(e) *The robot releases the apple on the tray.*

Figura 3.9: Fasi del funzionamento del robot [22]

3.2.4 Sistema di visione

Il sistema di visione non è solamente responsabile del riconoscimento delle mele, ma deve anche identificare la posizione dello stelo (se presente), o della cavità, o del calice. Per farlo si sfruttano algoritmi di identificazione degli oggetti.

Dopo che lo stelo o il calice sono identificati, questa informazione è utilizzata per calcolare l'orientazione della mela. L'informazione dell'orientazione sarà mandata al robot che prenderà e metterà la mela sul vassoio, con l'orientazione corretta.

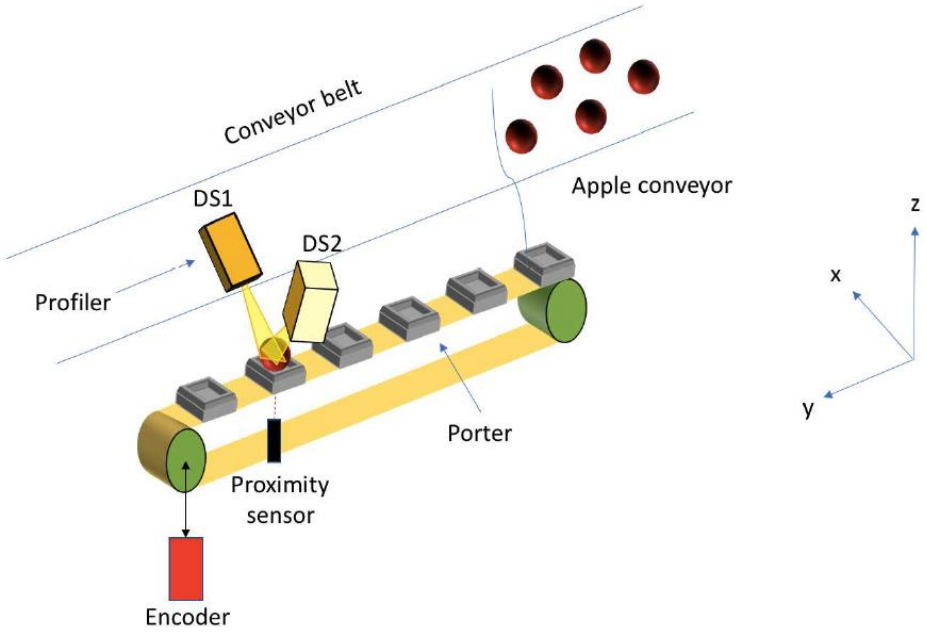


Figura 3.10: Schema del sistema di visione [22]

Gli elementi principali del sistema di visione sono:

- profilatore 3D;
- sensore di prossimità;
- encoder.

3.3 Problemi e possibili sviluppi

I profilatori forniscono immagini in bianco e nero. Questo accade perché il profilatore è uno scannerizzatore laser che può ricavare solo la distanza verticale da una mela e mapparla in scala di grigi. Inoltre, il profilatore è un dispositivo molto costoso.

Per queste ragioni si è pensato di sostituirlo con due telecamere stereoscopiche RGB. Questi sono alcuni dei vantaggi nel loro utilizzo:

- si ottiene l'informazione di colore persa dai profilatori;

- sono molto più convenienti di un profilatore e permettono di ottenere risultati comparabili;
- sono più piccole, rendendo possibile la riduzione della dimensione complessiva del sistema;
- si potrebbe implementare anche un controllo di qualità della superficie, per scartare le mele non vendibili, basandosi solamente sul colore della superficie.

Prima di costruire un nuovo impianto, sostituendo il profilometro con delle telecamere, è necessario ideare e creare un prototipo per poi collaudarlo in modo tale da favorirne l'inserimento in ambiente industriale, senza che ciò provochi troppi problemi di adattamento.

Nel paragrafo seguente si cercherà di capire qual è il vero compito delle telecamere all'interno di un sistema come il POM, e si cercherà di capire quali sono i problemi che incontreranno e come risolverli.

3.4 Telecamere stereoscopiche

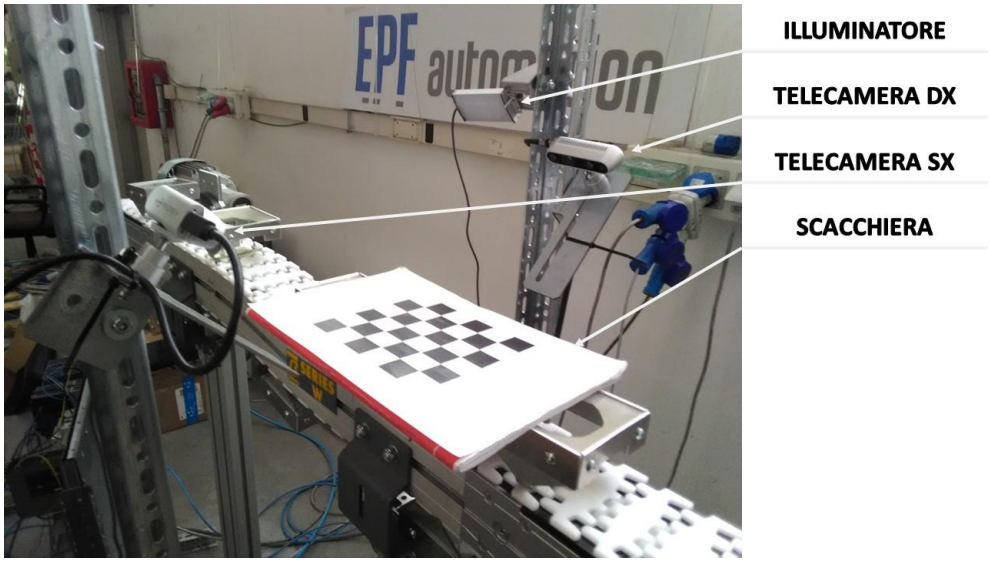


Figura 3.11: Posizionamento delle telecamere all'interno del sistema

I compiti principali delle telecamere sono essenzialmente 2:

- catturare le immagini 2D e 3D delle mele che passano sul nastro trasportatore.
- catturare le immagini 2D e 3D della scacchiera di calibrazione

Per eseguire queste funzioni sono necessarie delle telecamere stereoscopiche accompagnate da un sistema di illuminazione, in quanto le telecamere sono molto sensibili alla qualità dell'illuminazione ed il sistema deve essere in grado di operare in qualsiasi ora della giornata.

Come si può notare dalla figura sopra, le telecamere sono poste molto vicine al nastro. Ciò serve ad ottenere delle immagini 3D più precise, in quanto, come vedremo, la loro accuratezza diminuisce all'aumentare della distanza.

Allo stesso tempo, non è possibile posizionare le telecamere al di sotto di una cera distanza minima, sia per motivazioni stereoscopiche, sia per motivazioni di FOV. Infatti, la telecamera deve poter inquadrare l'intera scacchiera per poter eseguire la calibrazione. Come verrà spiegato nel capitolo 5, questa è una fase che serve ad impostare il sistema di riferimento delle telecamere.

Inoltre, per evitare riflessi e per facilitare il riconoscimento della scacchiera da parte del programma, le telecamere devono stare al di sopra di una certa inclinazione minima.

Dall'altra parte, le telecamere, per inquadrare la maggior porzione di mela possibile dovrebbero stare più orizzontali possibile.

In questa applicazione in particolare, le telecamere non solo devono essere in grado di ricavare un'immagine 3D abbastanza precisa (con un errore massimo di 1mm), ma devono anche poter elaborare le immagini abbastanza velocemente in quanto le mele sono in movimento sul nastro trasportatore.

Le immagini prese dalle telecamere devono essere trasferite al PC in modo da essere utilizzate all'interno del software che unirà le due point-cloud e consentirà la visualizzazione di entrambi i lati della mela.

In conclusione, le telecamere stereoscopiche all'interno di questo sistema hanno bisogno di:

1. rilevare l'immagine 3D in modo accurato (errore $\leq$ 1mm);
2. catturare l'immagine delle mele in movimento (alti FPS);
3. stare più vicine possibili all'oggetto da osservare (bassa distanza min. e alto FOV);
4. un programma di configurazione che sia compatibile con il software dell'applicazione.

Oltre alle questioni tecniche, bisogna aggiungere che queste telecamere non possono essere troppo costose, in quanto l'obbiettivo principale è la sostituzione del sistema POM con un altro che sia in grado di svolgere lo stesso compito, ma in modo più conveniente.

Conseguentemente il sistema è stato munito di due telecamere Realsense D435. Queste telecamere, non solo soddisfano tutte le necessità tecniche scritte sopra, ma si possono configurare tramite l'Intel RealSense Software Development Kit (SDK) 2.0, un programma facile da usare e che tra i suoi wrappers dispone di Python, il linguaggio di programmazione con cui è stato codificato il programma dell'intera applicazione.

3.4.1 Dati e specifiche tecniche

L'Intel Realsense Depth Camera D435 è una telecamera alimentabile tramite cavo USB e possiede un ampio campo visivo sia per il sensore di profondità che per il sensore RGB. È la scelta ideale per l'hardware di prototipazione e per lo sviluppo di nuovi software, grazie al suo basso costo. La telecamera è disegnata per essere portatile e facile da configurare. Perfetta per catturare immagini di oggetti che si muovono velocemente.

Tabella 3.3: Specifiche tecniche delle telecamere D435 [23]

Camera Features	
Use Environment	Indoor/Outdoor
Depth Technology	Active IR Stereo (Global Shutter)
Depth FOV (H x V x D)	91.2 x 65.5 x 100.6
Depth Output Resolution & Frame Rate	Up to 1280x720 Up to 90fps
Minimum Depth Distance (Min-Z)	0.2m
Maximum Range	10m+. Varies depending on performance accuracy, scene and light conditions
RGB Resolution	1080p @ 30fps

3.4.2 Configurazione tramite Realsense SDK

Storicamente, le telecamere di profondità sono sempre state costose e difficili da usare. Invece, le telecamere di profondità RealSense D400 sono poco costose e facili da usare, sia per l'uso interno che quello all'aperto.

Le telecamere RealSense D435 sono pronte all'utilizzo, basta collegarle al PC tramite il cavo USB.

Per procedere alla configurazione e all'utilizzo basta scaricare dal sito intelrealsense.com la piattaforma RealSense SDK 2.0. L'SDK è composta da 4 componenti principali: la libreria librealsense2, gli strumenti, i codici di esempio e i wrappers.

Librealsense2
Questa libreria è il componente centrale della SDK e fornisce un'API per configurare, controllare e accedere ai dati dello streaming dalle telecamere di profondità. L'API permette di iniziare a prendere confidenza con le funzioni di base della camera utilizzando un'API di alto livello, oppure di prendere il totale controllo di tutti i parametri della telecamera utilizzando un'API di basso livello.

Strumenti
Forniti dall'SDK ci sono due strumenti di applicazione e un set di cinque strumenti di debug. Gli strumenti di applicazione includono il RealSense Viewer (che permette la facile visualizzazione degli stream video e di profondità e la configurazione della camera) e il Depth Quality Tool (che permette agli sviluppatori di testare la qualità della sensibilità in profondità della camera).

Codice di esempio
Ci sono una ventina di esempi di codice che dimostrano come utilizzare l'SDK e le camere di profondità per diversi scopi, includendo l'allineamento dei frames di profondità con i corrispondenti frames a colori, come mostrare la distanza della camera dall'oggetto che si trova al centro dell'immagine e come applicare uno degli algoritmi già esistenti di network neurale profondo(DNN).

Wrappers
Viene fornito supporto per un ampio range di linguaggi e piattaforme software dai wrappers inclusi, quali Python, .NET, Node.js, Robot Operating System (ROS), LabView, Point-Cloud Library (PCL) e la piattaforma di gaming Unity.

Ne1 prossimo capitolo, nel paragrafo "configurazione delle telecamere realsense" si analizzeranno alcune delle caratteristiche della telecamera

che sono state configurate mediante questi strumenti e si vedrà come impostarla per ottenere la migliore accuratezza in diverse condizioni.

CAPITOLO 4

4. Sistemi di visione industriale

In questo capitolo si analizzerà l'importanza e la diffusione commerciale dei sistemi di visione e si andranno a descrivere nel dettaglio alcuni dei componenti di cui, normalmente, è costituita. Quindi, ne vedremo qualche applicazione in ambito industriale, ed infine, si andrà ad approfondire la tecnica della visione stereoscopica.

Per una buona messa a punto del programma che andrà a gestire le telecamere, è necessario capire, come funziona la visione stereoscopica e come deve essere strutturato un sistema di visione industriale.

4.1 Mercato

Le analisi di mercato svolte in questi ultimi anni mostrano che il ramo della visione artificiale è tra quelli più in ascesa dell'automazione industriale. Il motivo primario è l'incremento dell'impegno sul controllo della qualità e dell'efficienza, che porta all'aumento della redditività e della competitività.

Basandosi su un resoconto redatto dalla società di ricerche di mercato MarketsandMarkets si prevede che tra il 2016 e il 2022 il business della visione artificiale crescerà annualmente in media dell'8,15% fino a raggiungere i 14,43 miliardi di dollari nel 2022.

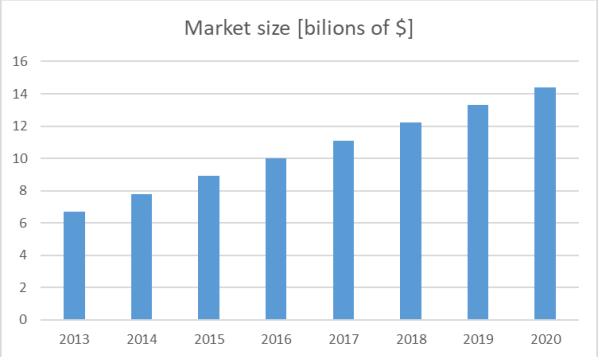


Figura 4.1: Mercato della visione artificiale [25]

La crescita del mercato è dovuta dall'aumento della domanda di sistemi robotici vision-guided in settori quali: automotive, farmaceutico, industriale, alimentare e packaging. A cui si aggiunge la crescita della richiesta di sistemi di visione per applicazioni specifiche.

Per ciò che riguarda l'hardware, il segmento fotocamera copre la maggior parte del mercato della visione; in futuro, questo segmento sarà soggetto a miglioramenti aggiuntivi con l'arrivo della visione artificiale 3D.

L'incremento dei salariati in Cina e altri Paesi in via di sviluppo, la standardizzazione e l'aumento della domanda di sistemi di visione artificiale provocano il mantenimento dell'elevata richiesta di sistemi di visione artificiale in tutto il globo. La zona APAC (Asia-Pacifico) continua a mantenere la fetta di mercato maggiore della visione artificiale.

4.2 Struttura di un sistema di visione

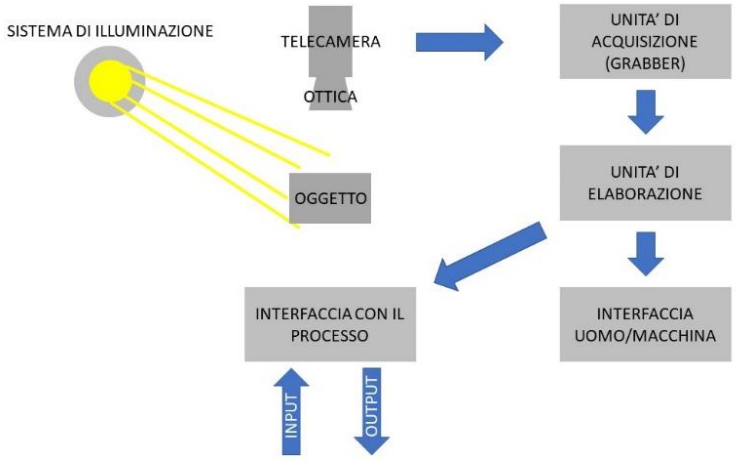


Figura 4.2: Schema di un sistema di visione generico

In genere, un sistema di visione è costituito dalle componenti seguenti:

- telecamere e ottiche
- sistema di illuminazione
- sistema di acquisizione ed elaborazione dell'immagine

Gli oggetti da osservare sono posizionati davanti a una o più telecamere e illuminati per mettere in evidenza i dettagli da identificare. Il sistema ottico crea un'immagine sul sensore della telecamera che di conseguenza genera un segnale elettrico. Successivamente questo segnale può venire digitalizzato e salvato in memoria.

Quindi, l'immagine può venire processata da un software che racchiude algoritmi di calcolo ed analisi, che permette di trovare le caratteristiche dell'immagine e evidenziarne alcuni aspetti in modo tale da effettuare i controlli e le verifiche per i quali il sistema è stato ideato.

4.2.1 I Meccanismi di acquisizione

L'acquisizione, la tipologia, la qualità e le altre caratteristiche dell'immagine sono influenzate dai sensori addetti all'acquisizione. I

sensori hanno il compito di trasformare l'energia della radiazione luminosa in un segnale elettrico mediante un materiale fotosensibile.

Esistono tre tipi di sensori di acquisizione principali:

- sensore singolo, ovvero un fotodiodo che abbinato ad uno spostamento consente di derivare immagini 2D;
- sensore lineare, costituito da una riga di sensori singoli che necessita di movimento per ricavare immagini 2D; la riga può diventare un anello nell'evenienza di TAC, MRI e PET;
- sensore a matrice, cioè una matrice di sensori CCD, impiegato nelle fotocamere e telecamere digitali.

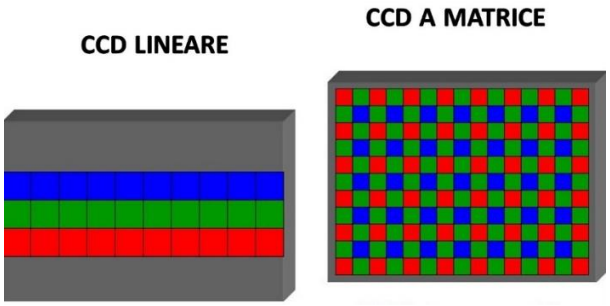


Figura 4.3: Schema funzionale dei sensori CCD [26]

L'acquisizione si può distinguere in: monodimensionale, bidimensionale, tridimensionale.

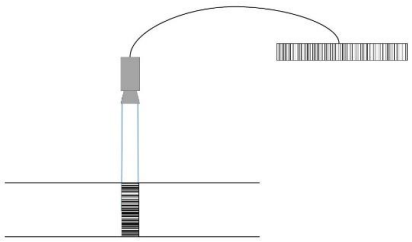


Figura 4.4: Schema dell'acquisizione monodimensionale

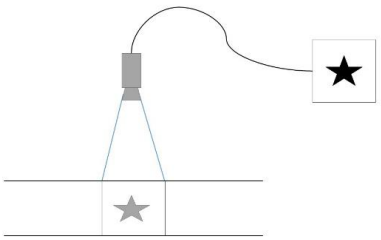


Figura 4.5: Schema dell'acquisizione bidimensionale

Monodimensionale

L'acquisizione monodimensionale, anche chiamata lineare, si serve di un sensore con una riga di elementi sensibili, detti pixel. Conseguentemente, l'immagine per ogni fotogramma è composta da una fila di punti. Si può catturare un'immagine bidimensionale procurandosi più righe con un movimento relativo tra l'oggetto ed il sensore.

Di solito, le telecamere con sensore lineare hanno velocità di acquisizione e risoluzioni molto alte. Il bisogno di attuare un movimento relativo tra oggetto e sensore fa diventare conveniente questa tipologia di acquisizione qualora il pezzo sia già mobile sulla linea di produzione. Per di più, permette il processamento anche di oggetti continui.

Bidimensionale

L'acquisizione bidimensionale, o matriciale, è la più utilizzata. Il sensore è composto da una matrice di pixel che vengono impressionati contemporaneamente, ottenendo un'immagine bidimensionale. Esistono diversi tipi di sensori matriciali, con risoluzioni che vanno da poche centinaia di pixel quadrati per i sensori di visione, alle telecamere Megapixel da 6 milioni di punti.

Questo tipo di acquisizione si presta ad acquisire oggetti che devono essere elaborati singolarmente. Inoltre, esistono telecamere specializzate nell'acquisizione ad alta velocità, ad alta risoluzione, o con sensibilità a diverse frequenze di luce.

Tridimensionale

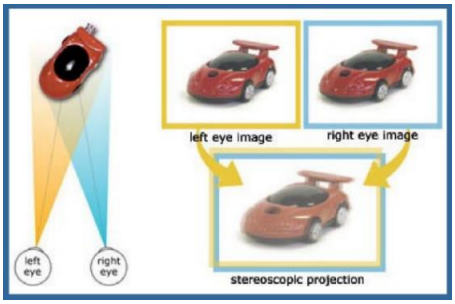


Figura 4.6: Acquisizione stereoscopica [27]

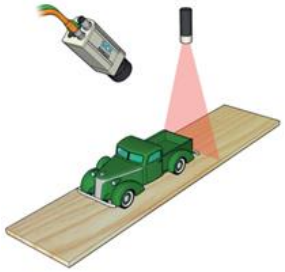


Figura 4.7: Triangolazione con luce strutturata [28]

Ci sono molti modi per ottenere un'immagine 3D. Le tecniche più comuni sono l'acquisizione stereoscopica e la triangolazione con luce strutturata.

La prima si basa sull'acquisizione di un'area da due telecamere da diverse angolazioni. La localizzazione dei punti comuni nelle diverse immagini, insieme alla conoscenza degli angoli di ripresa, permettono di ottenere informazioni di profondità per ogni punto dell'immagine.

La seconda, invece, si basa sulla proiezione di una o più linee sull'oggetto e l'acquisizione dell'immagine del profilo creato sull'oggetto mediante un sensore bidimensionale, posizionato con un angolo noto rispetto al piano della proiezione. Dall'analisi dell'immagine del profilo si ricava una sezione dell'oggetto. L'acquisizione di molteplici linee mediante un movimento relativo fra oggetto e telecamera permette di creare diverse sezioni in modo tale da comporre un'immagine tridimensionale.

La triangolazione con luce strutturata garantisce una buona precisione e risulta robusta ai disturbi ambientali, ma richiede un movimento relativo fra oggetto e sistema di triangolazione. L'acquisizione stereoscopica permette di creare un'immagine 3D senza necessità di parti in movimento, ma la precisione di misura e la resistenza ai disturbi sono inferiori.

4.2.2 La Telecamera

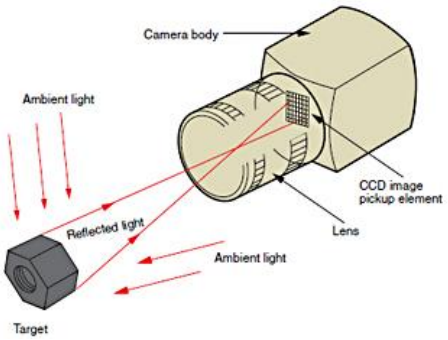


Figura 4.8: Schema di funzionamento di una telecamera [29]

Le telecamere trasferiscono le immagini a sistemi di supervisione, esame e stoccaggio tramite l'utilizzo di server video, registratori e monitor. In base alla tipologia di applicazione si possono impiegare telecamere analogiche, digitali, con sensore CCD o CMOS, statiche, manovrabili, dome (da soffitto), a scansione progressiva, a infrarossi ecc. Le telecamere attuali, di solito, sono dotate di protocolli e interfacce di rete di vario tipo (IP, PoE, telefoniche, mesh, fibra ottica, wireless ecc.), capacità di controllo via web e archiviazione di immagini a bordo (con schede di memoria).

Le telecamere professionali esigono performance di alto rango e richiedono tecnologie come l'alta definizione, l'ottica per lunghe distanze e in condizione day & night, streaming diretto su web, capacità di supportare interfacce wireless, cellulari (smartphone), PC e webcam.

Le telecamere industriali più richieste sono di tipo CCD (Charged Coupled Device), si basano su sensori a trasferimento di carica in cui le immagini sono digitalizzate e memorizzate in un chip tramite fototransistor che trasformano l'intensità luminosa in elettroni liberi.

Esistono anche le tecnologie CMOS (Complementary Metal Oxide Semiconductor) che realizzano sul medesimo substrato sia la porzione

fotosensibile che la circuiteria accessoria, il che permette la creazione di prodotti economici, come ad esempio le smart camera, che possiedono discrete capacità di calcolo e memorizzazione.

4.2.3 L'Ottica

Il compito del sistema ottico è quello di focalizzare l'immagine sul sensore della telecamera. Alcuni attributi caratteristici delle lenti (obiettivo) sono:

- Lunghezza focale: è la distanza tra la lente e il piano sul quale si forma il punto focale di un fascio di raggi che risultano paralleli all'asse ottico. Da ciò dipende l'estensione dell'immagine, ovvero l'ingrandimento, e lo spazio tra la telecamera e il pezzo da osservare.
- Rapporto focale o apertura: è il rapporto tra la lunghezza focale e il diametro utile della lente. Più è piccolo, più è grande la luminosità e la risoluzione dell'immagine.
- Profondità di campo: è l'intervallo all'interno del quale l'immagine si trova a fuoco.

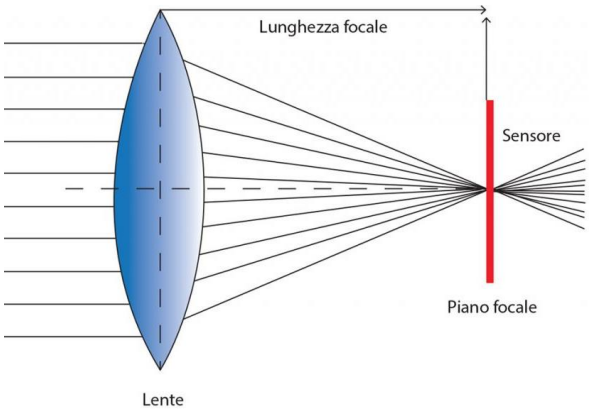


Figura 4.9: Schema semplificato di una lente [30]

Potrebbe essere necessario disporre di un sistema con grande profondità di campo qualora l'oggetto possa presentarsi ad un range di distanze diverse oppure se l'oggetto risulta abbastanza profondo.

Invece, si può utilizzare un sistema con bassa profondità di campo per semplificare l'elaborazione dell'immagine, mettendo fuori fuoco le aree inquadrate al di fuori della zona di interesse.

4.2.4 Il sistema di illuminazione

Questo sistema è una parte fondamentale del sistema in quanto definisce la maniera con cui l'immagine verrà acquisita ed elaborata. Le metodiche di illuminazione possibili sono:

- illuminazione direzionale;
- illuminazione diffusa (o omnidirezionale);
- illuminazione coassiale;
- retroilluminazione (diascopia).

La scelta del metodo dipende anche dalle proprietà fisiche della luce utilizzata.

Figura 4.10: Angoli di illuminazione [31]

Illuminazione direzionale

Risulta essere la più semplice da compiere perché qualsiasi sorgente sufficientemente densa può fornire un'illuminazione direzionale. In generale la luce direzionale è adatta alla illuminazione di superfici piane, poiché su superfici curve darebbe luogo a intensità di illuminazione non uniformi che possono inficiare l'analisi dell'immagine. Nel caso di oggetti riflettenti può dare luogo a riflessi che di solito sono considerati da evitare. Una sorgente direzionale tipica è il laser.

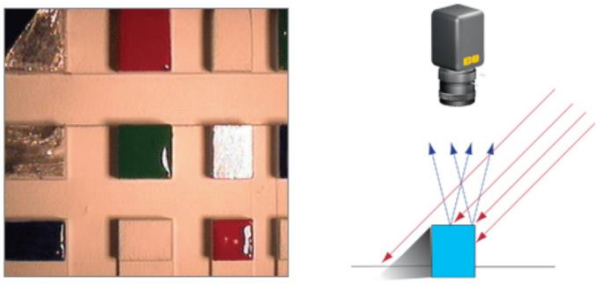


Figura 4.11: Esempio di illuminazione direzionale [32]

Illuminazione diffusa

È la più adatta agli oggetti riflettenti. Si può ottenere tramite sistemi in cui l'illuminazione viene riflessa da cupole diffusive e arriva sull'oggetto da molteplici direzioni. A volte risulta di difficile applicazione per l'ingombro, per la ridotta distanza che deve intercorrere fra illuminatore e oggetto e per il costo.

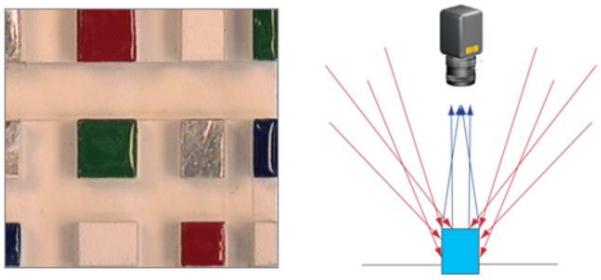


Figura 4.12: Esempio di illuminazione diffusa [32]

Illuminazione coassiale

In questa tecnica la sorgente – tipicamente una lampada circolare a fluorescenza o una corona di led – è posta intorno all'obiettivo della telecamera. In questo modo la luce viene proiettata in linea con l'asse ottico della telecamera. Si utilizza quando l'oggetto presenta una superficie opaca e diffusiva, per evidenziare zone chiare e scure. Su superfici riflettenti può essere usata per mettere in evidenza particolari con diversa riflessione. Questa illuminazione si utilizza con lo stesso scopo del flash.

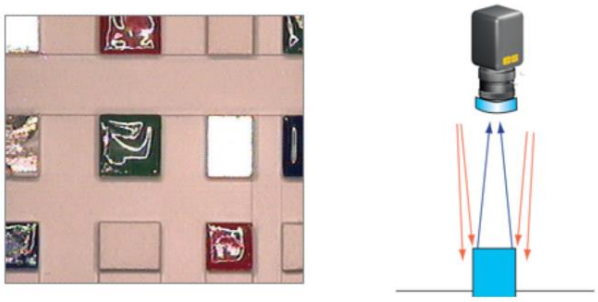


Figura 4.13: Esempio di illuminazione coassiale [32]

Retroilluminazione

L'oggetto viene posto fra la telecamera ed uno sfondo luminoso che costituisce la sorgente. Gli oggetti opachi appaiono neri su di uno

sfondo chiaro. Ciò permette di eseguire misurazioni 2D con la massima affidabilità.

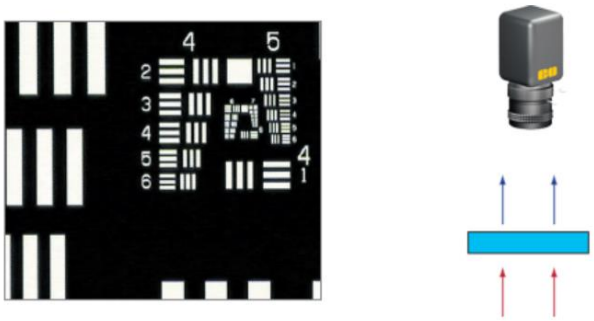


Figura 4.14: Esempio di retroilluminazione [32]

Se utilizzata con oggetti trasparenti, difetti quali macchie o particolari risultano visibili sia sulla faccia dell'oggetto rivolta verso la telecamera, sia su quella rivolta verso il pannello luminoso. La sola zona che resta esclusa dal controllo è quella vicina ai bordi.

4.3 Applicazioni in ambito industriale

A causa delle molteplici funzionalità e dell'alta capacità di calcolo, i sistemi di visione potrebbero essere impiegati in un numero di ambiti potenzialmente illimitati.

Tra gli utilizzi più comuni nel settore industriale appaiono:

- controllo qualità;
- orientamento, posizionamento e guida robot;
- verifiche su nastri in continuo;
- lettura di caratteri e codici.

4.3.1 Controllo qualità

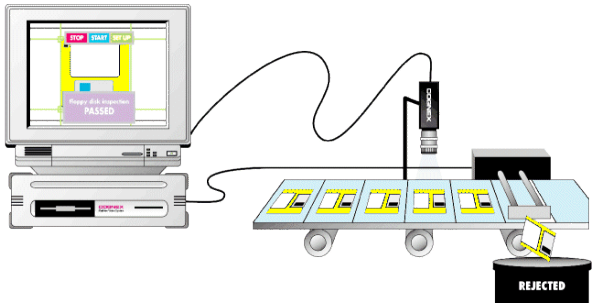


Figura 4.15: Schema di ispezione industriale [33]

La visione industriale risulta essere la tecnica meno intrusiva e più facile da implementare per poter eseguire in parallelo l'ispezione dei prodotti a fianco delle linee di produzione. Essa, è in grado di svolgere tutti gli interventi di controllo "a distanza", senza dover toccare fisicamente il prodotto in esame.

Se il sistema viene inserito nella linea nel punto appropriato può:

- impedire la perdita di parte del prodotto consentendo un rientro immediato sull'investimento;
- assicurare l'uniformità del prodotto che esce dalla linea;
- fornire molti dati statistici che possono essere utilizzati per diminuire la difettosità o migliorare la produttività.

Le combinazioni di misure, verifiche e letture che si possono effettuare sono moltissime e varie: misure dimensionali, verifiche di presenza/assenza, verifiche cromatiche, verifiche estetiche, lettura di codici e scritte, e ovviamente la combinazione di tutti i sopraccitati. Spesso gli algoritmi di controllo qualità sono accompagnati a quelli di localizzazione.

Controllo qualità 2D

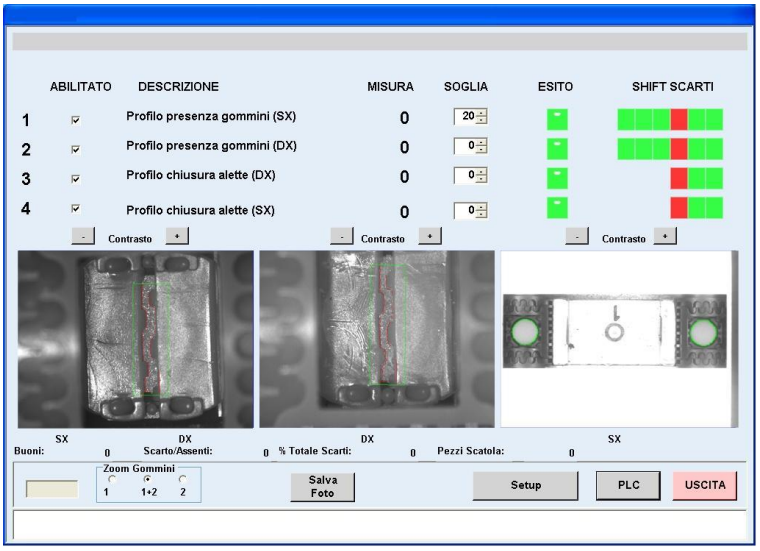


Figura 4.16: Finestra di interfaccia per il controllo del profilo 2d [34]

Le immagini possono essere catturate sia da telecamere matriciali, che lineari o di altra natura, per poi essere elaborate da un algoritmo bidimensionale di image processing.

Spesso, per effettuare la verifica, l'immagine viene scomposta nelle sue linee primitive come, cerchi, angoli, contorni, creste ed altro.

All'interno dell'immagine i difetti devono essere ben visibili, per questo il materiale deve essere irraggiato da un sistema di illuminazione adatto allo scopo.

Controllo qualità 3D

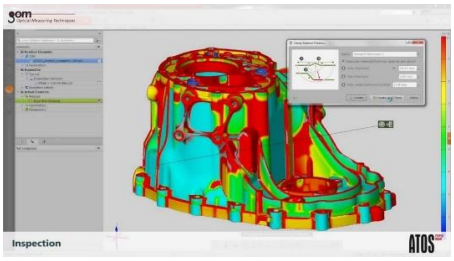


Figura 4.17: Finestra di interfaccia per il controllo del profilo 3D [35]

Per implementare un controllo qualità 3D bisogna ideare un sistema in grado di creare delle immagini 3D, in cui ogni pixel possegga l'informazione della distanza e non più un'intensità di colore come nel caso 2D.

Dopo aver creato questa mappa di profondità, si possono percorrere due vie:

- Si può eseguire un'elaborazione standard sui toni di grigio della mappa di profondità;

- Si può elaborare l'immagine in modo da trovarne le sue primitive e poi confrontare le loro dimensioni rispetto ad un riferimento nel mondo reale.

4.3.2 Guida robot

La guida robot è la divisione all'interno del quale i sistemi di visione hanno avuto maggior successo fino ad ora a causa della solidità crescente degli algoritmi di localizzazione.

Robot e cobot, per potersi muovere con sicurezza e disinvoltura all'interno degli spazi di lavoro e dedicarsi alle operazioni di picking con la massima precisione, anche ad alte velocità, richiedono camere di visione interamente programmabili, di modo che l'utilizzatore possa modificare le applicazioni a seconda delle proprie necessità, in modo del tutto autonomo.

Bin Picking

Figura 4.18: robot per bin picking e sistema di visione [36]

Il “bin-picking” consiste nell’insegnare a un robot ad identificare e prendere oggetti posti in modo casuale dentro delle casse. Questo incarico non può essere svolto da un robot industriale che non dispone di sensori in grado di riconoscere l’ambiente in cui lavora.

Per consentire al robot il riconoscimento dei pezzi da afferrare necessita di un sistema di visione, spesso tridimensionale, e di un algoritmo che consente l’identificazione degli oggetti.

Non solo, il robot deve anche poter manipolare gli oggetti in modo tale da prelevarli. Per questo, Il robot deve possedere un algoritmo che lo renda in grado di effettuare più tipi di prese in modo tale da poter trovare sempre una maniera per prendere l’oggetto portarlo a destinazione.

Inoltre, sarà necessario implementare un altro algoritmo che sia in grado di portare il robot al punto di prelievo assumendo la giusta modalità di presa per poi passare all’estrazione, eseguita percorrendo traiettorie che evitino collisioni gli oggetti presente dentro l’ambiente.

Riassumendo, un sistema di bin-picking deve:

- individuare gli oggetti;
- scegliere la presa migliore per afferrarli;
- ideare il percorso da seguire per la mano del robot.

Pick and place 2D

Questo tipo di applicazione di solito necessita di sviluppare degli algoritmi di localizzazione per operare con oggetti appoggiati su un piano ed aventi 3 gradi di libertà: X,Y, rotazione.

Le librerie software più avanzate possono simulare la deformazione del modello dovuta sia a cause meccaniche che per causa della prospettiva.

Di solito gli algoritmi di localizzazione 2D sono in grado di far fronte alle occlusioni parziali degli oggetti e ai cambiamenti di luminosità.

Soventemente l'algoritmo necessita che l'utente specifichi una zona dell'immagine da sfruttare come modello da ritrovare all'interno delle nuove immagini da acquisire. Successivamente si passa al robot l'informazione del punto di presa.

Quando si configura la guida robot, si deve calibrare il sistema di visione in modo che riferimenti corrispondano ad un dato frame del robot.

Le applicazioni di pick and place che possiedono formati diversi hanno bisogno di un'interfaccia utente per la loro gestione, per la definizione di modelli nuovi, per la modifica di quelli già disponibili, per l'adattamento dei punti di presa, per la calibrazione del sistema, per la gestione dei parametri di accettazione, ecc.

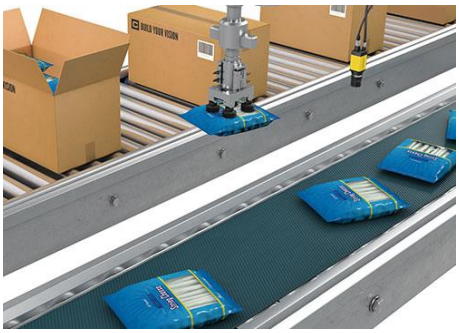
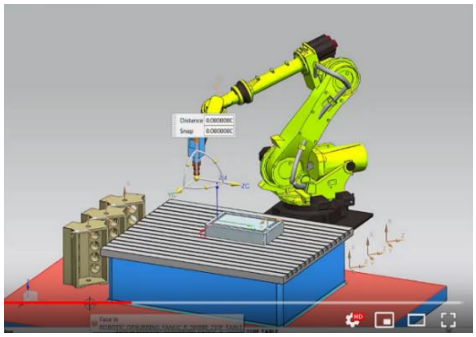
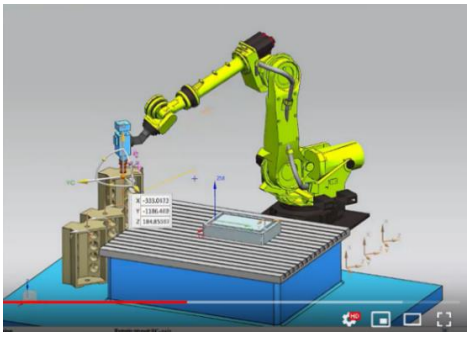


Figura 4.19: Esempi di robot per pick and place 2D [37] [38]

Pick and place 3D

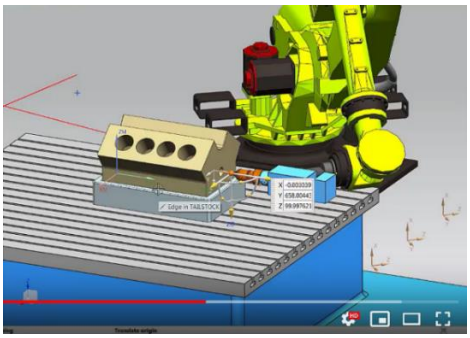


inizio (pezzo in stazione di carico)

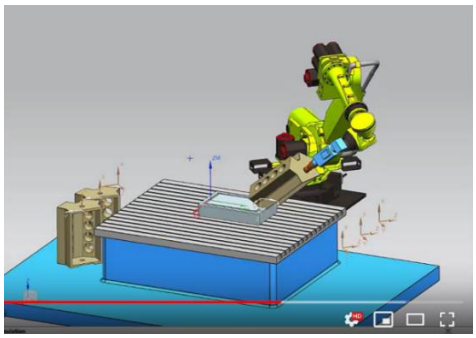


sollevamento

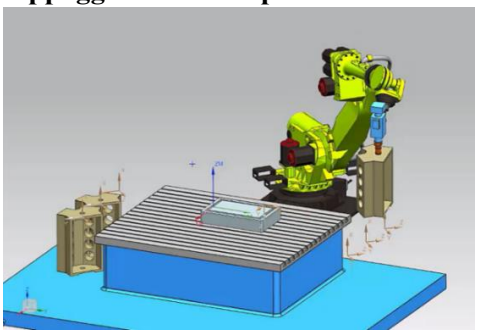
avvicinamento al banco



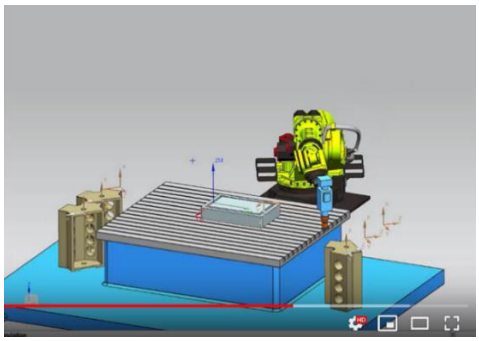
appoggio sul banco per lavorazione



ripresa del pezzo



allontanamento dal banco



fine (pezzo in stazione di scarico)

Il pick and place 3D è simile alla versione 2D, ma vi è una più grande pluralità negli algoritmi di localizzazione.
Si calibra il sistema 3D in modo analogo a quello 2D e dopo che il frame del robot è in linea con quello della telecamera, gli aggiustamenti proposti dal sistema di visione corrispondono con quelli del robot.

Figura 4.20: Esempio di un'operazione di pick and place 3d [39]

4.3.3 Lettura in continuo

In questo caso si utilizza una o più telecamere lineari per identificare i difetti sul materiale continuo in tempo reale e successivamente si memorizzano le misure e le immagini all'interno di un database. La scheda dei difetti, che si può consultare dalla rete aziendale, viene generata automaticamente. I dati si possono integrare con altri sistemi come, ad esempio, quello che mantiene la tracciabilità dei prodotti.

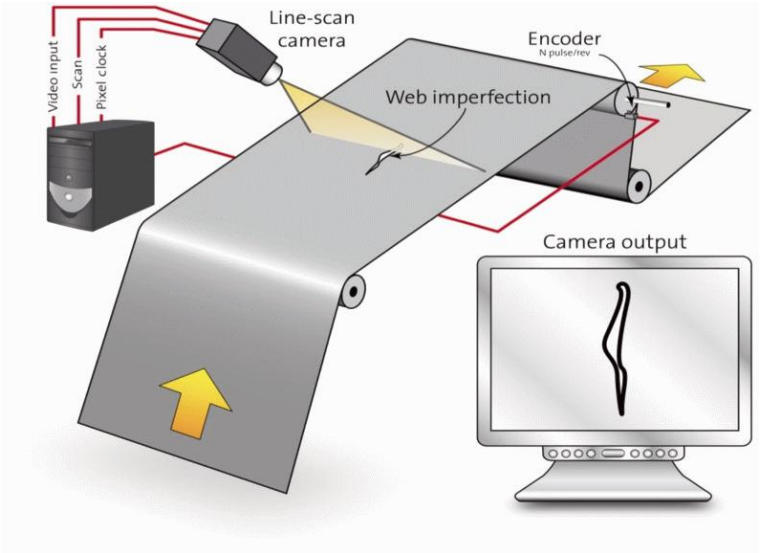


Figura 4.21: Schema della lettura in continuo [40]

4.3.4 Lettura di caratteri e codici

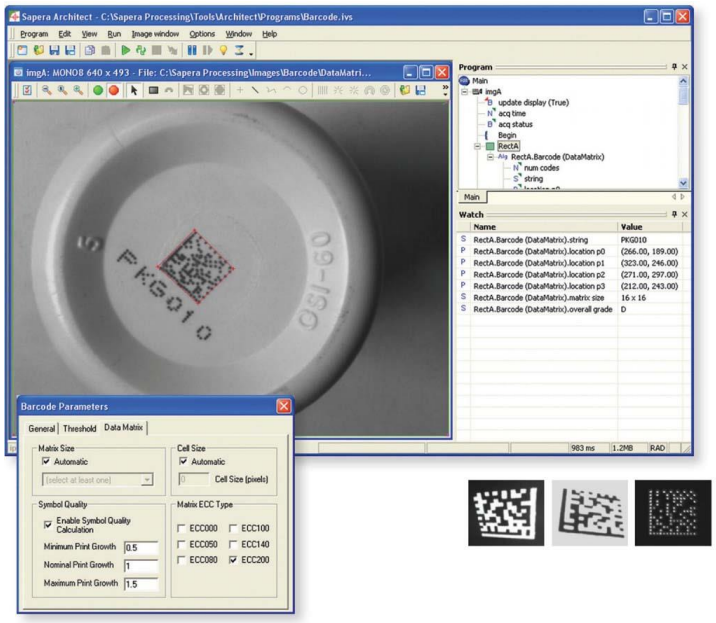


Figura 4.22: Interfaccia di un sistema di riconoscimento di codice a barre [34]

La maggior parte delle aziende impiegano tecnologie di identificazione AIDC (Automatic Identification and Data Capture) per ottenere una migliore visibilità delle proprie attività produttive. Queste tecnologie si basano sull'utilizzo di codici a barre (barcode) e RFID (Radio Frequency IDentification), sebbene vengano adoperati anche sistemi GPS, WSN, Mems, sensori ambientali e soluzioni di riconoscimento vocale.

Questi sistemi:

- evitano l'inserimento manuale di codici;
- consentono l'identificazione e la rilevazione automatica dei dati;
- riducono gli errori ed i tempi di inserimento.

Inoltre, soventemente, fanno parte dei sistemi MES (Manufacturing Execution System) ed ERP (Enterprise Resource Planning) negli ambiti in cui la produzione e la logistica richiedono l'implementazione di soluzioni integrate e allo stato dell'arte.

4.3.5 Applicazioni in altri campi

Qui sotto sono elencati alcuni esempi delle applicazioni di visione artificiale che si possono trovare in ambito industriale e non solo:

- controllo veicoli autonomi;
- video sorveglianza;
- organizzazione di informazioni;
- modellazione di oggetti o ambienti;
- interazione tra l'uomo e il processo;

Un altro campo applicativo è quello dell'elaborazione dei segnali. Difatti, diverse procedure che servono per processare segnali ad una variabile si possono estendere a segnali a 2 variabili in modo abbastanza semplice. Nondimeno, non tutte le tecniche utilizzate per le immagini si prestano ad essere adoperati su segnali a singola variabile.

4.3.6 Nuove tendenze

Dopo l'anno 2000 hanno iniziato ad affermarsi dei sistemi di visione compatti chiamati *smart-camera*. Fondamentalmente non c'è nessuna variazione qualitativa rispetto ai sistemi basati su PC, ma solo una maggiore praticità.

Dopo l'anno 2005 compaiono i primi *sistemi integrati di visione* che utilizzano software ad alto livello che si serve di *logiche ibride*. Questi sistemi di visione hanno un alto livello di adattabilità agli eventi esterni che li rendono molto più affidabili e versatili.

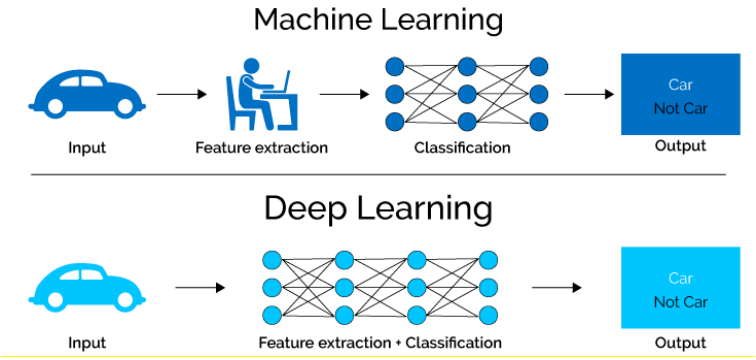


Figura 4.23: Deep Learning vs Machine Learning [41]

Machine Learning

Questo metodo è utilizzato per lo sviluppo di macchine automatizzate mediante l'esecuzione di algoritmi e set di regole bene definiti.

Nel Machine Learning i dati per l'allenamento della macchina e il set di regole che esegue l'algoritmo sono forniti dall'uomo.

Invece, l'esecuzione dell'algoritmo può avvenire senza alcun'interferenza umana. Esso trasforma i dati in patterns ed è in grado di analizzare il sistema in profondità per individuare il problema nella produzione automaticamente.

Deep learning

Gli algoritmi di Deep Learning imparano il compito affidatogli attraverso una rete neurale artificiale che mappa il compito stesso in modo gerarchico. Ogni complessità è definita da una serie di concetti più semplici. Nel contesto della Computer Vision, questo significa identificare in maniera gerarchica prima alcune caratteristiche dell'immagine, ad esempio le aree chiare e scure, poi categorizzare le linee, poi le forme prima di procedere verso un riconoscimento dell'immagine completa.

Già da qualche anno le soluzioni basate sull'intelligenza artificiale hanno cominciato a prendere piede, perché per addestrare reti neurali

convoluzionali (CNN) basta sfruttare algoritmi di apprendimento e immagini campione dell'applicazione specifica a cui si vuole applicare.

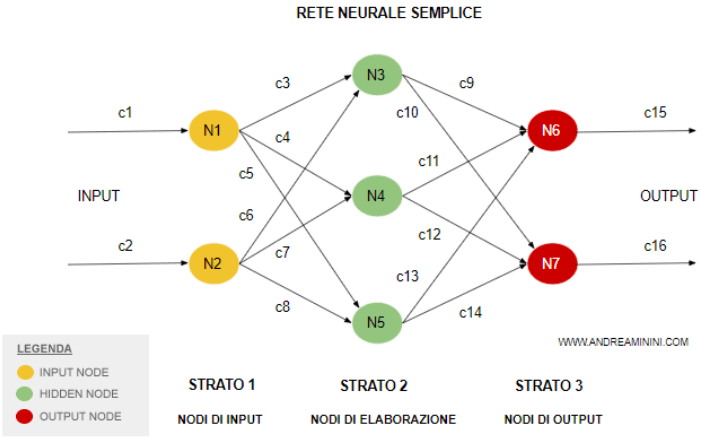


Figura 4.24: Schema di rete neurale semplice [33]

4.4 La visione stereoscopica artificiale

La visione stereoscopica artificiale permette l'elaborazione di informazioni 3D da immagini digitali, come quelle ottenute da una camera CCD. Confrontando le informazioni di una scena da due punti di vista, l'informazione 3D può essere estratta esaminando le posizioni relative degli oggetti nei due schermi. Questo è simile a ciò che avviene nel processo biologico della stereopsia.

Questi sensori che sono in grado di fornire informazioni tridimensionali sono largamente impiegati nei campi come la robotica, per estrarre informazioni sulla posizione degli oggetti 3D e per il riconoscimento di oggetti, dove l'informazione di profondità permette al sistema di separare gli oggetti sovrapposti.

4.4.1 Schema di funzionamento

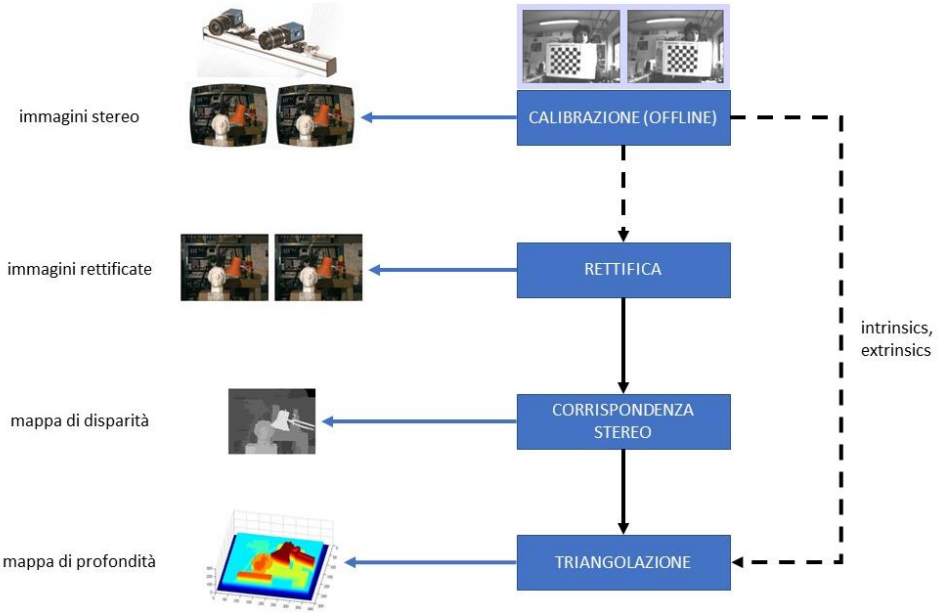


Figura 4.25: Diagramma di flusso della visione stereoscopica

Nella visione stereoscopica, si usano due telecamere disposte in posizioni e angolazioni diverse in modo da ottenere visuali diverse dello stesso scenario. Confrontando queste due immagini, l'informazione di profondità relativa può essere ottenuta sotto forma di una mappa di disparità, che mostra la differenza, in coordinate, per i punti corrispondenti. I valori in questa mappa di disparità sono inversamente proporzionali alla profondità del pixel.

In pratica, la visione stereoscopica si realizza in tre fasi principali:

1. rettifica;
2. point matching;
3. triangolazione.

Per eseguire le prime due fasi, prima si devono calcolare i parametri intrinseci ed estrinseci delle telecamere, il che può avvenire automaticamente tramite una calibrazione.

4.4.2 Calibrazione

Questa procedura mira a trovare:

- parametri intrinseci delle due telecamere (lunghezza focale, centro dell'immagine, parametri di distorsione delle lenti, ecc.);
- parametri estrinseci che legano le due telecamere (la matrice della trasformazione che permette l'allineamento delle due telecamere).

Figura 4.26: Riconoscimento della scacchiera di calibrazione da diverse angolazioni [42]

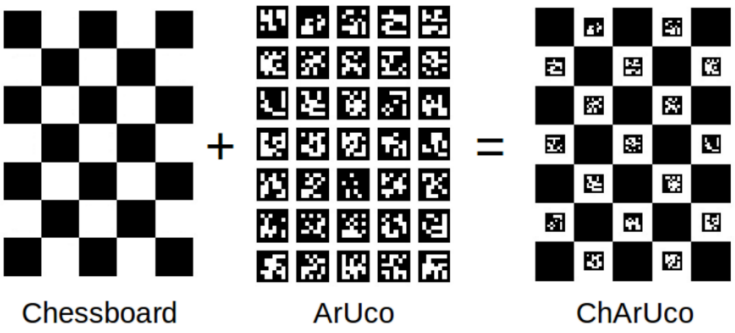


Figura 4.27: Definizione dei markers ArUco e ChArUco [43]

La calibrazione avviene previa acquisizione di pattern conosciuti di 10 o più punti. Di solito si utilizzano scacchiere o vari markers binari,

come ArUco o ChArUco. Il vantaggio principale nell'utilizzo di questi markers è che anche un singolo marker può bastare per ottenere la posa della camera in modo affidabile. Inoltre, la codifica binaria interna li rende molto affidabili in quanto permette l'uso di programmi per l'identificazione degli errori e per la loro correzione.

4.4.3 Rettifica

Se si dovessero cercare i punti equivalenti all'interno di una regione 2D, la quantità di pixel da elaborare sarebbe troppo alta e il processo richiederebbe troppo tempo. Per questo si riduce la regione di ricerca ad una linea, detta epipolare.

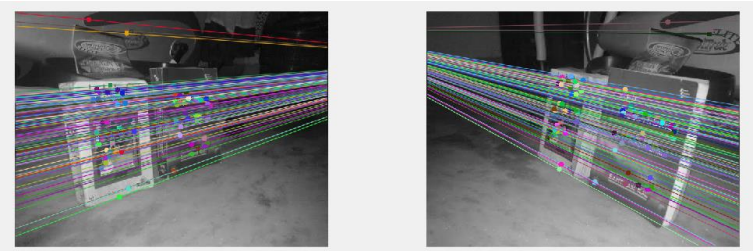


Figura 4.28: Linee epipolari di due immagini estrapolate da Open CV [44]

Il vincolo epipolare impone che, i punti corrispondenti ad un punto appartenente alla linea (rossa) di visione dell'immagine di riferimento (reference) giacciono sulla linea (verde) orizzontale appartenente al piano πT dell'immagine obbiettivo (target).

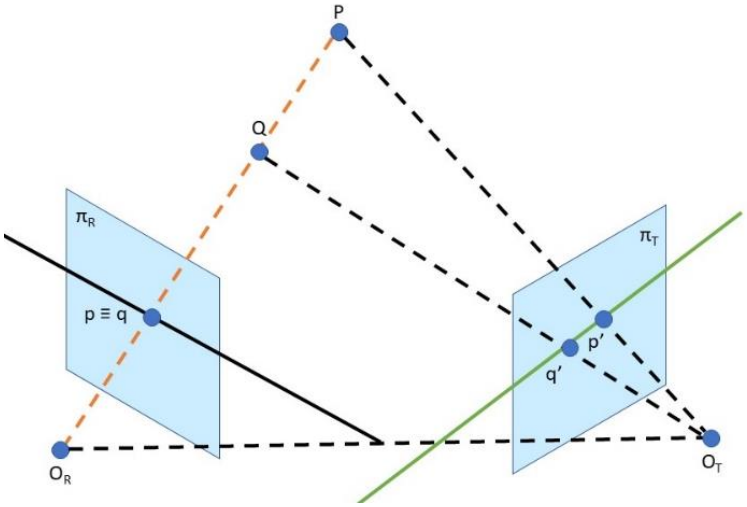


Figura 4.29: Schema del vincolo epipolare

Una volta capito che, lo spazio in cui cercare i punti corrispondenti non è 2D, bensì 1D, si può disporre (virtualmente) l'impianto stereo in una configurazione più conveniente, detta forma standard, in cui i punti corrispondenti sono vincolati alla stessa linea di scansione dell'immagine (linea gialla).

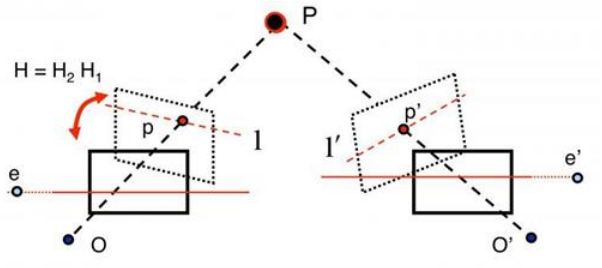


Figura 4.30: Spostamento dei piani delle immagini durante la rettifica [45]

Questo step è fondamentale per facilitare la ricerca dei punti corrispondenti, in quanto si passa dall'avere linee epipolari con

orientazione diversa per ogni immagine, all'avere linee epipolari orizzontali per entrambe le immagini.

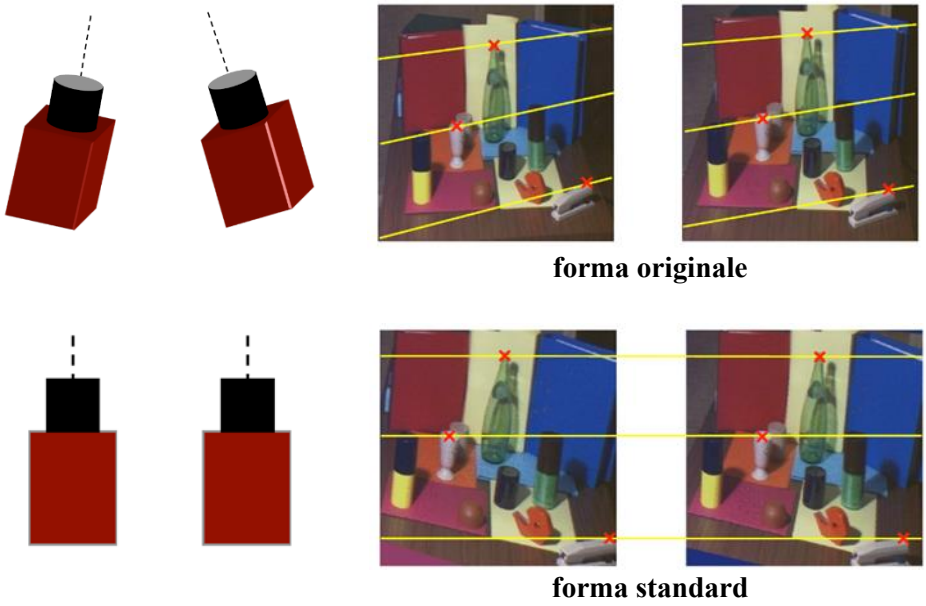


Figura 4.31: Schema dei piani delle immagini e delle linee epipolari prima e dopo la rettifica [46]

Oltretutto, sempre in questa fase, usando le informazioni della fase di calibrazione, si possono rimuovere le distorsioni dovute alle lenti delle singole telecamere.

4.4.4 Point Matching (accoppiamento dei punti)

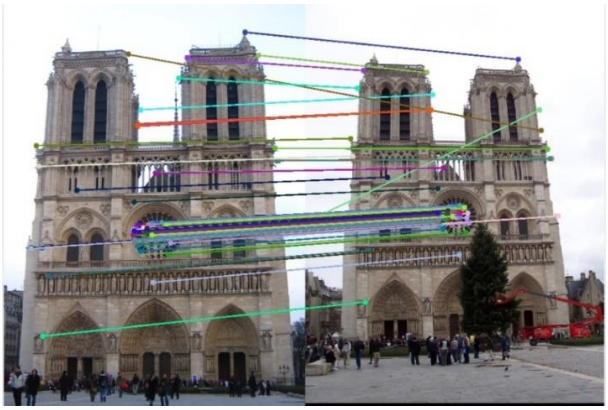


Figura 4.32: Punti corrispondenti nel paio di immagini stereo [47]

La corrispondenza stereo è una delle tematiche più investigate della computer vision. La sua evoluzione è continua e, ancora oggi, vengono rilasciati periodicamente nuovi algoritmi adatti ad utilizzi generici o specifici.

Questa fase ha l'obbiettivo di trovare i punti equivalenti nel paio di immagini rilevate. Per farlo si eseguono le operazioni seguenti:

1) calcolo del costo di accoppiamento;
2) aggregazione del costo;
3) calcolo della disparità;
4) ottimizzazione della disparità;
5) raffinamento della disparità;

All'inizio del processo viene ipotizzato che lo scenario sia interamente morbido, espressione con cui si indica che la profondità varia a tratti con continuità. Nella realtà questa condizione non è sempre rispettata.

Pre-processing

Spesso la fase della corrispondenza dei punti è preceduta da una fase di pre-processing in cui avvengono gli aggiustamenti seguenti:

- applicazione del filtro Laplaciano di Gaussian;
- sottrazione dei valori medi calcolati nei pixel circostanti;

- applicazione del filtro bilaterale;
- applicazione della trasformazione Census.

Questi interventi sono necessari per risolvere dei problemi visivi, tra i quali:

- distorsioni fotometriche e rumore
- distorsioni prospettiche
- identificazione di superfici speculari
- identificazione di regioni uniformi
- identificazione di pattern ripetitivi
- rilevamento di oggetti trasparenti
- presenza di occlusioni e discontinuità

Calcolo del costo di accoppiamento

Per ogni pixel si può calcolare un costo, ovvero un valore che misura quanto è diverso rispetto ai candidati nella seconda immagine. I metodi di calcolo utilizzati spaziano dalla semplice somma di differenze assolute, che considera le intensità di grigio, a combinazioni più complesse e affidabili che servono a ridurre la presenza di errori.

Costi di accoppiamento Pixel-based

Questi costi sono valutati su un punto dell'immagine di riferimento e sull'omologo dell'immagine obbiettivo.

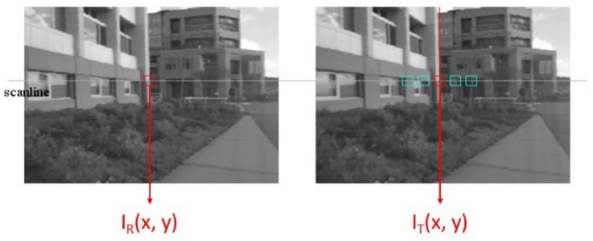


Figura 4.33: intensità di due punti candidati ad essere omologhi [48]

Differenze assolute

$$e(x,y,d) = |I_R(x,y) - I_T(x+d,y)| \qquad \textbf{Equazione 4.1}$$

Differenze quadratiche

$$e(x,y,d) = \left(I_R(x,y) - I_T(x+d,y)\right)^2 \qquad \textbf{Equazione 4.2}$$

Truncated Absolute Differences (TAD):

$$e(x,y,d) = min\{|I_R(x,y) - I_T(x+d,y)|, T\} \qquad \textbf{Equazione 4.3}$$

Con T = threshold, da scegliere in base all'applicazione.

Misura di dissomiglianza di Birchfield e Tomasi
Questa misura è insensibile al sampling dell'immagine perché utilizza le funzioni di intensità dei pixel circostanti interpolate linearmente.

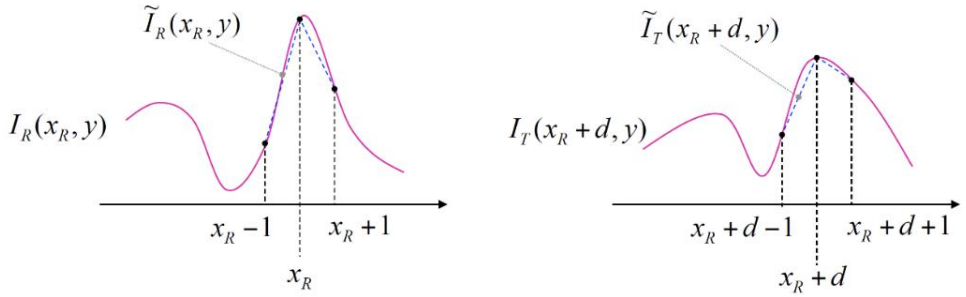


Figura 4.34: Calcolo della dissomiglianza (birchfield and tomasi) [49]

$$e(x_R, y, d) = \min\left\{ \min_{x_R - \frac{1}{2} \le x \le x_R + \frac{1}{2}} \left| I_R(x_R, y) - \tilde{I}_T(x + d, y) \right|, \min_{x_R - \frac{1}{2} \le x \le x_R + \frac{1}{2}} \left| I_T(x_R + d, y) - \tilde{I}_R(x, y) \right| \right\}$$

Equazione 4.4

Costi di accoppiamento Area-based (o Window-based)

Questi costi si valutano su una superficie, centrata su un pixel dell'immagine di riferimento e sulla superficie centrata nel pixel omologo. Ciò permette un calcolo più robusto, e ciò lo rende il metodo maggiormente impiegato.

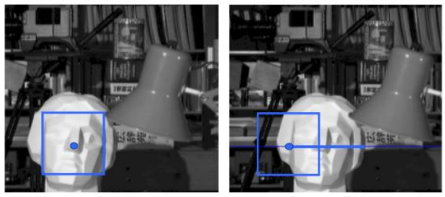


Figura 4.35: Superfici di pixel coinvolte nel calcolo [46]

Sum of Absolute differences (SAD)

$$C(x, y, d) = \sum_{x \in S} |I_R(x, y) - I_T(x + d, y)| \qquad \textbf{Equazione 4.5}$$

Sum of Squared differences (SSD)

$$C(x, y, d) = \sum_{x \in S} \left(I_R(x, y) - I_T(x + d, y) \right)^2 \qquad \textbf{Equazione 4.6}$$

Sum of truncated absolute differences (STAD)

$$C(x, y, d) = \sum_{x \in S} min\{|I_R(x, y) - I_T(x + d, y)|, T\}$$ **Equazione 4.7**

Con T = threshold da scegliere in base all'applicazione

Ogni elemento C(x,y,d) viene inserito nella Disparity Space Image (DSI), andando a creare una matrice 3D (W x H x (dmax - dmin))

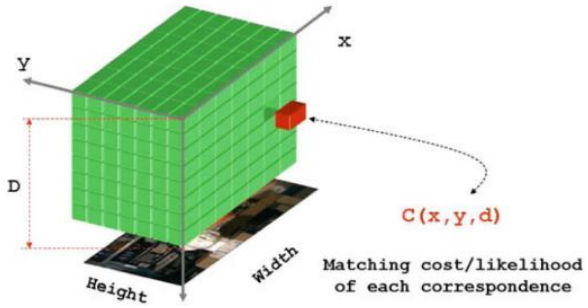


Figura 4.36: rappresentazione della disparity space image (DSI) [50]

Aggregazione del costo

Dopo aver definito i costi per ogni pixel, la fase successiva consiste nel raggrupparli per avere una visione più ampia della zona analizzata.

Fixed Window

Figura 4.37: Aggregazione del costo tramite FW [49]

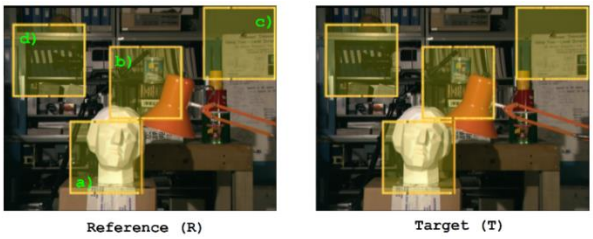


Figura 4.38: Errori del metodo FW [49]

Questa strategia è la più semplice e fallisce in moltissimi punti perché:

a) assume implicitamente che le superfici frontali siano parallele;
b) ignora la discontinuità di profondità;
c) non può gestire correttamente le regioni uniformi;
d) non può gestire correttamente i pattern ripetitivi.

Nonostante le sue limitazioni, la strategia FW è l'algoritmo più frequentemente utilizzato nelle applicazioni reali. Ciò è dovuto a diversi fattori, tra i quali:

- implementazione facile e veloce;
- alta velocità di calcolo grazie a schemi di calcolo incrementali;
- può essere eseguita in tempo reale su processori standard;
- ha requisiti di memoria limitati;
- l'implementazione Hardware può essere eseguita in tempo reale con basso consumo di energia (<1W).

Inoltre, bisogna considerare che si possono implementare delle tecniche di ottimizzazione (ad esempio Integral Images,oppure Box-Filtering) che migliorano notevolmente il risultato finale.

Shiftable Windows
Caratteristiche:

- il supporto non è vincolato ad essere posizionato centralmente (riducendo il problema di localizzazione dei bordi);
- il supporto è simmetrico;
- tempo di esecuzione: 12 sec.

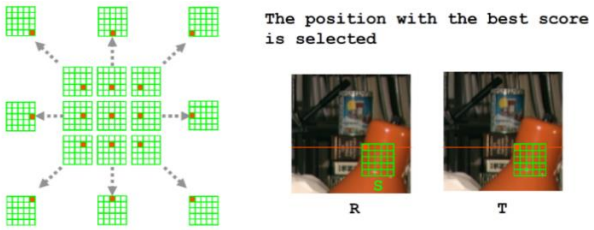
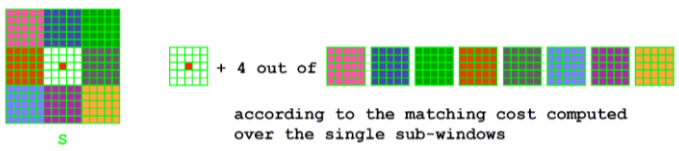


Figura 4.39: Aggregazione del costo tramite shiftable windows [49]

Multiple Windows
Caratteristiche:

- il numero di elementi nel supporto rimane costante;
- la forma del supporto non è vincolata;
- il supporto è simmetrico;
- proposto per 5, 9 e 25 sub-windows (5W, 9W and 25W);
- tempo di esecuzione (9W): 11 sec.



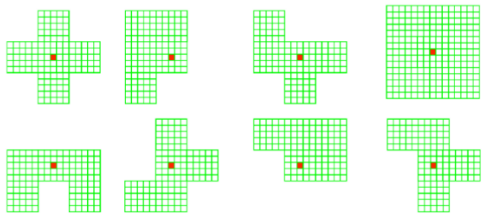


Figura 4.40: Alcune forme componibili con 9 sub-windows [49]

Variable Windows
Caratteristiche:

- impiega la funzione dei costi: Birchfield and Tomasi;
- supporto quadrato con dimensione variabile;
- la posizione del supporto può cambiare;
- il supporto è simmetrico;
- tempo di esecuzione: 16 sec.

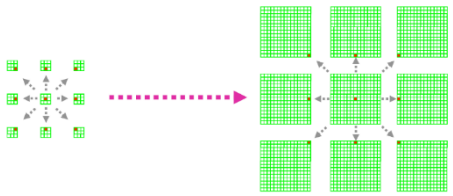


Figura 4.41: Aggregazione dei costi tramite variable windows [49]

Segmentation based
Nella visione stereoscopica, la segmentazione viene spesso utilizzata come step preliminare che consiste nella divisione dell'immagine in regioni composte da pixel vicini con intensità di colore simile.

Caratteristiche:

- segmentazione dell'immagine di riferimento (non simmetrica);
- forma e dimensioni non vincolate (all'interno del supporto maggiore);

- si usa la funzione di calcolo dei costi pixel-based: M-estimator;
- richiede la segmentazione esplicita;
- tempo di esecuzione: 2 sec;

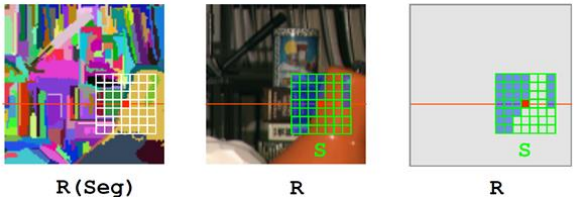


Figura 4.42: Aggregazione dei costi tramite segmentazione [49]

Calcolo della disparità

La disparità è la differenza tra le coordinate x di due punti corrispondenti; di solito è codificata in un'immagine in scala di grigio.

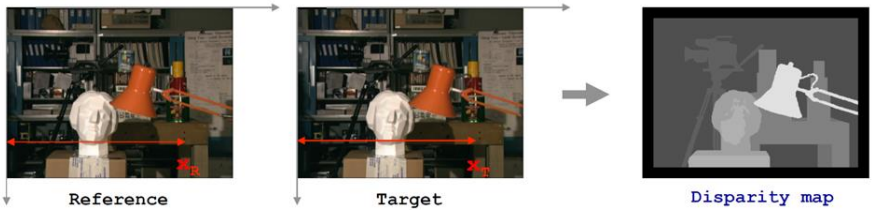


Figura 4.43: Mappa della disparità dato il paio di immagini stereo [49]

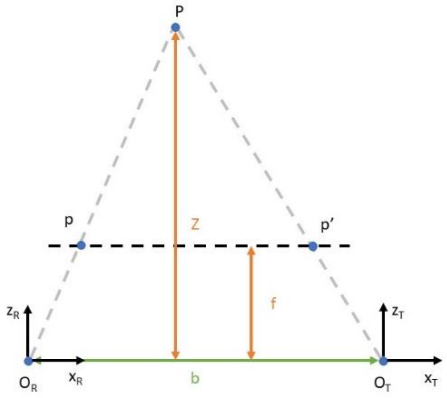


Figura 4.44: Geometria dell'impianto stereoscopico [49]

Portando l'impianto stereo nella forma standard e considerando i triangoli simili PO_RO_T e Ppp', si possono ricavare le seguenti relazioni che legano la profondità alla disparità di due punti corrispondenti:

$$\frac{b}{Z} = \frac{(b+x_T)-x_R}{Z-f} \quad \textbf{Eq. 4.10} \quad \Rightarrow \quad Z = \frac{b*f}{x_R-x_T} = \frac{b*f}{d} \quad \textbf{Eq. 4.11}$$

Con:

- Z = profondità della baseline
- b = lunghezza della baseline
- f = distanza tra i punti e la baseline
- d = x_R-x_T = disparità

Conoscendo alcuni parametri, utilizzando le relazioni sopracitate, è possibile ricavare la curva che rappresenta la relazione tra la distanza e la disparità.

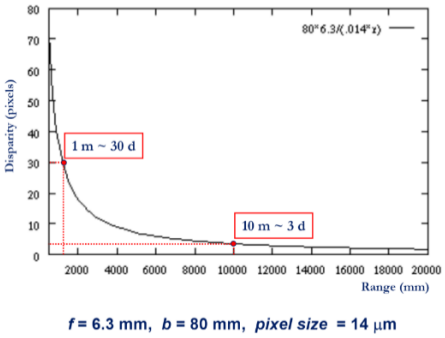


Figura 4.45: Relazione tra distanza e disparità [46]

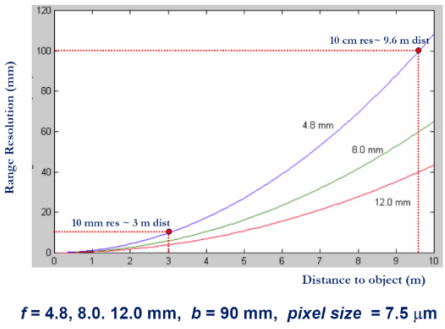


Figura 4.46: Relazione tra risoluzione e distanza [46]

Sempre dalle stesse relazioni, è possibile anche trovare l'andamento della risoluzione, in funzione della distanza. Ciò è particolarmente utile

nell'utilizzo del sistema stereoscopico perché permette di valutare la precisione (ideale) della mappa di disparità.

$$\Delta Z = \frac{Z^2}{b*f} * \Delta d$$ **Equazione 4.12**

Con:

- ΔZ = risoluzione della profondità
- Δd = distanza dall'oggetto

Calcolare la mappa di profondità, dati i punti corrispondenti esatti, non è particolarmente difficile. Il problema è che spesso la fase di point matching contiene degli errori, per cui la qualità della mappa di disparità dovrà essere corretta e/o migliorata utilizzando delle logiche che permettono di trovare e correggere gli errori più evidenti.

Ottimizzazione della disparità

Questa fase mira a trovare l'assegnazione dei valori della disparità che minimizza la funzione del costo sull'intero sottoinsieme di paia di punti stereo. Questa funzione è composta da due termini:

$$E(d) = E_{data}(d) + E_{smooth}(d)$$ **Equazione 4.13**

- Il termine E_{data} misura quanto bene l'assegnazione si adatta al paio stereo (in termini di costi di accoppiamento complessivi).
- Il termine di morbidezza/regolarizzazione E_{smooth} impone esplicitamente assunzioni di continuità a tratti dello scenario. Questo termine penalizza le variazioni di disparità e grosse

variazioni sono permesse solo su bordi di profondità sconosciuti, che di solito sono legati agli spigoli nell'immagine.

Raffinamento della mappa di disparità

Le mappe di disparità contengono degli outliers che devono essere identificati e corretti, inoltre, potrebbe essere necessario aumentarne la risoluzione e andare oltre quella dei pixel.

Interpolazione Sub-pixel

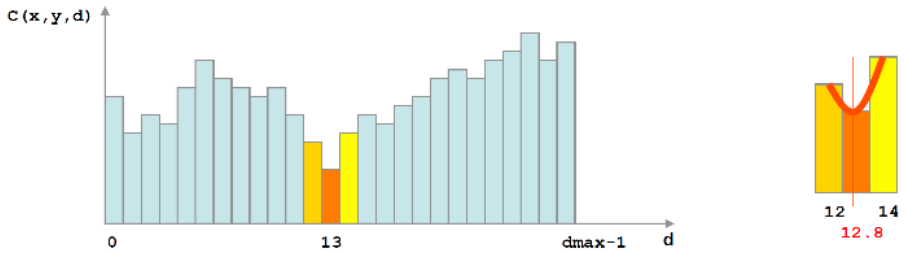


Figura 4.47: Raffinamento tramite interpolazione sub-pixel [49]

Tipicamente, si può ottenere disparità a livello sub-pixel interpolando tre costi di accoppiamento vicini con una funzione di secondo grado (parabola). Ciò risulta essere computazionalmente non dispendioso e ragionevolmente accurato.

Tecniche di filtraggio delle immagini

A volte le mappe di disparità sono rifinite semplicemente tramite tecniche di filtraggio senza imporre esplicitamente alcun vincolo.

Alcune operazioni di filtraggio comuni sono:

• filtraggio mediano;

• operatori morfologici;

• filtraggio bilaterale.

Raffinamento mediante segmentazione
Ogni segmento è modellato come un piano nello spazio 3D (3 DOF):

$$d(x,y) = \alpha \cdot x + \beta \cdot y + \gamma$$ **Equazione 4.15**

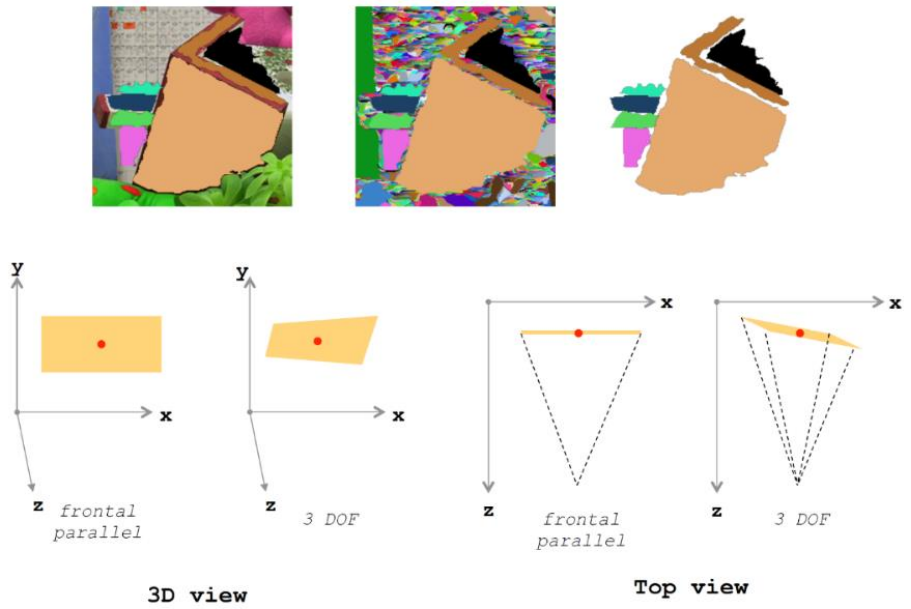


Figura 4.48: Modellazione di un segmento per il riconoscimento degli outliers [49]

4.4.5 Triangolazione

Data la mappa di disparità, la linea di base e la lunghezza focale (dalla calibrazione): la triangolazione calcola la posizione dei punti corrispondenti nello spazio tridimensionale.

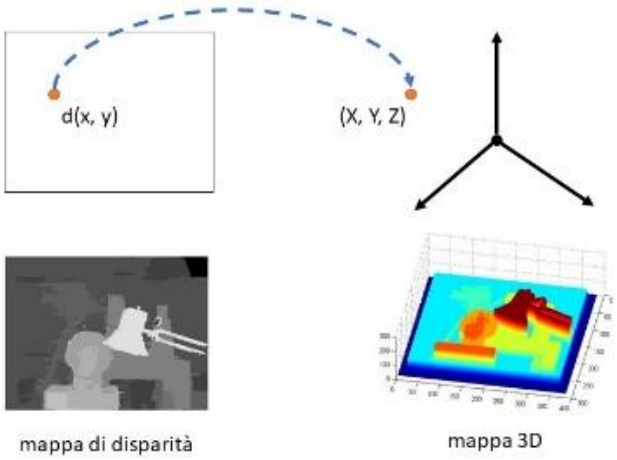


Figura 4.49: Passaggio dalla mappa di disparità a quella di profondità (3D)

Le coordinate di ogni punto si possono trovare utilizzando le seguenti formule:

$$Z = \frac{b*f}{d}$$ **Equazione 4.16**

$$X = Z * \frac{x_R}{f}$$ **Equazione 4.17**

$$Y = Z * \frac{y_R}{f}$$ **Equazione 4.18**

Ricordando che:

- Z = profondità della baseline
- b = lunghezza della baseline
- f = distanza tra i punti e la baseline
- d = x_R-x_T = disparità

4.5 Configurazione delle telecamere realsense

Come anticipato nel capitolo precedente, la configurazione delle telecamere deve avvenire servendosi della piattaforma Realsense SDK 2.0.

Le proprietà delle telecamere possono essere modificate mediante facilmente sfruttando delle applicazioni come:

- il RealSense Viewer: che permette la facile visualizzazione degli stream video e di profondità e la configurazione della camera;
- il Depth Quality Tool: che permette agli sviluppatori di testare la qualità della sensibilità in profondità della camera;

In alternativa, è possibile modificare le impostazioni delle telecamere direttamente dal programma scritto in Python, sfruttando la libreria pyrealsense2.

Nei paragrafi seguenti si spiegherà come sono state modificati alcuni settaggi della telecamera in modo tale da ottenere stream di colore e di profondità di qualità più elevata. Inoltre, quando necessario, ci si soffermerà a spiegare come funziona la telecamera, in modo tale da capire come variano certi parametri come il limite di precisione teorico ed il MinZ (profondità minima rilevabile)

4.5.1 Risoluzione dell'imager di profondità

Per la telecamera D435, la risoluzione che porta a miglior precisione nella mappa di profondità è la: 848x480. Quindi, questa è stata la risoluzione scelta per lo stream delle nostre telecamere. Di seguito si riporta cosa potrebbe accadere nel caso si apportassero delle modifiche. Risoluzioni più basse possono essere utilizzate ma ridurranno la precisione in profondità. L'accuratezza dei sensori di profondità stereoscopici dipende dall'abilità di accoppiare oggetti nelle immagini sinistra e destra. Maggiore la risoluzione di input, migliore l'immagine di input, migliore la precisione in profondità.

Se l'applicazione richiede una risoluzione minore per ragioni computazionali, si raccomanda l'utilizzo di algoritmi di post-processing per effettuare subsample (decimate) sulla mappa di profondità e sull'immagine a colori ad alta risoluzione dopo che i dati sono stati ricevuti. Le uniche ragioni che potrebbero obbligare l'utilizzo delle camere in modalità a bassa risoluzione sarebbero:

- Ridurre al minimo il campo operativo che scala con la risoluzione x-y
- Ridurre la larghezza di banda sul bus USB 3.0, per permettere a più camere di operare allo stesso tempo senza superare il limite di larghezza di banda

4.5.2 Esposizione

Bisogna controllare che l'auto-esposizione funzioni correttamente, o passare all'esposizione manuale per essere sicuri di ottenere una buona distribuzione del colore nell'immagine. Una cattiva messa a punto in questo campo può portare a basse performance.

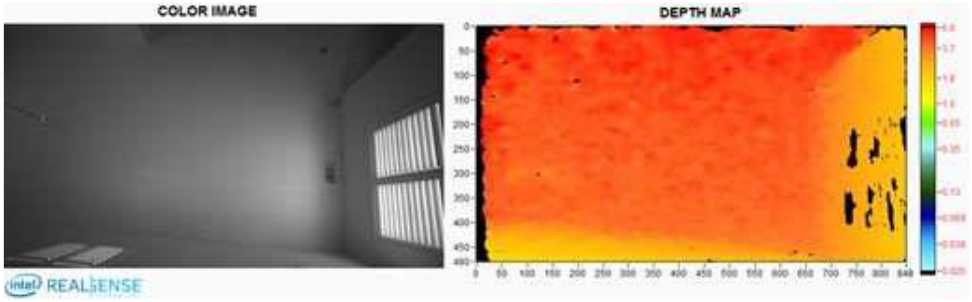


Figura 3.12: Esempio corretto [24]

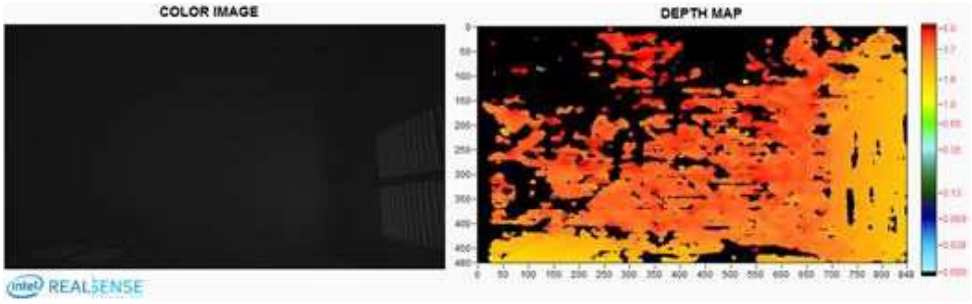


Figura 3.13: Esempio errato (esposizione troppo bassa) [24]

Per ottenere dei risultati migliori, bisogna aggiustare l'esposizione prima del GAIN, mantenendo il GAIN = 16 (valore minore). Aumentare il GAIN tende ad introdurre rumore e, mentre l'immagine a colori può sembrare migliore, la qualità della profondità diminuirà. Le unità di esposizione sono in microsecondi. Sovra esporre le immagini può essere altrettanto dannoso quanto sotto esporle, quindi bisogna stare attenti a trovare il valore di esposizione corretto.

Ci sono altre due opzioni da considerare quando si utilizza la feature di auto-esposizione. Quando l'auto-esposizione è accesa, medierà l'intensità dei pixel all'interno di una regione d'interesse (Region-Of-Interest (ROI)) e proverà a mantenere questo valore a un punto predefinito (Setpoint). Sia il ROI che il Setpoint sono impostabili tramite software.

Nel caso dell'applicazione in questione, si è deciso di mantenere GAIN =16, e l'esposizione è stata regolata in modo manuale, cercando di ottenere una depthmap con meno "buchi" possibile.

4.5.3 Verifica delle prestazioni

Nel nostro caso si è verificato fin da subito che le telecamere avevano un errore veramente piccolo a distanze inferiori a 500mm, quindi non c'è stato bisogno di effettuare alcuna procedura di ri-calibrazione. Comunque, è stato molto utile capire il limite teorico delle telecamere

e l'andamento dell'errore quadratico medio mostrato nel grafico sottostante.

Limite di precisione teorico

L'errore RMS rappresenta l'errore quadratico medio sui valori di profondità di un piano ad una determinata distanza:

$$Depth\ RMS\ error = \frac{Distance^2 * Subpixel}{focal\ length * Baseline}$$ **Equazione 3.1**

$$focal\ length = \frac{1}{2} * \frac{Xres}{tan\left(\frac{HFOV}{2}\right)}$$ **Equazione 3.2**

Con:

- Depth RMS error = errore quadratico medio in mm
- Distance = distanza in mm
- focal length = lunghezza focale in pixel
- Xres = risoluzione in X in pixel
- HFOV = campo visivo orizzontale in gradi

Per un obbiettivo ben texturato ci si può aspettare di ottenere Subpixel < 0,1 o addirittura il raggiungimento di 0,05.

La curva seguente si ottiene utilizzando:

- D415 con HFOV=65deg, Xres=1280, baseline=55mm.
- D435 con HFOV=90deg, Xres=848, baseline=50mm.

Entrambe con subpixel=0.08.

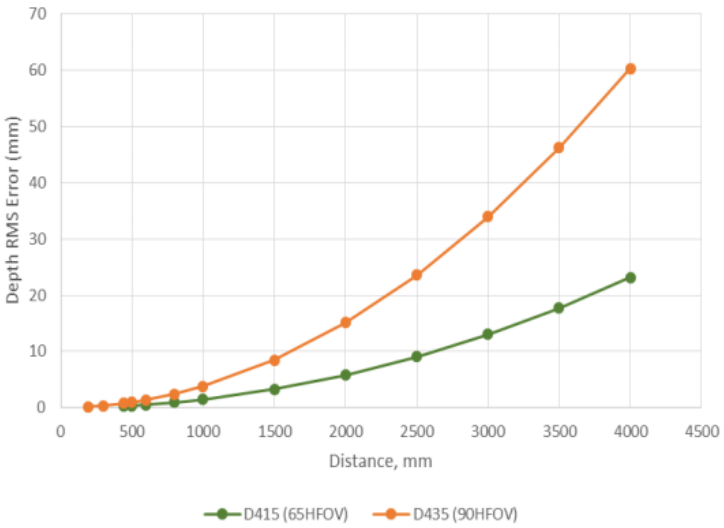


Figura 3.14: Andamento dell'errore quadratico medio [24]

4.5.4 Variazione delle prestazioni in funzione della distanza

L'errore sulla profondità scala con il quadrato della profondità. Quindi, bisogna provare a posizionare le telecamere più vicino possibile agli oggetti, ma non così vicino da oltrepassare il limite di distanza minimo per il funzionamento, ovvero il MinZ.

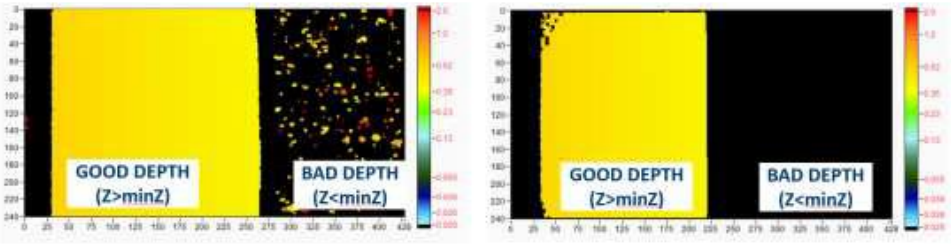


Figura 3.15: "High Density" vs "High Accuracy" [24]

Guardando al piano inclinato: “MinZ Noise” può essere ridotto utilizzando le pre-impostazioni di profondità: LEFT = “High Density”, RIGHT = “High Accuracy”

Nel caso dell’applicazione sviluppata, non è stato necessario ridurre il MinZ, in ogni caso, di seguito riporteremo come fare, soprattutto per capire più a fondo il funzionamento delle telecamere.

Se il MinZ diventa un problema (il che significa c’è la necessità di stare più vicino del MinZ), esso può essere ridotto diminuendo la risoluzione dei Depth imagers. Il MinZ scala linearmente con la risoluzione in X dei sensori di profondità. Nota che per la D435 a 848x480 il MinZ è ~16.8cm. La formula con cui si può ricavare il MinZ è:

$$MinZ = \frac{focal\ length * Baseline}{126}$$ **Equazione 3.3**

Con:

- focal length = lunghezza focale in pixel
- MinZ =profondità minima misurabile in mm
- Baseline = linea di base in mm

Un approccio alternativo alla riduzione del minZ è la diminuzione del parametro “Disparity shift” nell’API Advanced Mode. Normalmente il sensore di profondità è impostato per misurare la profondità di oggetti a distanze che vanno da minZ fino all’infinito. Se questo parametro venisse ridotto, non si potrebbero più misurare oggetti più lontani di MaxZ. Si noti che, quando viene aumentato il Disparity shift, MaxZ aumenta molto più velocemente del MinZ, a causa della relazione inversa tra la profondità e la disparità.

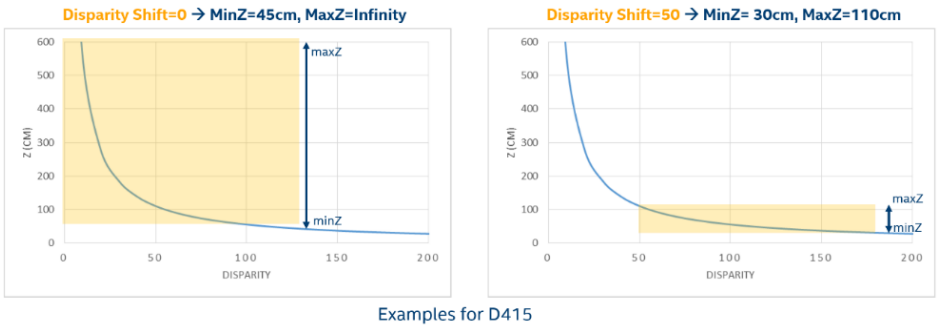


Figura 3.16: Andamento di minZ in funzione del disparity shift per la telecamera D415 [24]

4.5.5 Utilizzo di camere di profondità multiple

Per osservare l'intera superficie di una singola mela si è provato ad avvalersi di 2 telecamere, in quanto ognuna può essere in grado di vedere, solo una porzione di essa. Successivamente, dopo aver rilevato le pointcloud di diverse mele, si è scoperto che per ricavare la superficie completa della mela, sarebbe stato necessario aggiungere una telecamera ulteriore.

Il miglior modo per utilizzare molte camere di profondità è utilizzare ognuna in modo indipendente, ognuna con il proprio thread di cattura e post processing, e poi combinarle dopo che si sono calcolate le pointclouds. Ciò è esattamente il processo che è stato seguito nella creazione del programma.

4.5.6 Post-processing

Di default non si applica alcun post-processing della profondità, ciò viene lasciato fare ad apps di alto livello. Sono state aggiunte alcune semplici opzioni di post-processing nella RealSense SDK 2.0, ma non sono esaustive. Qua sotto si descrivono i passi di post-processing più utilizzati e quelli che sono stati utilizzati nell'applicazione in esame.

Sub-sampling
Di solito si consiglia di fare una media a nonzero per un pixel ed i suoi vicini. Tutti gli algoritmi stereo coinvolgono alcune operazioni di convoluzione, quindi ridurre la risoluzione (X,Y) dopo la cattura di solito è molto benefico per la riduzione dei calcoli richiesti alle app di alto livello.

Un fattore di 2 nella riduzione della risoluzione velocizzerà il processo di 4 volte, ed un fattore di 4 ridurrà i calcoli di 16 volte. Inoltre, il subsampling può essere utilizzato per fare un rudimentale hole-filling e smoothing dei dati utilizzando una media nonzero oppure una funzione a media non zero. Infine, il subsampling tende a velocizzare la visualizzazione della point-cloud.

Questo tipo di operazione si sarebbe rilevata utile per velocizzare l'acquisizione della mappa di profondità, soprattutto quando le mele si trovano in movimento sul nastro trasportatore. Invece, non si è potuto procedere in questo modo perché la precisione sulle coordinate che si sarebbe ottenuta sarebbe stata troppo bassa.

Temporal filtering
Quandunque sia possibile, conviene utilizzare la mediazione temporale per migliorare la mappa di profondità. Ciò è dovuto al fatto che è sempre presente del rumore temporale nei dati. Si consiglia l'utilizzo di un filtro IIR. In alcuni casi potrebbe essere benefico l'uso di "persistence", dove l'ultimo valore valido è ritenuto indefinitamente o all'interno di un certo frame di tempo. Questo filtro è stato utilizzato nella fase di calibrazione delle telecamere, perché questa operazione ha bisogno che la mappa di profondità sia la più precisa possibile, per cui anche il rumore temporale sarebbe potuto essere un problema.

Edge-preserving filtering
Questo ammorbidisce il rumore di profondità, mantenendo gli spigoli ma rendendo le superfici più piatte. Comunque, bisogna fare attenzione ad utilizzare parametri che non rimuovono gli elementi in modo troppo

aggressivo. Si consiglia di sperimentare con questo filtro incrementando gradualmente lo step size threshold fin quando sembra funzionare meglio per l'utilizzo voluto.

Questo filtro non è stato utilizzato perché le mele hanno una superficie morbida, ovvero priva di spigoli, per cui avrebbe avuto poco effetto.

Hole-filling

Alcune applicazioni non tollerano buchi nella profondità. Per esempio, nella fotografia a profondità migliorata è importante avere valori di profondità per ogni pixel, anche se è solo una stima. Per questo, diventa necessario riempire i buchi con la stima migliore basata sui valori vicini, oppure sull'immagine RGB.

Per la nostra applicazione si è potuto applicare questo un filtro sia nella fase di calibrazione che nella fase del rilevamento delle coordinate delle mele in movimento. A differenza del filtro temporale, infatti, questo filtro è "spaziale" per cui è adatto ad essere applicato anche a oggetti in movimento.

4.5.7 Effetto della luce ambientale

La maggior parte delle telecamere di profondità presenta problemi a funzionare alla luce del sole. Invece, le Dxx tendono a comportarsi ancora meglio nella luce chiara. Ciò è dovuto al fatto che la qualità della profondità nelle telecamere D4xx è direttamente legata alla qualità delle immagini di input. La luce del sole riduce il rumore del sensore e tende a tirar fuori le texture negli oggetti. Inoltre, l'esposizione può essere ridotta fino a circa 1ms il che riduce i problemi legati al movimento.

In realtà si è visto che queste telecamere sono molto sensibili ai fasci di luce, anche quando non sono particolarmente concentrati, quindi l'esposizione alla luce solare deve essere sempre tenuta sotto controllo.

CAPITOLO 5

5. Sviluppo software

In questo capitolo si andrà a descrivere il processo di sviluppo seguito nell'ideazione del software di gestione delle telecamere.

Per prima cosa è stato necessario definire il problema da risolvere. In questo caso, si tratta di sostituire un sistema in grado di fare una scansione completa delle mele in modo da poter fornire al robot le informazioni corrette di posizionamento ed orientazione per poterle afferrare tramite una ventosa abbastanza grande.

Al posto di utilizzare un sistema di scansione laser, avremmo dovuto usare un paio di telecamere stereoscopiche che, prima di operare, necessitano di una fase di calibrazione e possono essere configurate e programmate utilizzando diversi wrappers, tra cui Python, .NET, Node.js, Robot Operating System (ROS), LabView, ecc .

Oltre alle telecamere, avremmo dovuto utilizzare delle fotocellule in modo da catturare le immagini 3D delle mele, in modo ordinato, una alla volta.

Inoltre, bisogna tener presente che l'operatore deve essere in grado di gestire il sistema, quindi è assolutamente necessario disporre di un'interfaccia grafica che permetta di:

- visualizzare i risultati, ovvero le immagini 3D delle mele;
- interagire con il processo tramite pulsanti e bottoni.

Dato ciò, risulta necessario creare un programma utilizzando un linguaggio in grado di comunicare sia con le telecamere che con le fotocellule e che sia adatto alla creazione di interfacce grafiche.

Non solo, vista la mia inesperienza, il linguaggio di programmazione scelto, oltre ad essere in grado di operare in queste condizioni, deve anche essere abbastanza veloce da imparare da zero.

Conseguentemente il mio dottorando mi ha suggerito di imparare Python, che è molto impiegato nel campo dell'automazione ed è anche molto semplice da utilizzare per i principianti.

Scelto il linguaggio di programmazione, si è installato il relativo IDE (Integrated Development Environment) per poter compilare e debuggare il programma più velocemente.

A questo punto, si è deciso di dividere la fase di sviluppo in due parti:

- Nella prima fase, si sarebbe creato un codice in grado di utilizzare le telecamere per rilevare la posizione degli oggetti all'interno del loro campo visivo;
- Nella seconda fase, si sarebbe creato l'eseguibile in grado di coordinare il codice creato nella fase precedente con il passaggio dei facchini di fronte alle telecamere, servendosi del segnale mandato dalla fotocellula.

Le applicazioni create nelle due fasi necessitano di un'interfaccia grafica per funzionare correttamente, in quanto:

- Nella prima fase l'operatore dovrà cliccare sui punti di cui si vuole conoscere le coordinate;
- Nella seconda fase l'operatore dovrà gestire la sequenza con cui avviare le diverse parti del programma completo.

Quindi è stato necessario installare PyQt, ovvero una libreria grafica per Python che permette di realizzare applicazioni dotate di un'interfaccia grafica. Anche PyQt necessita dell'installazione del relativo IDE, chiamato QtCreator.

Ovviamente, ogni fase di sviluppo è stata accompagnata da una fase di testing per verificare sia la qualità del programma che l'affidabilità e la precisione delle telecamere stereoscopiche.

5.1 Python

Figura 5.1: Logo del linguaggio di programmazione python [42]

Mentre diversi linguaggi sono in declino, la crescita di Python sembra inarrestabile. Quasi il 14% di tutte le domande di Stack Overflow sono taggate "python", e il trend è in aumento. La sua popolarità può essere attribuita a diverse ragioni.

Python è nato negli anni Novanta. Questo non significa solamente che ha avuto tempo di crescere, ma anche che ha acquisito una comunità ampia che lo supporta.

Quindi, se si verificano dei problemi mentre si codifica in Python, ci sono alte probabilità di risolverli velocemente con una singola ricerca su Google. Semplicemente perché qualcuno avrà già incontrato lo stesso problema e avrà scritto qualcosa di utile su come affrontarlo.

Inoltre, il fatto che è stato presente per decadi ha dato ai programmatori il tempo per creare molti tutorial utili e strutturati.

Oltretutto, la sintassi di Python è facile da leggere per un essere umano. Python è un linguaggio di alto livello che riesce ad essere sia semplice che potente. La sintassi e i diversi moduli e funzioni che si trovano all'interno del linguaggio sono intuitivi, e il modo in cui è ideato il linguaggio si basa sul principio del della "sorpresa minore": il programma funziona in modo prevedibile.

Uno degli elementi più importanti di Python è la possibilità di impiegare molteplici paradigmi di programmazione: quello funzionale, ma anche

quello imperativo e quello object-oriented, con supporto all'ereditarietà multipla.

La ragione primaria per cui Python è popolare è la sua produttività rispetto ad altri linguaggi di programmazione come C++ e Java. Esso risulta molto più conciso ed espressivo e richiede meno tempo, sforzo e linee di codice per eseguire le stesse operazioni.

Il tempo di sviluppo più corto permette di risparmiare soldi ed aumentare la produttività, inoltre aumenta la competitività dell'azienda in quanto garantisce una velocità di prototipazione ed innovazione maggiore.

Siccome Python è stato presente per tutto questo tempo, gli sviluppatori hanno creato pacchetti per qualsiasi scopo. Ad esempio:

- Numpy: per elaborare numeri, vettori e matrici;
- SciPy: per calcoli ingegneristici;
- Pandas: per manipolare ed analizzare grandi quantità di dati;
- Scikit-Learn: per creare un'intelligenza artificiale.

Questo mette Python in cima alla lista dei recenti sviluppi nel campo dell'automazione, e ciò si nota soprattutto nel campo del Machine Learning, negli anni più recenti.

Purtroppo, Python presenta anche degli svantaggi. Python è veramente molto lento. In media, si ha bisogno di 2-10 volte tanto tempo per completare un compito con Python rispetto a qualsiasi altro linguaggio.

Ci sono diverse ragioni per questo. Una di esse è che è stampato dinamicamente il che significa che c'è bisogno di utilizzare molta memoria, e l'utilizzo di molta memoria si traduce in maggior tempo computazionale.

Un'altra ragione è che Python può eseguire sono un compito alla volta a causa del GIL (Global Interpreter Lock). In confronto, un comune web browser può eseguire una dozzina di processi differenti contemporaneamente.

Ma alla fine nessuno di questi problemi di velocità hanno vera importanza. I computers ed i servers sono diventati così veloci che comunque il tempo di esecuzione non supererà le frazioni di secondo.

In Python si utilizzano spazi bianchi ed indentazioni per indicare diversi livelli di codice. Ciò lo rende bello da vedere e facile da capire.

Altri linguaggi, come per esempio il C++, si affidano, invece, a parentesi e punto e virgola. Questo potrebbe non essere bello da vedere o semplice da usare ma rende il codice più facile da mantenere. Ciò è particolarmente utile per progetti più grandi.

Linguaggi più nuovi come Haskell risolvono questo problema: si affidano a spazi bianchi, ma offrono una sintassi alternativa per gli utenti che vogliono continuare senza.

Uno script di Python non viene prima compilato e poi eseguito. All'interno di Python si compila ogni volta che si esegue, quindi qualsiasi errore nel codice si manifesta durante il runtime. Ciò porta a basse prestazioni, consumo di tempo, ed il bisogno di molti più test.

Ciò è positivo per i principiati perché il testing può insegnare molto. Ma per sviluppatori esperti, dover debuggare un programma complesso in Python è piuttosto tedioso.

In conclusione, Python è ottimo per scrivere il codice dell'applicazione, per la sua versatilità e la sua facilità di utilizzo.

Però prima di utilizzarlo è stato necessario installare nel computer un ambiente di sviluppo adeguato, ovvero PyCharm.

5.2 Pycharm

Figura 5.2: Logo di PyCharm [43]

Per eseguire i programmi di Python, si ha il bisogno di un IDE (Integrated Development Environment). Un IDE consiste di un editor ed un compilatore che servono a scrivere e compilare programmi. In pratica, possiede una combinazione di elementi necessari allo sviluppo del software.

La presenza di un IDE rende il processo di sviluppo e di programmazione molto più semplice. Interpreta ciò che si sta scrivendo e suggerisce le parole chiave da inserire. Si possono distinguere le classi dai metodi in quanto l'IDE assegna loro colori diversi. L'IDE fornisce colori diversi anche per le parole chiave giuste e quelle sbagliate. Se si scrive una parola chiave sbagliata prova a predire la parola chiave corretta e completa il testo automaticamente.

Le ragioni principali per l'utilizzo di un IDE per lo sviluppo sono:

- un IDE possiede una finestra di text editor in cui si possono scrivere i programmi;
- presenta una finestra di project editor dove si possono immagazzinare i files necessari per il progetto del software;
- si possono fornire diversi input e controllare l'efficienza del programma ispezionando l'output che si riceve sulla finestra di output;
- se si presenta un errore, l'IDE mostra degli avvertimenti e dei suggerimenti sulla finestra di output in modo da risolverli più facilmente;

- un IDE ha un set di moduli e pacchetti che aiutano ad aggiungere elementi alle applicazioni software;
- aiuta ad aumentare l'efficienza nella creazione del software.

L'IDE più popolare e ampiamente utilizzato per la programmazione e lo sviluppo di applicazioni in Python è PyCharm. Esso può essere installato su Windows, Linux, oppure Mac OS. In aggiunta, contiene moduli e pacchetti che aiutano a sviluppare software in meno tempo e con minor sforzo.

L'installazione di PyCharm non richiede alcuno sforzo. Basta scaricare l'applicazione di set-up dal sito ufficiale cliccando sul link:

www.jetbrains.com/pycharm/download/#section=windows

In conclusione, tra gli ambienti di sviluppo disponibili, è stato scelto PyCharm perché presenta i seguenti vantaggi:

- installare PyCharm è molto semplice;
- è un IDE facile da utilizzare;
- ci sono molti plugins utili e scorciatoie produttive in PyCharm;
- integra librerie ed elementi IDE come l'auto-completamento e la colorazione;
- permette la visualizzazione del codice sorgente in un click;
- lo sviluppo del software è molto più veloce;
- l'elemento di identificazione dell'errore nel codice migliora ulteriormente il processo di sviluppo;
- la comunità degli sviluppatori di Python Developers è molto grande per cui si possono risolvere i problemi/dubbi facilmente.

5.3 Prima fase di sviluppo del programma

Per prima cosa è stato necessario sviluppare un programma che fosse in grado di eseguire:

- la calibrazione delle telecamere stereoscopiche;
- la rilevazione delle coordinate 3D di un punto a scelta dello scenario.

In questo modo, si potrà verificare il corretto funzionamento delle telecamere e si potrà testare la loro precisione sul campo.

Per farlo è stato necessario creare un'interfaccia grafica rudimentale in quanto una parte della calibrazione e l'intera seconda operazione necessita di collaborazione da parte dell'operatore.

Questa fase è stata fondamentale per impostare in modo definitivo la calibrazione delle telecamere. Questa operazione, infatti, rimarrà tale anche nelle fasi di sviluppo seguenti.

La calibrazione è stata divisa in due parti principali:

- nella prima parte, le due telecamere devono rilevare, tramite la scacchiera, il piano di riferimento su cui verranno appoggiati gli oggetti da osservare;

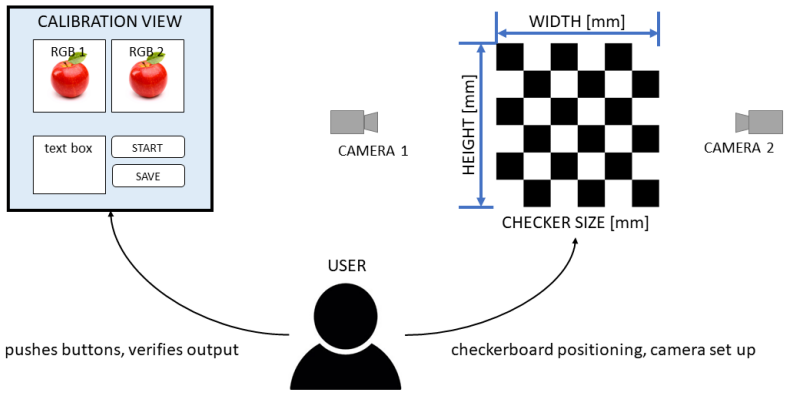


Figura 5.3: Schema di base della prima fase della calibrazione

- nella seconda parte, mediante un oggetto calibrato (in questo caso un triangolo), le telecamere assumono esso come sistema di riferimento comune per tutte le misurazioni successive.

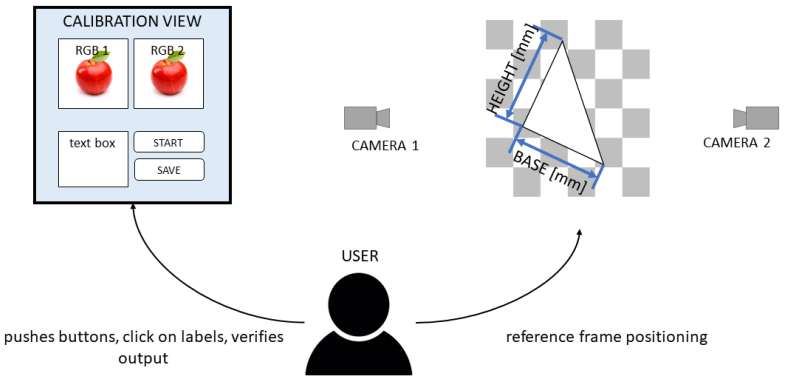


Figura 5.4: Schema di base della seconda fase di calibrazione

In questa parte dello sviluppo è stato fondamentale capire il funzionamento della programmazione a oggetti in quanto, sapendo che il programma sarebbe diventato più ampio e complesso, risultava necessario organizzare il programma in modo strutturato e modulabile, in modo tale che, in futuro, sarebbe stato più semplice aggiungere nuovi elementi.

5.4 Prima fase di test

Oltre a verificare l'effettivo funzionamento del programma e la precisione delle telecamere, questo test è stato fondamentale per capire come ottimizzare la disposizione delle telecamere.

Infatti, i test sono stati eseguiti utilizzando una struttura modulare e configurabile in modo tale da poter disporre le telecamere, a diverse distanze ed angolazioni rispetto alla scacchiera di riferimento.

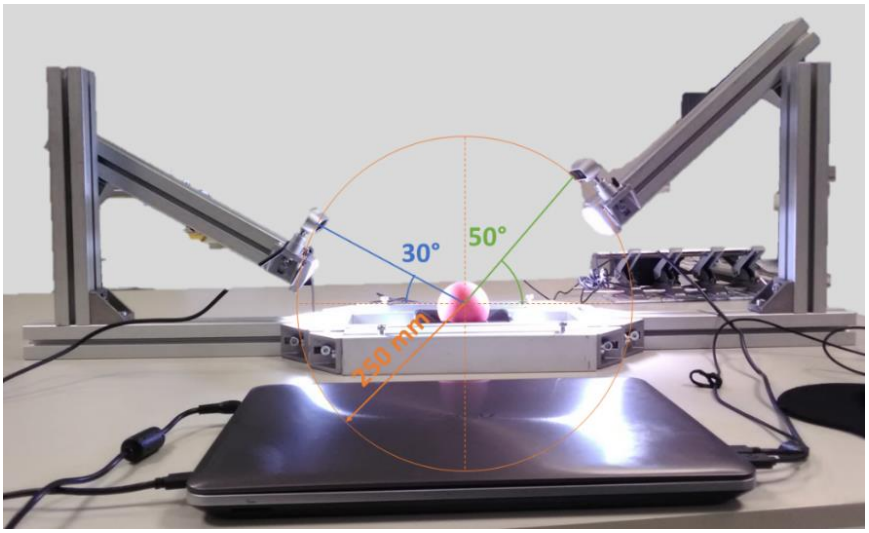


Figura 5.5: Esempio di configurazione della struttura per due angolazioni diverse alla stessa distanza

Questo tipo di studio è necessario, non solo per verificare l'adattamento delle telecamere a diverse condizioni, ma anche per capire quale sia la migliore configurazione per "vedere" la maggior porzione di mela possibile.

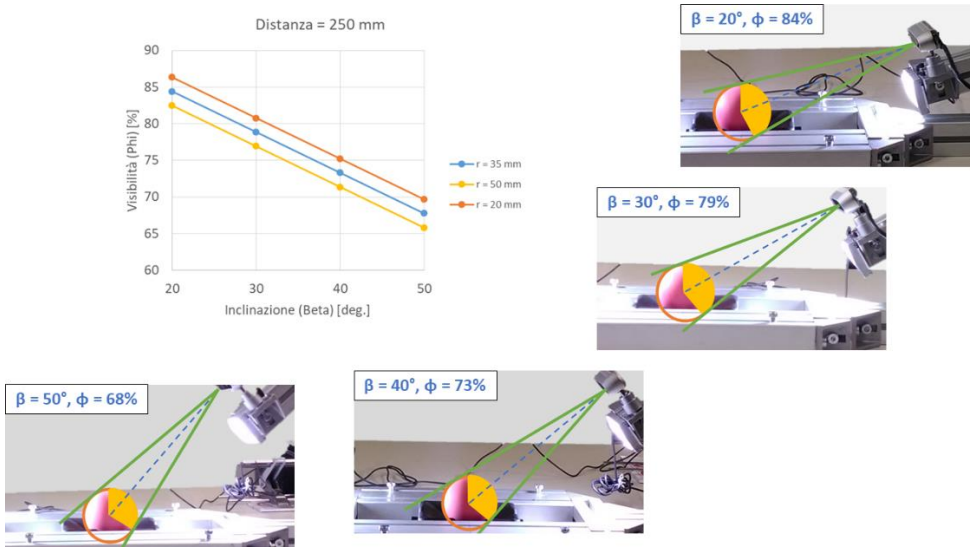


Figura 5.6: Visibilità di una sfera in funzione dell'inclinazione delle telecamere

Per quel che riguarda il grado di precisione delle telecamere, abbiamo verificato che l'errore massimo che si verifica in condizioni normali di funzionamento non è mai superiore a 1mm. Ciò era prevedibile in quanto era scritto nel grafico fornito da RealSense, ma era necessario verificarlo sul campo.

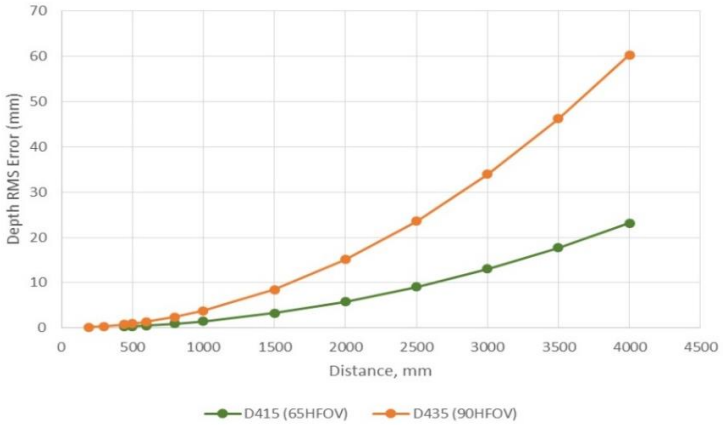


Figura 5.7: Limite di precisione teorico per le telecamere realsense in funzione della distanza [44]

Queste prove ci hanno permesso di capire l'importanza dell'illuminazione sulla qualità dell'immagine 3D elaborata dalle telecamere. Se l'intensità luminosa non è costante, e se il software delle telecamere non ha la giusta messa a punto per quel livello di luminosità, l'errore sulla misurazione può aumentare, o peggio ancora, le telecamere potrebbero non riuscire ad individuare affatto la scacchiera di calibrazione, o un qualsiasi altro oggetto.

Un altro fattore importante che può influenzare l'errore di rilevazione delle coordinate è la planarità della scacchiera di riferimento e l'esattezza delle misure dell'oggetto calibrato.

Inoltre, la precisione della seconda fase della calibrazione dipende fortemente dall'accuratezza dell'occhio umano, in quanto deve essere l'operatore a cliccare sulle estremità dell'oggetto calibrato in modo tale da passare le sue coordinate al programma. Probabilmente questo è uno dei fattori che causa gli errori maggiori, e in vista di un possibile sviluppo futuro, si consiglia di creare un codice che sia in grado di rilevare queste coordinate senza l'ausilio dell'operatore.

5.5 PyQt5

Figura 5.8: Logo di PyQt5 [45]

Nella fase successiva di sviluppo del programma, sarà necessario costruire l'interfaccia grafica definitiva con cui l'operatore sarà in grado di interagire con il programma del prototipo del sistema. Risulta quindi necessario installare PyQt, ovvero una libreria grafica per Python che permette di realizzare applicazioni dotate di un'interfaccia grafica.

PyQt mette insieme la struttura applicativa Qt e il linguaggio di programmazione Python.

Visto che sono poche le applicazioni di Qt che sfruttano tutte le funzioni presenti, il framework è stato diviso in moduli selezionabili ed assemblabili in modo da garantire una flessibilità elevata per gli utenti. Con l'arrivo dell'ultima versione, Qt ha presentato un'ulteriore classificazione dei moduli in modulo base (Qt Essentials) e modulo supplementare (Qt Add on).

I Qt Essentials rappresentano le fondamenta di Qt. Sono fruibili su tutte le piattaforme di sviluppo supportate e sono adatte all'impiego per la maggior parte dello sviluppo delle applicazioni.

I moduli "Qt Add-on" sono vantaggiosi per gli utenti di Qt che hanno bisogno di sviluppare applicazioni particolari. Mentre i moduli analizzati precedentemente, sono ideati per qualsiasi tipo di sviluppo e piattaforma di destinazione, la maggior parte di questi elementi si possono utilizzare solo su piattaforme ben definite.

Le classi di Qt impiegano un meccanismo di segnali/cave per la comunicazione tra oggetti rendendo facile la creazione di di componenti software ri-utilizzabili.

Qt supporta diversi linguaggi di programmazione, ma quello nativo del framework rimane C++ e la sua espansione mediante il preprocessore MOC permette l'utilizzo di nuovi strumenti e paradigmi, come i segnali e le cave già accennati. Essi garantiscono una comunicazione indirizzata dagli eventi tra i vari elementi del programma e costituisce un'opzione flessibile alle funzioni di richiamo dirette denominate "callback".

Qt include anche Qt Designer, un disegnatore di interfacce grafiche utente. PyQt è in grado di generare codice in Python da Qt Designer. È anche possibile aggiungere a Qt Designer nuovi controlli GUI scritti in Python.

Quindi, oltre a PyQt è necessario installare anche Qt. Nel nostro caso, abbiamo scelto di installare le versioni più recenti, quindi abbiamo installato PyQt5 e Qt5.

Inoltre, ci siamo serviti dell'ambiente di sviluppo integrato Qt Creator, che dispone di un editor visuale con cui abbiamo costruito la struttura portante delle finestre dell'interfaccia grafica del programma, senza dover scrivere una singola riga di codice.

5.6 Qt Creator

Qt Creator è un ambiente di sviluppo integrato (IDE) che offre, durante il processo di sviluppo, diversi strumenti e sistemi di automatizzazione. Fornisce assistenza all'utente nell'ideazione di un nuovo progetto, indirizzando le sue azioni durante il processo di creazione e fornendogli tutti i file necessari. Oltre a questo, l'IDE rende più veloce la scrittura del codice, tramite l'evidenziazione della sintassi, il completamento automatico del codice ed il controllo degli errori.

Questi sono gli strumenti che appartengono al Code Editor di Qt Creator:

- qmake: è un sistema di compilazione standard di Qt;
- Qt Designer: è il programma nativo per la progettazione e la creazione di interfacce utente grafiche con l'aiuto dei widget di Qt. Questo editor visuale permette di organizzare e personalizzare i widget;
- Qt Linguist: è uno strumento in grado di localizzare linguisticamente le applicazioni direttamente all'interno di Qt Creator;
- Qt Assistant: fornisce un accesso rapido alla documentazione ufficiale del framework.

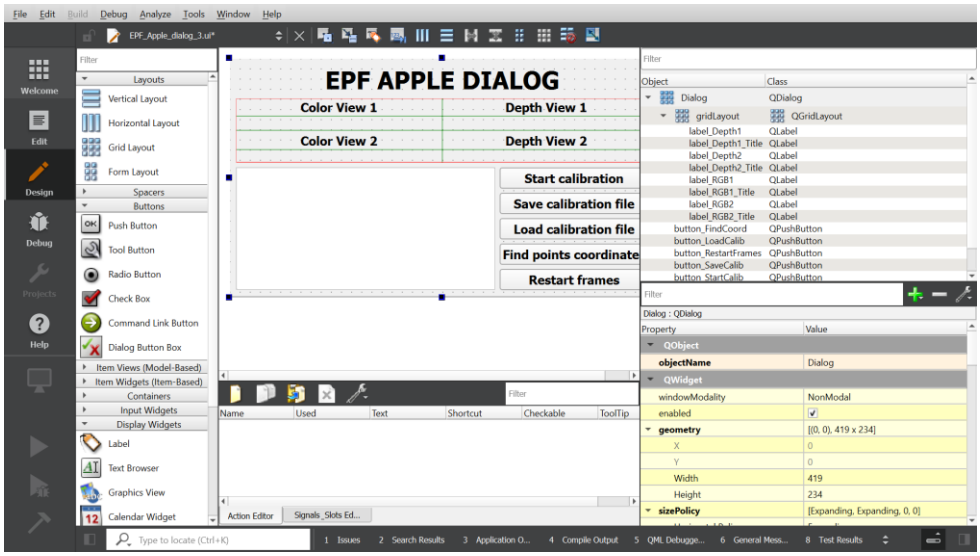


Figura 5.9: impostazione della finestra di calibrazione in QtCreator

5.7 Seconda fase di sviluppo del programma.

Ora si può procedere a creare il programma definitivo. Tramite PyQt5 sono state create tre finestre:

- Main App: alla quale si accede all'esecuzione del programma;
- Dialog Calibration: da aprire per eseguire la calibrazione delle telecamere;
- Dialog Snap: da aprire per eseguire la nuova parte di programma.

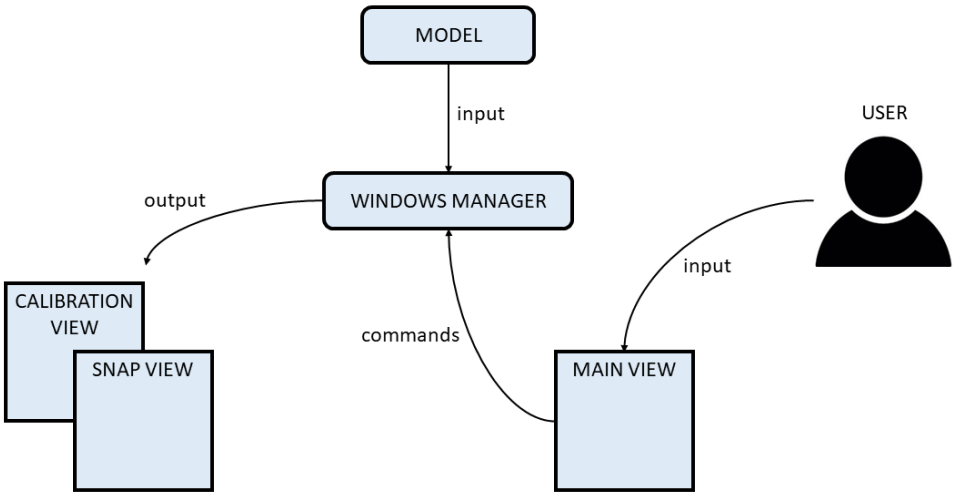


Figura 5.10: Schema di base della gestione delle finestre

Mentre le prime due gestiranno le due operazioni già programmate, la terza servirà a gestire la nuova parte di programma, all'interno del quale si deve:

- rilevare il passaggio del facchino tramite le fotocellule;
- catturare le foto 3D della mela sul facchino tramite le telecamere;
- elaborare le Point Cloud delle telecamere per avere un'immagine 3D totale della mela.

Quindi, a differenza della fase precedente, risulta necessario collegare le fotocellule al PC, ed installare l'estensione Snap7 per poterle gestire tramite Python. Dopo aver capito come operare le fotocellule si è passato alla programmazione effettiva delle finestre e delle nuove operazioni, sfruttando PyQt.

Inoltre, bisogna tener conto che l'utente, tramite le finestre, è in grado di modificare i dati con cui opera il programma, e le finestre devono sempre mostrare le informazioni sulla base dei dati "aggiornati". Conseguentemente è stato necessario dotare il programma

dell'architettura Model View in quanto è la più efficiente e la più adatta alla gestione del problema da affrontare.

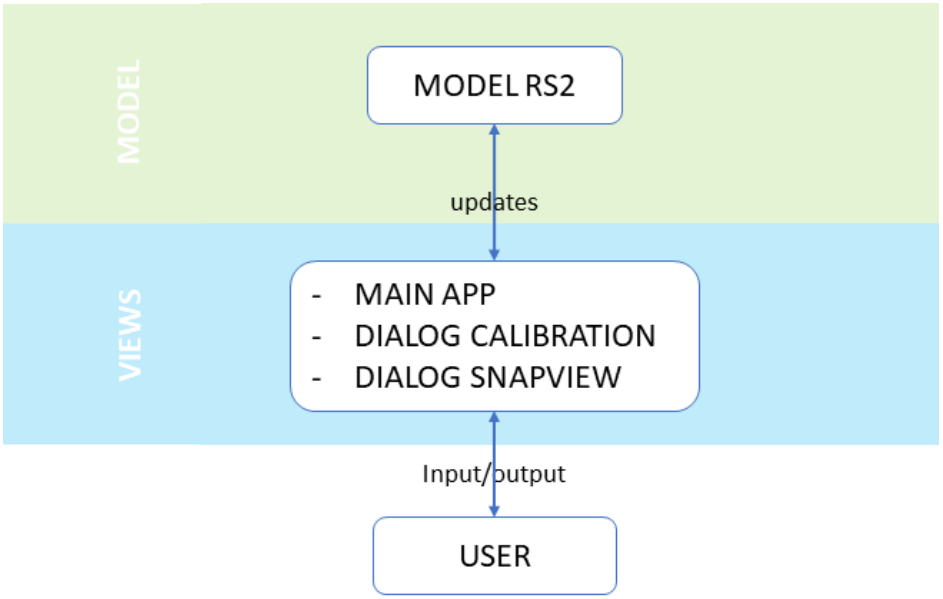


Figura 5.11: Schema di base del programma definitivo

Per la programmazione, è stato necessario capire come gestire al meglio il multithreading. Infatti, in tutte e tre le finestre è possibile visualizzare i frames delle telecamere, e può capitare che sia aperta più di una finestra contemporaneamente. Questo è uno degli aspetti più difficili da gestire all'interno di Python e difatti, è stato il problema la cui risoluzione ha impiegato il tempo maggiore.

Ciononostante, tramite alcuni strumenti forniti da PyQt, come i timer ed il sistema di segnali e cave, si sono riuscite a gestire in modo chiaro e ordinato tutte le operazioni del programma.

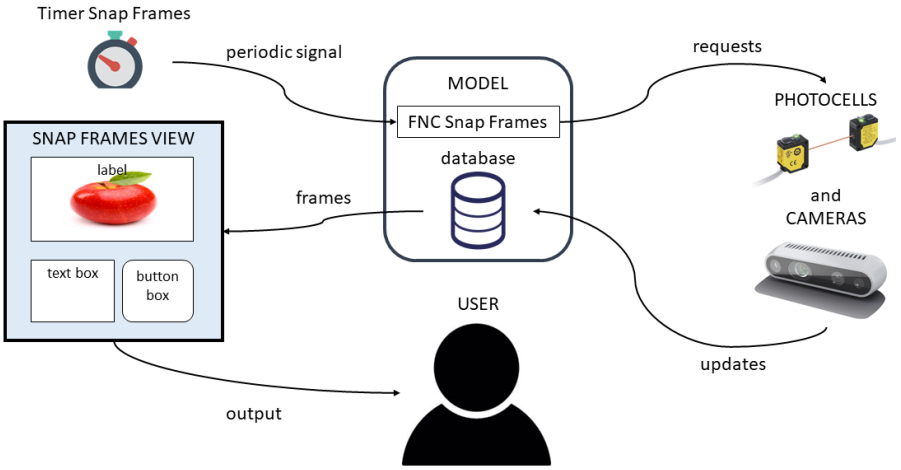


Figura 5.12: Schema di base del funzionamento della seconda parte di programma

Ovviamente è stato necessario applicare questi principi anche alla parte del programma creata precedentemente in quanto tutto deve funzionare in contemporanea. Questo processo, comunque, non è stato particolarmente difficile in quanto si è trattato solo di aggiungere qualche elemento (timer e segnali/cave), in quanto l'ossatura del programma era già presente.

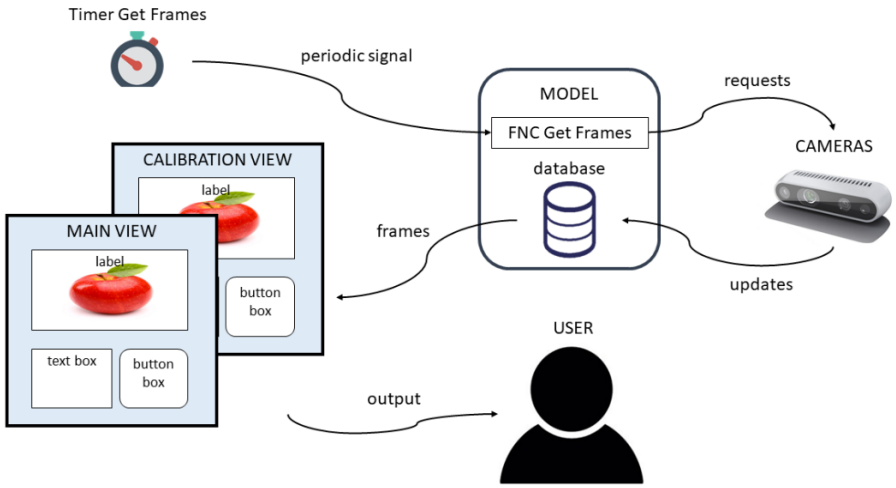


Figura 5.13: Schema di base per la cattura dei frames della telecamera nella prima parte del programma aggiornata

5.8 Seconda fase di test

Mentre la prima fase di test è stata eseguita successivamente alla prima fase di sviluppo del programma, la seconda fase di test è stata eseguita in parallelo alla seconda fase di sviluppo del programma.

Ciò è dovuto al fatto che, per poter debuggare la nuova parte di programma, era assolutamente necessario eseguirlo nelle condizioni di lavoro effettive. Quindi, ancora prima di procedere alla programmazione è stato necessario costruire un prototipo del sistema all'interno dell'officina dell'azienda EPF.

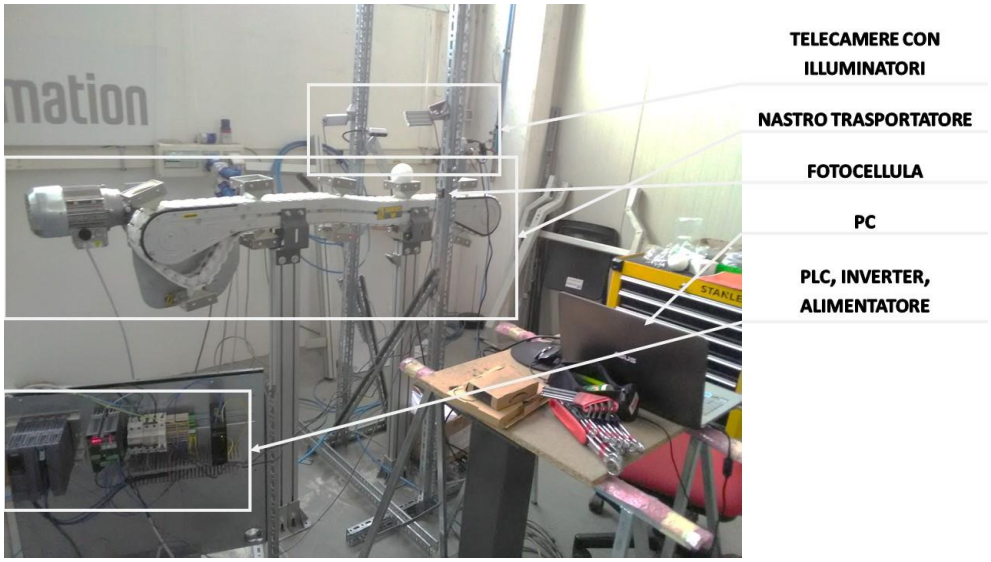


Figura 5.14: Vista complessiva del prototipo del sistema

Come detto nei capitoli precedenti questo sistema è costituito da:

- un nastro trasportatore (viola): per movimentare le mele;
- una fotocellula (rosso): per rilevare il passaggio dei facchini;
- due telecamere stereoscopiche con illuminatori (giallo): per catturare le immagini 3D delle mele;
- un pannello con un PLC, un inverter ed un alimentatore 24V (blu): per gestire il nastro trasportatore, la fotocellula e gli illuminatori;
- un PC (verde): per modificare il programma ed interagire con il processo.

Le telecamere sono state disposte nella migliore configurazione possibile in accordo con gli studi effettuati nella prima fase di test.

La fotocellula è stata posta sulla colonnina che regge una delle due telecamere in modo da scattare la foto quando la sfera è più vicina possibile ad esse, riducendo al minimo l'errore.

Vicino alle telecamere sono stati avvitati degli illuminatori in modo da mantenere un livello di luce più o meno costante.

Le telecamere sono state collegate direttamente al PC, mentre per la fotocellula è stato necessario interporre il PLC tra essa ed il PC.

Questo banco di prova è servito ad effettuare principalmente due test:

- test sulla fase di calibrazione
- test sulla cattura di immagini 3D

Mentre il primo test deve avvenire con il nastro trasportatore completamente fermo, il secondo test deve essere effettuato con i facchini in movimento. La velocità del nastro trasportatore è stata regolata tramite il PLC ad un livello "realistico" di circa 1 m/s.

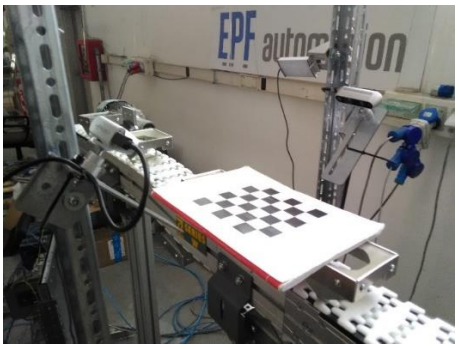


Figura 5.15: Test della fase di calibrazione

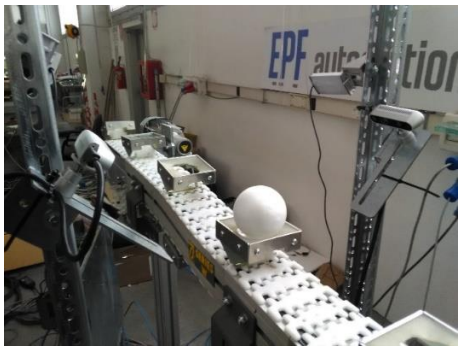


Figura 5.16: Test della cattura di immagini 3D

Questo periodo di test è avvenuto durante lo sviluppo del programma, ed è stato più di carattere qualitativo che quantitativo. Oltre a testare il normale funzionamento del programma, infatti, non è stato possibile determinare in modo quantitativo l'errore che si verifica nella costruzione della point cloud totale. Conseguentemente il risultato finale (la point cloud totale) è stato giudicato, fondamentalmente, in modo qualitativo.

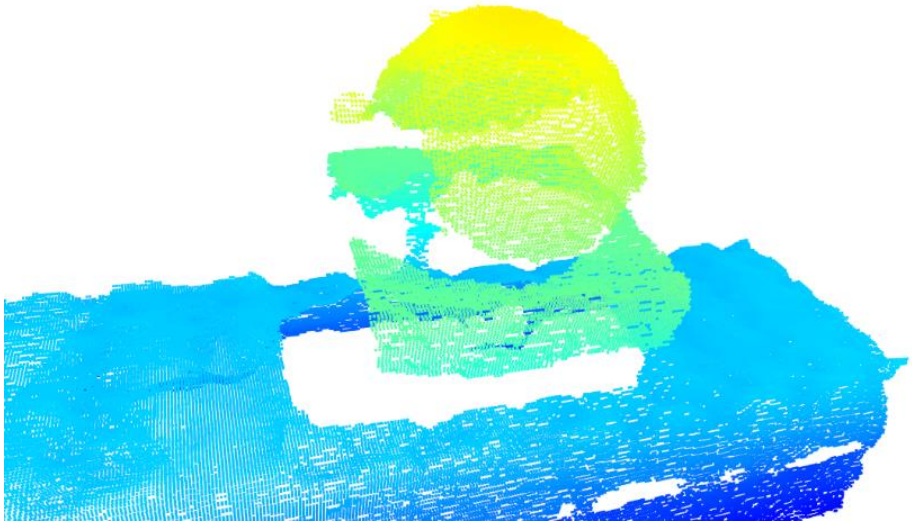


Figura 5.17: Esempio di pointcloud ottenuta dal test

Visto che si sono verificati dei problemi causati dall'illuminazione esterna non uniforme, si è provato a ridurli isolando la zona in cui avviene la cattura delle immagini, semplicemente utilizzando uno scatolone.

Figura 5.18: Isolamento della zona della cattura delle immagini 3D

In questo modo è stato possibile mantenere un livello di luce indipendente dall'ambiente circostante ed è stato più semplice capire quale fosse il suo effetto reale sulla densità dei punti sulla point cloud.

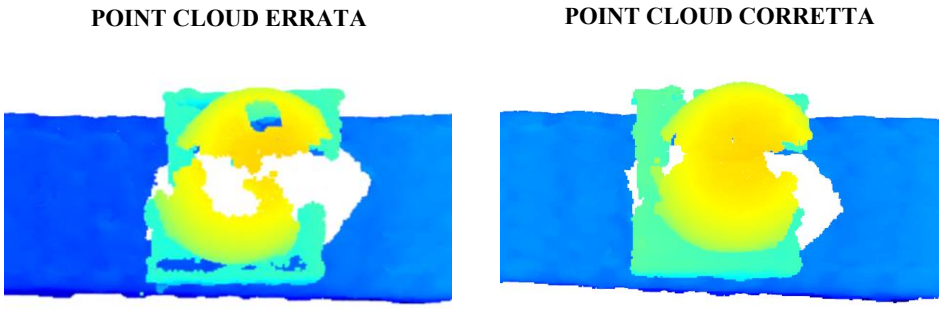


Figura 5.19: Effetto del movimento sulla pointcloud

In ogni caso, sia con lo scatolone, che senza, il problema più importante che si è riscontrato in questa fase è stato il movimento dell'oggetto osservato durante la cattura delle immagini. Questo fattore era assente nella prima fase di test ed ha imposto un ri-studio della configurazione software delle telecamere, in particolare dei filtri applicati alle immagini 3D. Risolto questo problema, il risultato ottenuto è stato giudicato soddisfacente e si è proseguito alla scrittura della tesi.

5.9 Descrizione del programma

Per comprendere il funzionamento e la struttura del programma è necessario capire, per prima cosa, le operazioni principali che esso deve essere in grado di eseguire.

Inoltre, bisogna tenere presente che questo programma deve essere utilizzato da un operatore per la gestione di diversi processi. Quindi necessita di un'interfaccia grafica che sia in grado di comunicare all'utente le informazioni più utili nel modo più efficiente possibile.

Questo programma dovrà svolgere compiti di diverso tipo, e durante ognuno le informazioni da comunicare all'utente saranno di tipo diverso. Di conseguenza, è stato necessario progettare delle finestre diverse per ogni macro-operazione.

Le funzioni principali del programma sono 3:

- inizializzazione delle telecamere e delle fotocellule e scelta della prossima funzione da eseguire;
- calibrazione delle telecamere;
- cattura delle foto 3D delle mele in movimento sul nastro trasportatore.

Oltre a queste 3 funzioni che sono supervisionate direttamente dall'operatore, ci sono altre 2 macro-operazioni che il programma deve eseguire:

- gestione delle finestre dell'interfaccia grafica;
- gestione dei dati legati alle telecamere;
- elaborazione delle pointcloud da inviare al robot.

Mentre le prime 2 sovrintendono i processi che avvengono nelle 3 funzioni principali, la terza è stata messa a parte in quanto non richiede l'interazione con l'operatore e quindi non richiede una propria finestra di interfaccia.

Qua sotto elencheremo i nomi delle parti di programma a cui corrispondono le funzioni principali elencate sopra:

- MAIN_APP.py;
- DIALOG_CALIBRATION.py;
- DIALOG_SNAPVIEW.py.

Alle altre 3 funzioni invece corrispondono:

- WINDOWS_CONTROLLER.py;
- MODEL_RS2.py;
- EPF_AbsoluteCoordinatesCalculator.py.

Ognuna di queste parti di programma racchiude al suo interno delle parti di codice chiamate Classi, all'interno dei quali ci sono dei Metodi che si occupano in modo specifico di eseguire operazioni elementari distinte. In questo paragrafo elencheremo le classi che vengono utilizzate da queste parti di codice per illustrare la struttura complessiva del programma, ma non andremo a spiegarle nel dettaglio.

- MAIN_APP.py permette l'esecuzione di:
  - MainWindow;
  - Thread;
  - Cameras (from MODEL_RS2.py).
- DIALOG_CALIBRATION.py permette l'esecuzione di:
  - DialogWindow;

    - Thread;
    - Cameras (from MODEL_RS2.py).
- DIALOG_SNAPVIEW.py permette l'esecuzione di:
    - Dialog_SnapView;
    - Thread;
    - Cameras (from MODEL_RS2.py).
- WINDOWS_CONTROLLER.py permette l'esecuzione di:
    - MainWindow (from MAIN_APP.py);
    - DialogWindow (from DIALOG_CALIBRATION.py);
    - Dialog_SnapView (from DIALOG_SNAPVIEW.py);
    - Cameras (from MODEL_RS2.py).
- MODEL_RS2.py permette l'esecuzione di:
    - Cameras (from MODEL_RS2.py).
- EPF_AbsoluteCoordinatesCalculator.py permette di eseguire:
    - Cameras (from MODEL_RS2.py).

Ogni funzione principale verrà spiegata nel dettaglio in seguito, all'interno dei paragrafi successivi.

Già da questo elenco però è possibile notare che MODEL_RS2.py assume un compito molto particolare in quanto senza di esso non può funzionare nessun'altra parte di programma. Ciò è dovuto al fatto che al programma è stata assegnata una struttura detta Model View, e come si può intuire dal nome MODEL_RS2.py costituisce il Modello del programma. Nel prossimo paragrafo approfondiremo l'architettura del programma e la relazione tra le diverse parti.

5.9.1 Il pattern Model/ViewController

Nel paragrafo precedente abbiamo visto quali sono le funzioni che il programma deve essere in grado di eseguire e abbiamo elencato le parti di programma legate ad ognuna di essa.

Ora, è necessario capire che per tutte e 3 le funzioni principali è necessario che l'utente veda, attraverso un'interfaccia grafica, ciò che sono in grado di vedere le telecamere, sia per interagire con il processo, che per verificarne il corretto funzionamento.

Come detto nel paragrafo precedente, a queste 3 funzioni assoceremo 3 finestre diverse che chiameremo Viste. Esse infatti hanno la funzione di mostrare all'utente solo alcune delle informazioni che riguardano le telecamere, in quanto non si vuole sovraccaricare lo schermo di informazioni inutili. Queste informazioni però devono sempre essere aggiornate, e se necessario deve essere possibile modificarle tramite degli input sulle etichette e sui bottoni dell'interfaccia grafica.

Come si può notare dall'elenco nel paragrafo precedente è necessario che ogni vista sia legata ad una fonte di dati, da cui prendere informazioni e su cui scrivere informazioni, in quanto per mezzo di essa è necessario anche scrivere e salvare i risultati ottenuti dai vari processi. Questa base di dati è detta Model.

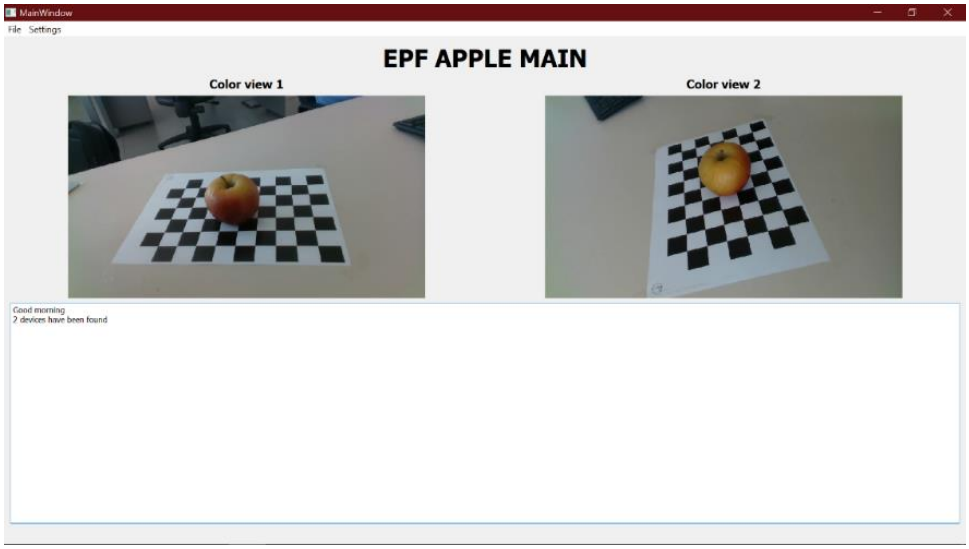


Figura 5.20: Esempio di una vista presente nel programma

In questo modo il programma ha assunto l'architettura Model/ViewController. Essa ha origine dal modello architetturale noto come Model–View–Controller (MVC) che è comunemente utilizzato per lo sviluppo delle interfacce utente e consiste, fondamentalmente, nel dividere un'applicazione in tre parti interconnesse. Ciò consente di separare la rappresentazione dei dati dai dati stessi ed il modo in cui possono essere modificati.

Il design del modello MVC separa le tre componenti principali:

- Model all'interno del quale si trovano i dati con cui la app sta lavorando.
- View è una qualsiasi rappresentazione dell'informazione come mostrata all'utente, che sia grafica o sotto forma di tabelle.
- Controller accetta gli input dall'utente, trasformandolo in comandi per il modello o le viste.

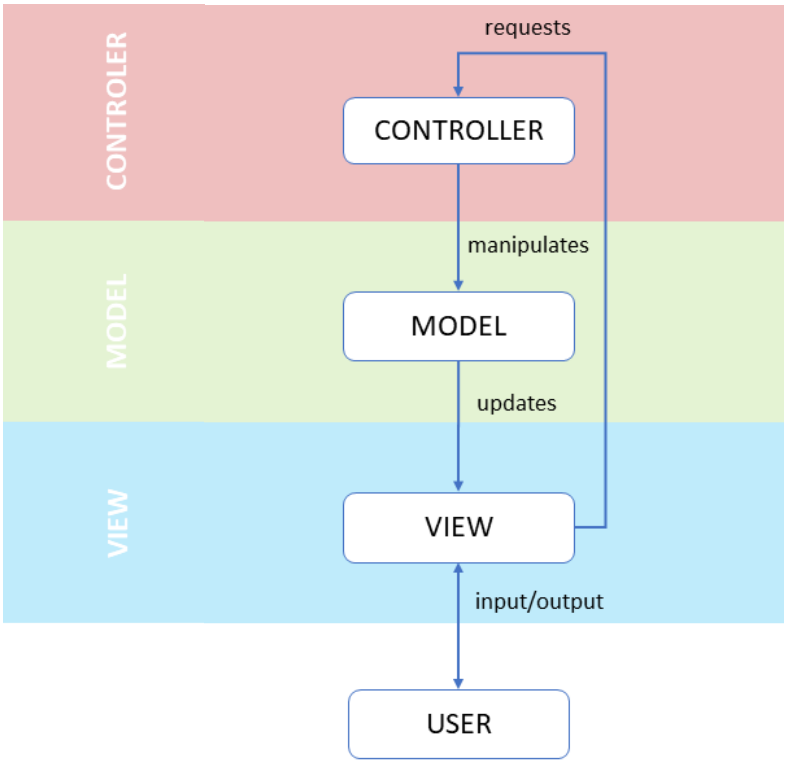


Figura 5.21: Architettura model – view – controller generica

All'interno di Qt la distinzione tra Viste e Controllore è più nebbiosa. Ciò è dovuto al fatto che Qt accetta gli eventi di input dall'utente (tramite l'OS) e delega la manipolazione agli widget (Controllore). Allo stesso tempo, gli widget trattano anche la presentazione dello stato corrente all'utente, mettendoli direttamente nelle Viste. Quindi, nel linguaggio di Qt le Viste ed il Controllore sono fusi insieme dando vita all'architettura Model/ViewController, anche chiamata "Model View" per semplicità.

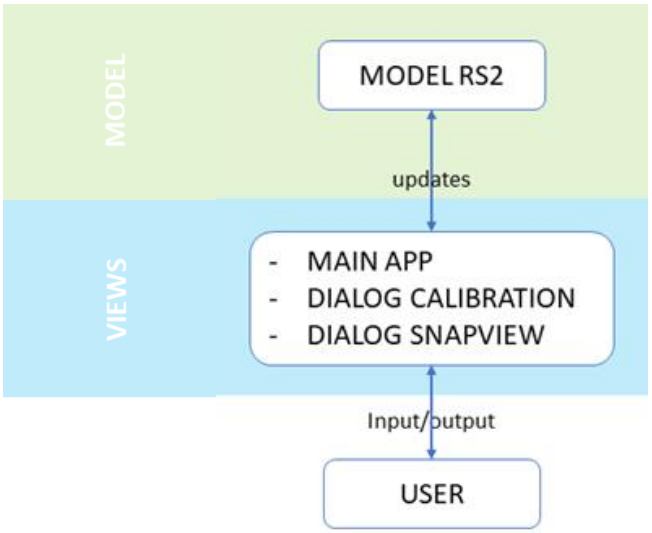


Figura 5.22: Architettura model – view applicata al programma

Nel nostro caso:

- Cameras è la parte di programma che definisce il Model;
- MainWindow, DialogWindow e DialogSnapView sono le 3 classi che si occupano dell'impostazione dell'interfaccia grafica (output) e dell'elaborazione degli input trasmessi dall'utente mediante il mouse e la tastiera, quindi rivestono la funzione di ViewController.

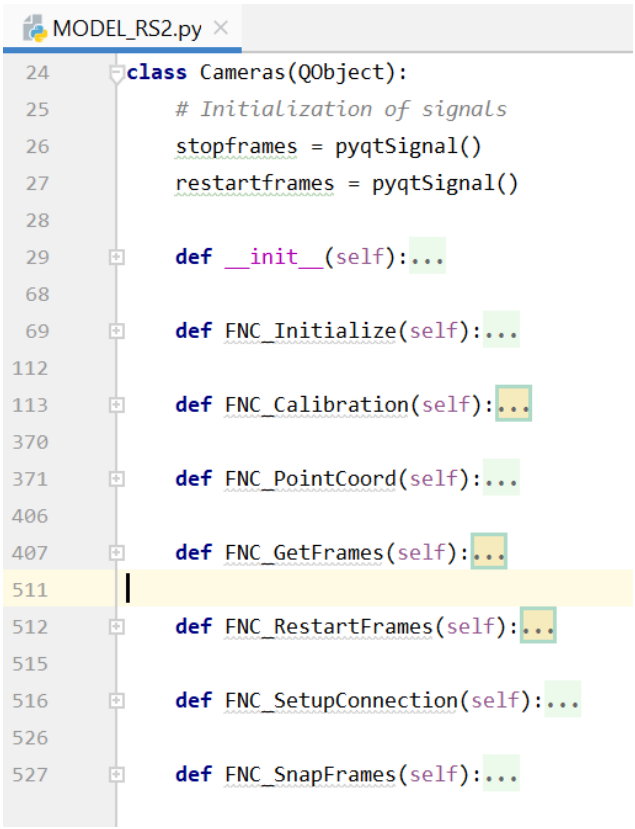


Figura 5.23: struttura di base del modello (cameras)

A queste due parti è stato aggiunto il WINDOWS_CONTROLLER per poter gestire nel modo più ordinato possibile l'apertura delle finestre. A causa della sua funzionalità, essa è l'applicazione necessaria a far partire il programma.

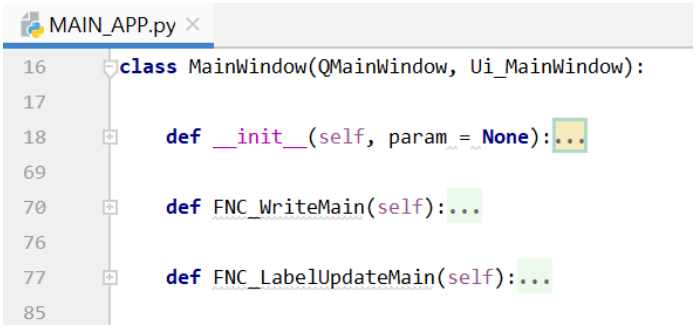
Da questa spiegazione si riesce quindi a capire perché tutte le parti di programma richiedono MODEL_RS2.py per poter funzionare. Al suo interno si trovano tutte le funzioni necessarie al funzionamento dell'intero programma:

- FNC_Initialize (accende e carica le impostazioni delle telecamere);

- FNC_Calibration (esegue la calibrazione delle telecamere);
- FNC_PointCoord (esegue la trasformazione da coordinate relative a coordinate assolute);
- FNC_GetFrames (preleva i frames dalle telecamere);
- FNC_RestartFrames (permette a Get_frames di ripartire dopo esser stato interrotto);
- FNC_SetupConnection (permette di impostare il collegamento con le fotocellule);
- FNC_SnapFrames (permette di catturare le depth map delle mele in movimento sul nastro trasportatore, utilizzando gli input delle telecamere stereoscopiche e delle fotocellule).

Invece, ogni View/Controller contiene al suo interno, oltre a qualche metodo specifico, anche una Vista:

- MainWindow utilizza Ui_MainWindow from EPF_Apple_mainwindow_4.py;
- DialogWindow utilizza Ui_Dialog from EPF_Apple_Dialog_Calibration.py;
- DialogSnapView utilizza Ui_Dialog_SnapView from EPF_Apple_Dialog_SnapView.py.



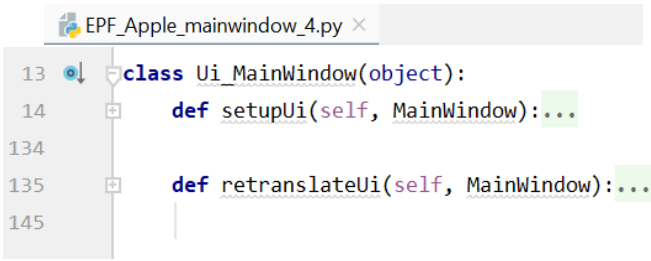


Figura 5.24: Struttura di base di una view/controller

Il codice delle Viste è stato generato da QtCreator ed ha come unico scopo l'impostazione grafica della finestra di interfaccia.

5.9.2 Programmazione a oggetti

Tra i vantaggi principali che offre Python, rispetto ad altri linguaggi di programmazione, c'è quello della possibilità di utilizzare la programmazione a oggetti.

Essa ha permesso la creazione del programma una parte alla volta, senza dover ripartire da zero, ogni volta che si devono aggiungere una o più funzionalità. Inoltre, ciò permette una maggior flessibilità e riutilizzo in quanto è possibile dividere il codice in parti ben definite, in base a funzionalità o ruolo all'interno dell'applicazione.

I principi su cui si basa la programmazione ad oggetti sono nati negli anni 60-70, ma questa tecnica ha iniziato a diffondersi solamente negli anni 90, ed è stata impiegata principalmente per la creazione di interfacce grafiche.

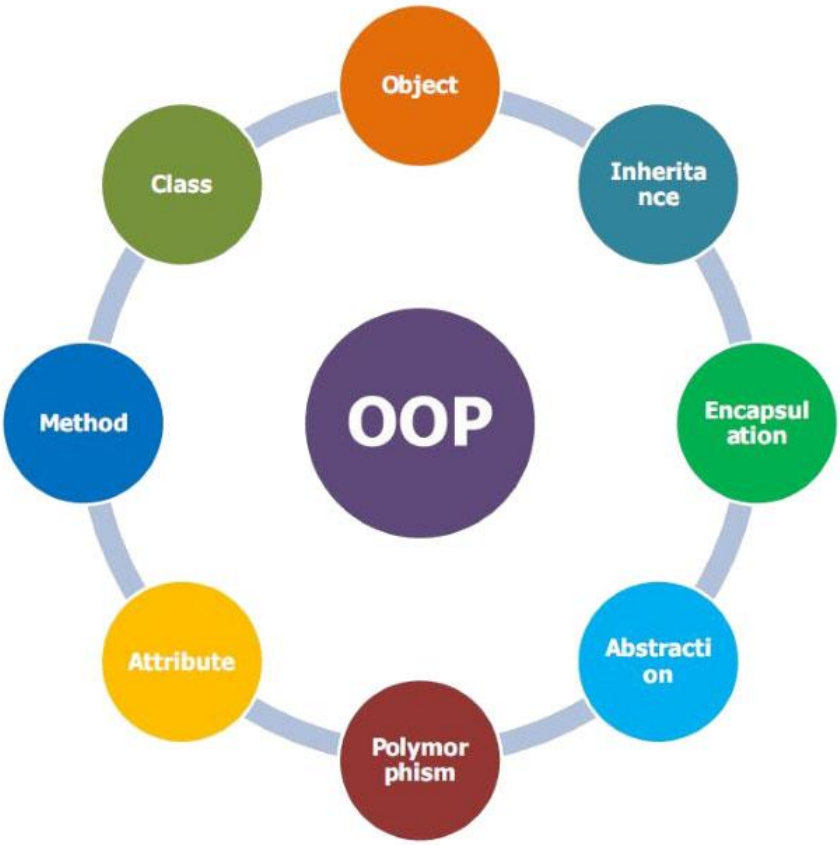


Figura 5.25: Concetti della tecnica di programmazione OOP [46]

L'object oriented programming (OOP) consiste nel mettere le strutture dati e le procedure che ci lavorano all'interno di una singola entità chiamata classe.

La realizzazione delle classi in Python serve ad isolare ed identificare nomi di oggetti. Nella definizione di una classe si danno ai nomi di variabili e di funzioni una struttura gerarchica, come quella che permette di separare le classi base da quelle derivate. Nel linguaggio di Python le classi si possono considerare come dei contenitori di nomi.

Le variabili che si definiscono all'interno di una classe sono dette locali, e ci si può accedere solo dalla stessa classe e dalle sue derivate. Se si vuole avere accesso dal di fuori della classe, allora è necessario riconoscerle esplicitamente come membri della stessa.

Le variabili che si trovano all'interno di una classe sono dette attributi, le funzioni, invece, sono denominate metodi. Sia gli attributi che i metodi sono definibili come membri della classe.

Da una classe è possibile creare degli oggetti chiamati istanze, che saranno tutti dello stesso tipo ma saranno contraddistinti da diversi attributi. I metodi definiti all'interno della classe sono validi per ogni istanza.

Gli attributi si possono considerare come i valori associabili ad un'istanza, mentre i metodi vanno a descrivere il comportamento dell'oggetto, come se fossero delle funzioni, ma specifiche ad ogni classe. I metodi possono avere accesso ad altri attributi e metodi dell'istanza a priori, senza doverlo esplicitare.

Ovviamente, nella programmazione a oggetti è fondamentale lavorare sulla definizione delle classi e le relazioni che si possono creare tra loro. Infatti, esse possono ereditare attributi e metodi da altre classi, per cui, all'interno di un programma, si possono trovare famiglie di classi imparentate tra di loro.

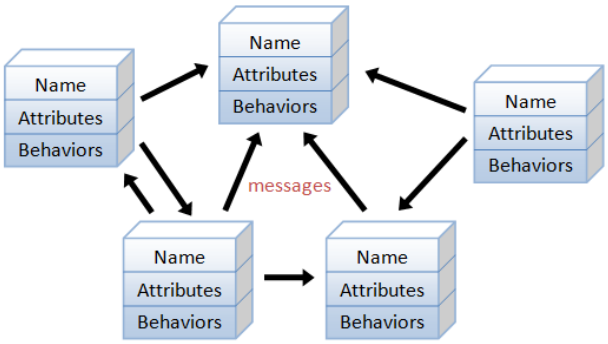


Figura 5.26: Interazione tra oggetti all'interno di un programma [47]

Andando a rivedere lo schema della struttura del programma nel primo paragrafo, e avendo capito il ruolo delle classi all'interno del modello Model/ViewController, come spiegato nel secondo paragrafo, ora dovrebbe risultare più chiaro il modo in cui le diverse parti sono collegate tra loro.

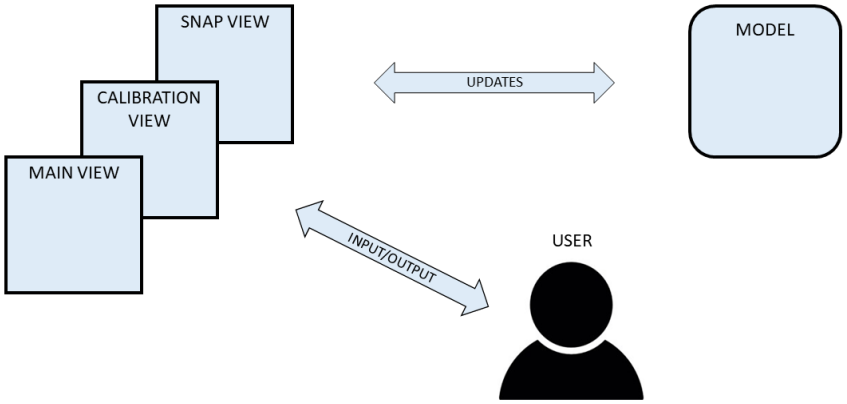


Figura 5.27: Schema di base del funzionamento del programma

L'ultimo aspetto fondamentale che è rimasto da chiarire è il ruolo delle classi Thread che si trovano all'interno di MAIN_APP.py, DIALOG_CALIBRATION.py e DIALOG_SNAPVIEW.py. Come vedremo nel prossimo paragrafo, esse sono fondamentali per il corretto funzionamento dei Timer, nel caso in cui sia sufficiente eseguire solo quelle parti di programma, senza dover ricorrere all'aiuto del codice all'interno del WINDOWS_CONTROLLER.py.

5.9.3 Programmazione Multitasking e Timer

Si è già menzionato quanto sia importante che i frames delle telecamere stereoscopiche siano visibili all'operatore durante l'esecuzione delle

funzioni principali del programma. Ciò significa che il programma deve essere in grado di eseguire diverse operazioni contemporaneamente: ad esempio, durante la calibrazione, ci sarà un momento in cui sarà necessario mostrare i frames delle telecamere e al tempo stesso si dovranno elaborare i dati della telecamera in modo da rilevare la presenza della scacchiera di calibrazione. Questo è solo un esempio, in quanto quasi tutte le operazioni elementari del programma devono avvenire mentre si mostrano i frames delle telecamere sull'interfaccia grafica.

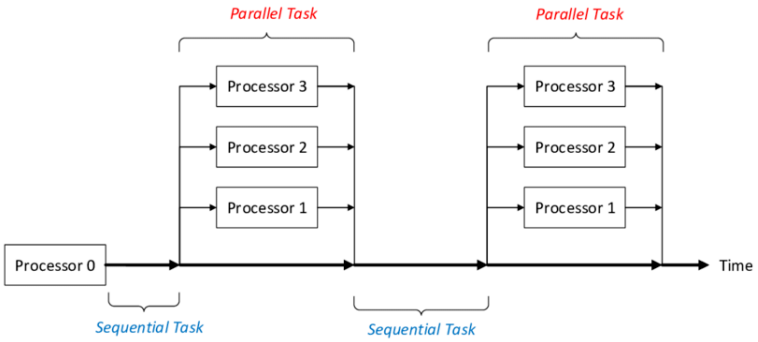


Figura 5.28: Concetto di base del multithreading [48]

In genere, l'esecuzione di più rami di codice in contemporanea, in informatica, avviene mediante il multithreading. All'interno di PyQt5, però, non è possibile eseguire un multithreading vero e proprio, ciononostante si può ottenere un risultato molto simile mediante l'utilizzo dei Timers.

Essi consentono di eseguire un'operazione periodicamente, con un intervallo di tempo definibile dall'utente. Ogni volta che scade il tempo, l'applicazione si interrompe e inizia ad eseguire l'operazione a cui è collegato il timer. Appena questa operazione giunge a termine l'applicazione riparte da dove si era fermata e il ciclo si ripete.

Da come si può intuire è fondamentale che l'operazione collegata sia relativamente breve e che sia libera di cicli infiniti, altrimenti l'intera applicazione rischierebbe di fermarsi. Inoltre, si intuisce che è fondamentale gestire la partenza e la chiusura dei timer nel modo corretto.

In questa applicazione i Timer sono stati utilizzati per:

- catturare i frame delle telecamere (Timer_Frames);
- mostrare i frame nelle etichette delle finestre (Timer_LabelUpdate);
- mostrare messaggi e gli avvisi all'utente nelle textbox delle finestre (Timer_Write);
- effettuare le operazioni di calibrazione (Timer_Calibration);
- ricevere il segnale dalle fotocellule e catturare i frame delle telecamere (Timer_SnapFrames).

Come si può notare sono molte le operazioni che devono essere eseguite periodicamente. Si potrebbe pensare che ad esse corrispondano altrettanti Timer. In realtà ad esse ne corrispondono di più: infatti bisogna tener conto che l'aggiornamento delle etichette e la scrittura all'interno delle textbox delle finestre, è un'azione legata a metodi che si trovano all'interno di ogni Vista. Ciò significa che ogni finestra ha almeno uno di questi timer legato ad uno di questi metodi.

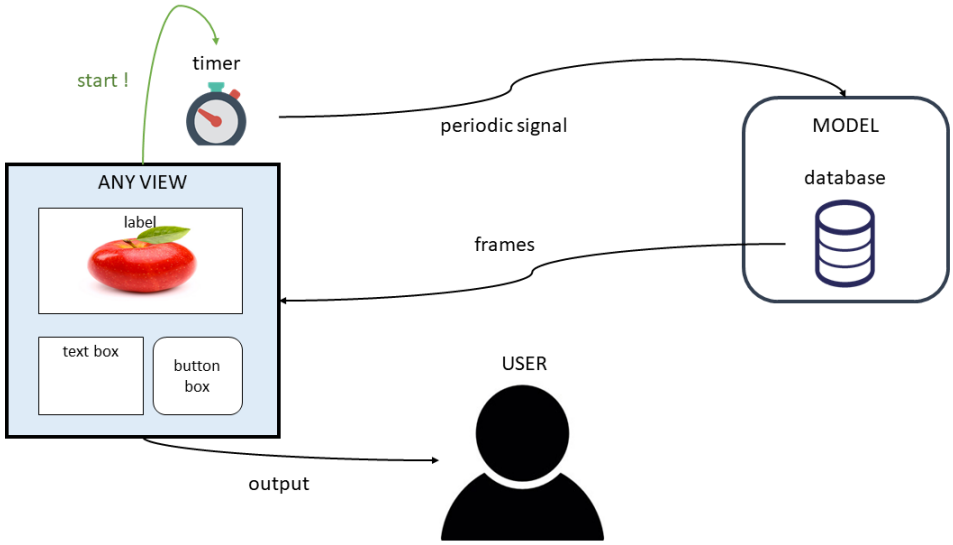


Figura 5.29: Schema della gestione dei timer per l'aggiornamento delle etichette all'interno delle viste

Risulta ovvio, invece, che Timer_Calibration e Timer_SnapFrames sono singoli in quanto sono legati a metodi presenti all'interno del Modello che si devono attivare dopo che viene aperta la finestra corrispondente.

Infine, esiste il problema della gestione di Timer_frames. Ciò è dovuto al fatto che il metodo a cui è legato, ovvero FNC_GetFrames, si trova all'interno del Modello, ma deve essere eseguito sia quando è aperta la finestra della calibrazione, sia quando è aperta la finestra principale. Siccome non è possibile far partire il segnale di partenza del Timer ogni volta che si apre una finestra diversa, è stata escogitata una strategia che coinvolge WINDOWS_CONTROLLER.py e le classi Thread.

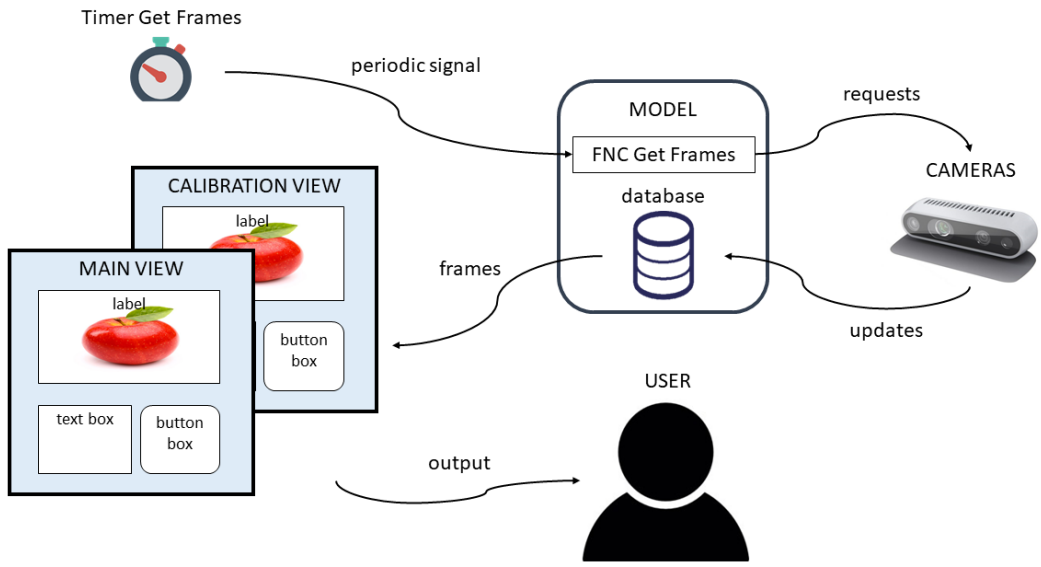


Figura 5.30: Schema della sequenza di avvenimenti attivati dal timer di FNC_getframes

Se si vuole utilizzare l'intera applicazione, è necessario eseguire per primo il codice all'interno del WINDOWS_CONTROLLER.py. Esso impone l'attivazione del Thread che si trova all'interno di MAIN_APP.py, il che comporta l'attivazione di Timer_Frames. Questo Timer non verrà attivato altre volte, di conseguenza i frame della telecamera saranno visibili all'interno di tutte le finestre senza alcun problema.

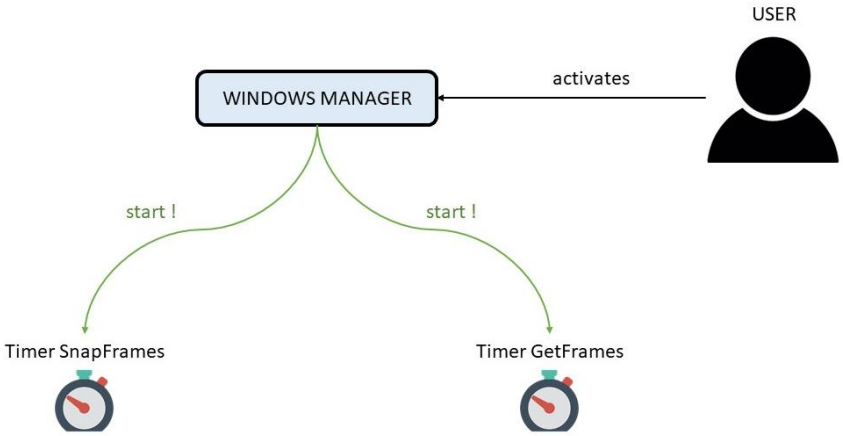


Figura 5.31: chema che indica l'attivazione dei timer operata da windows_controller.py

Se invece non si vuole utilizzare l'intera applicazione, ma solo una parte, allora si parte facendo correre il codice all'interno di DIALOG_CALIBRATION.py Ciò comporta l'attivazione del Thread all'interno di questa parte del codice, il che comanda l'attivazione del Timer_Frames. In questo modo si riescono a vedere i frames delle telecamere all'interno della finestra, senza dover ricorrere al Thread che si trova all'interno di MAIN_APP.py.

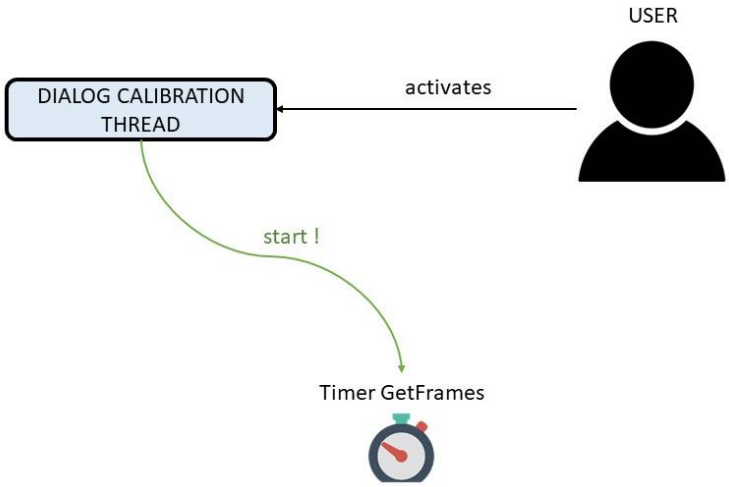


Figura 5.32: Schema del thread di calibrazione

Anche Timer_SnapFrames viene attivato all'interno di un Thread, ciò nonostante, questo thread viene attivato in ogni caso, sia che si voglia attivare solo la parte di codice corrispondente, sia che si voglia utilizzare l'intero codice. In realtà, ciò è dovuto al fatto che si era creato il thread dentro DIALOG_SNAPVIEW.py perché si pensava di utilizzare la funzione FNC_GetFrames. Solo successivamente si è deciso di utilizzare un metodo diverso.

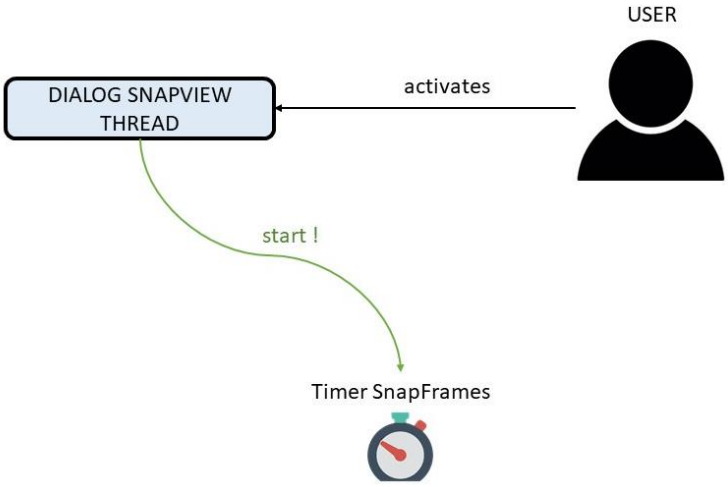


Figura 5.33: Schema del thread di snapview

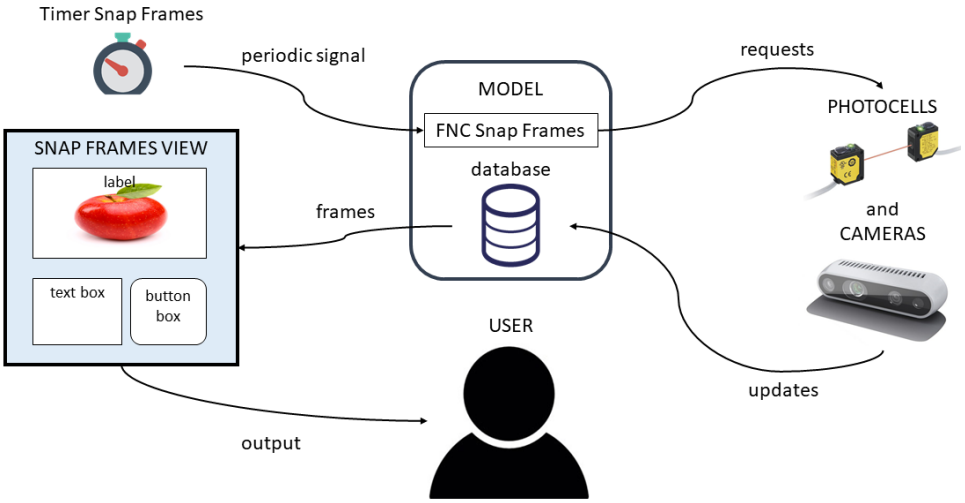


Figura 5.34: Schema della sequenza di avvenimenti attivati dal timer di FNC_snapframes

5.9.4 Programmazione con segnali/cave

Nel paragrafo precedente è stato spiegato come eseguire degli eventi periodici in Python tramite l'utilizzo dei Timers. Ora si approfondirà l'utilizzo di questi Timer andando a vedere come si possono sospendere gli eventi collegati ad essi.

Tutto ciò ci interessa in quanto, durante la calibrazione delle telecamere, c'è un momento in cui è necessario fermare la cattura dei frames. Questo accade quando si vanno a cliccare i punti di riferimento. Quest'operazione deve essere svolta su un'immagine ferma, fondamentalmente, per due motivi:

- La cattura delle immagini e la rilevazione delle coordinate, insieme alla creazione della matrice di trasformazione, sono, complessivamente, molto pesanti dal punto di vista computazionale, il che potrebbe portare l'applicazione a laggare, oppure a girare in modo poco fluido.

- Se, durante l'operazione, l'oggetto di riferimento si spostasse, anche solo di 1 mm, si verificherebbe un errore sul calcolo della matrice di trasformazione. Fermare i frames, significa bloccare la posizione dell'oggetto di riferimento e quindi evitare qualsiasi spostamento possibile.

Per farlo si ricorre all'utilizzo del meccanismo di segnali/cave disponibile all'interno del linguaggio Python.

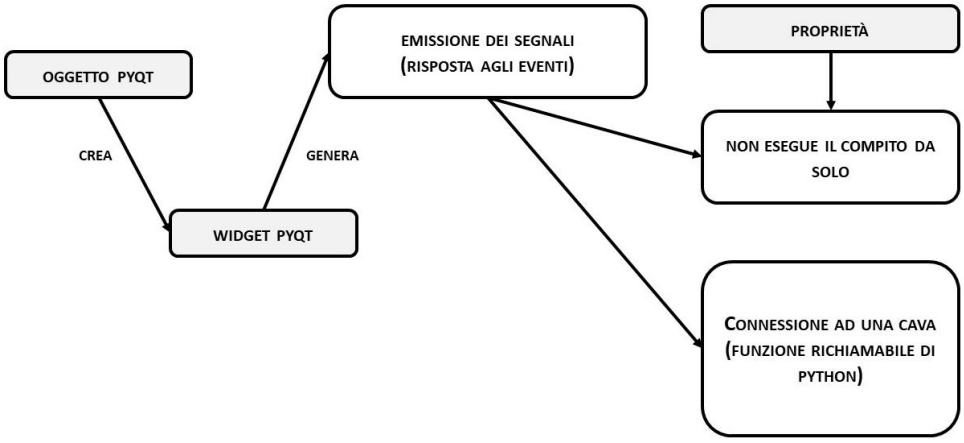


Figura 5.35: Schema di funzionamento del sistema segnali/cave

Ogni QObject supporta il meccanismo di segnali e cave. Quando viene emesso un segnale, PyQt lo scarta di default. Per prendere nota del segnale bisogna connetterlo ad una cava. In PyQt le cave si possono chiamare in qualsiasi modo (ad esempio, da qualsiasi funzione o metodo), e non è necessaria alcuna sintassi speciale quando viene definita.

Quindi, per risolvere il problema è bastato applicare al codice le seguenti modifiche:

- sono stati definiti i segnali stopframes e restartframes, all'interno di MODEL_RS2.py;

- sono state create, all'interno dei Thread di MAIN_APP.py e DIALOG_CALIBRATION.py, le funzioni FNC_StopFrames e FNC_RestartFrames in modo tale da fermare/riattivare il Timer_Frames;
- si è inserita la riga di codice necessaria ad emettere i suddetti segnali quando necessario.

FNC_StopFrames e FNC_RestartFrames si trovano all'interno di MAIN_APP e DIALOG_CALIBRATION perché questo sistema deve funzionare sia per l'applicazione completa che per la parte di codice addetta alla sola calibrazione.

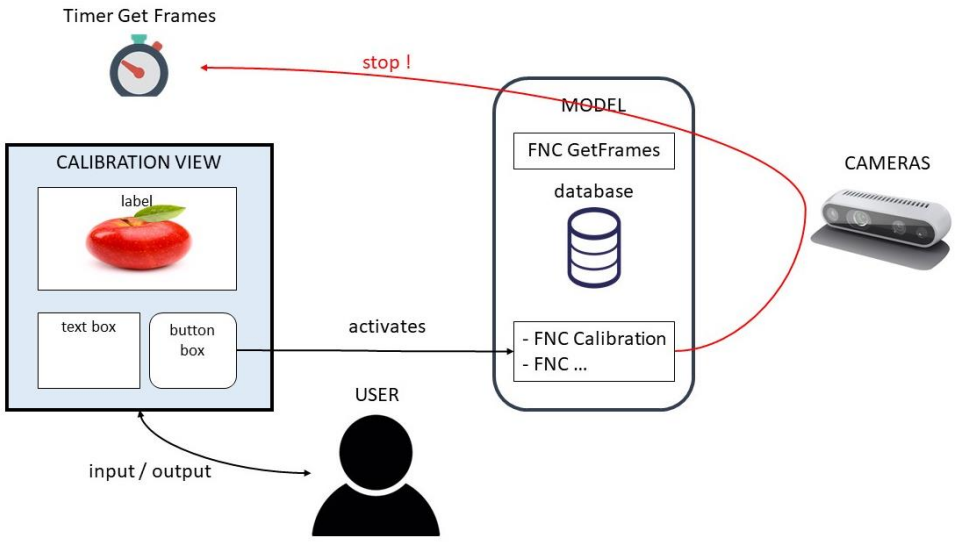


Figura 5.36: Schema della sequenza di avvenimenti che causa l'interruzione dello stream delle telecamere

In realtà il sistema di segnali e cave viene utilizzato anche quando l'utente clicca sul mouse o schiaccia i tasti della tastiera o quelli presenti sulla finestra di interfaccia. Infatti, tutti i widget, in PyQt, supportano il meccanismo di segnali e cave. In particolare, sono capaci di annunciare cambiamenti di stato, come quando una casella di controllo viene verificata o non verificata, e gli avvenimenti importanti, per esempio

quando un bottone viene cliccato (in qualsiasi modo). Tutti i widget di PyQt hanno un set di segnali predefiniti e non devono essere richiamati nel modo descritto sopra.

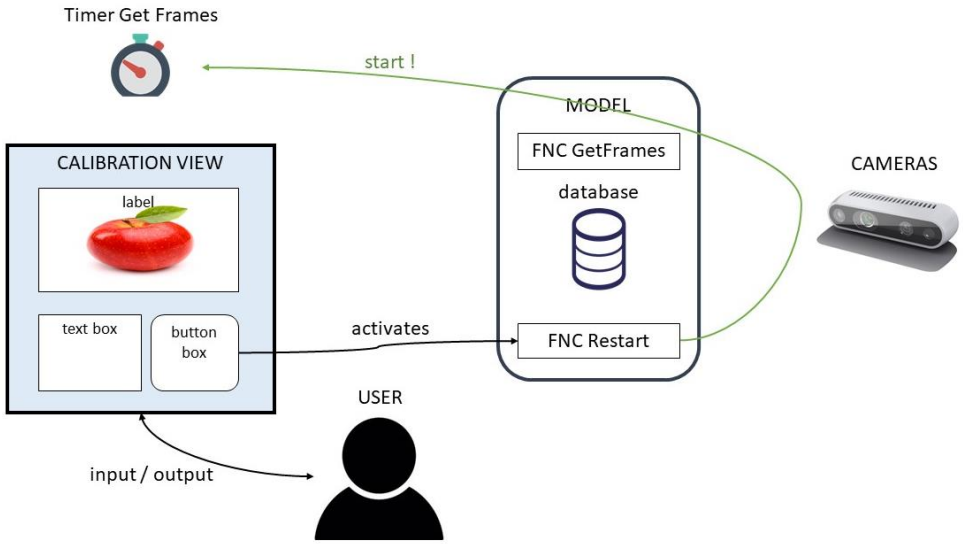


Figura 5.37: Schema della sequenza di avvenimenti che causa la ripartenza dello stream delle telecamere

Ad esempio, quando l'utente clicca con il mouse sull'etichetta, in cui si trova il frame della telecamera per rilevare le coordinate dell'oggetto di riferimento, viene inviato un segnale che porta l'informazione delle coordinate in cui si trovava il puntatore del mouse rispetto all'etichetta su cui si è cliccato, ed esso viene collegato ad una cava all'interno del quale sarà trasmessa l'informazione. Nonostante ciò non c'è bisogno di definire un segnale o una cava. Essendo eventi che accadono frequentemente durante l'utilizzo di interfacce grafiche, PyQT ha pensato di creare dei metodi specifici.

Qua sotto se ne elencano alcuni presenti nel programma:

- mousepressevent: che serve ad eseguire una funzione che necessita delle informazioni legate all'utilizzo del mouse (come nel caso detto sopra);
- clicked.connect: che permette di eseguire una funzione, subito dopo che l'utente ha schiacciato il bottone corrispondente dell'interfaccia grafica (dentro DIALOG_CALIBRATION.py serve a selezionare quale fase della calibrazione eseguire, oppure serve a far ripartire lo stream delle telecamere);
- setStatusTip: che permette di eseguire una funzione, subito dopo che l'utente ha cliccato sulla riga corrispondente di un menù a tendina (all'interno di MAIN_APP.py serve a comandare l'apertura delle altre finestre).

Si nota quindi che ogni widget ha dei metodi specifici che servono a rispondere in modo appropriato agli input dell'utente. Sono veramente tanti, per questo si è deciso di spiegare solo quelli più significativi.

Con il meccanismo segnali/cave si conclude l'ultimo paragrafo che descrive il modo in cui le diverse parti del programma sono collegate tra loro. Nei prossimi paragrafi seguirà la spiegazione dettagliata delle diverse parti del programma, ordinate nella sequenza con cui esse devono essere eseguite nel caso dello svolgimento completo dell'applicazione:

1. inizializzazione;
2. prima fase di calibrazione;
3. seconda fase di calibrazione;
4. cattura dei frames comandata dalle fotocellule;
5. elaborazione delle pointcloud.

5.9.5 Inizializzazione

Per procedere all'inizializzazione bisogna, per prima cosa, far partire il programma dell'applicazione completa. Quindi, come detto anche nei

paragrafi precedenti, è necessario eseguire WINDOWS_CONTROLLER.py.

Questa parte di codice carica il modello (la classe Cameras in MODEL_RS2.py) e subito dopo impone l'esecuzione dei metodi FNC_Initialize ed FNC_SetupConnection, che sono rispettivamente, il setup delle telecamere e poi quello delle fotocellule.

All'interno di FNC_Initialize, si trovano:

- la rilevazione delle telecamere, e l'ordinamento in base al proprio codice all'interno di una lista;
- il caricamento di un file che contiene le impostazioni della telecamera, riguardanti esposizione, fuoco, ecc.;
- l'avviamento dello streaming delle telecamere, sia quello a colori, che quello della profondità.

Invece, all'interno di FNC_SetupConnection avviene il collegamento con le fotocellule mediante il PLC. Per avvenire, è necessario inserire l'indirizzo IP corretto del Client, altrimenti non si riusciranno a ricevere i segnali delle fotocellule. L'informazione inviata dalle fotocellule non è altro che il numero di volte che un oggetto qualsiasi ha attraversato il raggio laser emesso dal sensore.

Subito dopo questi due funzioni, WINDOWS_CONTROLLER.py inizia a gestire l'apertura delle finestre:

- Carica e mostra la Mainwindow che si trova in MAIN_APP.py
- Attiva il Thread che si trova all'interno di MAIN_APP.py

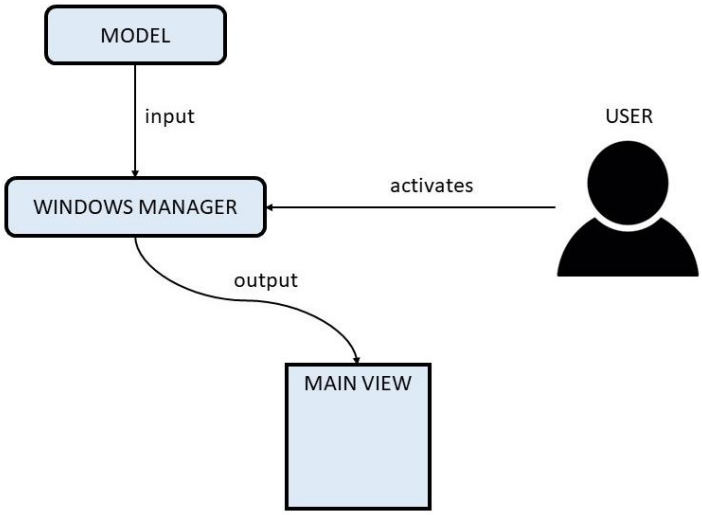


Figura 5.38: Schema dell'apertura della finestra principale

Quindi, ora che si è aperta la Vista principale è possibile verificare il funzionamento delle telecamere stereoscopiche, sia tramite le etichette che mostrano i frames dello streaming a colori, che tramite la textbox, che se tutto è andato come previsto, dovrebbe mostrare un messaggio che indica che l'inizializzazione è andata a buon fine.

A questo punto, tramite un menù a tendina che si apre cliccando sulla scritta Settings, è possibile selezionare la finestra di dialogo che si vuole aprire, nel nostro caso quella della calibrazione.

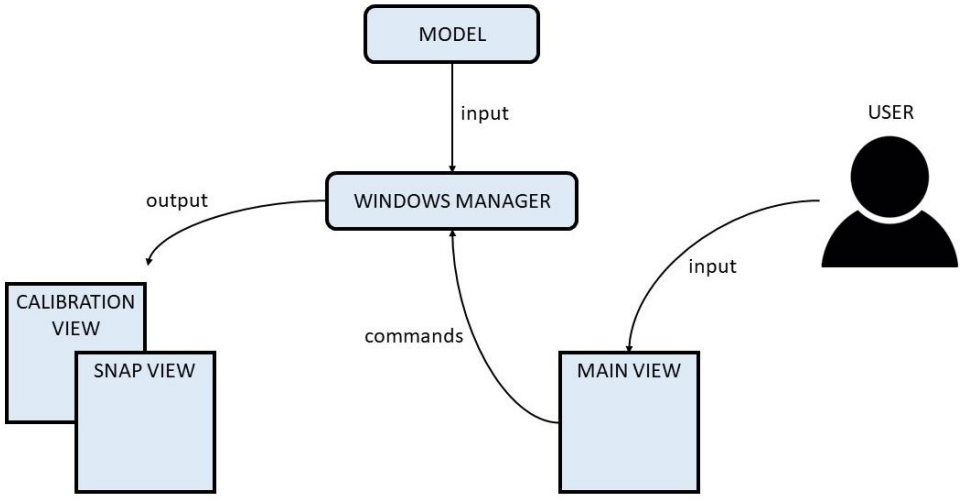


Figura 5.39: Schema dell'apertura delle finestre di dialogo

5.9.6 Prima fase di calibrazione

L'apertura della finestra di interfaccia della calibrazione avviene per mezzo del metodo FNC_ShowCalibration che si trova all'interno del WINDOWS_CONTROLLER.py. Esso permette l'esecuzione della classe DialogWindow all'interno del file DIALOG_CALIBRATION.py, il che consente l'inizio del processo della calibrazione.

Questa Vista permette la visualizzazione contemporanea di entrambi gli stream delle telecamere, ovvero sia quello a colori che quello della profondità.

Inoltre, sono presenti 5 bottoni:

- StartCalib (che attiva FNC_StartCalibration);
- SaveCalib (che attiva FNC_SaveCalibration);
- LoadCalib (che attiva FNC_LoadCalibration);
- FindCoord (che attiva FNC_FindCoordinates);

- RestartFrames (che attiva FNC_RestartFrames);

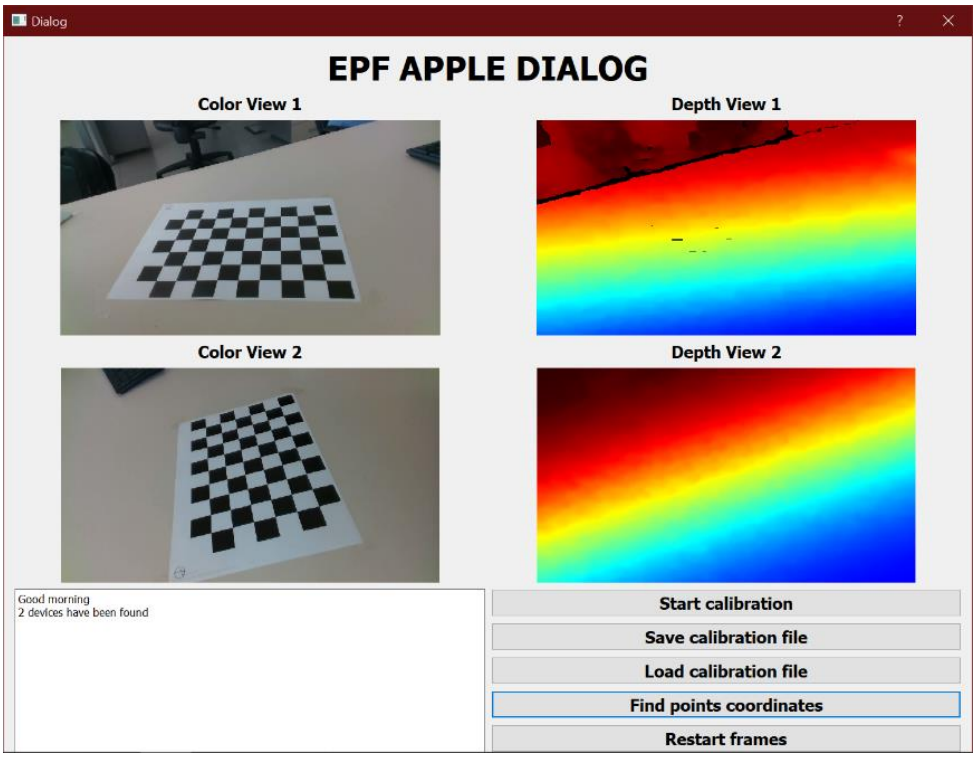


Figura 5.40: Finestra di interfaccia della calibrazione

Si può subito notare che è possibile caricare le matrici di trasformazione schiacciando il bottone LoadCalib nel caso in cui l'operazione di calibrazione fosse già avvenuta, ed è possibile salvare i dati frutto della calibrazione attuale tramite il bottone SaveCalib.

Schiacciando il bottone StartCalib si può avviare la calibrazione, mentre schiacciando il bottone FindCoord è possibile verificare che le matrici di trasformazione calcolate siano corrette provando a rilevare le coordinate di punti scelti dall'operatore sulle etichette dello stream a colori.

RestartFrames, invece, garantisce la ripartenza degli stream delle telecamere, dopo che essi sono stati interrotti per necessità di riduzione degli errori.

Il processo di Calibrazione è molto lungo e complesso, per cui è stato diviso in due fasi distinte: in questo paragrafo ci concentreremo sulla prima fase.

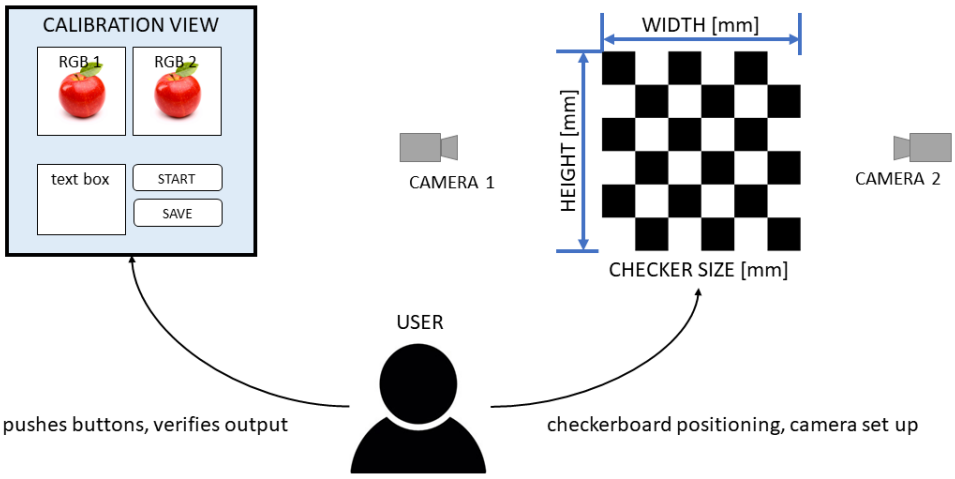


Figura 5.41: Schema funzionale della prima fase di calibrazione

La prima fase consiste nelle parti seguenti:

1) impostare i parametri della scacchiera nel codice della classe Cameras all'interno del file MODEL_RS2.py;
2) prendere gli INTRINSICS e gli EXTRINSICS delle telecamere;
3) stimare la posa della scacchiera e le coordinate relative utilizzando il metodo di Kabsch;
4) salvare il file della prima fase di calibrazione.

Per prima cosa è necessario che all'interno del codice che esegue la calibrazione, siano presenti i dati che riguardano la scacchiera, ovvero:

- altezza (in numero di scacchi);
- larghezza (in numero di scacchi);
- dimensione del lato di uno scacco.

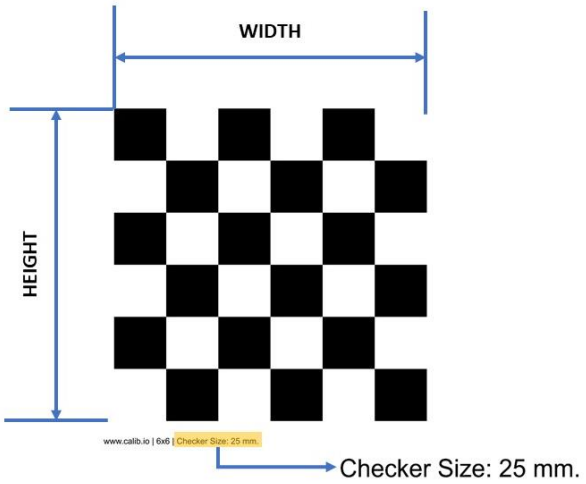


Figura 5.42: Parametri della scacchiera di calibrazione

Infatti, è possibile eseguire la calibrazione con scacchiere di diverse dimensioni, anche se in generale è stato verificato che:

- l'applicazione fatica ad individuare una scacchiera con scacchi avente lati uguali o inferiori a 10mm;
- la scacchiera non può essere troppo grande in quanto deve essere completamente visibile all'interno del campo di visione delle telecamere.

Oltre ai dati della scacchiera, per far avvenire la calibrazione è necessario conoscere dei parametri della telecamera chiamati Extrinsic

ed Intrinsics. Il programma li ricava dai frame inviati dalle telecamere nel momento in cui avviene la fase della loro inizializzazione.

Se tutti questi dati sono stati inseriti nel codice, allora è possibile proseguire cliccando sul tasto StartCalib. Finchè la scacchiera non sarà visibile dalle telecamere, il programma non sarà in grado di proseguire e mostrerà un avviso nella textbox della finestra.

Per procedere, la scacchiera deve essere posta sul piano di riferimento, su cui andremo ad appoggiare l'oggetto di riferimento. Mentre la scacchierà servirà ad individuare la localizzazione del piano di riferimento, l'oggetto di riferimento servirà ad individuare l'orientazione degli assi di riferimento e permetterà anche di rilevare le distanze nella unità di misura corretta (millimetri).

La rilevazione della scacchiera avviene confrontando la depthmap delle telecamere e lo stream a colori con i dati forniti all'utente sulla scacchiera e sugli Intrinsics ed Extrinsics delle telecamere.

Dopo la rilevazione della scacchiera, si passa alla stima della posa della stessa, mediante l'algoritmo di Kabsch. Con questa procedura si riesce a passare da un sistema di riferimento attaccato alla telecamera, ad un sistema di riferimento, sempre legato alla singola telecamera, ma appoggiato sulla scacchiera.

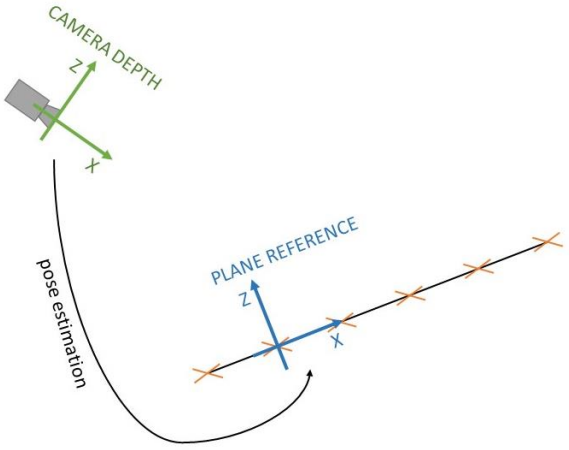


Figura 5.43: Passaggio dalla depth map al piano di riferimento

La matrice di trasformazione che permette di trovare queste nuove coordinate relative, insieme ai dati di intrinsics ed extrinsics, vengono salvati all'interno di una variabile di tipo dictionary chiamata first_calibration_results. A questo punto, nella textbox dell'interfaccia grafica apparirà il messaggio di avviso della fine della prima fase della calibrazione. La seconda fase inizierà automaticamente, non c'è bisogno di premere alcun bottone.

Per chi è interessato, nel prossimo paragrafo verrà spiegato il funzionamento dell'algoritmo di Kabsch. Chi, invece, non lo è, può proseguire al paragrafo successivo che tratta della seconda fase di calibrazione.

L'algoritmo di Kabsch

L'algoritmo di Kabsch, così chiamato da Wolfgang Kabsch, è un metodo che serve a calcolare la matrice di rotazione ottimale che minimizza l'RMSD (root mean squared deviation) tra due insiemi di punti.

L'algoritmo calcola solo la matrice di trasformazione, ma richiede anche il calcolo del vettore di traslazione. Quando sia il vettore di traslazione che la matrice di rotazione vengono calcolati, l'algoritmo viene chiamato "partial Procrustes superimposition".

L'algoritmo per il calcolo della rotazione di P in Q inizia dall'accoppiamento dei due set di punti. Nello spazio 3D ogni set di punti può essere rappresentato come una matrice N x 3:

$$\begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_N & y_N & z_N \end{pmatrix}$$

Entrambi i set di coordinate devono essere, per prima cosa, traslati, in modo che i centroidi coincidano con l'origine del sistema di coordinate. Ciò avviene sottraendo dai punti delle coordinate, le coordinate dei rispettivi centroidi.

La seconda fase consiste nel calcolo della matrice H di cross-varianza.

- in notazione matriciale come: $H = P^T Q$
- In notazione sommatoriale: $H_{ij} = \sum_{k=1}^{N} P_{ki} Q_{kj}$

Si può calcolare la matrice di rotazione ottimale R basandosi sulla formula matriciale: $R = (H^T * H)^{\frac{1}{2}} H^{-1}$

Se sono disponibili routines di decomposizione a valore singolo (SVD), la matrice di rotazione ottimale R, si può calcolare usando il seguente algoritmo:

- Prima calcolare l'SVD della matrice di covarianza H.
- Poi, decidere se c'è bisogno di correggere la matrice di rotazione in modo da avere un sistema di coordinate destrorso.
- Infine, calcolare la matrice di rotazione ottimale R come:

$$R = V \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{pmatrix} U^{\mathsf{T}}$$

5.9.6 Seconda fase di calibrazione

Come la prima fase, anche la seconda è costituita da diverse parti:

1) prendere le coordinate relative dei 3 punti dell'oggetto di riferimento, una volta per ogni camera;
2) calcolare le matrici di trasformazione per passare dalle coordinate relative a quelle assolute;
3) salvare il file della seconda fase di calibrazione.

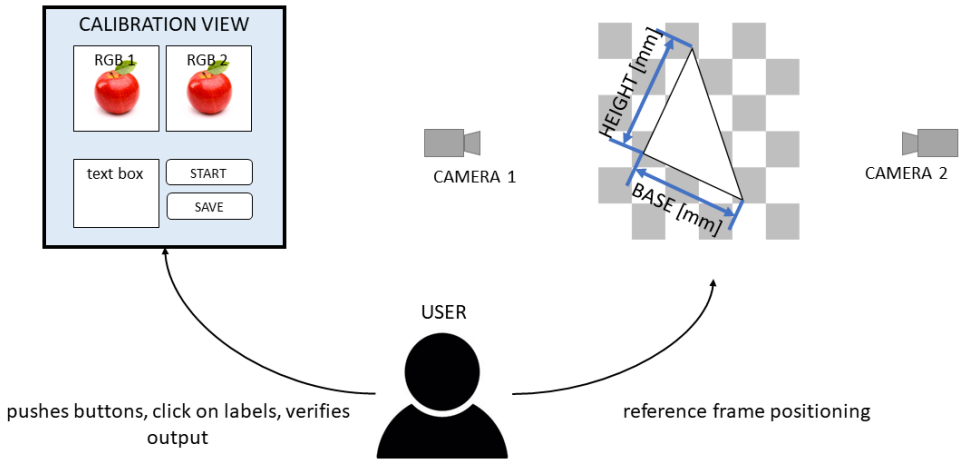


Figura 5.44: Schema funzionale della seconda calibrazione

All'inizio apparirà un avviso che indicherà all'operatore di cliccare sul primo punto di riferimento sulla prima telecamera. Ciò presuppone che l'operatore capisca che deve appoggiare l'oggetto di riferimento sul piano in modo tale da vedere dove si trovano i punti di riferimento da cliccare.

A questo punto si bloccherà lo stream delle telecamere in modo da non introdurre errori dovuti allo spostamento dell'oggetto di riferimento.

Dopo aver cliccato il primo punto, apparirà il messaggio per il secondo punto e poi quello per il terzo. Subito dopo sarà necessario ripetere l'operazione anche per la seconda telecamera. Ogni volta che si clicca su un punto dell'etichetta che contiene lo stream a colori della telecamera, l'applicazione dovrà:

- convertire le coordinate della posizione del puntatore nelle coordinate corrispondenti all'interno della depthmap
- convertirà le coordinate della depthmap con quelle relative collegate alla scacchiera utilizzando la matrice di trasformazione calcolata nella fase di calibrazione precedente.

Dopo aver rilevato la posizione dei 3 punti di riferimento rispetto ad entrambe le telecamere, si passa al calcolo delle matrici di trasformazione, che permetteranno di convertirli in coordinate assolute. SI ricorda che le coordinate assolute dei 3 punti sono state inserite all'interno del codice nel metodo FNC_Initialize che si trova dentro MODEL_RS2.py.

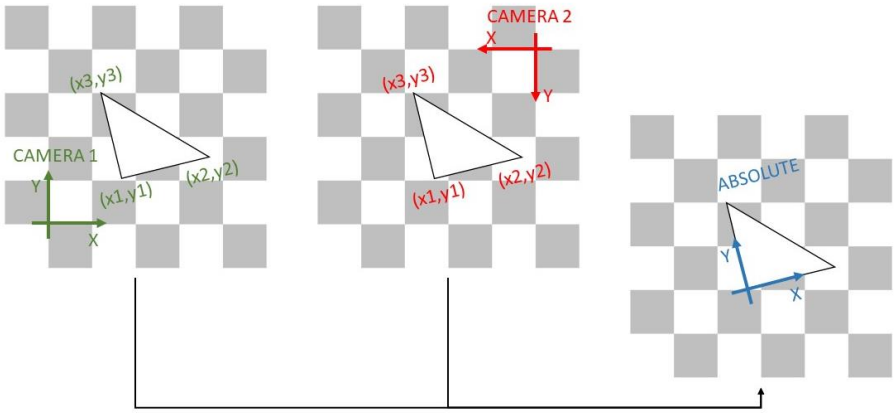


Figura 5.45: Passaggio da sistemi di riferimento relativi al sistema di riferimento assoluto

Dopo aver calcolato le due matrici (una per ogni telecamera), queste vengono salvate all'interno di una variabile di tipo lista chiamata second_calibration_results. Successivamente, apparirà il messaggio di avviso che indica la fine della calibrazione.

Se l'operatore lo ritiene opportuno a questo punto sarà in grado di procedere al salvataggio delle variabili first_calibration_results e second_calibration_results cliccando sul bottone SaveCalib. L'utente sarà in grado di salvare questi due file con estensione .json dovunque e con qualsiasi nome all'interno del PC in modo tale da facilitare il riconoscimento dei file ed il riutilizzo.

Il calcolo di queste matrici di trasformazione rigida o Euclidea avviene sempre seguendo l'algoritmo di Kabsch, che è stato spiegato nel paragrafo precedente.

5.9.7 Cattura dei frames comandata dalle fotocellule

Per passare a questa operazione è necessario aprire la finestra di dialogo corrispondente dalla finestra principale, esattamente come nel caso della calibrazione. Anche il modo in cui è codificata la gestione delle finestre è la stessa. Se necessario, è possibile tenere aperte tutte e 3 le finestre contemporaneamente senza alcun problema.

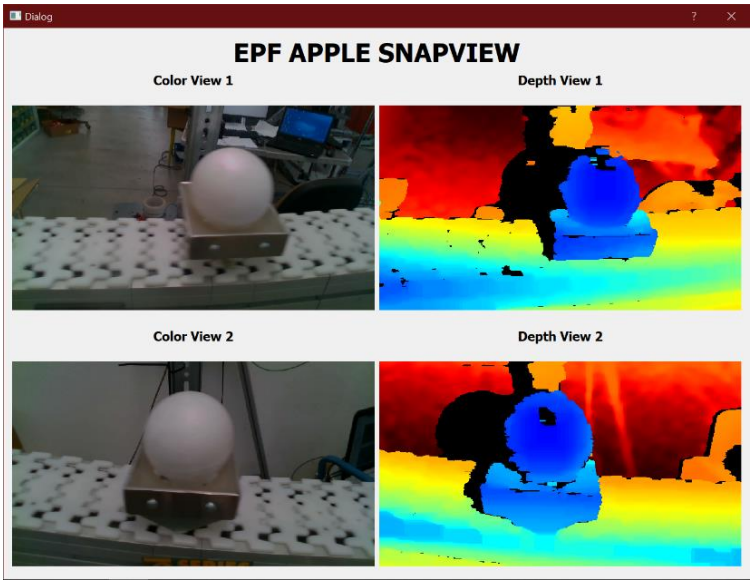


Figura 5.46: Finestra di interfaccia per la gestione di questa fase

Questo procedimento è quello più curato e critico, in quanto le operazioni devono avvenire all'interno di una finestra di tempo molto ristretta e la qualità delle immagini catturate deve essere abbastanza elevata. Si può dividere in 5 passaggi fondamentali:

- caricamento dei risultati della calibrazione;
- lettura del segnale del passaggio del facchino dalla fotocellula;
- cattura degli stream dalle telecamere stereoscopiche;
- calcolo della matrice delle coordinate relative del frame appena catturato;
- salvataggio delle immagini degli stream e della matrice delle coordinate relative.

La prima cosa che accade durante l'apertura di questa finestra è il caricamento delle matrici di trasformazione calcolate durante la fase di calibrazione. Questa fase è fondamentale, non tanto nel caso dell'applicazione complessiva, bensì nel caso dell'apertura di questa singola finestra. I dati caricati, infatti, vengono messi all'interno delle variabili first_calibration_results e second_calibration_results, che nel

caso dell'applicazione completa, arrivati a questo punto, dovrebbero già contenere i risultati della calibrazione.

Successivamente la cattura dei frames avviene esattamente allo stesso modo con cui avveniva per le altre due finestre, ma deve avvenire solo quando l'oggetto passa il laser del sensore. Quindi, anziché utilizzare il metodo FNC_GetFrames, si è deciso di creare un metodo nuovo, chiamato FNC_SnapFrames. Al suo interno non avviene solo la cattura delle immagini, bensì tutto ciò che è necessario ad effettuare questo processo.

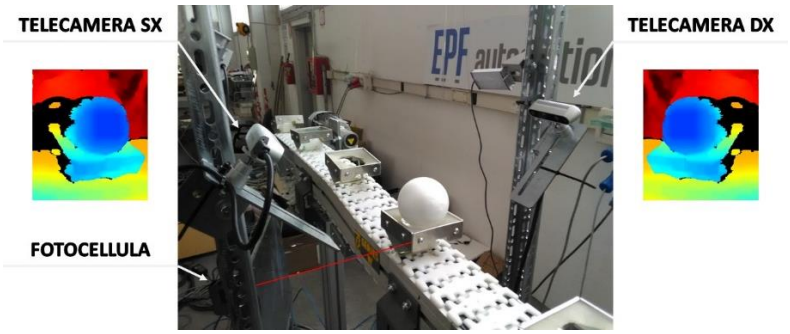


Figura 5.47: Cattura delle immagini 3D sul nastro trasportatore

Come FNC_GetFrames, questo metodo viene eseguito periodicamente in quanto viene richiamato da un Timer, ma in realtà solo la parte della lettura delle fotocellule avviene di continuo. Il resto avviene solamente se il segnale che arriva dalle fotocellule è maggiore di zero, ovvero solo se un oggetto è andato ad oscurare il laser. Dopo aver catturato i frames, dopo aver calcolato la matrice delle coordinate relative e dopo il salvataggio, si trova una linea di codice che azzera la variabile legata al segnale inviato dalle fotocellule. In questo modo il processo non si ripete e le "foto" vengono scattate e salvate una volta sola.

Si è già detto che la cattura dei frame avviene allo stesso modo, ma bisognerebbe precisare il fatto che c'è una piccola differenza nel post-processing. Nel caso della calibrazione è fondamentale che le mappe di profondità con cui vengono calcolate le matrici di trasformazione siano

prive di buchi. Per farlo si ricorre a due filtri: uno spaziale ed uno temporale. Invece, in questa istanza è assolutamente impossibile accettare un filtro temporale in quanto si stanno catturando delle immagini in movimento. Conseguentemente all'interno di FNC_SnapFrames si può trovare solo l'applicazione del filtro spaziale.

Il calcolo delle coordinate relative ricavate dalle depthmap provenienti dalle telecamere avviene come nella parte di codice di FNC_GetFrames che viene attivata durante la seconda fase della calibrazione, quando è necessario rilevare le coordinate relative dei punti di riferimento.

Sia i frames catturati dagli stream che la matrice delle coordinate relative vengono immagazzinati all'interno della cartella "frames" con un nome che indica il tipo di oggetto (color_frame, o depth_frame, oppure relative_points_matrix) ed il momento (anno/mese/giorno/ora/minuto/secondo) in cui è stato salvato. In questo modo sono facilmente ritrovabili per essere utilizzate nella prossima fase.

5.9.8 Elaborazione delle Point Cloud

Quest'operazione, a differenza di tutte le altre, avviene al di fuori dell'applicazione stessa. Al suo interno accadono le seguenti operazioni elementari:

- caricamento dei risultati della calibrazione;
- caricamento di una matrice di coordinate relative;
- calcolo della matrice di coordinate assolute corrispondenti;
- elaborazione delle Point Cloud relative ad ogni telecamera;
- elaborazione e visualizzazione della Point Cloud complessiva.

Queste operazioni non sono state inserite all'interno di DIALOG_SNAPVIEW.py, perché in realtà all'operatore non interessa visualizzare la PointCloud complessiva di ogni mela che passa sul nastro trasportatore. L'informazione delle coordinate assolute 3D dei punti che compongono la mela interessa solamente il robot, quindi è giusto che queste righe di codice siano scritte all'interno di un programma separato. Inoltre, è necessario specificare che separato non significa slegato: infatti anche questa parte interagisce con la classe Cameras, ovvero il Modello dell'applicazione complessiva.

Questa parte di programma non ha bisogno di una vera e propria interfaccia grafica per funzionare. Una volta che il programma è partito, l'utente non ha bisogno di fare più niente, se non aspettare che venga mostrata a schermo l'immagine 3D (navigabile) della PointCloud complessiva. Quindi non si può dire che abbia una vera e propria Vista.

Per creare un codice che consenta Il calcolo della matrice delle coordinate assolute, si è preso spunto dalla funzione FNC_PointCoord, che può essere eseguita schiacciando il bottone FindCoord all'interno della finestra della calibrazione. Questo metodo è in grado di calcolare le coordinate assolute di un punto a piacere all'interno di un frame. Ripetendo questo processo per tutti i punti che si trovano all'interno della matrice delle coordinate relative, e cambiando matrice di trasformazione quando si passa da una telecamera all'altra, si riescono

a mettere, all'interno di una matrice, le coordinate assolute di tutti i punti delle depthmap provenienti da entrambi i dispositivi.

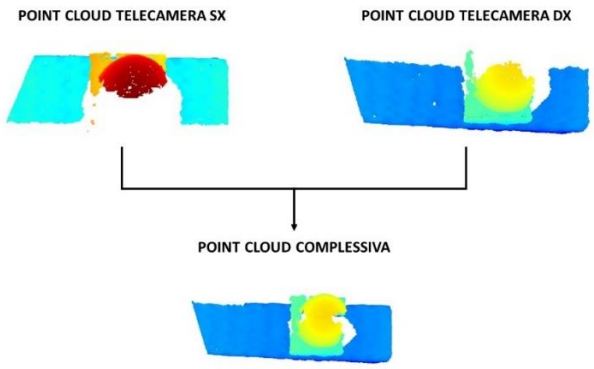


Figura 5.48: Elaborazione dell'immagine 3D complessiva

Infine, l'elaborazione delle PointCloud avviene con pochissime righe di codice in modo quasi automatico, utilizzando il modulo Open3D presente all'interno di Python. L'immagine della PointCloud complessiva è navigabile: è possibile spostare il punto di vista e aumentare/diminuire lo zoom. I punti hanno un colore che varia dal blu al rosso in base alla coordinata Z.

vista laterale **vista frontale**

Figura 5.49: Confronto tra due viste di una point cloud

Il risultato varia molto in base a come viene eseguito il post-processing dei frame nella fase precedente a questa. Inoltre, si può vedere che ci sono zone più povere di punti. Ciò dipende fortemente dal modo in cui sono orientate le telecamere rispetto agli oggetti. Le superfici perpendicolari alle telecamere si vedranno benissimo, mentre quelle più inclinate si vedranno molto meno. Se con due telecamere si riesce a vedere la maggior parte della mela, con una terza telecamera si dovrebbe riuscire a vederla nella sua interezza.

CAPITOLO 6

6. Collaudo del software

Per mettere a punto un sistema qualsiasi è sempre necessario sottoporlo ad un processo di collaudo. Infatti, per quanto le simulazioni si possano avvicinare alla realtà, esse non potranno mai sostituirla completamente.

In questo caso specifico, il sistema è stato provato più volte durante la fase di sviluppo, seguendone la crescita:

- Dapprima sono state testate le telecamere e si è verificato che il calcolo delle coordinate avvenisse correttamente
- Successivamente si è provato a disporre le telecamere in modi diversi, per capire quale potesse essere la configurazione più conveniente

Questi esperimenti servono a verificare il funzionamento del programma e a configurare le telecamere nel modo migliore possibile per eseguire il compito in modo preciso ed efficiente. Nei prossimi paragrafi di vedrà come una prova ben impostata può essere utile per identificare errori nel software e come può essere impiegata per indirizzare lo sviluppo.

6.1 Verifica del calcolo delle coordinate

Dopo aver scritto in Python il programma di calibrazione delle telecamere, si è deciso di testarne la precisione delle stesse sul campo, in modo tale da capire se sarebbe stato utile proseguire con lo sviluppo o meno.

Guardando i dati forniti dal costruttore delle telecamere, si sarebbe dovuto ottenere un errore inferiore ad 1mm, il che è sufficientemente piccolo, in quanto il robot, per l'afferraggio delle mele, si avvale di una ventosa e riuscirebbe sicuramente a compiere il suo compito.

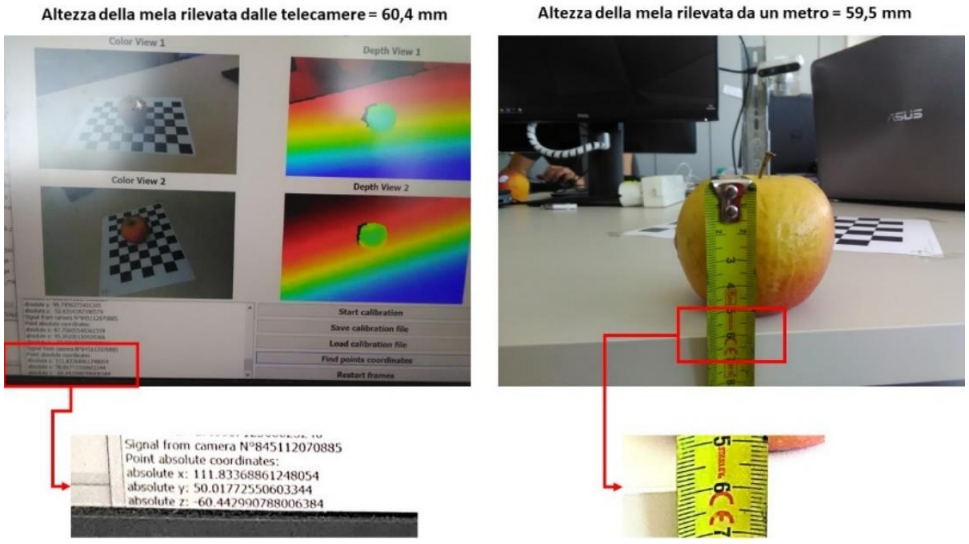


Figura 6.1: Precisione delle telecamere dopo la correzione del bug

Nonostante i dati forniti si è deciso di proseguire con la sperimentazione in quanto si era già notato, durante l'impostazione delle telecamere, che esse sono molto sensibili alla luce e quindi si voleva capire in che condizioni esse funzionassero al meglio. Inoltre, questo collaudo era comunque necessario per verificare il corretto funzionamento del programma. Infatti, la prima volta in cui sono state eseguite le prove,

l’errore effettuato dalle telecamere era molto superiore a ciò che era stato previsto dal costruttore, e successivamente si è riuscito a concludere che era dovuto ad un errore all’interno del programma.

Per svolgere questa verifica, non si è impostata una prova singola. Se ne sono impostate diverse, in modo tale che ognuna si concentrasse sull’analisi della precisione delle coordinate su assi diversi. Inizialmente si sono realizzate delle prove con lo scopo di approfondire la precisione delle coordinate X ed Y sui punti della scacchiera, infine sono stati posti degli oggetti sulla scacchiera in modo da analizzare la precisione anche sull’asse Z.

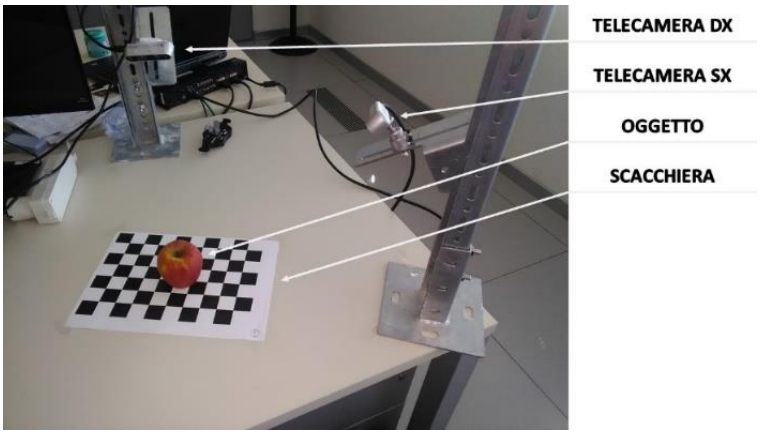


Figura: 6.2: Oggetti utilizzati per l’impostazione delle prove

Nella prima prova, per analizzare la distribuzione degli errori sulle coordinate X ed Y, sono state rilevate, tramite entrambe le telecamere, le coordinate di un certo numero di punti noti sulla scacchiera. La procedura è stata eseguita per due sistemi di riferimento differenti, indicati sulla figura sottostante, in modo da ottenere una maggiore comprensione del problema.

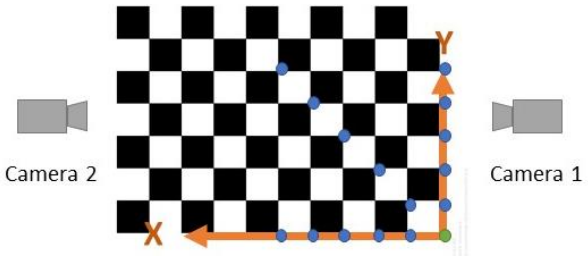


Figura 6.3: Schema della prova con s.r. all'angolo

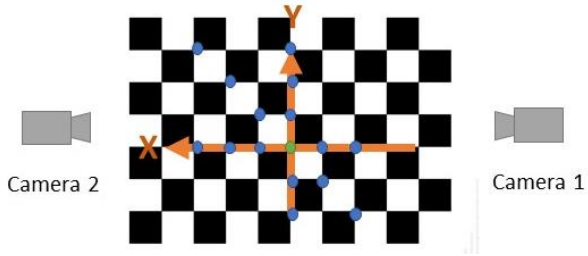
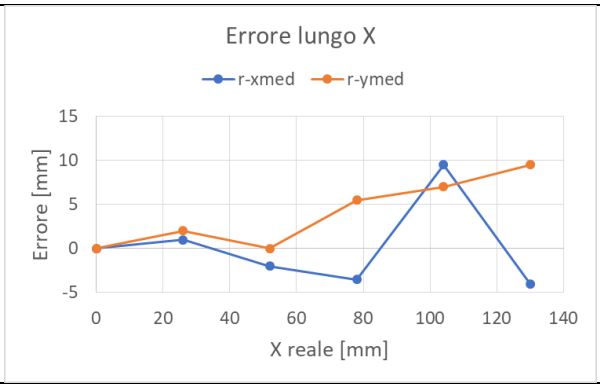
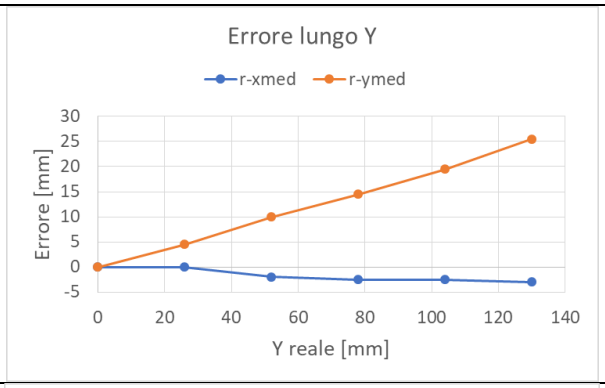


Figura 6.4: Schema della prova con s.r. centrato

Dopo aver rilevato le coordinate dei punti indicati in blu, si è diagrammato l'errore in funzione della distanza lungo le direzioni X ed Y, in modo tale da controllare se ci fosse una correlazione tra le due variabili.

Tabella 6.1: Andamento dell'errore per S.R. all'angolo




Errore lungo Y
r-xmed
r-ymed
Errore [mm]
30
25
20
15
10
5
0
-5
0
20
40
60
80
100
120
140
Y reale [mm]

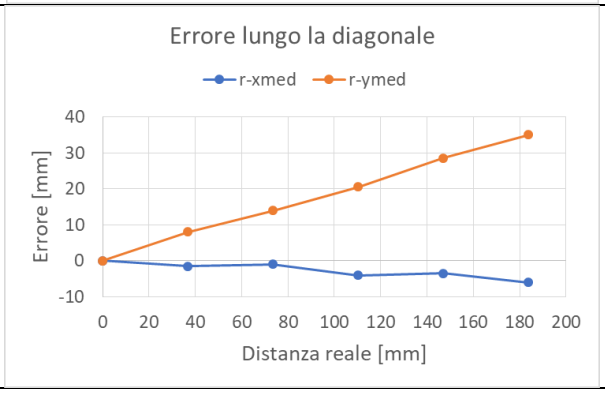
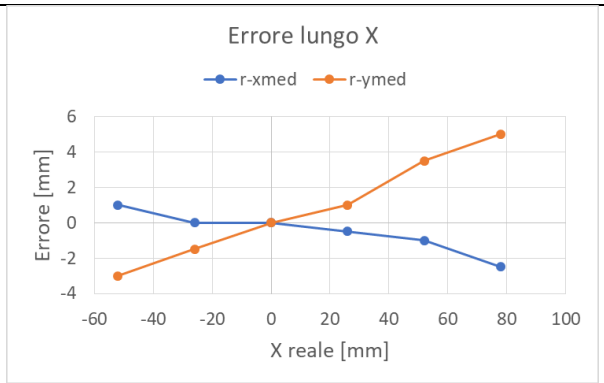
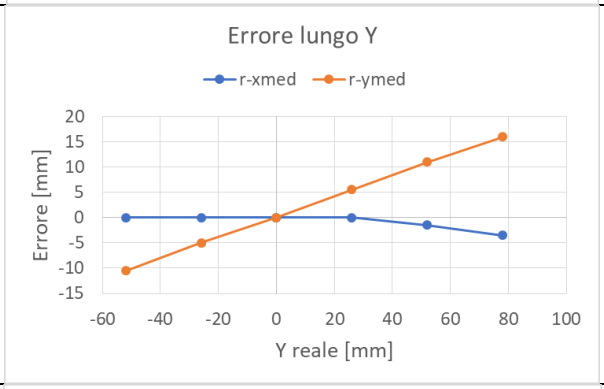
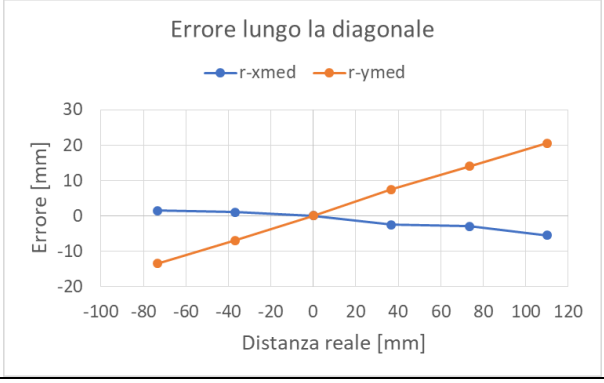

Errore lungo la diagonale
r-xmed
r-ymed
Errore [mm]
40
30
20
10
0
-10
0
20
40
60
80
100
120
140
160
180
200
Distanza reale [mm]

Tabella 6.2: Andamento dell'errore per S.R. centrato







Nella seconda prova, per approfondire ulteriormente l'analisi, si è ripetuta la stessa procedura della prima prova, cambiando l'orientazione della scacchiera. Oltre all'andamento dell'errore sulle coordinate, questo test ci ha permesso di capire che il programma assegna sempre l'asse x al lato lungo e l'asse y al lato corto. Questa

volta la prova è stata effettuata tenendo in considerazione un solo sistema di riferimento, in quanto la differenza di risultati tra sistemi di riferimento diversi era già stata trattata nella prova precedente.

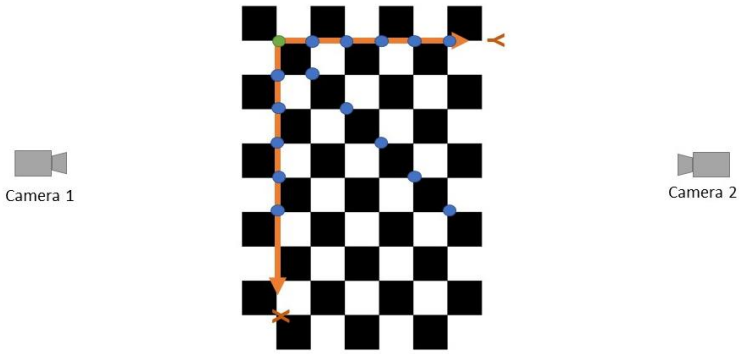
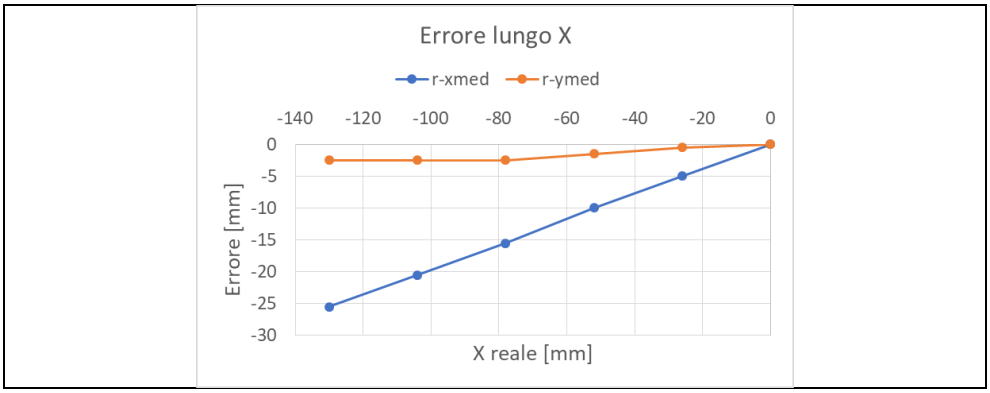
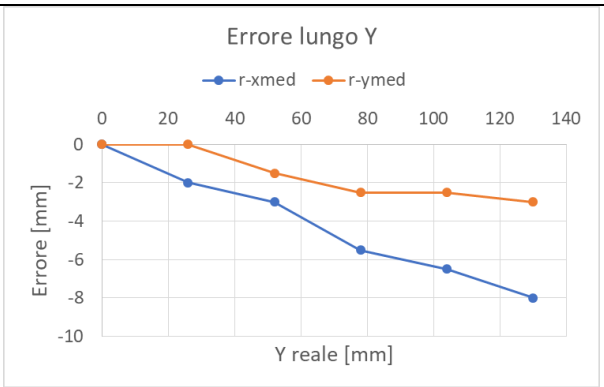


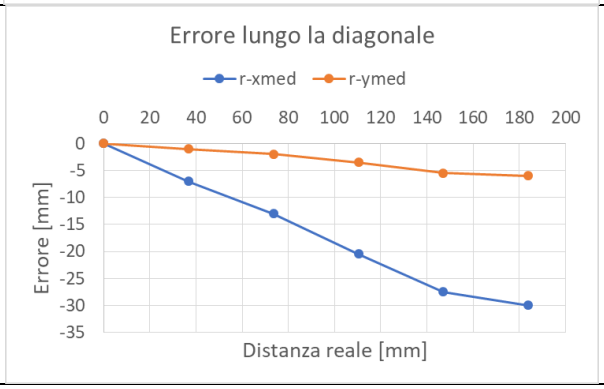
Figura 6.5: Schema della seconda prova

Dopo aver rilevato le coordinate dei punti indicati in blu, si è diagrammato l'errore in funzione della distanza lungo le direzioni X ed Y, in modo tale da controllare se ci fosse una correlazione tra le due variabili.

Tabella 6.3: Andamento dell'errore nella seconda prova







In entrambe le prove si può già intravedere che l'errore ha un andamento lineare con la distanza, ciò che però le differenzia è il fatto che mentre la prima presenta un errore maggiore in X, la seconda presenta un errore maggiore in Y. Si potrebbe concludere che l'errore maggiore si riscontra sempre nella lettura delle coordinate che variano lungo la direzione orizzontale del campo visivo della telecamera. Per accertare questa ipotesi si è deciso di impostare una terza prova, in cui si è posta la scacchiera in obliquo (quasi a 45°) rispetto alla direzione delle due telecamere.

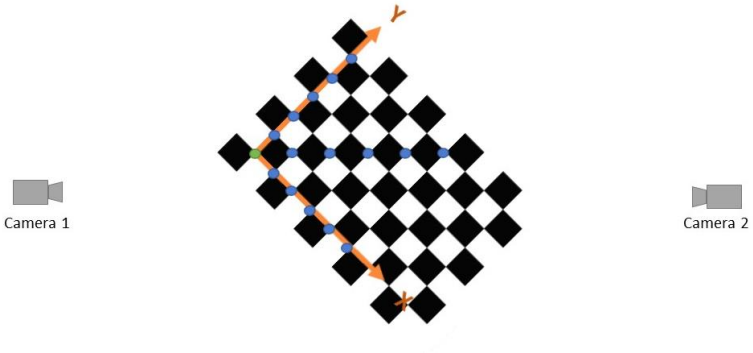
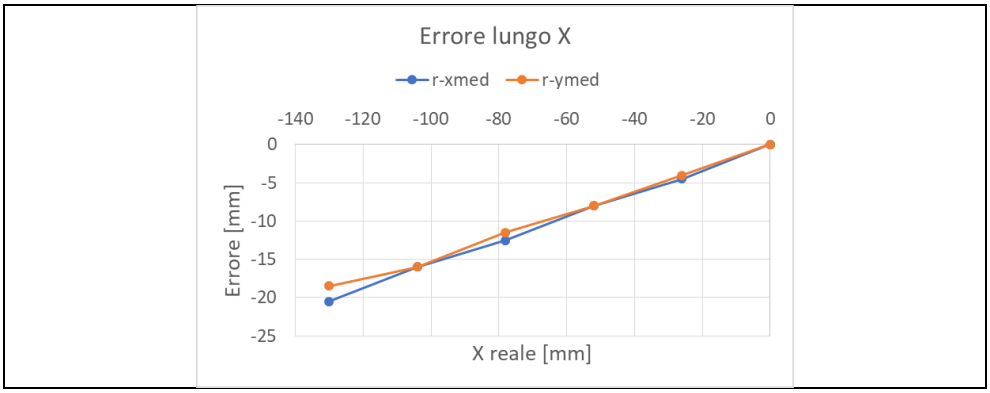
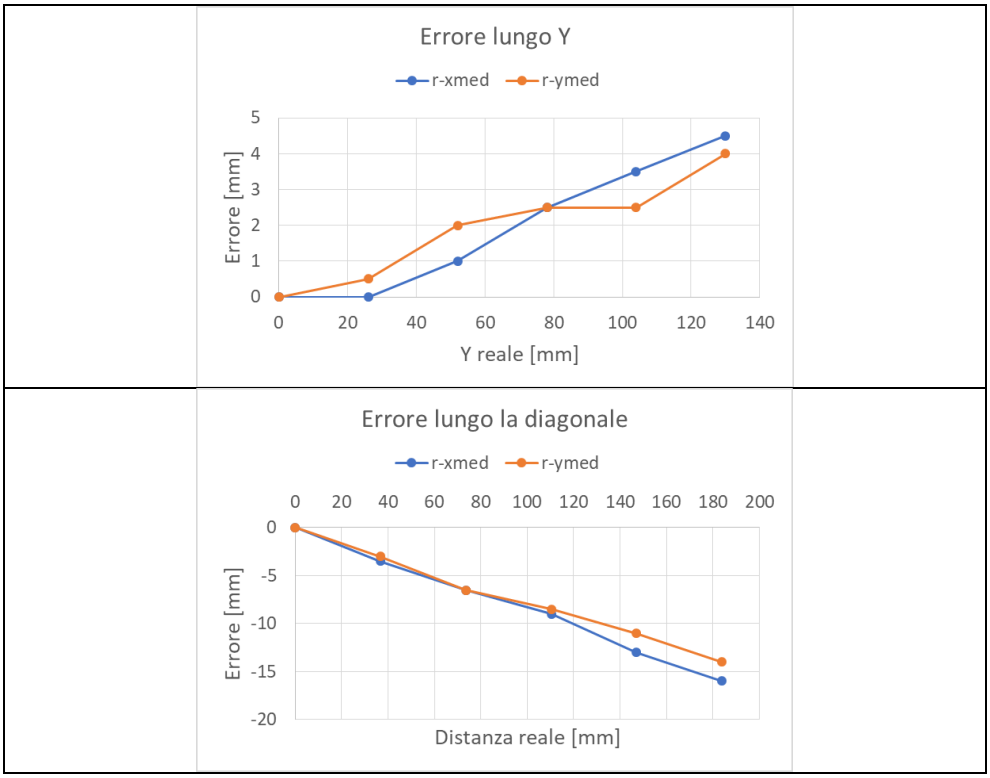


Figura 6.6: Schema della terza prova

Dopo aver rilevato le coordinate dei punti indicati in blu, si è diagrammato l'errore in funzione della distanza lungo le direzioni X ed Y, in modo tale da controllare se ci fosse una correlazione tra le due variabili.

Tabella 6.4: Andamento dell'errore nella terza prova





Dal grafico dell'errore lungo la diagonale si può concludere che l'ipotesi è verificata. Infatti, l'errore in X è lo stesso dell'errore in Y, proprio perché il piano di riferimento è inclinato di 45° rispetto alla direzione in cui punta la telecamera.

Quindi, dopo queste tre prove si può affermare che:

- In qualsiasi direzione ci si muova, l'errore determinante è sempre quello compiuto lungo il FOV orizzontale della camera stessa. A causa di questo errore l'impiego della telecamera risulta limitata a distanze dall'origine inferiori ai 3 o 4 cm.
- L'errore nell'altra direzione, invece, c'è ma è molto minore, quasi da risultare trascurabile in quanto l'applicazione non avrebbe bisogno di rilevare coordinate a distanze dall'origine maggiori di 10cm.

Prima di trarre delle conclusioni ed intervenire, è stata impostata un'altra prova per osservare il comportamento dell'errore anche sull'asse Z.

Per esaminare il comportamento del sistema in piani diversi da quello di riferimento sono stati utilizzati degli oggetti di altezza conosciuta. Con ogni oggetto si è ricreata una griglia di punti ad altezza costante posizionandolo su punti noti della scacchiera.

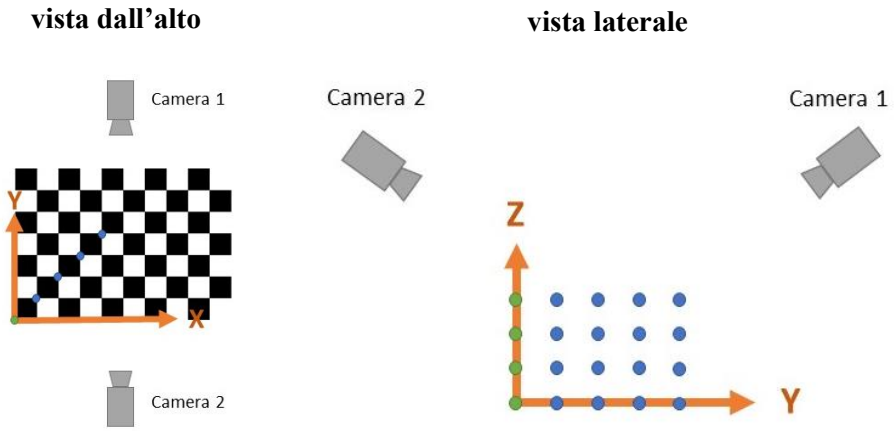
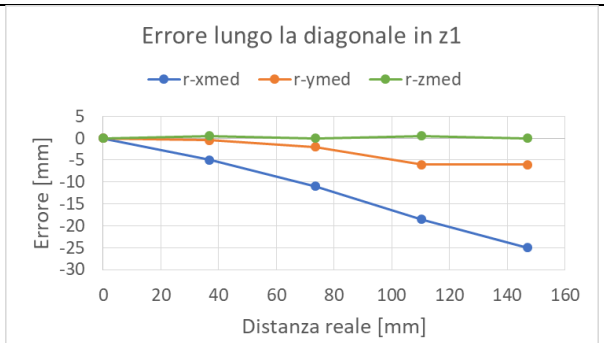
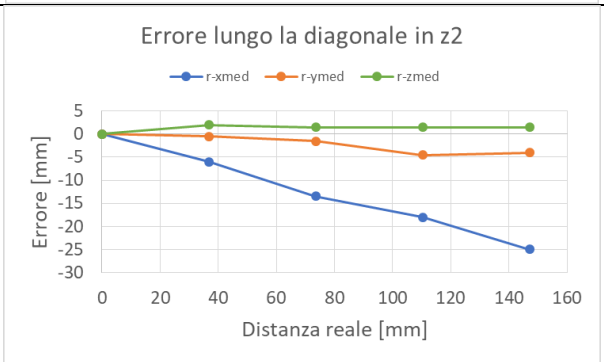


Figura 6.7: Schema della quarta prova

Dopo aver rilevato le coordinate dei punti indicati in blu, si è diagrammato l'errore in funzione della distanza lungo le direzioni X, Y e Z, in modo tale da controllare se ci fosse una correlazione tra le variabili.

Tabella 6.5: Andamento dell'errore lungo la diagonale





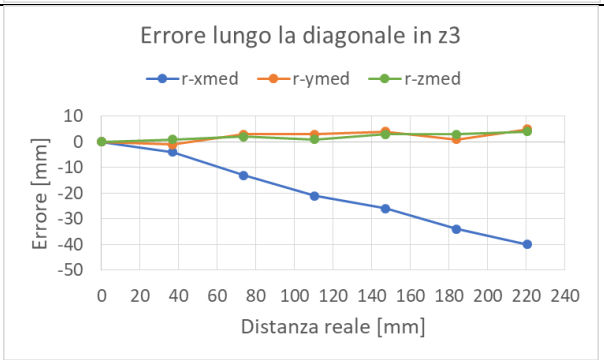
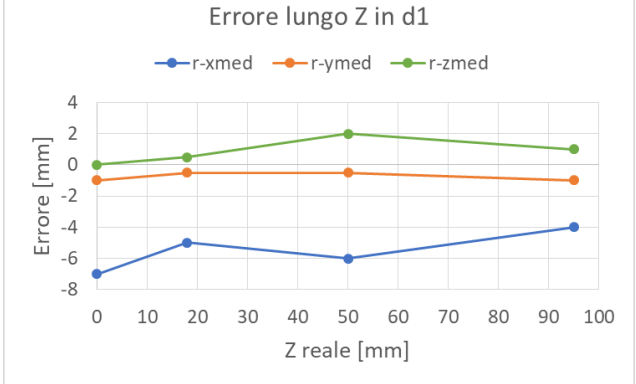
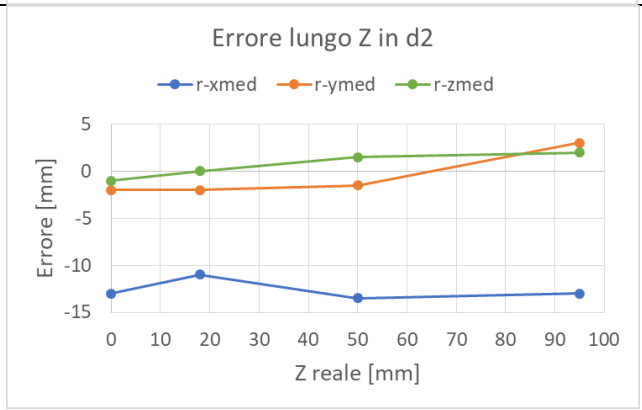
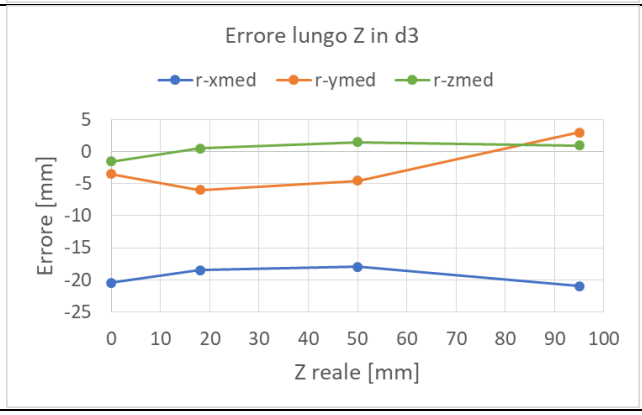


Tabella 6.6: Andamento dell'errore lungo Z







L'errore lungo la diagonale ha un comportamento molto simile a quello che si presentava nelle prove precedenti. Andando poi a vedere come varia l'errore in funzione della coordinata Z si nota che esso varia, ma

di molto poco, infatti il risultato è una traccia orizzontale, il che indica un errore praticamente costante. Un altro dettaglio positivo che si può notare è che l'errore in Z delle coordinate è sempre molto basso ed inferiore a 2mm.

Il fatto che l'errore rilevato sulle coordinate X ed Y fosse molto più grande rispetto a quello previsto dal costruttore e la proporzione lineare con la distanza così evidente sono state le ragioni principali che hanno fatto pensare che ci fosse un problema all'interno del software. Infatti, per quanto è possibile che le telecamere contengano un difetto, è molto improbabile che siano così tanto al di fuori dalla tolleranza, in quanto non sarebbero utilizzabili in alcuna applicazione. Inoltre, la linearità dell'errore, come poi si è scoperto essere effettivamente, indica che l'errore nel calcolo delle coordinate è legato ad una variabile precisa, e non è dovuto a fattori esterni casuali.

Successivamente si è scoperto che i dati del FOV della telecamera utilizzati nel calcolo delle coordinate assolute erano sbagliati, in particolare l'horizontal FOV era quello che presentava un errore relativo maggiore. Dopo aver corretto il codice, le prove sono state eseguite di nuovo, e per qualsiasi punto si è rilevato un errore massimo di 1mm. La prova si poteva considerare superata con successo, e si è potuto proseguire con lo sviluppo del programma.

6.2 Analisi della disposizione delle telecamere

Prima di costruire il sistema completo, ovvero quello dotato di nastro trasportatore e fotocellule, è stato necessario scegliere in che modo posizionare le telecamere. A prima vista questo sembra essere un problema banale, ma in realtà non lo è in quanto variando l'inclinazione della telecamera rispetto all'orizzontale si è verificato che:

- varia la distanza minima al quale si può posizionare la telecamera;
- varia la porzione di mela visibile dalla telecamera.

Andando a vedere il grafico riportato all'inizio del paragrafo precedente, si nota che la telecamera inizierebbe ad avere un errore eccessivo, ovvero superiore a 2mm solo se le telecamere superassero la distanza di 500mm.

Invece, per ciò che riguarda la visibilità della mela, sarebbe necessario che almeno il 70% fosse visibile, per far sì che il picciolo o il calice della mela siano visibili anche nel caso in cui essa sia posta orizzontalmente.

6.2.1 Studio analitico della visibilità della mela

Per calcolare matematicamente la visibilità della mela è stato necessario formulare delle ipotesi semplificative:

- si è ipotizzato che la mela abbia una forma sferica. Ciò non è realistico, ma se è visibile la sfera che comprende la mela al suo interno, allora è visibile anche la mela stessa;
- si è corretto l'angolo visivo verticale della telecamera di un angolo $\Delta\alpha$=10 deg. (presente agli estremi del campo visivo) in modo da tener conto di:
    - imprecisioni intrinseche nella costruzione della telecamera (+/- 3 deg.);
    - imprecisione di lettura delle coordinate assolute ai margini del campo visivo dovuta all'assenza dell'illuminazione del laser;

Indicando con:

- γ metà dell'angolo che sottende l'area non visibile
- r il raggio della circonferenza che circoscrive la mela
- x l'altezza della mela non visibile

Si può trovare la visibilità della mela indicata con:

- $\varphi_{\%}$ calcolata come l'arco visibile in percentuale
- $H_{\%}$ calcolata come l'altezza visibile in percentuale

Nel proseguimento della discussione la visibilità della mela sarà sempre identificata come $\varphi_{\%}$.

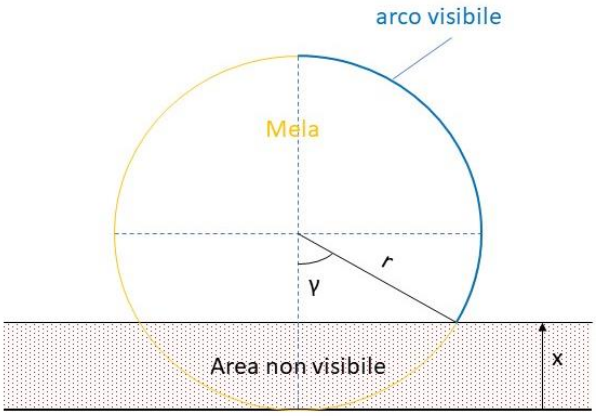


Figura 6.8: Geometria della visibilità di una sfera

Conoscendo l'altezza non visibile x si possono calcolare:

$$\gamma = \cos^{-1}\left(\frac{r-x}{r}\right) \qquad \textbf{Equazione 6.1}$$

$$H_{\%} = \frac{2r-x}{2r} * 100 \qquad \textbf{Equazione 6.2}$$

$$\varphi_{\%} = \frac{180°-\gamma}{180°} * 100 \text{ (visibilità)} \qquad \textbf{Equazione 6.3}$$

Invece, partendo dalla visibilità ϕ si possono calcolare:

$$\gamma = 180° - \frac{\varphi * 180}{100} \qquad \textbf{Equazione 6.4}$$

$$x = r * (1 - cos\gamma)$$ **Equazione 6.5**

La visibilità della mela dipende dal campo visivo della telecamera.

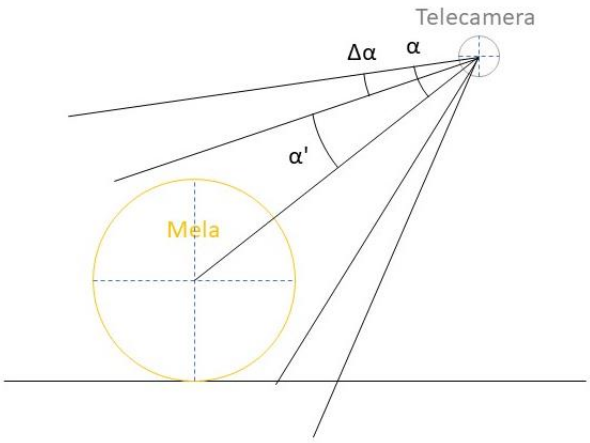


Figura 6.9: Geometria del FOV di una telecamera

Per eliminare la parte non utilizzabile basta applicare la formula seguente:

$$\alpha = \frac{V\ FOV}{2}$$ **Equazione 6.6**

$$\alpha' = \alpha - \Delta\alpha$$ **Equazione 6.7**

Da cui risulta:

Tabella 6.7: Calcolo del FOV utile

TELECAMERA			
VFOV	alfa*2	gradi	65,5
VFOV/2	alfa	gradi	32,8
margine inutilizzabile	delta	gradi	10
VFOV/2 utile	alfa'	gradi	22,8

A questo punto, dalla figura sottostante si possono ricavare le formule per trovare la distanza minima al quale si può piazzare la telecamera garantendo una determinata visibilità della parte inferiore:

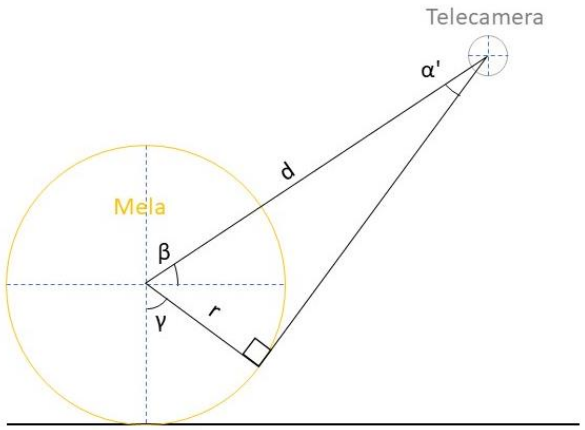


Figura 6.10: Geometria del limite di visibilità inferiore

$$d_{min_1} \geq \frac{r}{\sin \alpha'}$$ **Equazione 6.8**

$$d_{min_2} \geq \frac{r}{\cos(\beta+90-\gamma)}$$ **Equazione 6.9**

Si possono applicare gli stessi principi per trovare le formule da cui si può trarre la distanza minima al quale si può piazzare la telecamera consentendo una determinata visibilità della parte superiore:

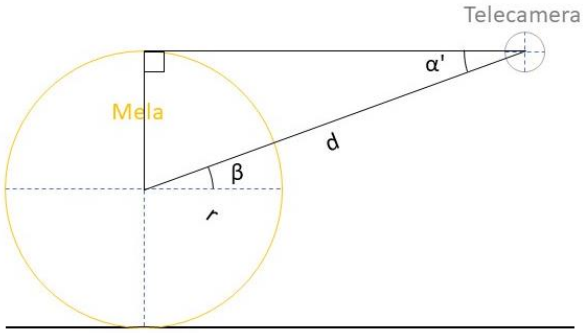


Figura 6.11: Geometria del limite di visibilità superiore

$$d_{min_1} \geq \frac{r}{\sin \alpha'}$$ **Equazione 6.10**

$$d_{min_3} \geq \frac{r}{\cos(90-\beta)}$$ **Equazione 6.11**

Le due parti sono diverse a causa del modo in cui è stato definito l'arco visibile. Se l'area non visibile si fosse trovata superiormente la situazione sarebbe capovolta. Il motivo per cui si è deciso di porre la parte non visibile nella parte inferiore, è che essa è la zona che comunque risulterebbe coperta dal facchino del nastro trasportatore, quindi risulterebbe comunque non osservabile.

Dopo aver trovato le relazioni che legano la visibilità della mela all'inclinazione delle telecamere e alla grandezza della mela, non resta che diagrammarle. Prima però è necessario conoscere il range di dati al quale si deve applicare questa analisi matematica.

Mentre il FOV verticale della telecamera è dato con certezza: 65,5° +/- 3°, la grandezza della mela può variare all'interno di un range abbastanza ampio.

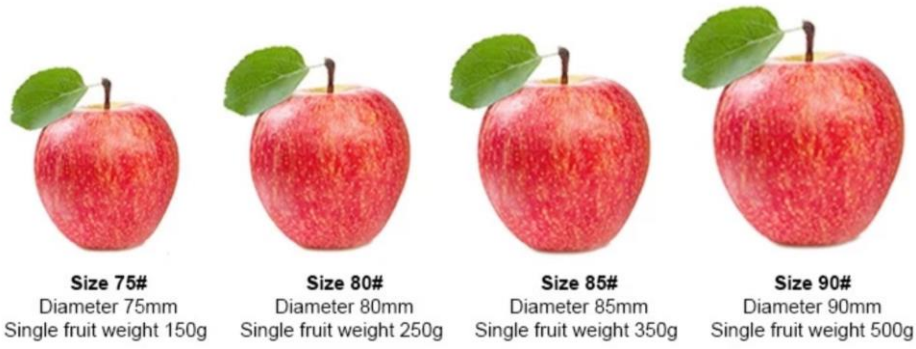


Figura 6.12: Esempio di taglie di un tipo di mela presenti in un catalogo [49]

All'interno della normativa europea [50], il limite minimo del diametro delle mele è definito come 60 mm, ma non viene specificato alcun limite massimo. Ciò nonostante, da alcuni cataloghi commerciali si deduce che, normalmente, le mele possono raggiungere un diametro massimo di 90mm. Quindi, per sicurezza, si è considerato un raggio massimo di 50mm.

Ora che è stato definito anche il range di grandezza della mela, è possibile impostare dei diagrammi con cui sia possibile trovare la configurazione ottimale delle telecamere.

Per prima cosa si è calcolato un grafico che mostra l'andamento della distanza minima in funzione dell'inclinazione, garantendo diversi valori di visibilità, per una mela di raggio pari a 50mm.

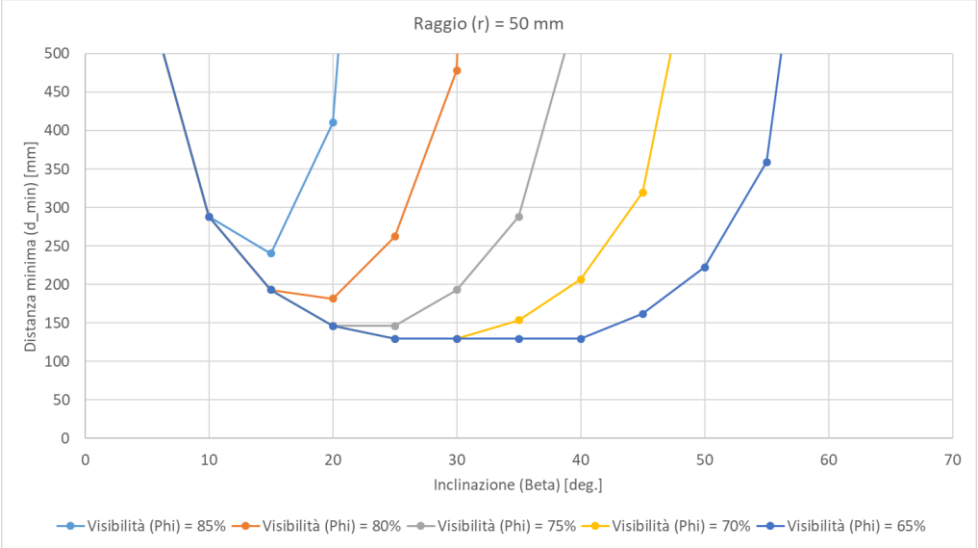


Figura 6.13: Grafico di posizionamento per raggio della mela costante

Osservando questo grafico si può notare che:

- All'aumentare della visibilità richiesta si riduce il range di inclinazione utilizzabile
- Ponendo la telecamera molto vicino (150mm), si otterrebbe una visibilità minima del 65%
- Imponendo una distanza minima di 200mm, si potrebbe ottenere una visibilità massima dell'80%

Successivamente si è costruito un altro grafico, per capire cosa sarebbe accaduto se si volesse garantire una visibilità almeno del 70%, andando a variare la grandezza delle mele.

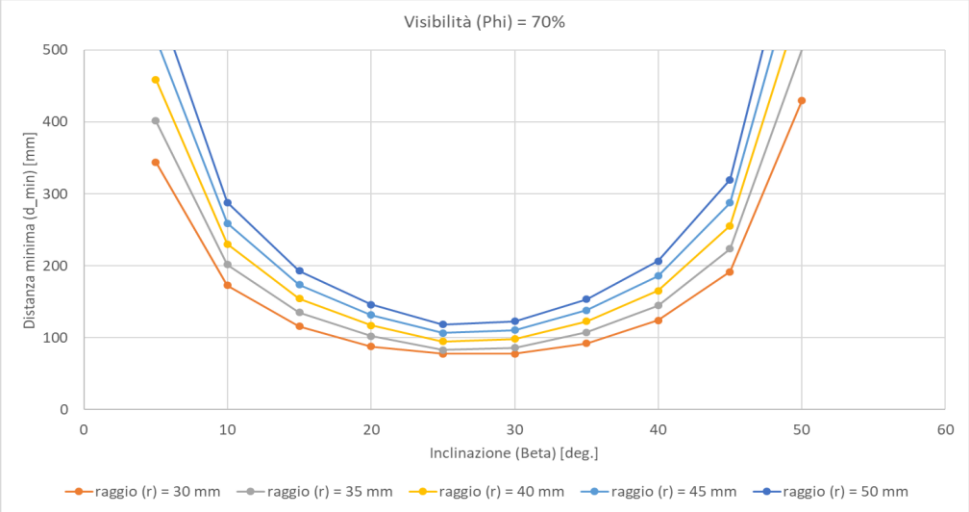


Figura 6.14: Grafico di posizionamento per visibilità costante (intermedia)

Si può notare che:

- Indipendentemente dal raggio si ottiene un'inclinazione ottimale tra i 25° ed i 30°
- Le curve sono molto vicine tra loro, soprattutto nella zona vicina all'inclinazione ottimale.

A questo punto rimane da capire come varierebbe questo grafico, se si richiedessero visibilità diverse.

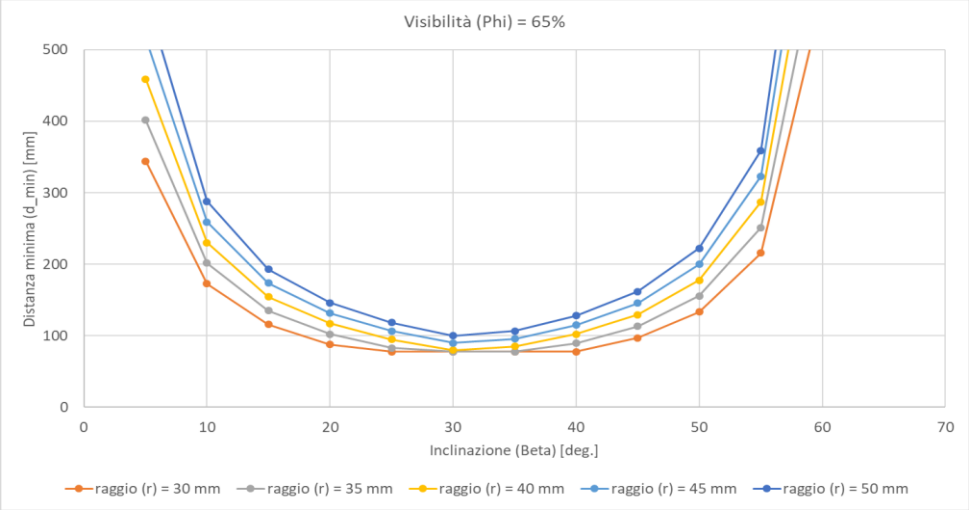


Figura 6.15: Grafico di posizionamento per visibilità costante (minima)

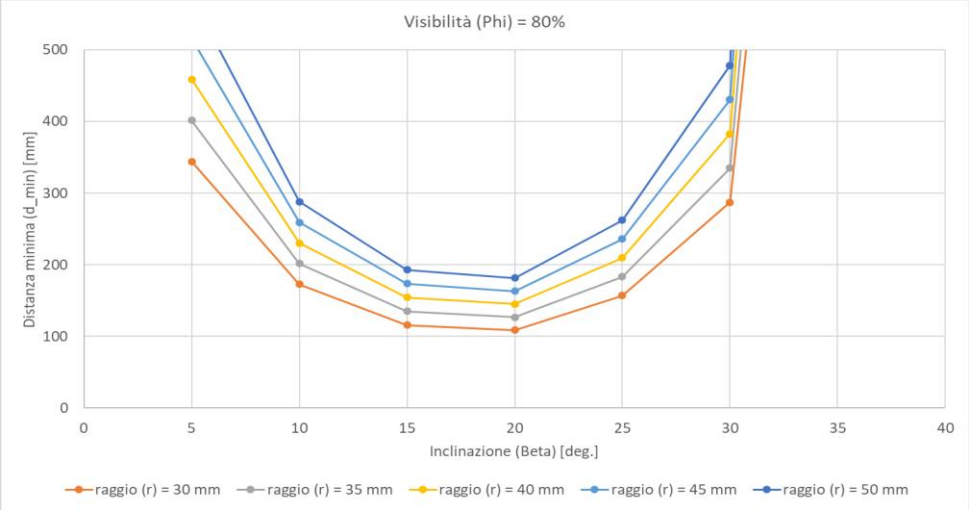


Figura 6.16: Grafico di posizionamento per visibilità costante (massima)

Da questi 3 grafici si desume che, richiedendo maggior visibilità:

- diminuisce l'inclinazione ottimale;
- si riduce notevolmente il range di inclinazione utilizzabile;

- aumenta la distanza minima all'interno di tutto il range di inclinazione utilizzabile;
- cresce la sensibilità alla variazione del raggio della mela.

I diagrammi trovati in questo paragrafo non servono solo a capire quale può essere la configurazione ottimale data una certa inclinazione. Infatti, è altamente improbabile che effettivamente si riesca as imporre questa condizione. In realtà servono soprattutto a dimostrare che il sistema è abbastanza flessibile, e che si può garantire almeno il 65% della visibilità della mela per un ampio range di inclinazioni ed anche mantenendo una distanza minima abbastanza bassa, ovvero di 150mm.

6.2.2 Verifica sperimentale della visibilità della mela

Prima di proseguire allo sviluppo del programma, si è deciso di verificare la veridicità di questi diagrammi allestendo una struttura che fosse in grado di porre le telecamere in diverse configurazioni. L'architettura di questo banco di prova è modulabile, in modo tale da porre le telecamere alle angolazioni e distanze volute abbastanza velocemente.

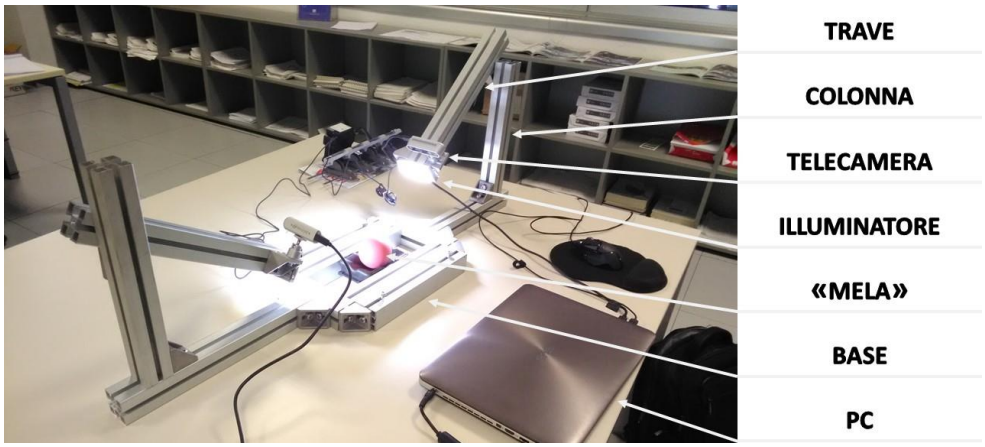


Figura 6.17: Foto del banco di prova delle telecamere

A parte la componente centrale dov'è posta la pallina di gomma (gialla), e le basi orizzontali che poggiano sul tavolo (bianche), tutte le altre parti sono mobili, sia le colonne verticali (arancione), che le travi oblique (blu). Nello schema sotto, le parti rosse sono i vincoli da svitare per permettere il movimento, mentre il rettangolino grigio rappresenta la telecamera.

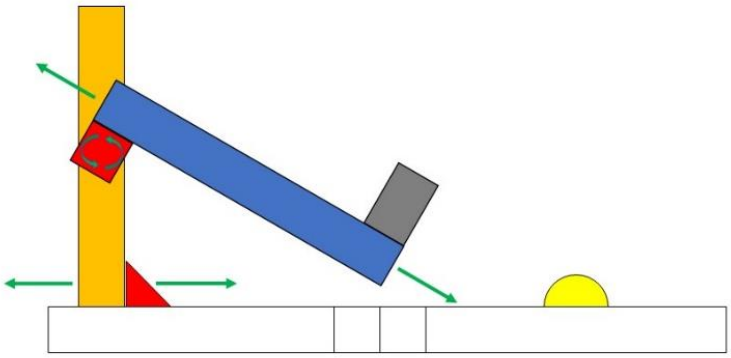


Figura 6.18: Schema del banco di prova modulare

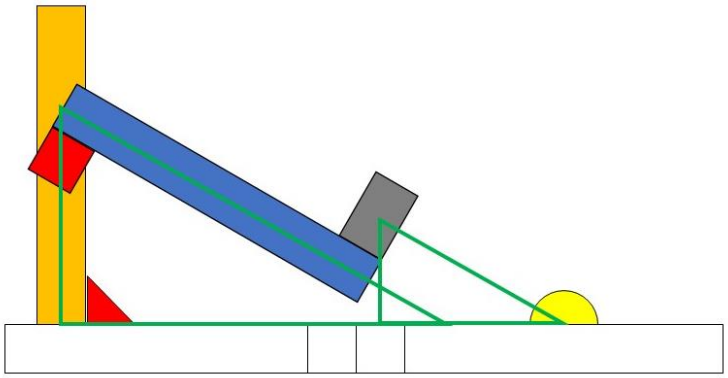


Figura 6.19: Geometria del banco prova

Come si può notare, la linea visiva della telecamera, la linea della base orizzontale, e la linea verticale che congiunge la telecamera alla base, formano un triangolo rettangolo. Quindi, data un'inclinazione ed una distanza è possibile calcolare i lati di questo rettangolo, in modo tale da sapere dove porre la telecamera.

Se la telecamera è posta nella giusta posizione e con la giusta orientazione, allora la linea della trave sarà parallela a quella della telecamera, e le linea della trave, quella della colonna, e quella della base formeranno un altro triangolo rettangolo, simile a quello precedente.

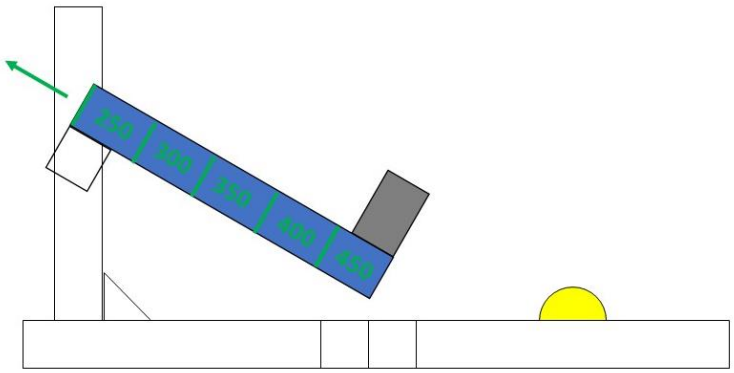


Figura 6.20: Variazione della distanza ad inclinazione costante

Se si parte dalla posizione più vicina, per ottenere quelle più lontane, basta far scorrere la trave verso l'esterno, mantenendo la stessa inclinazione, fino a raggiungere la distanza obbiettivo.

La struttura costruita ci ha consentito di provare l'inclinazione delle telecamere a 20, 30, 40 e 50 gradi, e per ogni inclinazione ci ha permesso di porle a distanze di 250, 300, 350, 400, 450 mm. Se si vanno a confrontare questi valori con quelli presenti nei diagrammi precedenti, si nota che sarebbe stato interessante fare delle prove con distanze inferiori a 250mm. A causa di motivi costruttivi ciò non è stato

possibile, ciò nonostante le prove sostenute sono risultate più che sufficienti per dimostrare che i calcoli eseguiti per la costruzione dei diagrammi fossero giuste.

In particolare, è stato interessante analizzare ciò che sarebbe accaduto alla distanza di 250mm, in quanto l'applicazione reale avrebbe richiesto alle telecamere di trovarsi ad una distanza molto simile.

Come si può notare dal grafico seguente, a questa distanza si ottiene la visibilità massima andando a ridurre al minimo l'inclinazione, mentre l'inclinazione massima di 50° consentirebbe comunque una visibilità minima del 68%.

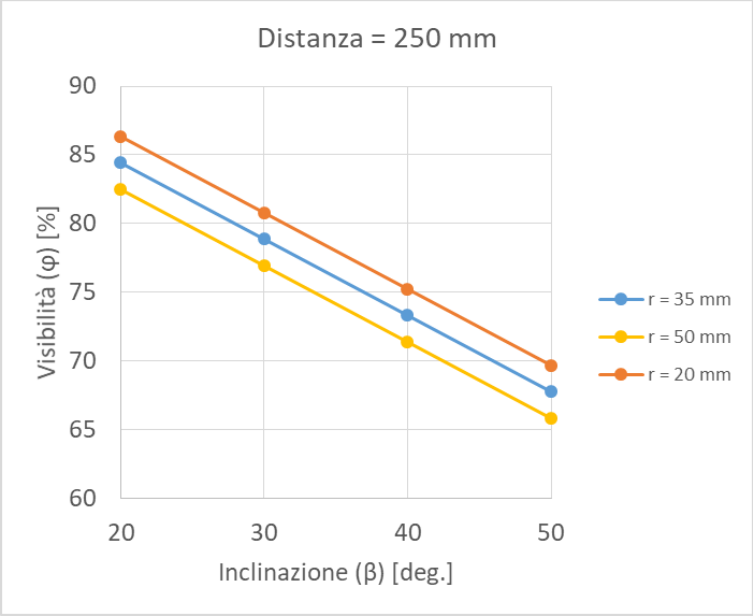


Figura 6.21: Grafico della visibilità alla distanza di 250mm

Nelle pagine seguenti mostreremo le foto fatte durante la prova, che dimostrano che i risultati teorici ottenuti con excel si sono verificati anche nella realtà.

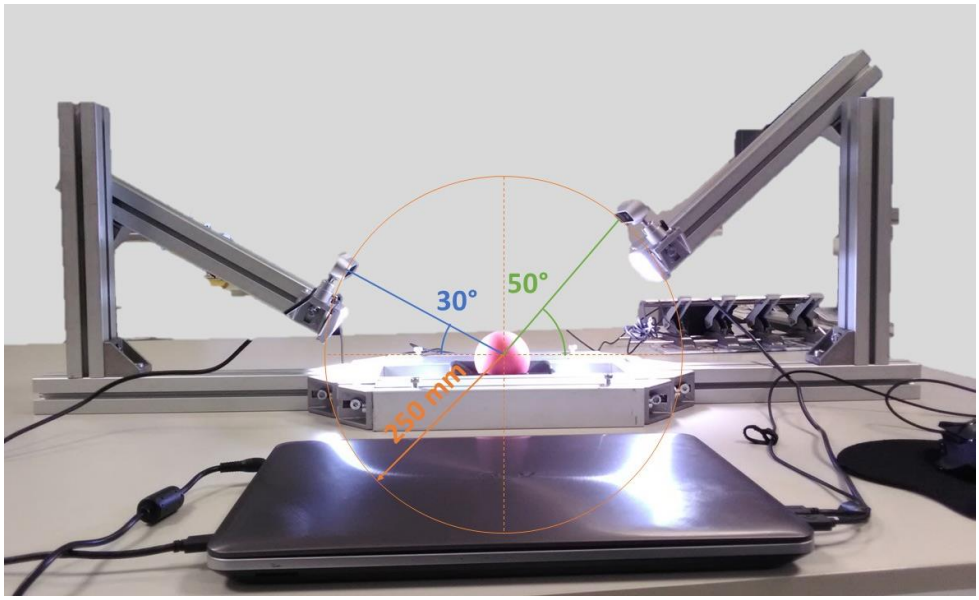


Figura 6.22: Impostazione del banco prova per: β1 = 30°, β2 = 50°

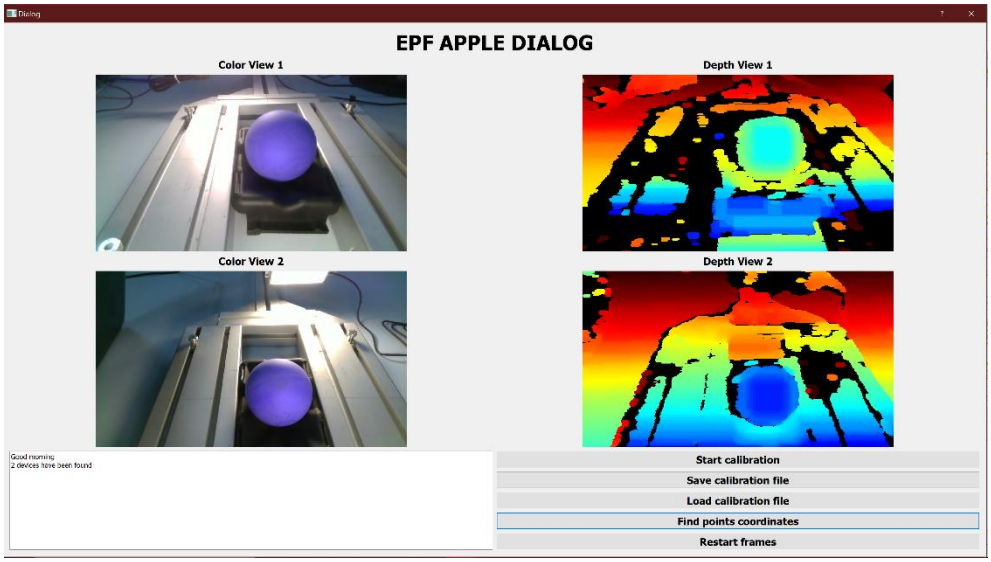


Figura 6.23: Visualizzazione delle depthmap per: β1 = 30°, β2 = 50°

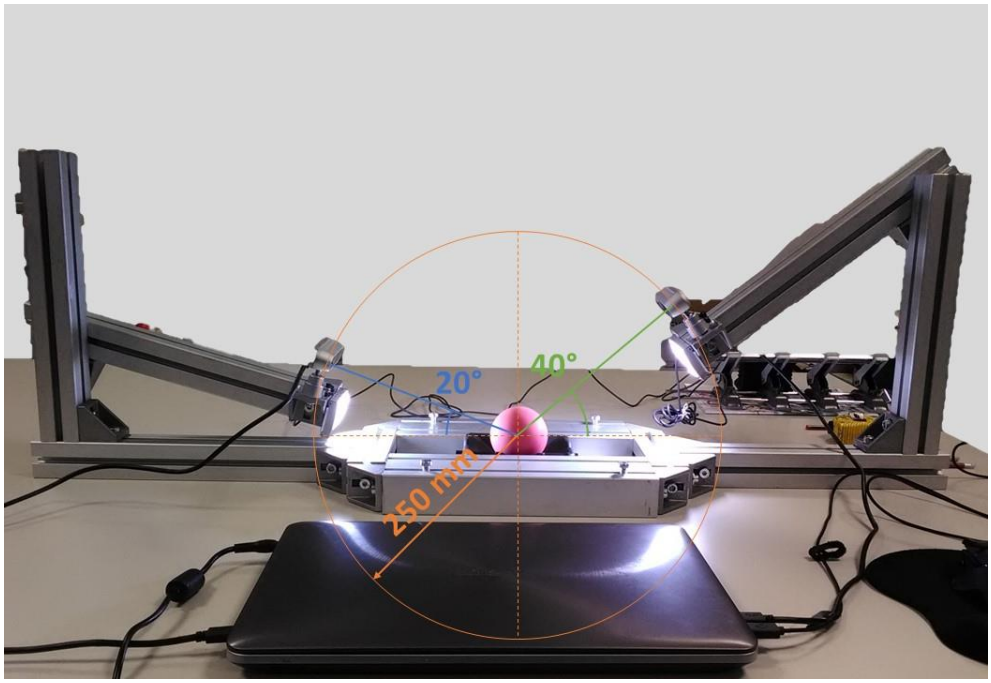


Figura 6.24: Impostazione del banco prova per: β1 = 20°, β2 = 40°

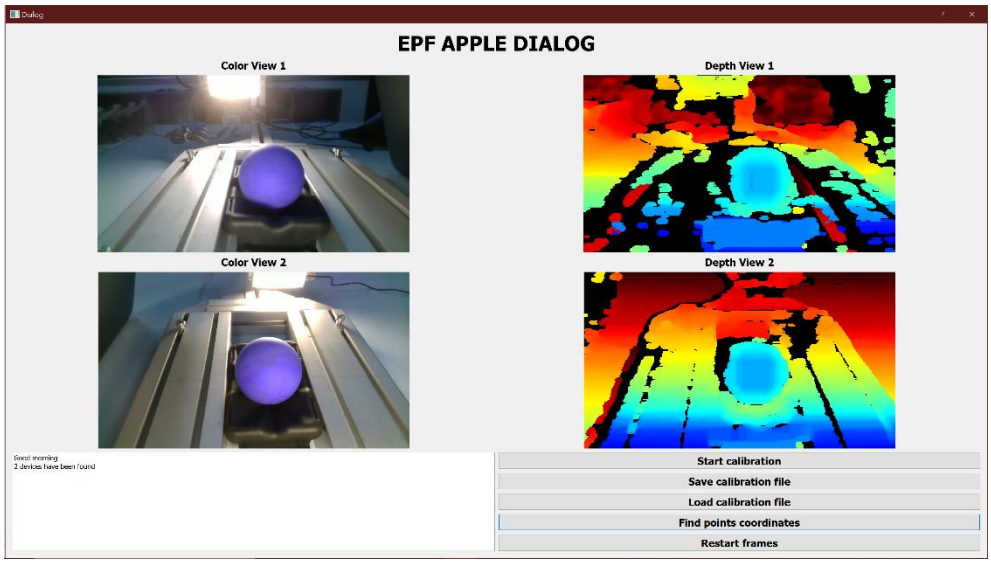


Figura 6.25: Visualizzazione delle depthmap per: β1 = 20°, β2 = 40°

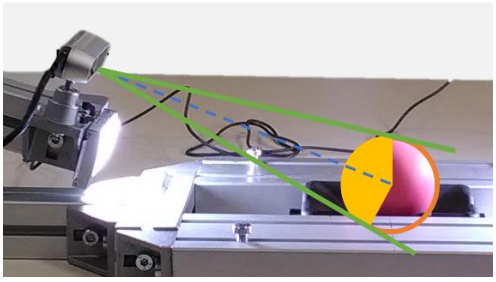


Figura 6.26: β = 20°, φ = 84%

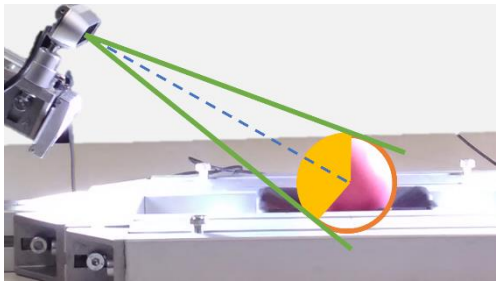


Figura 6.27: β = 30°, φ = 79%

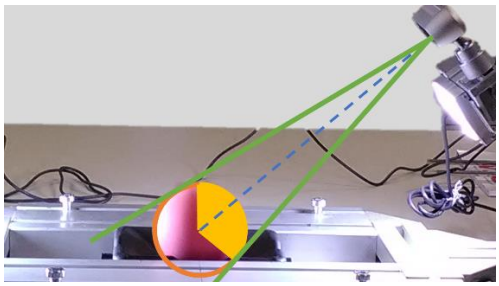


Figura 6.28: β = 40°, φ = 73%

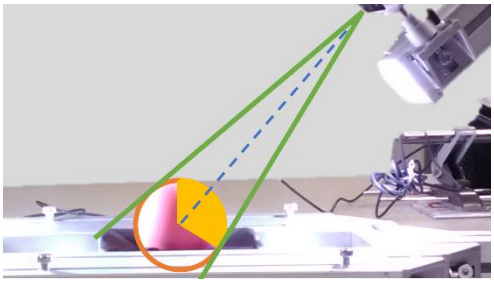


Figura 6.29: β = 50°, φ = 68%

Come si può notare dalle foto i risultati ottenuti sono molto positivi in quanto anche nella condizione peggiore si è molto vicini a garantire una visibilità del 70% della mela.

Però si può anche notare che la visualizzazione delle depthmap è di scarsa qualità quando l'inclinazione è molto bassa, ovvero 20 o 30 gradi. Probabilmente ciò è dovuto al fatto che il laser della telecamera non viene riflesso quando l'inclinazione è troppo bassa. Infatti, mentre la superficie della mela risulta sempre priva di "buchi", è il piano circostante che presenta più vuoti.

Per questo motivo nell'applicazione reale non si utilizzeranno inclinazioni troppo basse, nonostante ciò non si supereranno i 50°, quindi si riuscirà sempre a garantire una buona visibilità, in modo tale da consentire l'individuazione dello stelo o del calice della mela.

CAPITOLO 7

7. Banco prova

Dopo aver eliminato i bug del programma per la gestione delle telecamere, si è deciso di costruire un prototipo su cui fosse possibile sperimentare il funzionamento delle telecamere in condizioni il più possibile simili a quelle presenti nell'impianto reale.

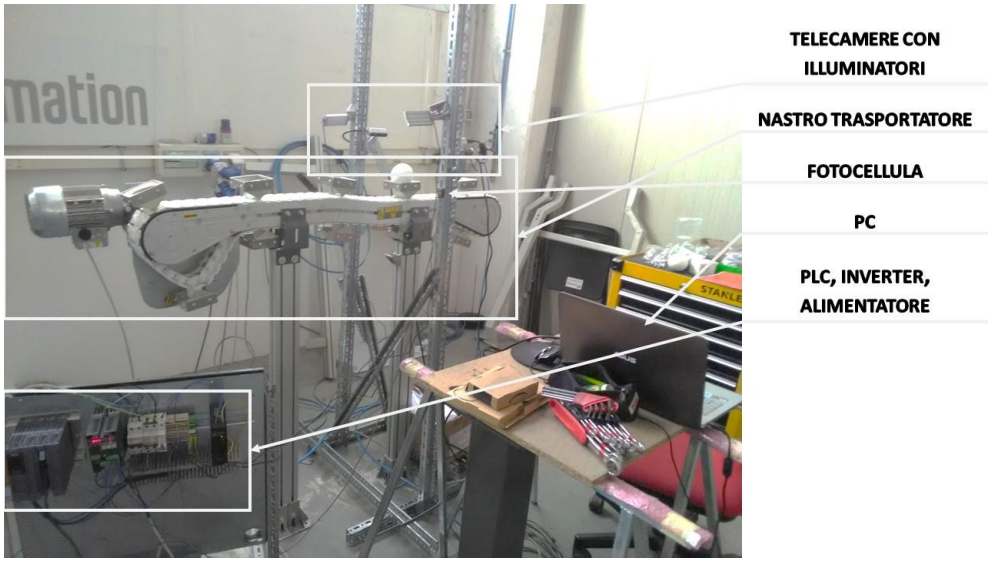


Figura 7.1: Foto del sistema completo

A parte il robot, in questo banco prova sono presenti tutti i componenti che faranno parte del sistema completo:

- un nastro trasportatore di piccole dimensioni: per la movimentazione delle mele;
- una fotocellula: per rilevare il passaggio dei facchini;
- un PLC: per la gestione della fotocellula;

- due telecamere stereoscopiche: per catturare le immagini 3D delle mele;
- due illuminatori: per mantenere costante il livello di luminosità
- un alimentatore: per fornire la corrente elettrica ai componenti che lavorano a bassa tensione (fotocellula, PLC ed illuminatori);
- un PC: per la gestione delle operazioni.

Successivamente, si è applicato uno scatolone per cercare di isolare l'area in cui avviene la cattura delle immagini dalla luce ambientale. In questo modo è stato possibile regolare il livello di illuminazione all'interno del carter ponendo gli illuminatori in posizioni diverse, permettendo di capire meglio l'effetto della luce sulla qualità della depth map generata dalle telecamere.

Figura 7.2: Foto della versione finale del prototipo

Nei paragrafi seguenti si vedranno le caratteristiche ed il funzionamento di ogni componente, per poi andare a verificare l'effetto del movimento delle mele sul programma di rilevazione delle coordinate.

7.1 Fotocellula

Un sensore fotoelettrico è un apparecchio impiegato per valutare la distanza, oppure individuare la mancanza o la presenza di un oggetto servendosi di una sorgente ottica, e un ricevitore fotoelettrico.

In questo caso, la fotocellula è fondamentale in quanto devono rilevare il passaggio delle mele davanti alle telecamere.

Per la nostra applicazione è bastato applicare una singola fotocellula esattamente sulla colonnina che regge una delle telecamere Realsense. In questo modo non c'è stato bisogno di aggiungere strutture di metallo supplementari e allo stesso tempo, questa posizione permette di fotografare i facchini nel momento in cui sono più vicini alle telecamere, garantendo la precisione massima.

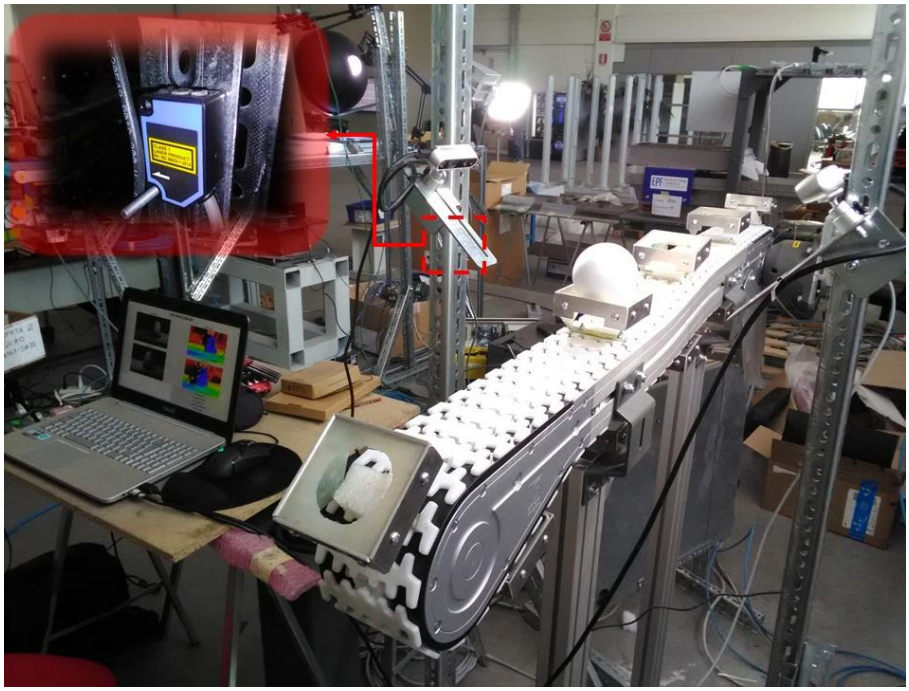


Figura 7.3: fotocellula nel sistema

La fotocellula è stata collegata ad una delle porte di input del PLC, e per portare il segnale al computer, ovviamente, si è collegato il PC ad una delle porte di output, tramite un cavo di tipo ethernet. A differenza delle telecamere stereoscopiche, la lettura dei dati della fotocellula tramite PLC non è stata altrettanto "automatica". Per configurare il collegamento ethernet con il PLC è stato necessario installare un wrapper di Python per snap7. Nel paragrafo "Configurazione" verrà spiegato nel dettaglio in come si è impostata la connessione.

Per garantire nel tempo il funzionamento delle fotocellule, basterà assicurarsi che la polvere non blocchi il laser e che i cavi che le collegano con il PLC siano in buono stato.

7.1.1 Funzionamento

Le fotocellule si possono distinguere in 3 categorie principali che differiscono per modalità di funzionamento: a barriera, a catarifrangente, e a riflessione diretta (tasteggio diretto).

In questo caso, si è scelto di impiegare delle fotocellule a riflessione diretta. Questo tipo di sensori sono racchiusi in un'unica scatola che comprende le ottiche e l'elettronica di controllo.

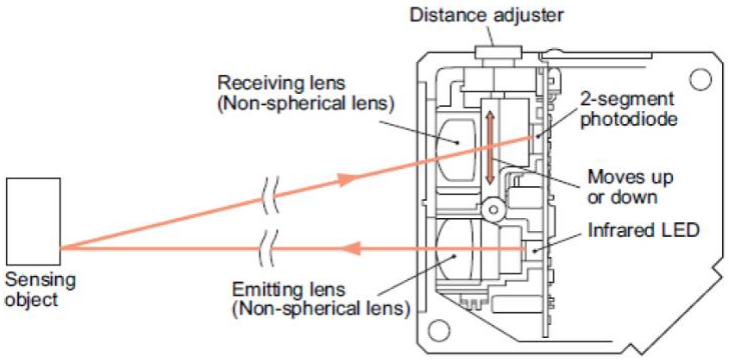


Figura 7.4: Schema di una fotocellula a riflessione diretta [51]

Nei sistemi a riflessione diretta l'emettitore emette una luce che poi può essere riflessa verso ricevitore dall'oggetto che deve essere rilevato. A differenza degli altri tipi di sensore, l'oggetto viene rilevato quando la luce arriva al ricevitore.

Gli emettitori sono quasi sempre dei LED a causa della loro economicità, affidabilità e durata. Nella maggioranza dei casi viene utilizzata la luce infrarossa, ma è in aumento l'utilizzo del LED Laser.

In questo caso, la fotocellula utilizza un laser di classe 1, il che significa che è sempre sicura in quanto le radiazioni emesse sono al disotto degli standard massimi consentiti.

7.1.2 SNAP7

Prima di procedere all'impostazione del collegamento conviene presentare la struttura complessiva di Snap 7 in modo da capirne meglio il funzionamento generale.

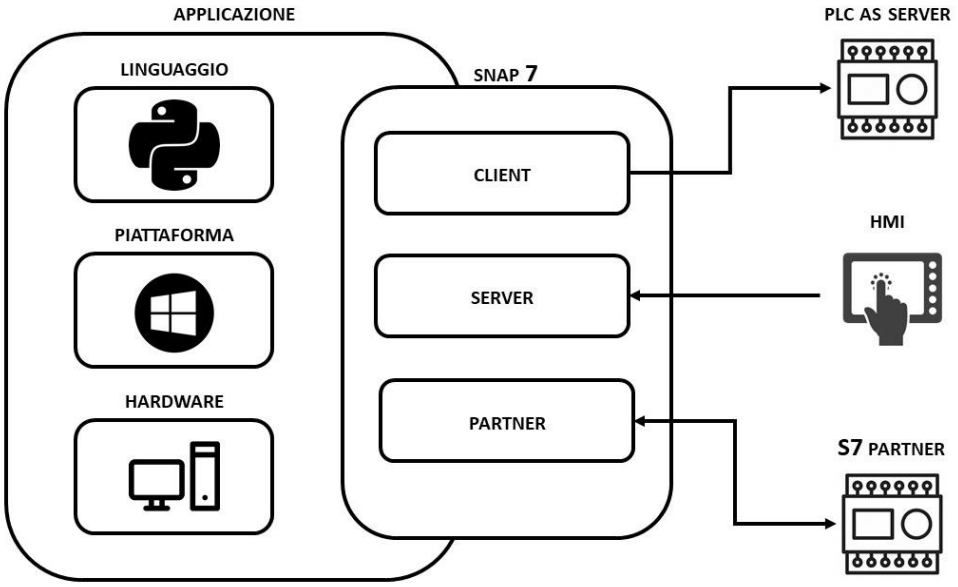


Figura 7.5: Schema di base di snap7

Questo programma open source è una libreria che serve ad interfacciarsi con i PLC Siemens S7. Risulta essere più di un semplice driver per leggere e scrivere dati su un PLC, lo sviluppatore l'ha definita "suite" perché contiene tre componenti indipendenti:

- Client
- Server
- Partner

Il Client, anche se è il componente più "ovvio", presenta alcuni elementi innovativi come lo smart-connect e funzioni asincrone che possono compiere grossi trasferimenti di dati in un thread separato. Inoltre, implementa alcuni nuovi elementi come l'impostazione/eliminazione

della password, ed il caricamento di un database senza doverne conoscere la dimensione in anticipo.

Il Server permette il collegamento di un pannello HMI, Scada o Server OPC all'applicazione che viene vista come se fosse un Siemens CPU (S7315-2PN / DP) e si può andare on-line con Simatic Manager/Tia Portal per vedere le cartelle e le informazioni del sistema. Il Server gestisce fino a 1024 clienti contemporaneamente con accesso costante ai dati condivisi. Il Log in implementa una coda circolare e un callback asincrono per sincronizzarsi con l'attività dei clienti.

Il Partner implementa l'architettura peer-to-peer (oppure client-client in accordo con Siemens). Questo componente consente di fare lo store.on-demand: non è il PC ad interrogare in polling il PLC per controllare se ha già i dati, ma è il PLC che manda i dati quando decide di farlo. Il PC riceve i dati in callback completamente asincrono. Questo è possibile attraverso una connessione TCP/IP, utilizzando la robustezza del protocollo S7. Le funzioni utilizzate sono BSend / BRECV e permettono di trasferire fino a 64k nello stesso processo. L'impiego di questo componente non è facile (perché non è facile utilizzare i Partners, in accordo con Siemens), comunque esiste una guida molto dettagliata sul come farlo e ci sono esempi pronti all'uso.

Configurazione

Per prima cosa bisogna tener conto del fatto che il collegamento S7 con le fotocellule deve essere configurato in STEP 7 (TIA Portal) nella vista di rete. Chiarito questo, si possono elencare i passi che portano alla formazione di questa connessione.

1. Nell'editor Dispositivi & Reti" aprire la vista di rete.
2. Fare clic su "Collegamenti". Selezionare un collegamento, ad es. collegamento TCP.

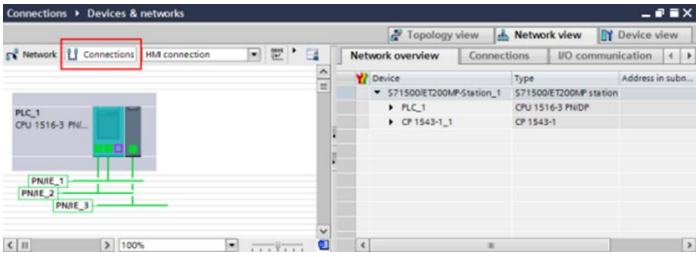


Figura 7.6: 2° passo della configurazione [52]

3. Con il tasto destro del mouse fare clic sulla CPU e nel menu contestuale della CPU selezionare il comando "Aggiungi nuovo collegamento".

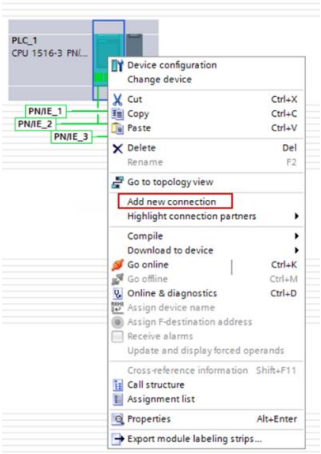


Figura 7.7: 3° passo della configurazione [52]

4. Nella finestra di dialogo "Crea nuovo collegamento" vengono configurati nuovi collegamenti. Selezionare il tipo di collegamento, ad es. collegamento TCP. Nella parte sinistra della finestra di dialogo selezionare il partner di collegamento, ad es. "Non specificato". Nella parte destra della finestra di dialogo selezionare l'interfaccia locale, tramite la quale si deve comunicare. Fare clic sul pulsante "Applica" e chiudere la finestra di dialogo.

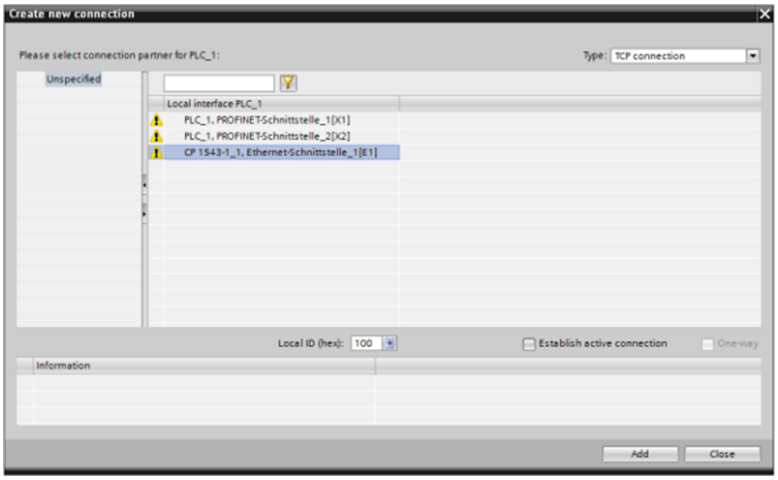


Figura 7.8: 4° passo della configurazione [52]

5. Nell'area a tabella della vista di rete passare nella scheda "Collegamenti", dove vengono visualizzati tutti i collegamenti configurati. Un collegamento viene visualizzato su sfondo rosso, ad es. se l'indirizzo IP del partner non è configurato.

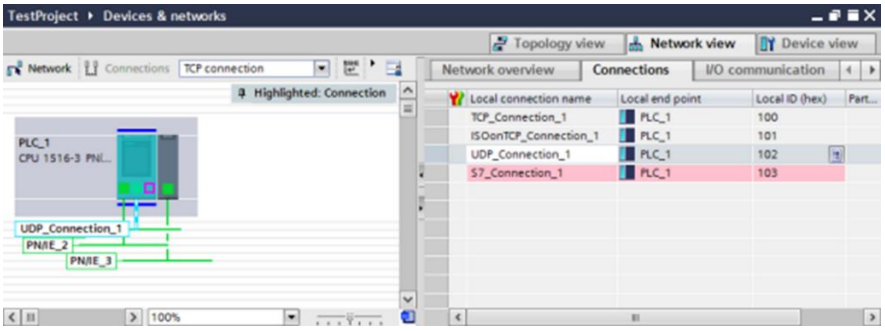


Figura 7.9: 5° passo della configurazione [52]

6. Completare la configurazione dei collegamenti. Marcare il collegamento. Nella finestra d'ispezione vengono visualizzate le proprietà del collegamento. Registrare qui l'indirizzo IP del partner e della porta o di TSAP.

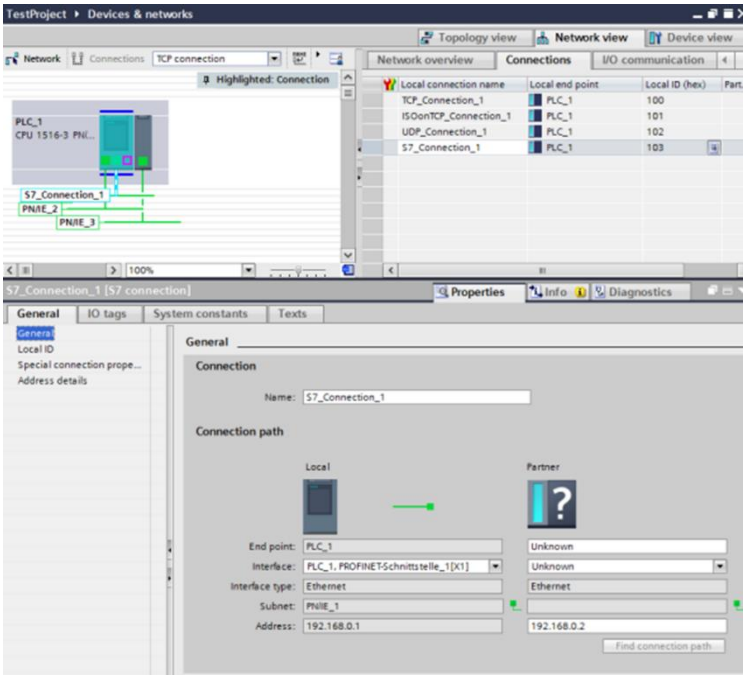


Figura 7.10: 6° passo della configurazione [52]

7. Con un collegamento S7 il TSAP viene configurato in Address details. Questo è richiesto per un collegamento S7 unilaterale con PUT/ GET. Impostare la risorsa di collegamento del partner su 03 e il rack/slot sulla CPU del partner. Anche se un CP è inserito nel partner, la CPU è sempre indirizzata con il rack/slot.

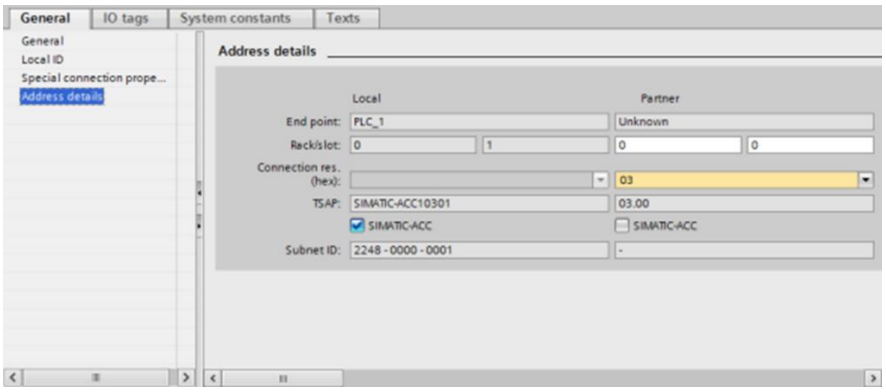


Figura 7.11: 7° passo della configurazione [52]

8. Nella CPU S7-1500 e CPU S7-1200 attivare con un collegamento S7 la funzione "Permit access with PUT/GET communication from remote partner (PLC, HMI, OPC, ...)". Attivare la funzione nelle proprietà della CPU in "Protection > Connection mechanisms".

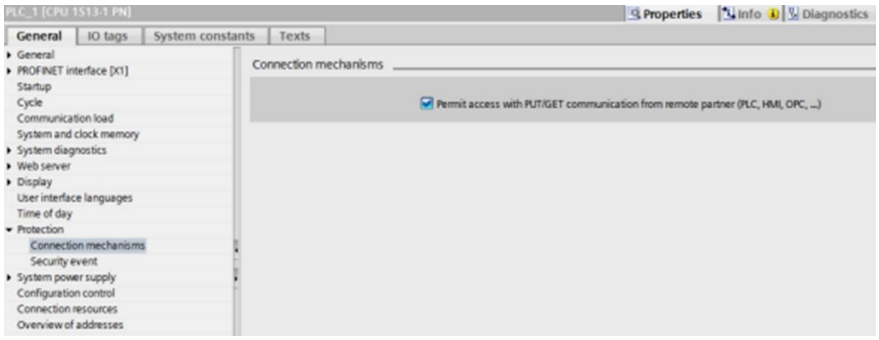


Figura 7.12: 8° passo della configurazione [52]

9. Caricare la configurazione nel controllore.

Eseguendo correttamente queste istruzioni, è stato possibile leggere i segnali inviati dalle fotocellule. Inoltre, mediante Python è stato possibile elaborarli in modo tale da catturare un'immagine 3D dalle telecamere stereoscopiche ogni volta che i facchini del nastro trasportatore superano il laser delle fotocellule. Questa parte verrà trattata in futuro nel capitolo sulla "Descrizione del programma".

7.2 Nastro trasportatore

In generale, il nastro trasportatore è un dispositivo impiegato per trasportare oggetti o materiali presenti in grandi numeri e deputati all'elaborazione industriale.

Nell prototipo in questione il nastro trasportatore è molto piccolo e semplice, rispetto a quello presente nell'impianto reale, ma esegue lo stesso lavoro: movimentare i facchini su cui sono poste le mele che devono essere prelevate dal robot.

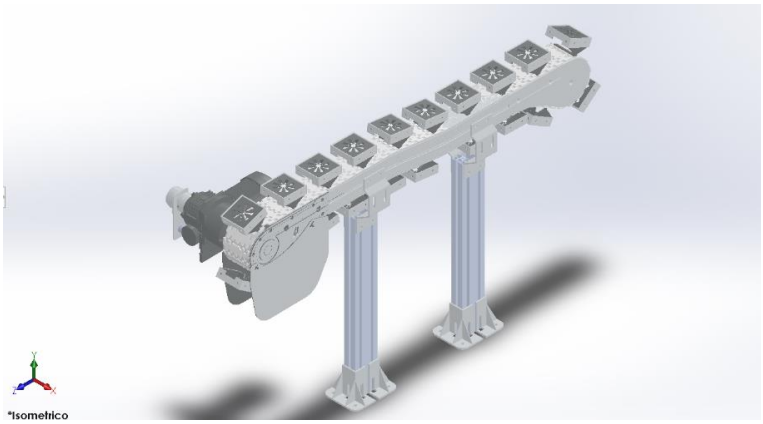


Figura 7.13: immagine 3D del nastro trasportatore

Questo nastro è mosso da un motore elettrico asincrono trifase di 0,37 kW di potenza, il quale è comandato da un inverter che converte la tensione e la frequenza fisse dell'alimentazione di rete in una tensione variabile con frequenza variabile. Ciò consente la regolazione della coppia e della velocità del motore elettrico.

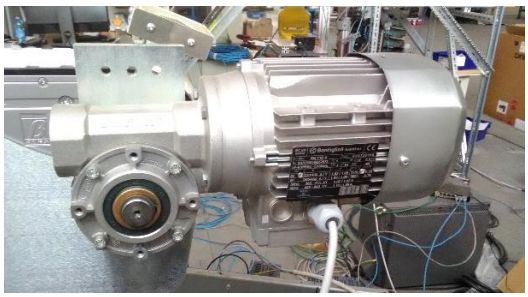


Figura 7.14: Motore elettrico

Figura 7.15: Inverter

7.2.1 Caratteristiche tecniche

In questo paragrafo andremo ad analizzare nel dettaglio i singoli componenti del nastro trasportatore:

- catene;

- telaio in profilati di alluminio
- gruppo di traino
- gruppo di rinvio
- motore elettrico e inverter

Catene

Figura 7.16: Catene in resina acetalica [53]

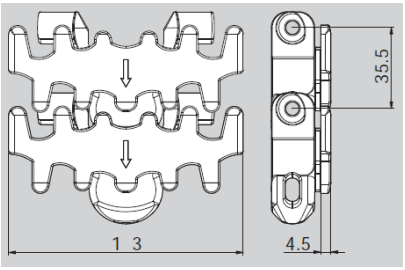


Figura 7.17: Catena con maglie piane [53]

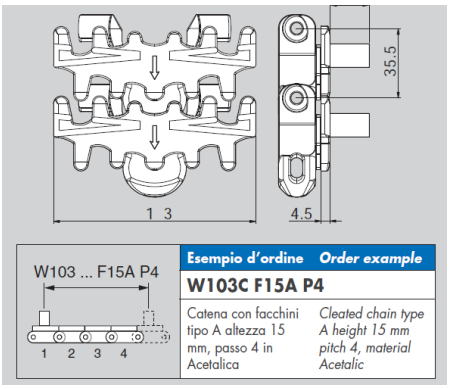


Figura 7.18: Catena con maglie a facchini [53]

La catena per trasportatori è realizzata in resina acetalica, che consente alte velocità di impiego, bassa rumorosità e raggi di curvatura fino a 170 mm. Il perno di giunzione è in acciaio inossidabile.

Telaio

La trave e le strutture di sostegno del nastro trasportatore sono realizzate in barre di alluminio anodizzato che vengono fornite in qualsiasi lunghezza fino a 6 m.

Questo materiale è adatto all'applicazione nella maggior parte degli ambienti di lavoro.

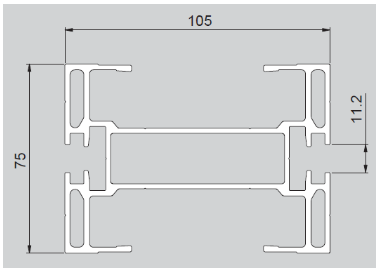


Figura 7.19: Sezione della trave [53]

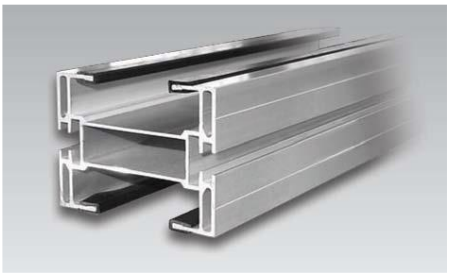


Figura 7.20: Trave 3D [53]

Tabella 7.1: Dati tecnici dei profilati d'alluminio [53]

Lega	AFNOR 6060
Peso specifico	2,7 kg/dm^3
Modulo elastico	66000 N/mm^2

Gruppo di traino

Figura 7.21: Vista laterale sx del gruppo di traino [53]

Figura 7.22: Vista laterale dx del gruppo di traino [53]

Il gruppo di traino costituisce la motorizzazione di testa del trasportatore. Sono disponibili due configurazioni di trasmissione: con motoriduttore a fissaggio diretto con limitatore di coppia con la trasmissione a destra o a sinistra.

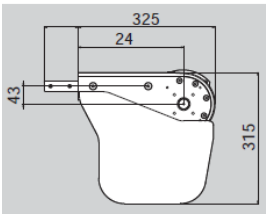


Figura 7.23: Disegno 2D del gruppo di traino

Tabella 7.2: Dati tecnici dei gruppi di traino [53]

Forza di trazione massima	1250 N
Passo catena	35,5 mm
Numero di denti della ruota motrice	12
Diametro primitivo	137,2 mm

Gruppi di rinvio

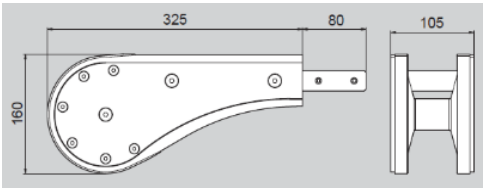


Figura 7.24: Disegno 2D del gruppo di rinvio [53]

Figura 7.25: Vista laterale sx del gruppo di rinvio [53]

Figura 7.26: Vista laterale dx del gruppo di rinvio [53]

Il gruppo di rinvio costituisce l'elemento terminale di un trasportatore con ritorno della catena sulla parte inferiore della trave.

Motore elettrico asincrono trifase IE1

Figura 7.27: Motore elettrico 3D [54]

Figura 7.28: targa del motore elettrico

I motori normalizzati IEC della serie BN sono conformi a tutti gli standard internazionali applicabili, incluse le EMC (Emissioni Elettromagnetiche) ed LV (Bassa Tensione).

Questo motore può essere applicato al gruppo di traino del nastro trasportatore tramite un accoppiamento con flangia, come illustrato nella figura sottostante.

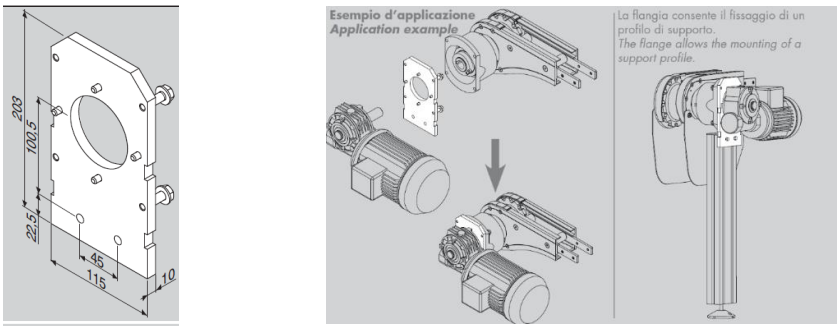


Figura 7.29: flangia di fissaggio del motore elettrico al gruppo di traino [53]

Data la targa del motore sopra, e dalla scheda tecnica in appendice, si può vedere che:

- il motore possiede 0,37 kW di potenza;
- ha classe di protezione IP55;
- è un motore asincrono trifase;
- richiede di essere comandato da un inverter.

Inverter

Un inverter è un dispositivo elettronico capace di convertire una corrente continua in ingresso in una corrente alternata in uscita e di regolarne i valori in ampiezza e frequenza.

Per la regolazione della velocità dei motori trifase, si impiegano inverter a frequenza variabile, perché la velocità di rotazione del motore dipende dalla frequenza della tensione con cui è alimentato.

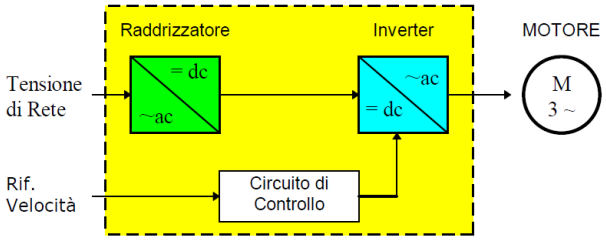


Figura 7.30: schema di funzionamento dell'inverter [55]

All’interno dell’inverter, per prima cosa, la tensione alternata della rete viene raddrizzata in corrente continua, per poi essere subito riconvertita in corrente alternata trifase a frequenza variabile.

Il valore della frequenza si deve scegliere in base alla velocità di rotazione a cui si vuole far girare motore.

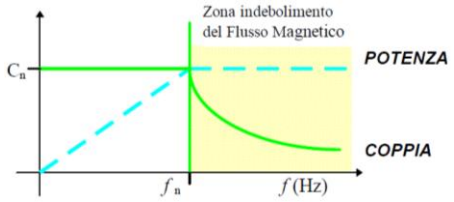


Figura 7.31: Variazione di potenza e coppia per il controllo di velocità [55]

Quindi, aumentando la frequenza oltre a quella di targa, si entra nella zona di indebolimento del flusso magnetico (detta anche a potenza costante), che comporta un calo graduale della coppia motrice.

Per controllare il senso di rotazione del motore, la velocità e la coppia, si ricorre alla tastiera di programmazione integrata nell’inverter.

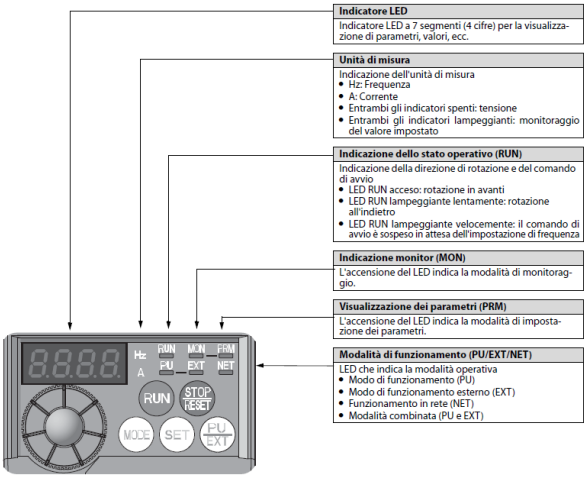
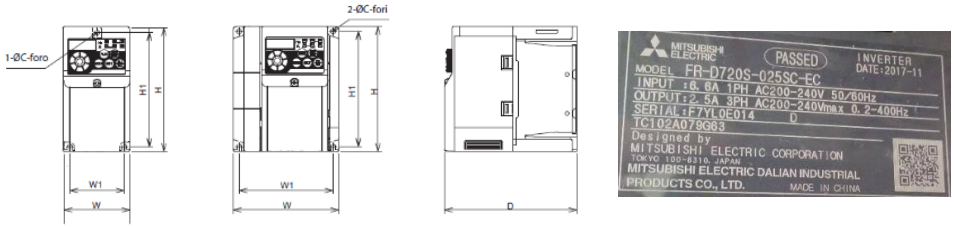


Figura 7.32: tastiera di programmazione integrata [56]

Tramite questa tastiera è anche possibile controllare le modalità di funzionamento dell'inverter, oltre che ai parametri fondamentali meccanici ed elettrici del motore.



Inverter		W	W1	H	H1	D	C
Classe 200 V	FR-D720S-008SC	68	56	128	118	80,5	
	FR-D720S-014SC						
	FR-D720S-025SC					142,5	
	FR-D720S-042SC					162,5	
	FR-D720S-070SC	108	96			155,5	
	FR-D720S-100SC	140	128	150	138	145	

Figura 7.33: Ingombro dell'inverter [56]

Per l'installazione, oltre a dover considerare l'ingombro, bisogna tener conto di una distanza di sicurezza minima dovuta al calore dissipato durante il funzionamento.

7.3 PLC

Il PLC (Programmable Logic Controller) è uno strumento digitale rivolto al controllo dei processi industriali.

Lo si può equiparare ad un PC provvisto di interfacce di ingresso e uscita, in grado di dialogare con dispositivi elettronici di ogni tipo. Rispetto ai semplici personal computer è dedicato al funzionamento in ambiente industriale in cui si possono trovare condizioni problematiche come: temperature elevate, alta umidità, disturbi elettrici, vibrazioni ecc.

Figura 7.34: PLC

Se ci si trovasse a che fare con un'applicazione industriale reale, il PLC si ritroverebbe a gestire numerosi dispositivi dell'impianto. Invece, per ciò che riguarda il prototipo in questione, il PLC si occuperà solamente di gestire il traffico di dati tra la fotocellula ed il PC.

In pratica, senza il PLC sarebbe impossibile catturare le immagini delle mele in movimento sul nastro trasportatore. Ciò è dovuto al fatto che la

fotocellula non può essere attaccata direttamente al PC, bensì ha bisogno di essere gestita tramite un PLC.

In questo caso, la fotocellula invierà un segnale di tipo on-off al PLC, per indicare il passaggio di un facchino mosso dal nastro trasportatore. Questo segnale verrà elaborato dal PLC, per creare una variabile da inviare al computer, che indicherà il numero di volte che è avvenuto il passaggio dei facchini.

7.3.1 Caratteristiche tecniche del PLC

General	
Tipo	Steuerung
Modello	SIMATIC
Versione	CPU 1512C-1 PN
Entrata	32 digital + 5 analog
Uscita	32 digital + 2 analog
Implementation	
Serie	S7-1500
Interfaces	
Interfacce	PROFINET IRT
Electrical values	
Tensione CC	24 V DC V=
Measures	
Larghezza	110 mm
Altezza	147 mm
Profondità	129 mm
Produttore	SIEMENS
Codice articolo del produttore	6ES7512-1CK01-0AB0
Peso dell'imballaggio	1.532 kg
RoHS	conforme
EAN / GTIN	4047623408642

Figura 7.35: Scheda tecnica del PLC SIMATIC S7-1500 [57]

Le CPU di questo PLC sono scalabili in prestazioni, memoria e quantità. Possono essere utilizzate per salvare spazio con ingressi ed uscite integrati ed offrono funzioni di tecnologia integrata (conteggio, misura e posizionamento).

Questa soluzione è particolarmente adatta ad le applicazioni complesse che hanno bisogno di alte prestazioni di calcolo e comunicazione, unite ad alta flessibilità e potenti funzioni di controllo del movimento.

Questi PLC implementano potenti funzionalità di regolazione come i PID per consentire il controllo da un unico dispositivo di tutte le funzionalità richieste da un impianto o da una macchina industriale.

In poche parole, questo tipo di PLC può fare molto di più rispetto a quello che gli viene richiesto, in questo caso particolare.

Inoltre, è abbastanza semplice da utilizzare grazie alle informazioni visualizzabili dal display che consentono di capire velocemente ciò che sta accadendo.

Importante ricordare che la programmazione del PLC è assistita dal TIA Portal (Totally Integrated Automation Portal), l'ambiente di sviluppo che ha permesso di configurare il collegamento tra la fotocellula ed il PC.

7.4 Alimentatore

Ha la funzione di provvedere ai circuiti dei dispositivi a cui è connesso le correnti elettriche opportune, mantenendo un buon isolamento. In questo caso l'alimentatore deve fornire l'energia necessaria al PLC, alla fotocellula e agli illuminatori.

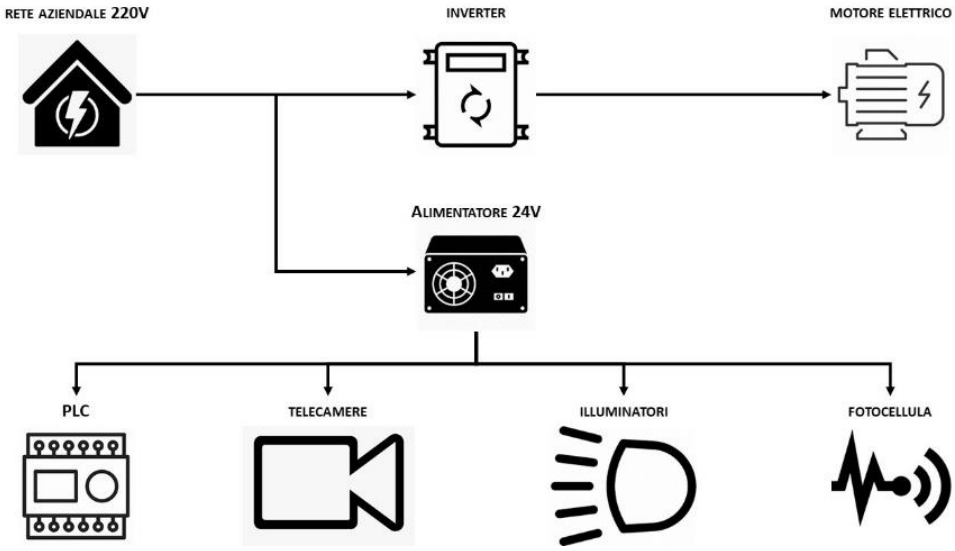


Figura 7.36: Schema di alimentazione del prototipo

Per farlo. È necessario convertire la corrente alternata della rete in corrente continua e modificare i livelli di tensione e di intensità di corrente.

Per prima cosa, la tensione di rete viene raddrizzata e livellata con un condensatore.

Successivamente, sfruttando la tecnologia PWM (Pulse Width Modulation), si modula questa corrente continua in funzione delle esigenze del carico.

Quindi, la tensione modulata viene applicata all'ingresso del trasformatore e la tensione in uscita, viene raddrizzata e livellata.

7.4.1 Caratteristiche tecniche dell'alimentatore

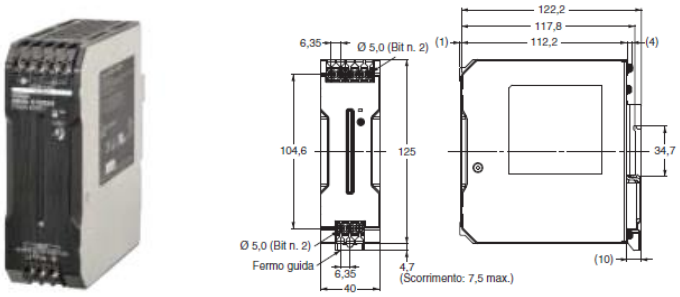


Figura: 7.37: Ingombro dell'alimentatore [58]

Tabella 7.3: Caratteristiche principali dell'alimentatore

Potenza	120 W
Tensione di ingresso	100/240 V
Tensione di uscita	24 V
Corrente in uscita	5A

L'alimentatore S8VK-C12024 è di taglia relativamente piccola, infatti non ci serve ad alimentare utenze che richiedono potenze elevate. In compenso, risulta essere abbastanza ridotto da lasciare abbastanza spazio libero sul quadro elettrico per tutti gli altri componenti, in particolare il PLC, alcuni interruttori magnetotermici e qualche connettore.

Come il PLC, esso necessita di essere montato ad una certa distanza dagli altri apparecchi elettrici per garantire la dissipazione del calore ed evitare il surriscaldamento. Questo vincolo deve essere preso in considerazione nel caso di un impianto reale, ma nel caso del prototipo risulta essere meno importante in quanto non viene utilizzato per un arco di tempo abbastanza lungo.

In ogni caso, per sicurezza, l'ambiente di installazione non deve:

- essere soggetto a urti o vibrazioni;
- presentare sorgenti di disturbi intensi ad alta frequenza e colpi di corrente;
- essere esposto alla luce solare diretta;

- presentare liquidi, corpi estranei o gas corrosivi che potrebbero penetrare all'interno del prodotto.

7.5 Cattura di oggetti in movimento

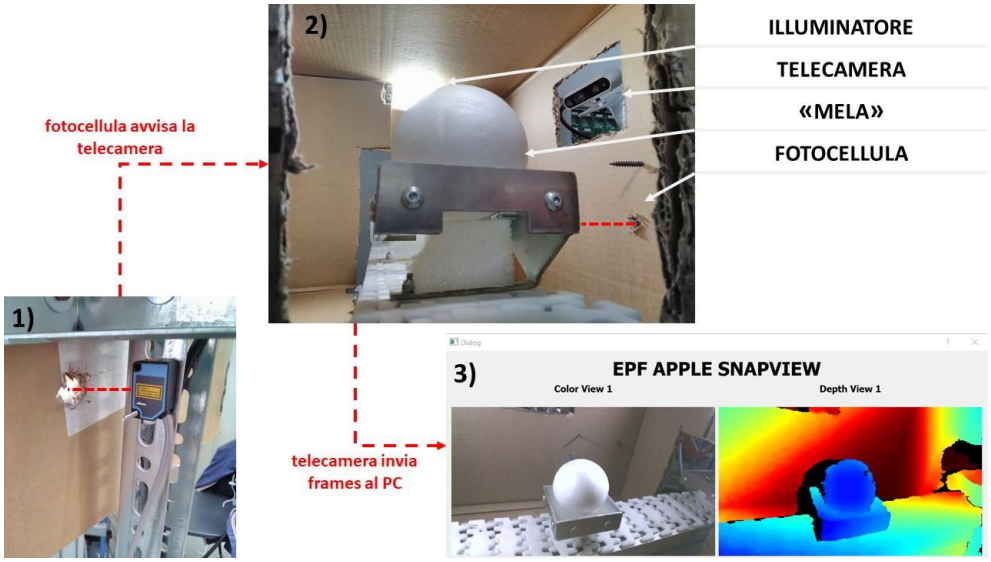


Figura 7.38: Svolgimento dei test

Questa è la fase di collaudo finale, quindi è necessario provare il sistema completo nelle sue condizioni di lavoro effettive.

Questi test sono avvenuti all'interno dell'officina dell'azienda EPF, per cui luce, vibrazioni, polvere, ecc. non sono ottimali, esattamente come dovrebbe accadere nella realtà. Inizialmente, sono stati aggiunti degli illuminatori per far sì che l'intensità della luce irradiata sulle mele rimanga circa costante, successivamente è stato installato uno scatolone per isolare al meglio possibile l'area di rilevazione dall'ambiente circostante.

Con questo sistema si è effettuata ancora una volta il test della calibrazione, che questa volta è riuscito senza problemi, senza dover

aggiungere ulteriori righe di codice. Di questo test è inutile parlarne in quanto è identico a quello descritto nel paragrafo “calcolo delle coordinate” e non sono presenti nuovi fattori di disturbo rispetto alla situazione precedente.

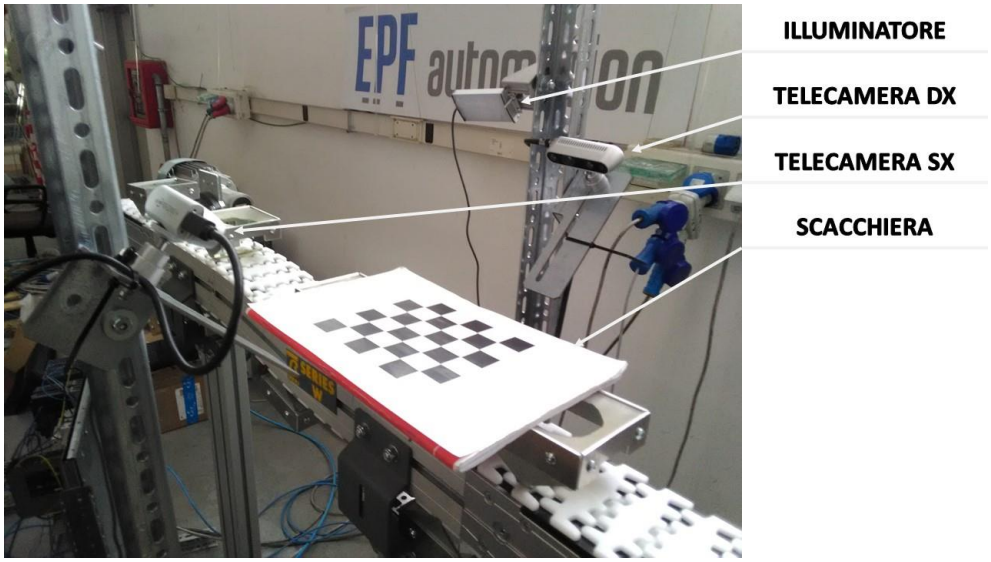


Figura 7.39: Calibrazione sul nastro trasportatore

Quando però, si è deciso di installare lo scatolone, è sembrato opportuno integrare la scacchiera di calibrazione con un facchino. Ciò ha reso necessario l’utilizzo di una scacchiera molto più piccola, sia in dimensioni assolute, che nelle dimensioni di ogni scacco.

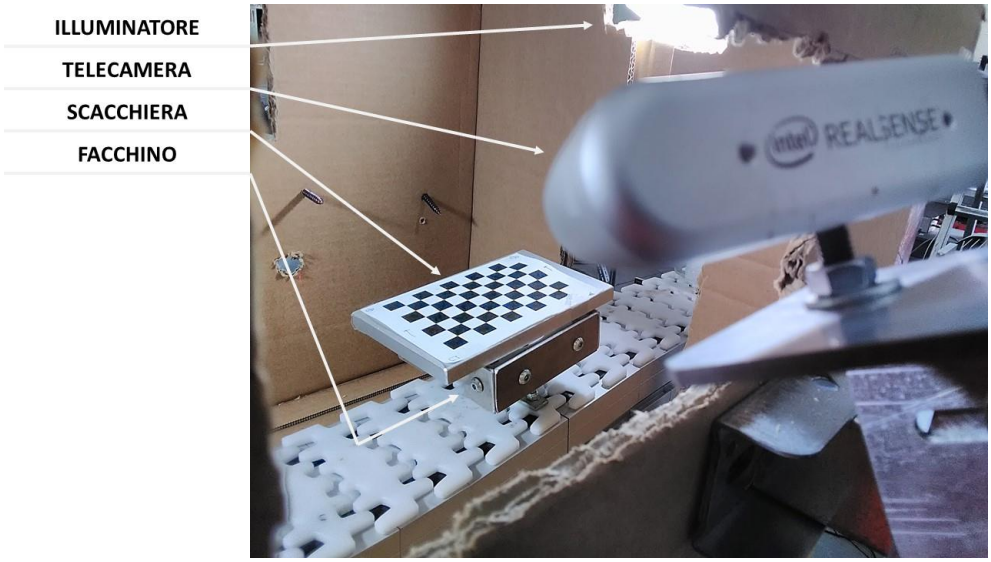


Figura 7.40: Calibrazione dentro lo scatolone

Purtroppo, si è notato che questo tipo di scacchiera era difficilmente rilevabile dalle telecamere e, se rilevata correttamente, portava un incremento dell'errore RMS fino anche a 4mm, il che non è assolutamente accettabile per l'applicazione in questione.

Conseguentemente, si è dovuto fare un passo indietro, e anziché utilizzare la scacchiera integrata con il facchino, si è continuato ad utilizzare la scacchiera "libera" che è più facilmente rilevabile e garantisce un errore RMS molto più basso.

Dopo essersi assicurati che il sistema fosse in grado di rilevare le coordinate assolute degli oggetti, si è potuto procedere al vero test finale, ovvero, catturare le immagini 3D delle mele trasportate dai facchini in movimento.

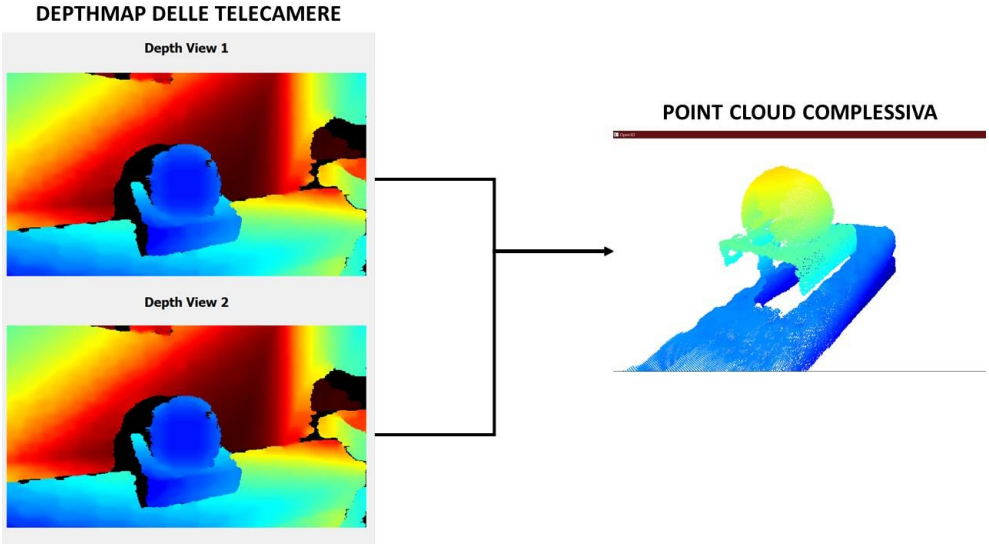


Figura 7.41: Elaborazione della point cloud complessiva

Quando il facchino oscura il laser della fotocellula il computer riceve un segnale che indica al programma che è arrivato il momento, per le telecamere, di scattare la foto 3D della mela. Tutto ciò avviene automaticamente, ma per vedere il risultato, è necessario processare l'immagine delle coordinate relative da un'altra parte del programma che si dedica interamente all'elaborazione delle point cloud.

La finestra di interfaccia del programma che cattura le depth map delle mele in movimento è in grado di mostrarle a schermo, ed è in grado di aggiornare le immagini ad ogni ciclo, ma la qualità delle singole depth map non basta per giudicare il sistema. Quando si crea la point cloud complessiva, si uniscono i punti delle immagini 3D catturate dalle 2 telecamere, e quest'azione comporta dei nuovi problemi.

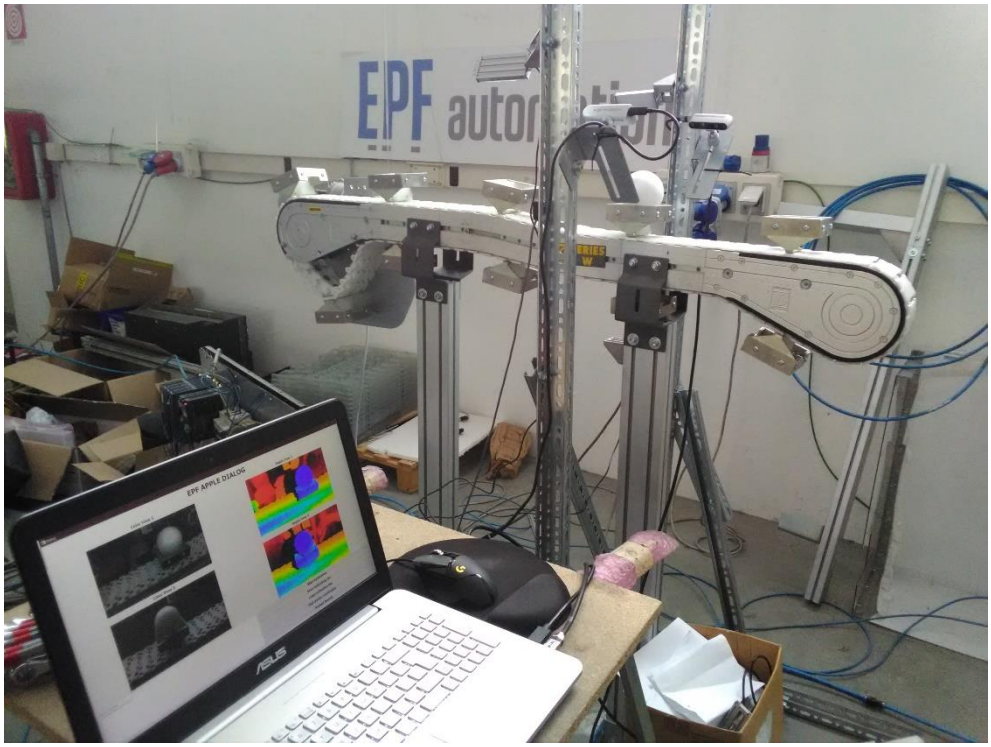


Figura 7.42: Visualizzazione dei frame delle telecamere durante il processo

Infatti, le prime volte che è stato effettuato il test, tutto è sembrato andare per il meglio. Il programma impiega pochi centesimi di secondo ad effettuare tutte le operazioni necessarie per cui la cattura delle immagini è istantanea dopo il passaggio davanti alle fotocellule. Visualizzando le immagini sulla finestra di interfaccia si riusciva a capire che effettivamente il programma stava funzionando, in quanto il ritmo con cui apparivano le immagini sullo schermo rispecchiava ciò che avveniva nella realtà. Anche la qualità delle immagini di profondità sembrava abbastanza alta.

Poi, però, andando ad elaborare queste matrici di coordinate relative sono sorti i primi dubbi. Mentre le point cloud delle singole telecamere non sembravano presentare errori, la point cloud complessiva aveva un aspetto “sfasato” come si può notare nella figura sotto.

vista superiore **vista isometrica**

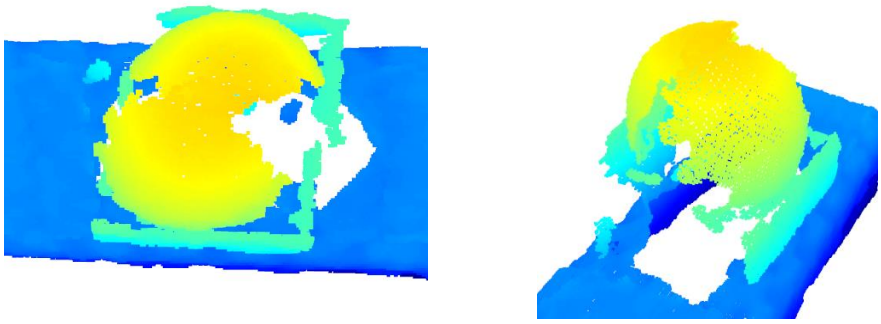


Figura 7.43: point cloud errata ottenuta dalle prove iniziali

Inizialmente si è pensato che le telecamere prendessero le immagini dallo stream di profondità in momenti troppo distanti nel tempo, ma tramite la funzione "time" si è potuto verificare che ogni ciclo di cattura dell'immagine dura pochi centesimi di secondo, il che non può comportare un errore di accoppiamento delle immagini così evidente.

Questo test è stato effettuato più volte, per vedere se lo "sfasamento" delle due immagini variasse in qualche modo. Come prevedibile, aumentando la velocità del nastro trasportatore si è verificato un aumento dello sfasamento. Inoltre, si è potuto constatare che la direzione dello sfasamento fosse sempre la stessa.

Quindi, se la direzione è costante, tra i fattori di errore possibili si possono togliere sia le vibrazioni che la luce solare, in quanto entrambi dovrebbero portare disturbi variabili nel tempo.

Se l'aumento della velocità incrementa l'errore, significa che la variabile che porta questi problemi è comunque legata al tempo. Visto che il problema non stava nella cattura delle immagini, si è spostata l'attenzione sul processamento delle immagini.

Durante la fase di calibrazione, quando si rileva la depth map della scacchiera e dell'oggetto di riferimento, è necessario che essa sia

precisa e priva di buchi. Per eliminare in ogni modo questa eventualità, alla mappa di profondità vengono applicati due filtri: uno spaziale ed uno temporale.

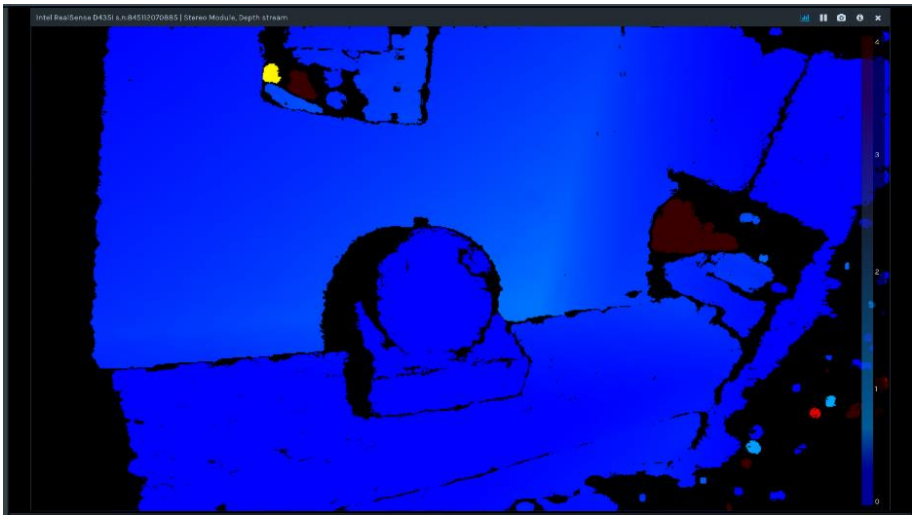


Figura 7.44: Foto della depth map senza filtri

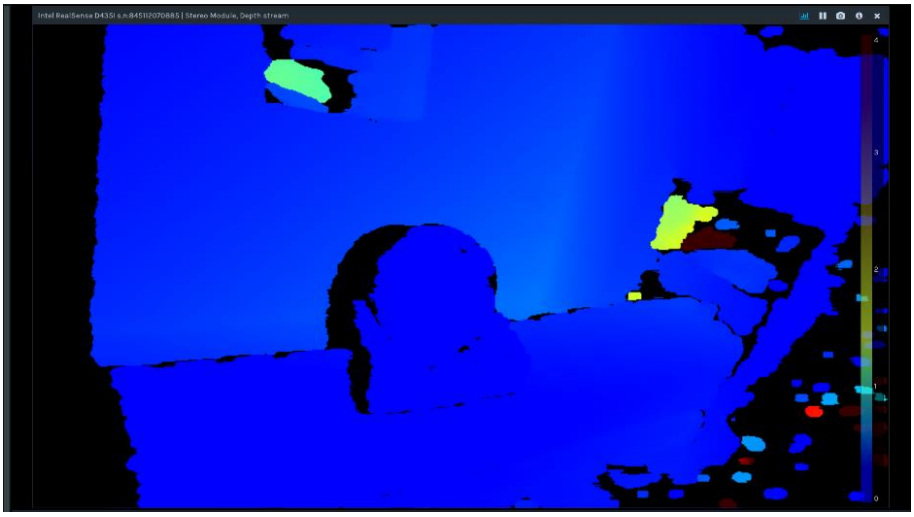


Figura 7.45: Foto della depth map con filtro spaziale e temporale

Il filtro spaziale consente di riempire i “buchi” nella mappa di profondità, andando a considerare i valori presenti intorno ad essi. Invece, il filtro temporale, opera sullo stesso buco, ma ad esso va a sostituire il valore medio che era presente nella sua “storia”.

L’operazione eseguita dal filtro temporale non è sbagliata nel caso della calibrazione in quanto la scacchiera e l’oggetto di riferimento permangono nel tempo, non restano davanti alle telecamere per pochi decimi di secondo.

Invece, in questo caso il filtro temporale può fare solo danni, in quanto è più importante avere che si riferisce ad un momento preciso, piuttosto che un’immagine priva di buchi. “Fondere” insieme le immagini di momenti diversi, ha causato la comparsa di una scia che si è manifestata nello sfasamento nella pointcloud, a causa del fatto che le telecamere si trovassero una di fronte all’altra.

Dopo aver eliminato il filtro temporale dalla parte di codice che tratta questa fase, il collaudo è stato ripetuto e ha dato esiti diversi.

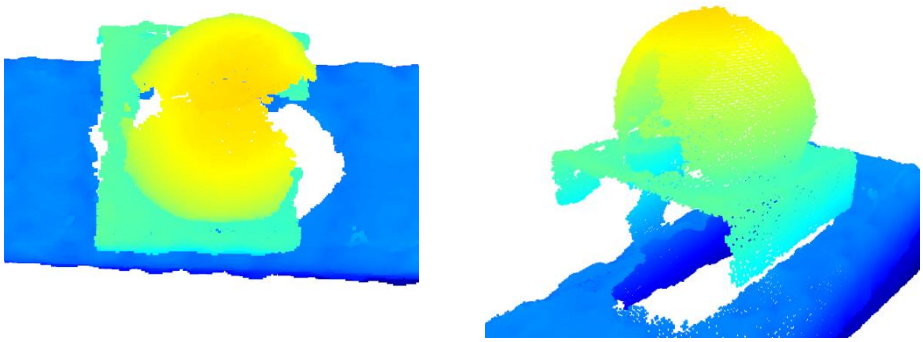


Figura 7.46: Point cloud complessiva senza filtro temporale (caso 1)

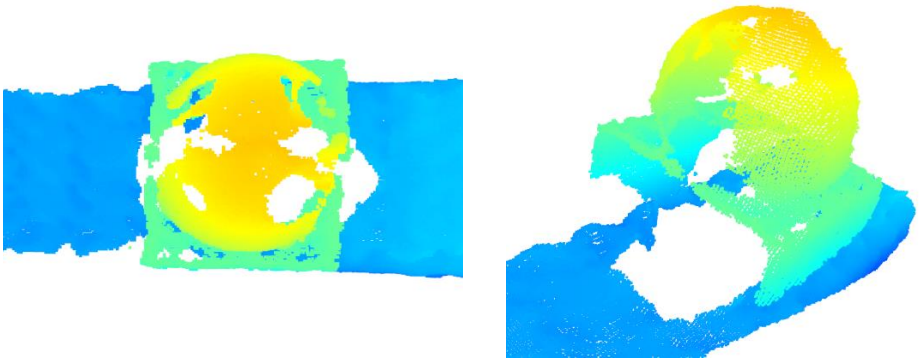


Figura 7.47: Point cloud complessiva senza filtro temporale (caso 2)

In nessun caso è avvenuto lo sfasamento dovuto al movimento del facchino sul nastro trasportatore, ma alcune volte, come si può vedere nel caso 2, la point cloud ha presentato dei vuoti. Siccome ciò è avvenuto in modo quasi randomico, è venuto il dubbio che ciò fosse dovuto all'illuminazione non omogenea dell'area del prototipo. Infatti, l'illuminazione proviene da delle finestre posizionate sopra di esso, per cui è possibile che si generino dei fasci di luce concentrata che impediscano alle telecamere di rilevare i dati di profondità correttamente. Per cui si è deciso di montare uno scatolone in modo da isolare (nel modo migliore possibile) la zona di rilevamento delle depth map dall'ambiente circostante. Dall'immagine sotto, osservando i frame RGB catturati dalle telecamere (Color View) si può notare che, effettivamente, l'isolamento dalla luce è più che soddisfacente.

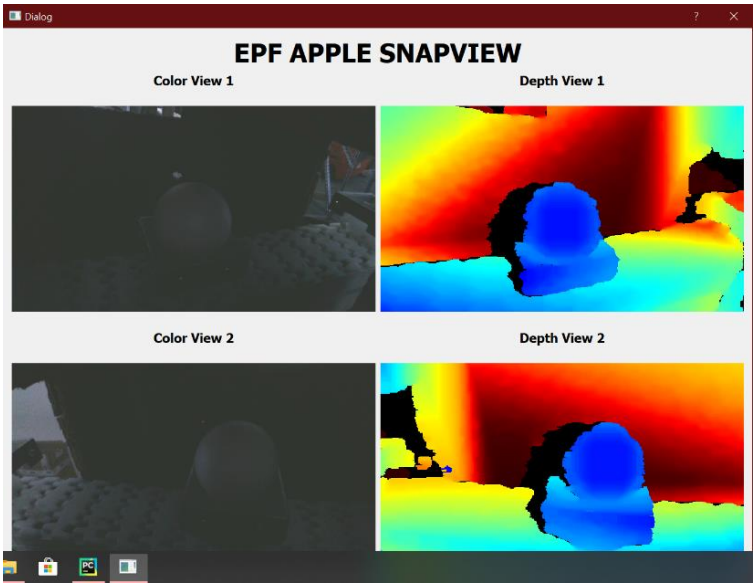


Figura 7.48: Visualizzazione dei frame delle telecamere al buio

Provando diversi livelli di illuminazione all'interno della zona di rilevazione, si è scoperto che i raggi di luce, sia quelli solari, che quelli degli illuminatori, tendono ad ostacolare la corretta rilevazione dei dati di profondità. Come si potrà notare dalle immagini successive, si è arrivati alla conclusione che, il modo migliore per ottenere delle buone point cloud è quello di fare rilevazioni completamente al buio.

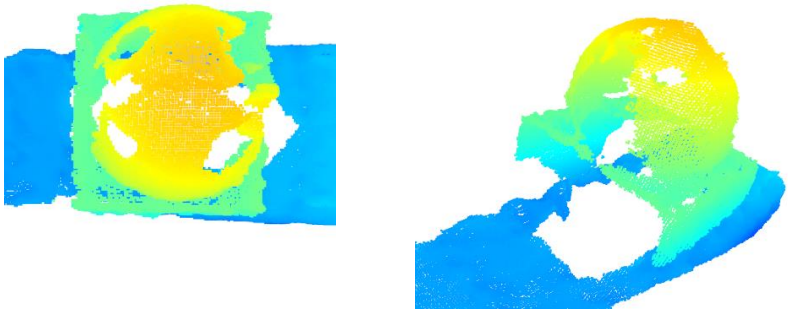


Figura 7.49: Point cloud con illuminazione forte

Figura 7.50: Point cloud con illuminazione debole

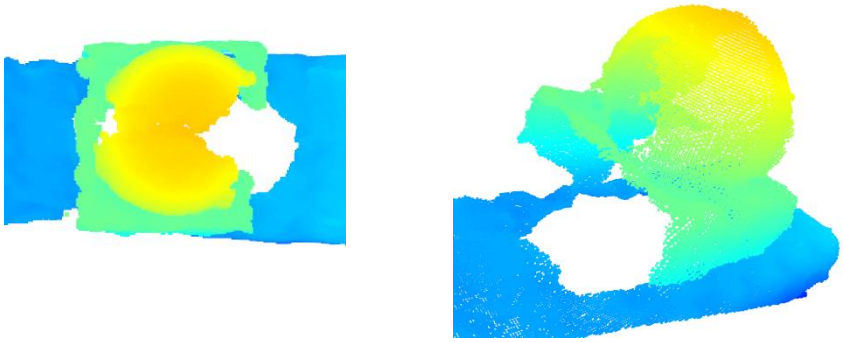


Figura 7.51: Point cloud senza illuminazione

Nonostante questo, non è consigliabile fare rilevazioni al buio. Anche se la qualità della depth map risulta essere molto alta, sembra ritornare lo sfasamento dovuto al movimento. Probabilmente, ciò si può spiegare ricordando che l'accoppiamento dei punti del paio di immagini delle telecamere dipende anche dalla qualità delle immagini RGB, le quali sono molto difficili da utilizzare, perché l'intensità di colore di questi frame, in questo caso, risulta essere quasi uniforme.

Quindi, il caso ottimale, si otterrebbe con un'illuminazione diffusa e uniforme, in modo tale che:

- non si generano fasci di luce che disturbano il laser delle telecamere;

- i frame RGB siano facilmente utilizzabili per l'accoppiamento dei punti del paio di immagini stereoscopiche.

Questo caso si è realizzato casualmente in alcuni dei test in cui lo scatolone era ancora assente. Ciò è dovuto al fatto che, probabilmente, nell'ora giusta della giornata, la luce solare non colpiva direttamente il nastro trasportatore e si diffondeva in modo uniforme all'interno dell'area.

Figura 7.52: Point cloud con illuminazione diffusa

Dopo aver rilevato un numero moderato di point cloud si può dire che:

- si può stimare che l'errore massimo sulla superficie della mela sia di circa 1mm, però si possono notare delle criticità sui bordi tangenti ai coni di visione delle telecamere, probabilmente, dovuti ad una difficoltà nel riflettere il laser delle telecamere su superfici quasi parallele ad esso;
- verticalmente, si riesce ad osservare sempre almeno il 70% della superficie della mela;
- orizzontalmente, non si riesce ad osservare la superficie della mela completamente, ma il problema sarebbe facilmente risolvibile aggiungendo una terza telecamera.

Conclusioni

Il confezionamento delle mele è un processo molto critico a causa della cura con cui le mele devono essere maneggiate. Inoltre, per aumentare l'efficienza e la ripetibilità dell'operazione diverse aziende hanno optato per automatizzare il processo mediante l'applicazione di robot accoppiati a sistemi di visione avanzati.

Nel terzo capitolo è stato analizzato il sistema: POM – Automatic Smart Packing System sviluppato dall'azienda EPF e si è valutata la possibilità di ottenere un miglioramento mediante la sostituzione del profilatore 3D, con una coppia di telecamere stereoscopiche.

Le telecamere stereoscopiche avrebbero avuto la funzione primaria di rilevare le coordinate delle mele sul nastro trasportatore in modo tale da permettere il picking mediante il robot già presente all'interno del sistema POM.

Per prima cosa è stata valutata la precisione e la flessibilità delle telecamere, tenendo conto dell'utilizzo in ambito industriale. Il che significa che:

- le coordinate devono essere rilevate con un errore ≤ 1 mm
- i fattori ambientali non devono introdurre errori, oppure devono essere controllabili

Il programma di rilevamento delle telecamere non era già disponibile per cui è stato necessario crearlo su Python. Fondamentalmente si è trattato di creare un software che permette l'esecuzione di una calibrazione che avviene in due fasi:

1) per prima cosa, ogni telecamera riesce ad individuare un proprio sistema di riferimento, semplicemente vedendo all'interno del proprio campo visivo, una scacchiera di dimensioni date;
2) successivamente, si può utilizzare un oggetto 3D (a scelta) per individuare la direzione e l'unità di lunghezza, degli assi del

sistema di riferimento che dovrà essere comune ad entrambe le telecamere.

Mentre il primo punto avviene automaticamente, il secondo necessita di un operatore che clicchi sullo schermo del PC sui punti "notevoli" dell'oggetto 3D.

Creato, questo software, è stato possibile confermare le schede tecniche delle telecamere, che indicavano:

- l'errore teorico del dato di profondità rilevato dalle telecamere per oggetti ad una distanza inferiore a 0,5m è sempre ≤ 1 mm.

Inoltre, durante questi test, si sono provate molte impostazioni dei settaggi delle telecamere per cercare di avere una mappa di profondità priva di "buchi" ed allo stesso tempo con una buona risoluzione.

Fin da subito si è notato che l'illuminazione è uno dei fattori ambientali più determinanti nella qualità delle immagini 3D.

Oltre a questo, è stata costruita una struttura modulare per verificare l'effetto dell'inclinazione delle telecamere sulla visibilità della superficie della mela. Infatti, per prendere la mela è necessario che il robot individui lo stelo o il calice, per cui l'immagine senza una di esse risulterebbe inutile.

Dopo uno studio teorico ed una verifica pratica si è verificato che:

- se le telecamere hanno un'inclinazione inferiore ai 50° rispetto all'asse orizzontale, allora sarà sempre possibile vedere almeno il 70% della superficie della mela;
- al di sotto dei 30° di inclinazione si rischia di abbassare la qualità della mappa di profondità a causa dei riflessi della luce sull'oggetto da osservare.

Passato questo punto, la verifica delle telecamere in condizioni statiche si poteva definire concluso, ed è iniziata la costruzione di un vero e

proprio prototipo per controllarne la precisione e l'affidabilità anche nella cattura di immagini 3D in movimento.

Per farlo è stato necessario scrivere un software che svolgesse in modo completo la gestione del sistema di visione nella stessa situazione dell'applicazione reale.

Il prototipo assemblato è costituito da:

- un piccolo nastro trasportatore mosso da un motore elettrico;
- una fotocellula: per rilevare il passaggio dei facchini;
- un PLC: per inviare il segnale dalla fotocellula ad un PC;
- due telecamere stereoscopiche: per catturare le immagini 3D delle mele sui facchini;
- due illuminatori: per regolare il livello di luce nella zona di rilevazione delle immagini 3D;
- un carter: per isolare l'area di rilevazione delle immagini 3D dalla luce esterna;

Mettendo una sfera che rappresenta la mela ideale sui facchini del nastro trasportatore, il sistema ha il compito di:

1) rilevare il passaggio del facchino di fronte alle telecamere stereoscopiche;
2) catturare i frame degli stream RGB e di profondità delle telecamere;
3) mostrare i frame catturati all'operatore;
4) costruire le matrici di coordinate relative sulla base delle mappe di profondità rilevate dalle telecamere e salvarle in memoria.

Date le matrici di coordinate relative, è stato necessario costruire un altro eseguibile in grado di elaborarle in modo da ricavare le matrici di coordinate assolute corrispondenti e combinarle in un'unica point cloud.

Eseguendo queste operazioni a diversi livelli di luminosità si è verificato che:

- le impostazioni di post processing delle immagini devono essere regolate in modo da evitare che il movimento “spezzi” l’unione delle immagini, in particolare è assolutamente da evitare l’applicazione di un filtro temporale;
- i fasci di luce, anche debole, infastidiscono il laser delle telecamere stereoscopiche, creando dei vuoti aggiuntivi notevoli nelle mappe di profondità;
- il sistema lavora meglio in condizioni di luce diffusa in quanto sia le immagini RGB che quelle 3D rilevate in questo modo sono di alta qualità;
- la precisione delle telecamere rimane quella rilevata in condizioni statiche: in generale si ottiene un errore $\leq$ 1mm, ma si può notare che ci sono dei problemi nel rilevare i bordi tangenti al cono visivo delle telecamere;
- verticalmente si riesce a vedere sempre più del 70% della superficie della mela, mentre orizzontalmente, purtroppo non si riesce ad ottenere la mela completa.

Le point cloud ottenute con questo sistema di visione si potrebbero migliorare facilmente aggiungendo una terza telecamera. Infatti, oltre ad ottenere l’intera superficie della mela, si avrebbero meno problemi sui bordi tangenti ai coni visivi delle telecamere, in quanto ai punti errati di una telecamera si sommerebbero i punti corretti di un’altra.

Purtroppo, non è stato possibile farlo all’interno dell’ambito di questa tesi, ma sarebbe stato interessante implementare un programma che consentisse ad un’intelligenza artificiale l’individuazione dello stelo e del calice della mela sulle point cloud ottenute dal prototipo.

Dopo questo passaggio, si potrebbe costruire un altro prototipo, questa volta munito anche di robot, in modo da verificare il funzionamento completo di questa evoluzione del sistema POM.

Oltretutto, sempre applicando un algoritmo di intelligenza artificiale, si potrebbe implementare un programma che consenta l’analisi delle

immagini a colori delle telecamere stereoscopiche per valutare la qualità delle mele e la presenza/assenza di danni sulla superficie.

Bibliografia

[1] https://www.fas.usda.gov/data/fresh-apples-grapes-and-pears-world-markets-and-trade (22/06/2020)
[2] http://www.assomela.it/ (22/06/2020)
[3] https://www.colturaecultura.it/content/post-raccolta-1 (22/06/2020)
[4] http://www.laimburg.it/it/ (22/06/2020)
[5] https://www.istitutoagrariosartor.edu.it/wp-content/uploads/2016/10/Maturit%C3%A0-frutta-IA-2013.pdf (22/06/2020)
[6] https://present5.com/la-maturazione-della-frutta-dalpiaz-ferruccio-cosa-avviene/ (22/06/2020)
[7] https://www.amazon.it/Newtry-Penetrometro-Sclerometro-maturazione-Penetrometer/dp/B07239NZMC (22/06/2020)
[8] https://www.fmach.it/ (22/06/2020)
[9] https://www.semanticscholar.org/paper/Analysis-of-Volatile-Compounds-by-Advanced-and-Lubes-Goodarzi/f5318baaac7eb8f347c5bc2df5fb1fdcf7f58a42 (22/06/2020)
[10] https://www.sciencedirect.com/science/article/pii/S002364381500184X (22/06/2020)
[11] https://www.cooperazionetrentina.it/Ufficio-Stampa/Notizie/E-in-Valle-di-Non-il-primo-impianto-al-mondo-per-la-conservazione-ipogea-delle-mele (22/06/2020)
[12] https://www.sciencedirect.com/science/article/abs/pii/S092552141530199X (22/06/2020)
[13] https://www.seeedstudio.com/blog/2019/08/21/how-to-overcome-the-disadvantages-using-different-co2-sensors/ (22/06/2020)
[14] http://www.michell.com/uk/documents/appnotes/Oxygen_Measurement_in_Natural_Gas.pdf (22/06/2020)

[15] https://www.interempresas.net/Poscosecha/Articulos/141277-Novedad-tecnologica-para-el-almacenamiento-de-manzanas-atmosfera-controlada-dinamica.html (22/06/2020)
[16] https://www.maf-roda.com/en/page/apples.php (22/06/2020)
[17] https://www.greefa.com/product/infeed/ (22/06/2020)
[18] https://www.youtube.com/watch?v=6CN1vcMpVPI (22/06/2020)
[19] http://www.icoel.com/prodotto/mele/ (22/06/2020)
[20] https://www.futura-grading.com/depliant/depliant-grading-2018-en.pdf (22/06/2020)
[21] https://www.fanuc.eu/it/it/robot/robot-filter-page/robot-collaborativi/collaborative-cr7ial (22/06/2020)
[22] Tesi POM: Automatic Smart Packing System
[23] https://www.stemmer-imaging.com/media/uploads/intel/IN/INTEL-FR-RealSense-Depth-Camera-technology-review-for-acquisition-of-3D-data.pdf (22/06/2020)
[24] https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/BKMs_Tuning_RealSense_D4xx_Cam.pdf (22/06/2020)
[25] https://www.glassonline.com/machine-vision-market-worth-9-50-billion-by-2020/ (22/06/2020)
[26] https://slideplayer.it/slide/541259/ (22/06/2020)
[27] https://docdomusdejanas.wixsite.com/docdomusdejanas/single-post/2016/12/13/MODELLI-3D-E-COME-CREARLI (22/06/2020)
[28] http://www.ce.unipr.it/~rizzini/student_theses/TesiBarbieri_triennale.pdf (22/06/2020)
[29] http://elettronica-plus.it/sistemi-di-visione_67341/ (22/06/2020)
[30] https://www.imagesspa.it/ottica/ (22/06/2020)

[31] https://www.lumenstore.it/servizio-clienti/guida-ai-led (22/06/2020)
[32] https://www.edmundoptics.com/knowledge-center/application-notes/illumination/choose-the-correct-illumination/ (22/06/2020)
[33] http://web.tiscali.it/QUADRA/vision.htm (22/06/2020)
[34] http://www.qualivision.biz/sistemivisione.php (22/06/2020)
[35] https://chuyentaomau.com/may-scan-3d-kiem-tra-3d-quang-so-1293/ (22/06/2020)
[36] https://www.tecnelab.it/news/tecnologie/i-sensori-precisi-e-affidabili-di-wenglor-sensoric-sono-ideali-per-le-operazioni-di-bin-picking (22/06/2020)
[37] https://www.cognex.com/library/media/industry/food-and-beverage/assembly-verification/vision-guided-robotics-pick-and-place.jpg?h=357&w=500&la=en&hash=B41226620E140F604BFF15E3D290156496D258A5 (22/06/2020)
[38] https://www.datalogic.com/_img_path_900/upload/image_gallery/jpg/products/UR_Robot_pickandplace.jpg (22/06/2020)
[39] https://www.youtube.com/watch?v=kzBXjJiKZRA (22/06/2020)
[40] https://www.automationmagazine.co.uk/articles/a-new-imaging-approach-for-web-inspection/ (22/06/2020)
[41] https://semiengineering.com/deep-learning-spreads/ (22/06/2020)
[42] https://www.artera.net/it/blog/programmazione/programmare-con-python/ (22/06/2020)
[43] http://www.webprecious.com/why-pycharm-is-becoming-important-for-every-python-programmer/ (22/06/2020)
[44] https://dev.intelrealsense.com/docs/tuning-depth-cameras-for-best-performance (22/06/2020)

[45] http://www.meccanicaeautomazione.it/la-libreria-pyqt5/ (22/06/2020)
[46] https://vitolavecchia.altervista.org/programmazione-orientata-agli-oggetti-cose-incapsulamento/ (22/06/2020)
[47] https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/Python1a_OOP.html (22/06/2020)
[48] https://www.researchgate.net/figure/Function-principle-of-multithreading-One-master-thread-is-responsible-for-executing-the_fig22_275271188 (22/06/2020)
[49] www.stemilt.com (22/06/2020)
[50] https://www.applesfromeurope.eu/for-professionals/commercial-quality-of-apples (22/06/2020)
[51] https://www.panasonic-electric-works.com/pew/it/downloads/ti_construction_working_principle_en.pdf (22/06/2020)
[52] https://support.industry.siemens.com/cs/document/109486139/come-si-configura-un-collegamento-in-step-7-(tia-portal)-?dti=0&lc=it-IT (22/06/2020)
[53] http://www.bettsistemi.com/w-series-components-for-chain-conveyors-system-p46.html (22/06/2020)
[54] https://www.bonfiglioli.com/italy/it/prodotto/serie-bn (22/06/2020)
[55] http://www.plcforum.info/didattica/dalpra/Guida_Inverter.Pdf (22/06/2020)
[56] https://it3a.mitsubishielectric.com/fa/it/dl/9778/260532.pdf (22/06/2020)
[57] https://www.reichelt.com/it/it/s7-1500-cpu-1512c-1-pn-s7-1512c-pn-p267952.html?PROVID=2788&gclid=Cj0KCQjww_f2BRC-ARIsAP3zarHK-1LCrQPaCyTG2SU-8jH8tnb66A0OklJ6yX-m3Ie9PxpJ38jSfI0aAkyyEALw_wcB&&r=1 (22/06/2020)
[58] https://assets.omron.eu/downloads/datasheet/it/v4/t058_s8vk-c_switch_mode_power_supply_datasheet_it.pdf (22/06/2020)