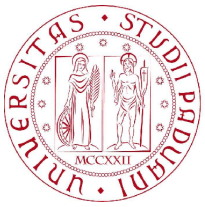


UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
TESI DI LAUREA

BIN PICKING ROBOT: ALGORITMI DI VISIONE E FRAMEWORK SOFTWARE

RELATORE: Ch.mo Prof. Emanuele Menegatti

LAUREANDO: *Stefano Tonello*

Padova, 9 marzo 2010

Sommario

La maggior parte dei sistemi attuali basati su robot manipolatori utilizzati in ambiente industriale non includono dei sensori avanzati per la percezione dell'ambiente. I manipolatori sono quindi *cechi* e vengono programmati solo per riprodurre continuamente la stessa sequenza di movimenti. In queste condizioni, hanno la necessità di operare in un ambiente particolarmente strutturato. D'altra parte, questo non è sempre facile da ottenere, soprattutto per lotti di produzione limitati e quindi soprattutto per le piccole e medie imprese. La cella di lavoro può infatti essere riconfigurata, i prodotti possono variare in forma e dimensione oppure possono essere posizionati alla rinfusa dal processo di produzione precedente. La flessibilità delle celle di lavoro può crescere considerevolmente permettendo al robot manipolatore di avere percezione dell'ambiente in cui opera. In questo contesto, la tesi si pone come obiettivo la progettazione di un sistema di visione che consenta la soluzione del problema del bin picking. Il bin picking consiste nell'individuazione degli oggetti disposti alla rinfusa all'interno di un contenitore. L'oggetto così individuato potrà quindi essere afferrato da un robot manipolatore, per poi essere soggetto al processo industriale designato. In questo modo vengono posti meno vincoli tra i diversi processi produttivi, facilitando la loro integrazione ed aumentando di conseguenza la flessibilità della cella di lavoro.

In particolare, nel presente studio verrà posta attenzione sui sistemi basati su modelli, nei quali il riconoscimento consiste nel cercare all'interno dell'immagine acquisita la posa dei prodotti le cui specifiche geometriche sono fornite in ingresso al sistema. L'obiettivo dell'algoritmo è quello di trovare la posizione ed orientazione dell'oggetto target all'interno del contenitore con una precisione sufficiente per consentire al robot manipolatore di afferrare l'oggetto. Nella presente tesi è stato sviluppato un nuovo algoritmo che, in base agli esperimenti effettuati, si è dimostrato avere un efficienza computazionale superiore e di essere più robusto alle variazioni delle condizioni di illuminazione degli approcci più diffusi per la

risoluzione del problema presenti in letteratura.

Mediante la progettazione di un framework specifico per le applicazioni di visione, la tesi si propone inoltre di fornire un ambiente di sviluppo che consenta di ottenere vantaggi quali la maggiore facilità di debugging e manutenzione del codice e una maggiore semplicità nel riutilizzo dei componenti, oltre che a fornire strumenti per lo sviluppo di architetture più complesse che consentano la creazione di sistemi distribuiti, paralleli e con vincoli real-time.

Indice

Elenco delle figure

Capitolo 1

Introduzione

1.1 Scopo della tesi

Lo scopo della tesi è lo sviluppo di un sistema di visione che consenta la soluzione del problema del bin picking. Il bin picking consiste nell'individuazione degli oggetti disposti alla rinfusa all'interno di un contenitore. L'oggetto così individuato potrà quindi essere afferrato da un robot manipolatore, per poi essere soggetto al processo industriale designato.

La maggior parte dei sistemi attuali basati su robot manipolatori utilizzati in ambiente industriale difficilmente includono dei sensori avanzati per la percezione dell'ambiente. La sensoristica comunemente presente in una cella di lavoro include solamente sensori elementari e specializzati per un singolo obiettivo. Ad esempio sono presenti sensori per la verifica della presenza di un pezzo o per controllo di spessore della lamiera. I manipolatori sono quindi *cechi*, e vengono programmati solo per riprodurre continuamente la stessa sequenza di movimenti. In queste condizioni, hanno la necessità di operare in un ambiente particolarmente strutturato. D'altra parte, questo non è sempre facile da ottenere, soprattutto per lotti di produzione limitati e quindi soprattutto per le piccole e medie imprese. La cella di lavoro può difatti essere riconfigurata da un prodotto all'altro, i prodotti possono variare in forma e dimensione, oppure possono essere posizionati alla rinfusa dal processi di produzione precedente, il quale può includere macchine che non consentano la corretta e precisa pallettizzazione del prodotto. La flessibilità delle celle di lavoro può crescere considerevolmente utilizzando gli opportuni sensori per permettere al robot manipolatore di avere percezione dell'ambiente in

cui opera.

L'aggiunta di sensoristica avanzata alle attuali celle di lavoro, come telecamere standard o telecamere 3D basate su luce strutturata, ha ottenuto sempre più interesse nella sia nella comunità scientifica che in quella industriale. Gli obiettivi principali a lungo termine [2] sono quelli di semplificare l'iterazione tra uomo e robot nei processi industriali, la creazione di celle di lavoro più flessibili delle attuali, e, in ultima istanza, la creazione di *cognitive factories*, ovvero industrie contenenti robots che collaborino attivamente ed autonomamente per l'esecuzione del processo produttivo interagendo con gli esseri umani. In questo contesto, lo sviluppo di un sistema di bin picking consente di ottenere una maggiore flessibilità vincolando in modo meno stringente il processo produttivo che precede quello attuale, facilitando l'integrazione tra i diversi processi produttivi. Non basandosi su algoritmi che sfruttino la sensoristica avanzata sarebbe stato necessario d'altra parte o porre vincoli più stringenti sul processo produttivo precedente oppure impiegare operai che compiano manualmente un lavoro duro, monotono e in molti contesti pericoloso.

Per quanto concerne la creazione del sistema di visione, il problema del bin picking richiede la soluzione del problema del riconoscimento degli oggetti all'interno della scena. In particolare, nel presente studio verrà posta attenzione sui sistemi basati su modelli, nei quali il riconoscimento consiste nel cercare all'interno dell'immagine acquisita la posa delle istanze di prodotti le cui specifiche geometriche sono fornite in ingresso al sistema. Buona parte dei sistemi di produzione industriali che impiegano robot manipolatori possiedono queste caratteristiche. L'obiettivo dell'algoritmo è quello di trovare la posizione ed orientazione dell'oggetto target nello spazio tridimensionale con una precisione sufficiente per consentire al robot manipolatore di afferrare l'oggetto. Una delle complicazioni date dal bin picking rispetto a sistemi basati su part-feeder e nastro trasportatore è data dal fatto che si possono verificare diverse configurazione degli oggetti presenti nel contenitore: gli oggetti possono toccarsi, possono sovrapporsi e assumere un'orientazione arbitraria rispetto alla telecamera. Queste caratteristiche rendono il problema particolarmente difficile nel caso generale di identificazione di diversi modelli di oggetti di forma arbitraria. Per semplificare l'approccio iniziale, si è scelto di sviluppare in primo luogo un sistema che consenta l'individuazione all'interno del contenitore di un singolo oggetto planare, come può essere ad

esempio una lamiera appena uscita dal processo di taglio. La soluzione proposta sarà quindi estesa per il riconoscimento di oggetti di forma arbitraria. Parallelamente, per entrambi i sotto problemi, verrà valutato l'impatto sia algoritmico che computazionale dato dal considerare il riconoscimento di più tipologie di oggetti contemporaneamente.

Per l'implementazione del sistema si è scelto di approfondire la creazione di un framework specifico che potesse consentire la creazione di sistemi di visione comprensivi di eventuali iterazioni con periferiche esterne. Negli ultimi anni la comunità scientifica della visione computazione ha raggiunto la maturazione proponendo una serie di algoritmi e tecniche standard per la soluzione ai problemi più comuni. In conseguenza sono nate e sono state sviluppate diverse librerie contenenti i più comuni algoritmi di visione, tra le quali le OpenCV è sicuramente la più conosciuta e diffusa in ambito accademico. In contrasto alla maturità di tali strumenti, pochi sforzi sono stati rivolti alla progettazione e all'utilizzo di architetture software che consentano di ottenere benefici quali la riusabilità del codice, la modularità, la scalabilità e l'interoperabilità tra le diverse tecnologie. Mediante la progettazione e lo sviluppo di un framework specifico per le applicazioni di visione, la tesi si propone di fornire un ambiente di sviluppo che consenta di ottenere vantaggi quali la maggiore facilità di debugging e manutenzione del codice, una maggiore semplicità nel riutilizzo dei componenti utilizzati, oltre che a fornire strumenti per la creazione di architetture più complesse che consentano la creazione di sistemi distribuiti, multi-threading e con vincoli real-time.

1.2 Struttura della tesi

La presente tesi è suddivisa in sei capitoli, di cui il primo è costituito da questa introduzione.

Nel capitolo 2 descriveremo formalmente il problema del bin picking introducendo le specifiche del sistema. La trattazione procederà quindi sulla progettazione del sistema per poi descrivere la sua realizzazione.

Nel capitolo 3 verrà descritto il problema del model matching, partendo dalla descrizione gli approcci precedenti disponibili in letteratura. La trattazione procederà descrivendo come l'algoritmo proposto possa risolvere le problematiche degli approcci precedenti, considerando in prima istanza il caso della rilevazione

di un oggetto planare utilizzando un sistema basato su una singola telecamera. In seguito sarà descritto come sia possibile rimuovere tali vincoli, considerando l'utilizzo di un sistema stereo oppure un oggetto di forma arbitraria.

Nel capitolo 4 affronteremo le problematiche relative allo sviluppo di un framework per lo sviluppo di applicazione di visione computazionale. Partendo dalla descrizione di diversi casi d'uso, tra i quali sarà presente anche il sistema di bin-picking, verrà descritto come sia possibile mediante l'utilizzo di un framework ottenere vantaggi come una maggiore facilità di debugging o manutenzione del codice, la semplicità di riutilizzo dei componenti o come sia possibile ottenere semplicemente l'esecuzione parallela, distribuita e con vincoli real-time del sistema sviluppato.

Nel capitolo 5 verranno descritti gli esperimenti effettuati al fine di confrontare l'algoritmo proposto nel capitolo 3 con gli algoritmi attualmente disponibili in letteratura, ponendo particolare enfasi sulla capacità di rilevare l'oggetto all'interno dell'immagine anche in presenza di variazioni sulle condizioni di illuminazione.

Infine, nel capitolo 6 verranno presentate le conclusioni e i possibili sviluppi futuri per il sistema di bin picking.

Capitolo 2

Cella robotizzata per il bin-picking

Nel presente capitolo verrà definito formalmente il problema del bin picking e descritto lo studio condotto per la progettazione e realizzazione del sistema di visione basato su telecamere standard che permetta la sua implementazione in ambito industriale.

Il problema del bin picking sarà prima trattato nella sua accezione più generale nella sezione 2.1, per poi fornire le specifiche del sistema nella sezione 2.2. La studio proseguirà quindi nella progettazione del sistema, la quale sarà affrontata nel dettaglio nella sezione 2.3. Infine, nella sezione 2.4 verrà descritta la sua realizzazione ed integrazione in ambiente industriale.

2.1 Il problema del bin-picking

Il problema del bin-picking consiste nell'identificazione e il successivo prelievo mediante l'utilizzo di robot manipolatore degli oggetti posti alla rinfusa all'interno di un contenitore. Da questa breve descrizione si possono individuare i due componenti fondamentali del sistema: da un lato abbiamo il sistema che si occupa dell'identificazione e localizzazione spaziale degli oggetti contenuti nel contenitore. Le informazioni ottenute vengono quindi utilizzate dal secondo componente del sistema, ovvero dal robot manipolatore, che si occupa del prelievo vero e proprio dell'oggetto dal contenitore.

Le applicazioni del problema del bin-picking sono molte e con specifiche diverse tra loro. Nel fornire qualche esempio, possiamo considerare i problemi di assemblaggio, packaging e piegatura lamiera. Nei sistemi di assemblaggio, spesso l'ingresso del sistema è costituito da contenitori dove i componenti che devono essere assemblati sono disposti alla rinfusa. Questo rappresenta uno dei problemi più complessi di bin-picking, dato che il sistema deve:

- Individuare il pezzo necessario all'interno della scatola, la quale contiene diverse tipologie di componenti. Nel caso generale inoltre questi componenti possono avere dimensione e forma arbitrarie.

- Determinare quale sia il modo migliore per afferrare gli oggetti individuati (problema del *grasping* degli oggetti).

- Definire il percorso che il robot manipolatore deve compiere per afferrare l'oggetto nella modalità determinata. Il percorso deve essere tale da non andare a danneggiare gli altri oggetti contenuti nella scatola.

Il problema dell'identificazione dei pezzi all'interno del contenitore non è ancora stato risolto nella sua accezione più generale, anche se sono stati compiuti negli ultimi anni diversi sforzi a riguardo [14]. La soluzione può essere trovata più semplicemente ponendo vincoli sulla tipologia di oggetti che il sistema può trattare. Ad esempio si può presupporre che gli oggetti contenuti all'interno della scatola sono convessi e quindi non si possono verificare situazioni in cui gli oggetti compenetrino tra loro.

Anche il problema di determinare il modo migliore di afferrare l'oggetto è complesso nel caso più generale, ma può essere semplificando ponendo vincoli sulla struttura dei componenti contenuti nella scatola. Ad esempio, sapendo a priori la forma degli oggetti, ad esempio che gli oggetti contenuti nelle scatole sono tutti dei bulloni, è possibile determinare a priori, mediante la specifica da parte di un operatore, quali siano le modalità di presa possibili. Questo approccio, per quanto semplice, può essere utilizzato in diversi processi industriali reali. Considerando invece la soluzione al problema del grasping di oggetti di forma arbitraria, è necessario ricorrere ad algoritmi più complessi (si veda [13]). Ad esempio, in [50] e [39] vengono utilizzati degli algoritmi di apprendimento supervisionati per la soluzione del problema, mentre in [54] vengono predetti i punti

di presa come funzione dell'immagine. Questi algoritmi non sono d'altra parte ancora sufficientemente stabili per poter essere utilizzati in un sistema industriale.

Infine, il problema della determinazione del percorso che il robot manipolatore deve compiere per prelevare l'oggetto individuato nella modalità determinata, ovvero il problema del *path planning* o *trajectory planning*, è d'altra parte un problema affrontato in generale nella programmazione dei robot manipolatori, per il quale esistono in letteratura diversi algoritmi stabili ed efficienti (si veda [11]). Come unica complicazione bisogna d'altro canto osservare che, nel caso generale, per la soluzione del problema del path planning è necessario avere a disposizione la ricostruzione tridimensionale dell'intero contenitore, in modo da evitare che il robot vada ad urtare gli altri oggetti contenuti nella scatola. Questo vincolo può essere rilassato per la soluzione di alcuni problemi specifici.

Nei sistemi di packaging, l'obiettivo è quello di riordinare all'interno di una scatola i prodotti prelevati dal sistema di bin-picking. Per questi sistemi, nella maggior parte dei casi, gli oggetti contenuti all'interno della scatola in ingresso sono solo di una tipologia. Questo pone una semplificazione nell'individuazione del prodotti. Inoltre, grazie al fatto che gli oggetti sono equivalenti tra loro, è possibile scegliere più liberamente l'oggetto da prelevare tra quelli presenti, ad esempio scegliendo l'oggetto sovrastante il mucchio. In questo modo, anche le successive fasi di grasping e planning risultano, in diversi gradi, semplificate.

Considerando infine i sistemi di piegatura lamiera, un'ulteriore semplificazione viene data dal fatto che gli oggetti considerati sono planari. Questo riduce il problema di identificazione da un problema nello spazio tridimensionale ad un problema nello spazio bidimensionale. Inoltre, la presa del pezzo avviene solitamente utilizzando una pinza a ventose. Questa configurazione riduce il complesso problema di grasping nello spazio tridimensionale al più semplice problema di ricerca di una superfice bidimensionale dove sia possibile appoggiare le ventose. Come nel caso del packaging, gli oggetti sono di una sola tipologia e quindi è possibile scegliere l'oggetto da prelevare tra quelli presenti. Infine, dato che gli oggetti planari hanno la tendenza ad orientarsi in modo parallelo al piano del contenitore e dato che il robot preleva i pezzi provenendo da una direzione normale a tale piano, ci sono minori probabilità che il percorso di prelievo calcolato dall'algoritmo di planning causi lo scontro tra il robot e gli altri oggetti presenti nel cassone. Questa configurazione semplifica la complessità degli algoritmi di

planning necessari alla soluzione del problema.

Dalla trattazione degli esempi, risulta chiaro che il problema del bin-picking per la piegatura della lamiera pone notevoli semplificazioni rispetto al problema del bin-picking nella sua accezione più generale. Nonostante questo, considerando il solo sistema di visione, esso contiene tutti gli elementi fondamentali comuni agli altri sistemi. Per questo motivo, nella presente tesi verrà considerato in primo luogo il caso della piegatura della lamiera. Si noti inoltre che, per quanto esposto, lo stesso sistema può essere utilizzato per celle di lavoro che compiano operazioni differenti della piegatura lamiera, come potrebbero essere celle di saldatura o verniciatura, fermo restando il vincolo di planarità degli oggetti da prelevare dal contenitore. Nel corso della trattazione saranno considerate le eventuali estensioni necessarie per poter rilassare i vincoli precedentemente fissati. In particolare, il sistema trovato verrà esteso al caso di oggetti di forma arbitraria, per i quali il vincolo di planarità non potrà più essere garantito. Sia nel caso planare che in quello tridimensionale verrà inoltre considerato l'impatto al sistema dato dal rilassamento del vincolo per cui solo una tipologia di oggetti può essere contenuto all'interno della scena. Questo apre il sistema alla soluzione del più generale problema del bin-picking, come quello dell'assemblaggio di prodotti descritto in precedenza. Le estensioni verranno d'altro canto considerate solo per quanto riguarda gli algoritmi per l'identificazione degli oggetti nella scena. Le problematiche relative al grasping di tali oggetti o gli algoritmi di planning necessari alla generazione delle traiettorie non saranno considerati oltre nella presente tesi.

2.2 Specifiche del sistema

In questa sezione verranno descritte le specifiche del sistema di bin-picking per il quale sono stati sviluppati gli algoritmi descritti all'interno della presente tesi. In primo luogo verranno descritti i componenti principali di una tipica cella di piegatura nella sezione 2.2.1, per poi passare nella sezione 2.2.2 ad una raccolta ed analisi dei requisiti che un sistema di bin picking integrato nella cella descritta deve possedere. In seguito, nella sezione 2.2.3, verranno definiti formalmente i parametri che formeranno le basi per il dimensionamento dei componenti del sistema durante la sua progettazione. Infine, nella sezione 2.2.4 verrà presentato un caso concreto di utilizzo del sistema di bin-picking.

2.2.1 La cella di lavoro

I componenti principali presenti in un impianto di piegatura lamiera sono:

- **Caricatore**: si occupa di rendere disponibile i prodotti in un punto determinato affinché il robot possa raggiungerli.
- **Allineatore**: è un piano inclinato che consente di portare il prodotto in una posizione predeterminata con grande accuratezza e ripetibilità.
- **Pressopiegatrice**: macchinario che effettua la piegatura della lamiera. E' composta da un banco fisso e da una parte in movimento verticale, detta pestone. La pressa viene attrezzata in più stazioni di piega, utilizzando matrice e coltelli di diverse forme che garantiscono l'esecuzione dei diversi tipi di pieghe.
- **Robot Manipolatore**: componente centrale della cella, esegue la sequenza programmata di movimenti interagendo con gli altri componenti del sistema.
- **Gripper**: organo di presa montato sul TCP (Tool Center Point) del robot manipolatore. Permette la presa del prodotto.
- **Pallet di scarico**: superfice dove viene posto il prodotto piegato.

I passi eseguiti durante il processo di piega sono:

- **Caricamento**: il prodotto da piegare viene inserito all'interno della cella di lavoro. Per questo passo ci sono diverse alternative:
  - Il prodotto viene posizionato da un operatore in una posizione predeterminata. Questa può essere nel caricatore oppure direttamente nell'allineatore.
  - I prodotti sono già impilati nel caricatore o direttamente nell'allineatore. Anche in questo caso è richiesto l'intervento umano, ma, a differenza del caso precedente, l'intervento è richiesto solo ogni lotto di prodotti.
  - I prodotti sono caricati da un altro robot che si occupa di inserirli all'interno della cella di lavoro. Questa opzione viene utilizzata nel

caso si disponga di un sistema automatico di scarico nel processo di taglio laser. In questo caso non è richiesto l'intervento umano, ma è necessario accoppiare strettamente la fase di taglio a quella di piega.

- I prodotti sono presenti in contenitori e viene utilizzato il robot manipolatore per effettuare lo carico. Questo è il caso di interesse, in quanto per l'esecuzione del passo è necessario il sistema di bin-picking oggetto della presente tesi. Si noti che questa soluzione è la più flessibile in quanto non richiede l'intervento umano e non pone stretti vincoli di accoppiamento con la fase precedente di taglio. D'altro canto è la soluzione di più complessa realizzazione, in quanto richiede l'introduzione di sensoristica avanzata all'interno della cella di lavoro, altrimenti non necessaria.

- **Esecuzione delle pieghe**: dopo che il robot ha prelevato il prodotto in posizione ripetibile ed accurata dall'allineatore è possibile eseguire i diversi passi di piega. Per ogni piega si hanno i seguenti passi:

  - Selezione attrezzaggio macchina. Il prodotto viene portato dal robot alla stazione della pressopiegatrice in grado di eseguire la piega con le caratteristiche richieste. Si noti che le stazioni di piega sono state precedentemente configurate in base alle caratteristiche del prodotto.
  - Posizionamento in pressa. Il pezzo viene portato nella posizione di piega dal robot manipolatore. Durante questo passo vengono utilizzati particolari parti della pressa con uno o più gradi di libertà, detti riscontri, che permettono di allineare il pezzo nel punto desiderato con una maggiore accuratezza di quella fornita dal robot manipolatore.
  - Esecuzione della piega. La pressopiegatrice si occupa dell'esecuzione della piega, muovendo il pestone verso il basso. In questo passo il robot può essere inattivo oppure accompagnare il movimento del prodotto al fine di limitare le flessioni del materiale.
  - Ripresa del pezzo. Infine, il robot preleva il prodotto piegato per poi passare alla piega successiva.

- **Pallettizzazione**: il passo di pallettizzazione consiste nel posizione il prodotto finito su dei pallet seguendo un layout di deposito predeterminato. I

prodotti così depositati possono quindi essere soggetti alla successiva fase di lavorazione. Si noti come il passo di pallettizzazione deve essere eseguito nelle modalità richieste dalla fase di lavorazione successiva, così come la fase di caricamento è legata al passo precedente. Per questa ragione, nel qual caso la fase di lavorazione successiva utilizzi un sistema di bin-picking, non è necessario ordinare con particolari layout i pezzi nel pallet, ma sarebbe possibile disporre casualmente gli oggetti all'interno di un contenitore di scarico. Anche questo è un caso in cui un sistema di bin-picking, o più in generale un sistema che adoperi sensoristica avanzata all'interno delle fasi di lavorazione, può aggiungere flessibilità all'interno dell'intero processo produttivo

2.2.2 Sistema di bin-picking

Come abbiamo visto, il sistema di bin-picking si inserisce nel passo di caricamento della cella di piegatura descritta nella sezione precedente. Dato il contesto considerato, possiamo definire i seguenti requisiti di alto livello desiderati per il sistema:

- **Utilizzo del robot manipolatore presente nella cella**. Dato che la cella di lavoro utilizza già un robot manipolatore, per non aumentare ulteriormente i costi dell'impianto risulta necessario impiegare lo stesso robot anche per il prelievo dei prodotti dal contenitore. Questa scelta pone dei vincoli sul posizionamento e l'ingombro che il sistema di bin picking deve avere. Difatti, da un lato il contenitore deve essere raggiungibile nel raggio d'azione del robot manipolatore, dall'altro il sistema di bin picking non deve ostruire i movimenti che il robot deve compiere. Si noti che questo requisito fa si che, a meno di accorgimenti particolari, il sistema di bin-picking può essere utilizzato solo per celle di lavoro che trattano prodotti di dimensioni medio-piccole.

- **Mascheramento tempo di elaborazione**: il tempo ciclo, ovvero il tempo necessario all'impianto per l'esecuzione di un ciclo di lavorazione su un singolo prodotto, è uno dei parametri fondamentali per la valutazione dell'efficienza di una cella di lavorazione. Il tempo ciclo governa la produttività economica di una determinata cella di lavoro, in quanto fornisce un indi-

cazione sul numero di prodotti che essa è in grado di gestire per unità di tempo. Per questo motivo è ritenuto un parametro di massima importanza nell'ambito industriale, dove ridurre il tempo ciclo indica spesso la selezione di una soluzione piuttosto che un altra. Data questa premessa, è auspicabile che il robot non si debba fermare durante il passo di caricamento. Questo può essere ottenuto mascherando il tempo di elaborazione degli algoritmi di identificazione del prodotto all'interno del contenitore, ovvero eseguendo l'elaborazione mentre il robot manipolatore è impegnato in altre lavorazioni, nel caso specifico piegare il pezzo prelevato precedentemente.

- **Continuità di operazione**: il sistema di bin-picking deve essere tollerante ai cambiamenti delle ambiente in cui opera. Questo requisito è particolarmente importante nel caso si utilizzino sistemi di visione. I rischi maggiori sono di due tipi. Da un lato le polveri presenti negli ambienti industriali possono andare a ridurre la capacità emissiva degli illuminatori così come le prestazioni delle lenti della telecamera. Inoltre, se sono presenti situazioni particolarmente severe, è possibile che l'accumulo di polveri possa diminuire il contrasto tra gli oggetti posizionati nel cassone. In secondo luogo, come già detto, vi sono dei vincoli sulla creazione del sistema di bin-picking. In particolare, si potrebbe avere un sistema dove l'illuminazione non è totalmente controllata. In questi casi fonti di luce esterne devono essere considerate all'interno degli algoritmi di elaborazione delle immagini.
  E' necessario inoltre considerare che la calibrazione del sistema può avere un lento movimento di deriva, causato ad esempio dalle vibrazioni a cui è inevitabilmente soggetto il sistema in un ambiente industriale, che può peggiorare la qualità complessiva del sistema. Per questo motivo devono essere previsti opportuni accorgimenti per limitare questo problema. In ogni caso è necessario prevedere le opportune procedure di correzione della calibrazione del sistema da eseguirsi in automatico o mediante la supervisione di un operatore.
  Infine, gli strumenti hardware utilizzati, in primo luogo gli illuminatori ma anche le telecamere, hanno una variazione delle prestazioni nel tempo. Ad esempio, un illuminatore a neon riduce nel tempo la sua emissione luminosa. Nonostante questo, il sistema deve continuare ad operare per un lasso di tempo il più lungo possibile. La lunghezza di vita dei componenti deter-

mina difatti direttamente il costo di manutenzione dell'impianto, il quale deve essere il più basso possibile.
Tutti questi fattori dovranno essere tenuti opportunamente in considerazione in particolare nell'implementazione degli algoritmi di visione. Gli algoritmi sviluppati dovranno essere quindi il più possibile tolleranti alle variazioni dell'ambiente e delle caratteristiche dei dispositivi utilizzati.

- **Costo**: seppur rispettando tutti i requisiti elencati precedentemente, il costo del sistema finale deve essere il più ridotto possibile. Si consideri che questo introduce un trade-off tra le scelte architetturali del sistema e la sua economicità, in particolare riguardo alla continuità di operazione. La scelta di strumenti hardware a costo più basso potrebbe rendere il sistema meno durevole, e quindi richiedere maggiori costi di manutenzione. Nella definizione del costo non si deve quindi considerare solo il costo iniziale, ma anche i costi di manutenzione successivi.

In aggiunta a queste specifiche di alto livello, che tutti i sistemi di bin-picking da integrare alle celle di piegatura lamiera devono avere, sono presenti altre specifiche che dipendono dal caso concreto di utilizzo. Fortunatamente, come vedremo, le variazioni di quest'ultime spesso inducono solo ad una variazione delle caratteristiche dell'hardware scelto, mentre sia la struttura del sistema che gli algoritmi di visione utilizzati rimangono al più inalterati.

2.2.3 Parametri del sistema

Le specifiche del sistema dipendono dalle caratteristiche dei prodotti che devono essere prelevati dal cassone e delle caratteristiche della cella utilizzata. Si possono individuare i seguenti fattori che influenzano le specifiche del sistema:

- **Dimensione del contenitore**: la dimensione del contenitore influenza il campo visivo del sensore utilizzato. A partità di risoluzione spaziale del sensore, più grande il campo visivo minore è la precisione assoluta di misura del sensore.

- **Materiale dei prodotti**: il materiale del prodotto, più precisamente le sue caratteristiche ottiche, influenzano la scelta del sistema di illuminazione da utilizzare.

- **Tolleranza piano X-Y**: la tolleranza sul piano X-Y indica la precisione con la quale deve essere identificato l'oggetto. Solitamente, questo valore è condizionato dalla minima distanza attesa tra l'organo di presa e i bordi o fori dell'oggetto da prelevare. Scegliendo un valore opportunamente maggiore si garantisce che eventuali imprecisioni nell'individuazione della posizione dell'oggetto non influiscano sulla presa dello stesso. Questo valore dipende quindi dalle caratteristiche geometriche dei prodotti che si desidera prelevare.

- **Tolleranza nell'asse Z**: la tolleranza sull'asse Z indica la precisione con la quale si vuole determinare la profondità dell'oggetto all'interno del contenitore. Questo valore influenza indirettamente il tempo ciclo. Per capire questo, è necessario innanzitutto introdurre brevemente come il robot manipolatore si comporta durante la presa di un pezzo. Quando al robot viene fornito il comando di presa del pezzo nella posizione indicata, il robot si muove velocemente fino a raggiungere una posizione appena sopra quella fornita. Quindi procede verso il prodotto utilizzando la funzionalità detta di ricerca. Durante la ricerca, il movimento non è più governato dalle posizioni fornite, ma il robot procede nella direzione indicata fino a quando non incontra degli ostacoli, rilavati ad esempio tramite sensori di pressione. L'ostacolo ricercato è in questo caso il prodotto da prelevare. La distanza a cui il robot si deve posizionare prima di iniziare la funzionalità di ricerca dipende dalla tolleranza di misura nell'asse Z. Tanto più è grande la tolleranza, maggiore è la distanza iniziale per la ricerca. Per questa ragione aumentare la tolleranza significa indirettamente aumentare il tempo ciclo.

- **Tolleranza nell'inclinazione**: la tolleranza nell'inclinazione indica con quale precisione si desidera ottenere l'inclinazione del prodotto rispetto al piano X-Y del contenitore. Solitamente questo valore non viene fornito in gradi, ma è calcolato dalla differenza in altezza tra il punto del prodotto con maggiore profondità e il punto con minore profondità. Questa definizione deriva dalle caratteristiche dell'organo di presa utilizzato. Particolari organi di presa sono difatti in grado di esercitare forza attrattiva sul prodotto ancora prima che il piano di presa raggiunga effettivamente il prodotto. Ci sono essenzialmente due tipologie di organi di presa con queste caratteri-

stiche: le *ventose a risucchio* e i magneti. Per vedere come questo influenzi la tolleranza nell'inclinazione, consideriamo per semplicità di trattazione il caso più in cui vi siano solo due punti di presa per l'oggetto. Supponiamo inoltre che il sistema di bin-picking abbia determinato che l'oggetto si trova parallelo al piano X-Y. La trattazione può essere semplicemente estesa a casi più complessi. L'organo di presa entra in gioco nella fase di ricerca descritta precedentemente. Dato che il prodotto è supposto parallelo al piano X-Y, anche il robot si posizionerà con l'organo di presa parallelo a tale piano per la ricerca. Quando l'organo di presa raggiunge il primo punto di presa nel prodotto la ricerca termina. Perché il prodotto possa essere afferrato correttamente, l'altro punto di presa deve essere ad una distanza data dalla tolleranza specificata dall'organo di presa. Quindi la precisione di determinazione dell'inclinazione deve essere almeno inferiore alla differenza in altezza tra i due punti di presa che l'organo di presa può tollerare. Senza porre nessun vincolo sulla posizione dei punti di presa, questa deve essere nel caso pessimo definita come la differenza in altezza tra il punto del prodotto con maggiore profondità nell'asse Z e il punto con minore profondità. Si noti come il parametro di tolleranza nell'inclinazione sia legato sia alla tolleranza dell'organo di presa sia alla dimensione massima del prodotto.

- **Inclinazione massima prodotto**: sebbene sarebbe teoricamente possibile individuare i prodotti in qualunque inclinazione rispetto al piano X-Y del contenitore, vincolando l'inclinazione massima attesa per il prodotto si ottengono diversi benefici. Da un lato, per il sistema di visione si ottiene una deformazione prospettica meno accentuata. Dall'altro, inclinando meno il polso del robot manipolatore, e quindi provenendo in direzione parallela all'asse Z, si diminuiscono le probabilità di urtare altri oggetti contenuti nel contenitore, semplificando gli algoritmi di grasping e path planning. Infine, come vedremo, all'aumentare dell'inclinazione diminuiscono le tolleranze nel piano X-Y, nell'asse Z e nell'inclinazione, richiedendo quindi un sensore di risoluzione più elevata.

Dalla discussione precedente possiamo ora dedurre i parametri del sistema che formeranno le basi quantitative per la selezione dell'hardware da utilizzare. Come parametri di ingresso abbiamo:

- Dimensione contenitore ($C_x \times C_y \times C_z$): rispettivamente lunghezza, larghezza e profondità del contenitore.
- Dimensione massima prodotto ($P_x^{max} \times P_y^{max}$): dimensioni massime per il prodotto.
- Dimensione minima prodotto ($P_x^{min} \times P_y^{min}$): dimensioni minime per il prodotto.
- Tolleranza di presa (T_{grip}): distanza minima tra il contorno della superficie di presa e il bordo o un foro del prodotto.
- Tolleranza organo di presa (T_{pump}): distanza minima tra il prodotto e l'organo di presa per cui il prodotto può essere considerato afferrato. Si suppone che la tolleranza dell'organo di presa non dipenda dall'inclinazione dello stesso.
- Tolleranza di ricerca (T_{search}): distanza tra il punto atteso dell'oggetto e il punto di inizio della ricerca del prodotto.
- Inclinazione massima (θ_{max}): inclinazione massima del prodotto rispetto al piano X-Y.

Dal valore di questi parametri possiamo ora calcolare i parametri interni del sistema:

- Distanza massima tra due punti di presa (G_{max}): la distanza massima tra due punti di presa è data da:
  $G_{max} = \sqrt{(P_x^{max})^2 + (P_y^{max})^2}$
- Tolleranza asse Z (T_z): la tolleranza nell'asse Z determina la precisione con cui si vuole ottenere le profondità media dell'oggetto. Essa è strettamente collegata alla tolleranza di ricerca mediante la relazione: $T_z = T_{search} \cdot sin(\theta_{max})$
- Tolleranza all'inclinazione (T_{incl}): la tolleranza all'inclinazione è data dalla tolleranza dell'organo di presa. Per ottenere il corrispondente valore in gradi di inclinazione abbiamo:
  $T_{incl} = arcsin(\frac{2 \cdot T_{pump}}{G_{max}})$

- Tolleranza piano X-Y (T_{xy}): la tolleranza nel piano X-Y viene calcolata in base alla tolleranza di presa nel caso pessimo in cui il prodotto sia posto, rispetto al piano calcola, con un inclinazione pari a T_{incl}. Abbiamo (si veda 2.1):
  $T_{xy} = T_{grip} \cdot cos(\theta_{max})$

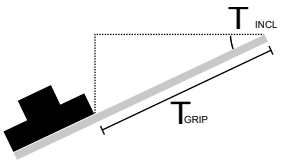
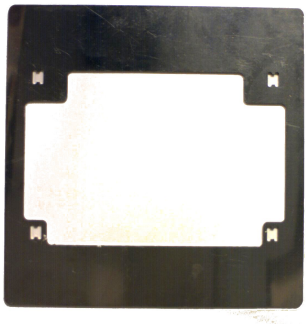
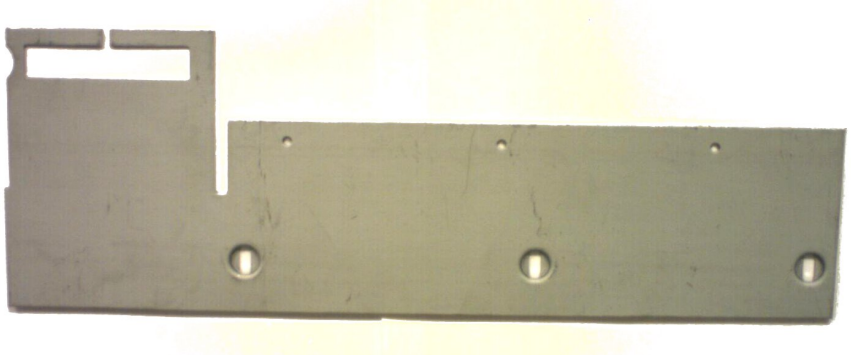


Figura 2.1: Calcolo tolleranza di presa.

2.2.4 Caso concreto di utilizzo



(a)



(b)

Figura 2.2: Due dei prodotti presi come campione per gli esperimenti.

Come caso concreto di utilizzo del sistema proposto si è considerato il processo di piegatura di prodotti destinati alla creazione di elettrodomestici per cucina. Il caso in esame riserva delle particolari complessita per due ragioni fondamentali: in primo luogo, la dimensione degli oggetti è molto variabile, passando da oggetti di dimensione 10 x 10 cm, ad oggetti di dimensione 40 x 20 cm. Questa situazione fa si che da un lato aumenti la dimensione massima del prodotto, e dall'altro diminuisca la tolleranza di presa nei prodotti più piccoli. In secondo luogo, la tipologia di materiale di alcuni dei prodotti ha causato non poche difficoltà nella scelta del sistema di illuminazione. Alcuni prodotti sono difatti di acciaio Inox, materiale particolarmente riflettente. Nelle figure 2.2 sono riportati due dei prodotti presi come campione tra i 39 prodotti forniti come specifica da parte del partner industriale del progetto di ricerca.

I parametri che definiscono le specifiche del sistema, così come definiti nella sezione 2.2.3, sono riportante nella tabella 2.1.

Tabella 2.1: Specifiche del sistema.

Parametro	**Valore**
Dimensione contenitore	60 x 40 x 30 cm
Dimensione massima prodotto	40 x 20 cm
Dimensione minima prodotto	10 x 10 cm
Tolleranza di presa	1 cm
Tolleranza organo di presa	1 cm
Tolleranza di ricerca	3 cm
Inclinazione massima	30°
Distanza massima tra due punti di presa	44.72 cm
Tolleranza asse Z	1.5 cm
Tolleranza all'inclinazione	2.5°
Tolleranza piano X-Y	1 cm

2.3 Progettazione del sistema di bin picking

Per la progettazione del sistema sono state prese in considerazione diverse configurazioni. In primo luogo è stata scelta la tipologia di sensore da utilizzare all'interno del sistema. Le alternative sono l'utilizzo di telecamere 3D basate su luce strutturata oppure un sistema di visione basato su telecamere standard. I sistemi di bin picking basati su luce strutturata hanno dimostrato una maggior resistenza alle particolari condizioni che si trovano normalmente in ambiente industriale. In particolare essi sono meno sensibili alle variazioni di illuminazione dell'ambiente. Forte di questi vantaggi, buona parte dei primi sistemi di bin picking presenti in commercio sono basati proprio su luce strutturata. D'altra parte, il costo di tali sistemi è di un ordine di grandezza superiore al costo di un sistema basato su telecamere standard. Considerando il requisito di ridurre i costi di sistema (si veda 2.2.2), si è optato per l'utilizzo di sistemi basati sul telecamere standard. La trattazione dettagliata della scelta è riportata nella sezione 2.3.1.
Deciso il sensore, si apre la scelta del posizionamento delle telecamere rispetto agli altri componenti del sistema. La telecamera può essere agganciata al TCP del robot manipolatore oppure essere posizionata in una posizione prefissata. Posizionando la telecamera nel TCP del robot si ha il vantaggio di avere maggiore flessibilità nella selezione del punto di vista per catturare le immagini. Se opportunamente sfruttato, questo vantaggio può tradursi in una minore risoluzione richiesta per la telecamera. D'altra parte, ponendo la telecamera nel robot manipolatore non è più possibile mascherare il tempo di elaborazione, in quanto il robot partecipa attivamente alla fase di determinazione della posizione del prodotto nel contenitore. Inoltre, il posizionamento della telecamera vincola ed è vincolata dalla geometria del gripper utilizzato, limitando in questo modo le possibilità di progettazione di quest'ultimo. Infine, la precisione di determinazione della posizione del prodotto rispetto al robot manipolatore è afflitta da un peggioramento dato dalla ripetibilità di posizionamento del robot (si noti che l'errore non dipende invece dall'accuratezza di posizionamento del robot manipolatore, in quanto noi siamo interessati solo alla posizione relativa tra prodotto e robot, non alla posizione assoluta del prodotto). Queste problematiche hanno fatto propendere per un posizionamento fisso della telecamera.
Una scelta ortogonale al posizionamento della telecamera è data dalla scelta della risoluzione del sensore da utilizzare e dal numero di telecamera da utilizzare

all'interno del sistema. Come vedremo nella sezione 2.3.2, l'utilizzo di soluzioni con due o più telecamere permette di ottenere delle precisioni migliori per quanto riguarda il calcolo della profondità degli oggetti senza dover aumentare eccessivamente la risoluzione di ogni singolo sensore. D'altra parte l'utilizzo di più telecamere è maggiormente costoso, sia in termini di hardware che in termini di tempo di elaborazione per gli algoritmi.
Infine, è necessario determinare il migliore sistema di illuminazione che permetta da un lato di aumentare il più possibile il contrasto delle immagini prodotte, garantendo in questo modo di migliorare le prestazioni degli algoritmi di elaborazione delle immagini, dall'altro di essere il più resistente possibile alla variazione di luminosità nell'ambiente di lavoro. Per raggiungere questi obiettivi sono state previste due ipotesi. Da un lato si ha l'utilizzo di illuminatori fissi, dall'altro l'utilizzo di un sistema meccanico che permetta la copertura del contenitore, e quindi l'isolamento del ambiente circostante, durante l'acquisizione delle immagini. Quest'ultima ipotesi permette sicuramente di ottenere un sistema più robusto alle variazioni di illuminazione, ma introduce un nuovo costo dato dal sistema meccanico di isolamento.
Nelle sottosezioni che seguono tratteremo nel dettaglio le scelte progettuali che hanno condotto alla realizzazione del sistema di bin picking.

2.3.1 Scelta della tipologia di sensore

Per la scelta della tipologia di sensore da utilizzare all'interno del sistema sono state vagliate due ipotesi: l'utilizzo di telecamere 3D basate su luce strutturata oppure l'utilizzo di telecamere standard. Nel seguito della sezione analizzeremo nel dettaglio le caratteristiche di entrambi i tipi di sistemi. La trattazione porrà quindi le basi per la successiva valutazione comparativa tra i due sistemi.

Telecamere 3D con luce strutturata

La tecnica più utilizzata nel campo dei sistemi di visione per effettuare misure della scena senza prevedere il contatto è basato sull'utilizzo di telecamere che incorporino una sorgente di luce strutturata per il loro funzionamento. Il termine luce strutturata è definita come la proiezione di un particolare pattern per illuminare gli oggetti all'interno della scena. E' possibile utilizzare come pattern un punto, una linea, una griglia, oppure una forma più complessa, in base ai requisi-

ti del problema specifico che desideriamo risolvere. Il dispositivo per l'emissione della luce può essere un LCD (Liquid Crystal Device) oppure un diodo laser. L'utilizzo di emettitori LCD ha inizio negli anni '80. Questi sistemi sono diventati molto popolari soprattutto in ambito accademico per il loro costo ridotto [52] e le meno stringenti misure di sicurezza necessarie per il loro utilizzo rispetto ai dispositivi basati su laser [51]. D'altro canto, l'utilizzo di proiettori basati su diodo laser è sicuramente la scelta migliore per applicazioni industriali. I sistemi laser difatti possono essere creati di dimensioni più ridotte e avere più potenza di illuminazione, riducendo in questo modo l'effetto delle variazioni nell'illuminazione ambientale. Inoltre, i fasci di luce laser hanno una banda limitata. E' quindi possibile utilizzare dei filtri passa banda [29] in modo da ridurre ulteriormente le influenze date dall'ambiente esterno. Il pattern di luce proiettato nella scena viene quindi letto da un normale sensore di visione (CCD o CMOS). In base alle deformazioni che il pattern subisce è possibile inferire le caratteristiche tridimensionali dell'oggetto osservato.Questi sistemi vengono chiamati il letteratura *active vision system*.

In figura 2.3 è riportato lo schema concettuale di un sistema con luce strutturata che utilizza una singolo piano come pattern di illuminazione. L'illuminatore posto in alto si occupa di proiettare un singolo fascio di luce nella scena. La telecamera, posizionata a sinistra in figura, si occupa di rilevare la posizione nella scena del fascio luminoso. Come output del sistema abbiamo una linea spezzata che codifica le informazioni di altezza degli oggetti nella scena.

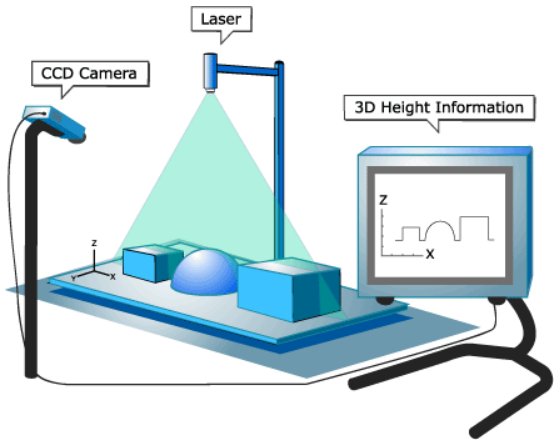


Figura 2.3: Schematizzazione di un sistema di visione basato su luce strutturata.

Al fine di ricavare la profondità dell'oggetto rispetto alla telecamera, è necessario utilizzare il processo di triangolazione. La triangolazione si basa sulla trigonometria piana dove un lato del triangolo e i suoi angoli sono noti ed è necessario determinare la lunghezza degli altri lati. Nel trattare la valutazione della profondità tramite triangolazione consideriamo il sistema semplificato di figura 2.4, dove si ha una sorgente di luce laser che illumina un singolo punto dell'oggetto. La trattazione può essere facilmente estesa al caso in cui viene illuminato un'intera linea dell'oggetto, come nel caso in esame. I seguenti parametri definiscono le specifiche del sistema:

- α descrive l'angolo formato tra l'emettitore laser e la telecamera.
- b, detto *baseline*, descrive la distanza tra la telecamera e l'emettitore laser.
- β può essere calcolato in base alla posizione del raggio laser percepito dalla telecamera. Si ha che $\beta = arctan(\frac{f}{x})$, dove f è la lunghezza focale della telecamera e x è la distanza tra il punto nell'immagine e il centro della stessa.

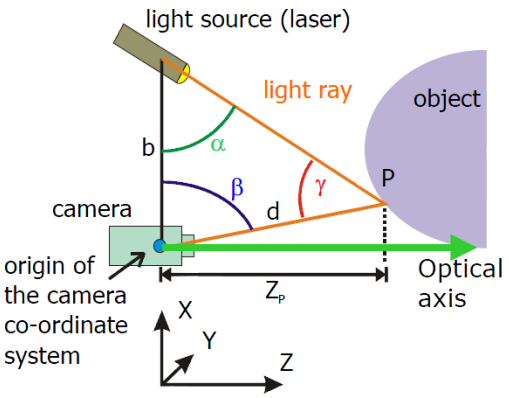


Figura 2.4: Calcolo della profondità tramite triangolazione.

Volendo determinare d, ovvero la distanza tra telecamera e l'oggetto, siamo nella condizione in cui conosciamo due angoli e un lato di un triangolo, e vogliamo determinare un altro lato. Innanzitutto si osservi che, utilizzando le regole del seno e del coseno abbiamo:

$$sin\gamma = \frac{h}{d}; \qquad sin\alpha = \frac{h}{b} \tag{2.1}$$

Da cui otteniamo:

$$\frac{d}{sin\alpha} = \frac{b}{sin\gamma} \tag{2.2}$$

Essendo α, β e γ angoli di un triangoli, essi sommano a 180°:

$$sin(\alpha + \beta) = sin(\pi - \gamma) = -sin(\gamma) \tag{2.3}$$

Unendo le due equazioni precedenti otteniamo infine:

$$d = \frac{b \cdot sin\alpha}{sin\gamma} = \frac{b \cdot sin\alpha}{sin(\alpha + \beta)} \tag{2.4}$$

Per la creazione della rappresentazione tridimensionale dell'intero oggetto è necessario che o l'oggetto si muova rispetto al sensore (ad esempio utilizzando un nastro trasportatore) oppure che il sensore si muova rispetto all'oggetto. Nel caso in esame consideriamo quest'ultima situazione. Il movimento può avvenire con due modalità. Da un lato è possibile muove il sensore parallelamente al piano dell'oggetto, oppure ruotare il sensore sul proprio asse. La prima soluzione è preferibile, in quanto permette di avere il piano contenitore sempre alla stessa distanza dal sensore. Si noti che questa soluzione richiede l'utilizzo di un sistema meccanico composto da un asse traslativo che consenta di muovere la telecamera. La presenza di questo sistema meccanico può peggiorare la precisione del sistema, in quanto è necessario aggiungere all'errore di misurazione intrinseco del sensore l'errore dato dal posizionamento dello stesso.

Figura 2.5: Sick IVP Ruler E.

Sono presenti in commercio diversi dispositivi che si basano sull'utilizzo di luce strutturata per la determinazione del profilo 3D degli oggetti. La telecamera 3D *Ruler E* (si veda 2.5) della Sick IVP è sicuramente una delle periferiche più conosciute e diffuse. Il sensore *Ruler E* è disponibile in tre versioni, le quali differiscono per la dimensione del campo visivo e risoluzione. Il campo visivo è

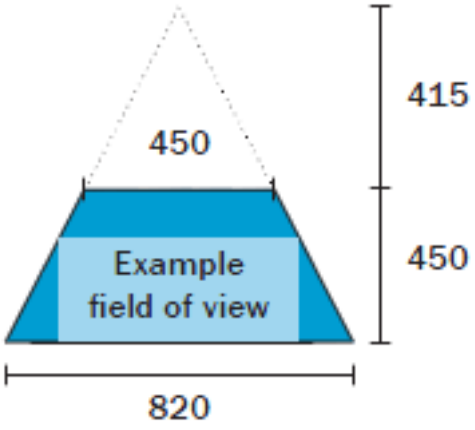


Figura 2.6: Campo visivo del sistema con luce strutturata.

di forma trapezoidale, come riportato in figure 2.6. Nella scelta della versione da utilizzare è necessario tenere in considerazione che il contenitore utilizzato deve rientrare all'interno del campo visivo. Ad esempio con il *Ruler E600* siamo in grado di inquadrare completamente il contenitore utilizzato in questo lavoro, il quale ha lunghezza 40 cm e altezza 30 cm. Con tale sensore è possibile ottenere un risoluzione in altezza di 0.2 mm, la quale soddisfa pienamente i requisiti richiesti dal sistema. Come già detto, è necessario creare una slitta che muova il sensore nella direzione della lunghezza del contenitore. Fortunatamente, la telecamera *Ruler E* dispone di un ingresso dedicato che gli consente di leggere direttamente i dati forniti dal motore utilizzato nella slitta per fornire le misure in millimetri. Per il dimensionamento della slitta è necessario valutare la risoluzione nel piano X-Y che si desidera ottenere. Avendo nel nostro caso la una tolleranza di 10 mm, è necessario che il sensore effettui una misura almeno ogni 5 mm. Considerando che una velocità eccessiva può portare ad un incremento degli errori nel calcolo della posizione dell'oggetto, da valutazioni empiriche date da sistemi che utilizzano tale sensore si può stimare che il tempo necessario per una scansione sia nell'intervallo tra i due e tre secondi. A questo va aggiunto il tempo di elaborazione delle informazioni in modo da ottenere la posizione degli oggetti.

Per maggior informazioni sul prodotto *Sick IVP Ruler E* si rimanda a [6].

Sistema di telecamere

In alternativa all'utilizzo di sensori basati su luce strutturata è possibile utilizzare telecamera standard per la determinazione della posizione tridimensionale dell'oggetto.

Per descrivere geometricamente come la scena ripresa viene percepita della telecamera, è possibile ricorre al semplice modello di telecamera a *pin-hole*. In questo modello, la luce proveniente dalla scena passa attraverso un singolo punto a proietta un'immagine invertita sul matrice del sensore (si veda 2.7). Nel modello di telecamera a pin-hole, il punto di apertura è anche l'origine O del sistema di riferimento tridimensionale della telecamera, dove il piano X-Y è parallelo al piano del sensore e l'asse Z ha direzione dell'asse ottico (ovvero la direzione verso cui la telecamera sta guardando). Questo sistema di coordinate è chiamato sistema di coordinate standard della telecamera, e la sua origine è detta punto focale o più semplicemente centro della telecamera (si veda figura 2.8). Il piano dell'immagine è parallelo al piano X-Y del sistema di riferimento della telecamera ed è ad una distanza f (detta lunghezza focale della telecamera) nella direzione negativa dell'asse Z. L'origine del sistema di coordinate del piano dell'immagine è detto punto principale. Si noti che il piano dell'immagine ha assi X e Y di direzione opposta rispetto al sistema di riferimento della telecamera. Un punto nella scena $\mathbf{P}$ con coordinate (X_P, Y_P, Z_P) verrà proiettato nel punto $\mathbf{p} = (x_p, y_p)$ nel piano dell'immagine (si veda figura 2.8). La relazione tra i due sistemi di riferimento è data da:

$$x_p = \frac{X_P f}{Z_P} \quad , \quad y_p = \frac{Y_P f}{Z_P} \tag{2.5}$$

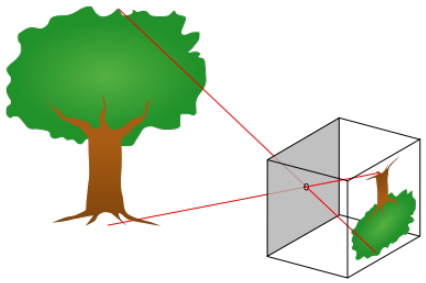


Figura 2.7: Modello di telecamera a pin-hole.

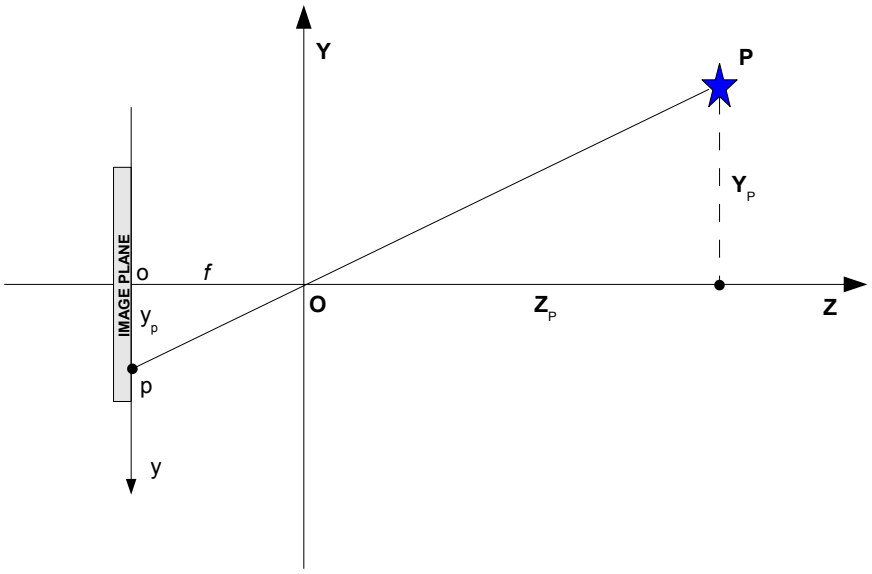


Figura 2.8: Il sistema di riferimento della telecamera visto dall'asse X.

Ricorrendo all'utilizzo delle coordinate omogenee, la trasformazione che consente di calcolare il punto nell'immagine dato il punto nella scena è esprimibile in forma matriciale come:

$$\begin{bmatrix} sx_p \\ sy_p \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix} \tag{2.6}$$

dove $s \neq 0$ è un fattore di scala diverso da 0.

Per calcolare le coordinate (u, v) di un pixel a partire dalla posizione del punto nel piano dell'immagine possiamo utilizzare le seguenti relazioni:

$$u = u_c + \frac{x_p}{w_p} \quad , \quad v = v_c + \frac{y_p}{h_p} \tag{2.7}$$

dove u_c e v_c sono le coordinate del punto principale in pixel e w_p and h_p sono la larghezza e l'altezza di un pixel, rispettivamente.

Nel sistema di coordinate dell'immagine, il sistema di equazioni 2.6 diventa:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_c & 0 \\ 0 & \alpha_v & v_c & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix} \tag{2.8}$$

dove $\alpha_u = \frac{f}{w_p}$ e $\alpha_v = \frac{f}{h_p}$. La matrice 3×4 $\mathbf{A}$ nell'equazione 2.8 è detta *matrice di proiezione prospettica*. I parametri f, u_p, v_p, u_c e v_c sono detti *parametri intrinseci* della telecamera, in quanto essi dipendono solamente dalle caratteristiche proprie della telecamera. Utilizzando la rappresentazione matriciale $\tilde{\mathbf{u}} = [sv\ su\ s]^T$ e $\tilde{\mathbf{P}} = [X_P\ Y_P\ Z_P\ 1]^T$, abbiamo che:

$$\tilde{\mathbf{u}} = \mathbf{A}\tilde{\mathbf{P}} \tag{2.9}$$

In generale le coordinate di un punto 3D non vengono espresse nel sistema di coordinate standard della telecamera, ma nel sistema di riferimento della scena. I due sistemi di riferimento sono legati da una roto-traslazione rigida nello spazio tridimensionale. La rototraslazione è composta da una matrice 3×3 ortonormale $\mathbf{R}$ che rappresenta la rotazione della telecamera, e da un vettore di tre elementi $\mathbf{t}$ che rappresenta la traslazione. Detto (X'_P, Y'_P, Z'_P) il punto espresso nelle coordinate della scena, possiamo calcolare le coordinate nel sistema di riferimento della telecamera:

$$\begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix} = \mathbf{T} \begin{bmatrix} X'_P \\ Y'_P \\ Z'_P \\ 1 \end{bmatrix} \quad , \quad \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \tag{2.10}$$

La matrice $\mathbf{R}$ e il vettore $\mathbf{t}$ vengono detti *parametri estrinseci* in quanto dipendono dalla posizione della telecamera rispetto al sistema di riferimento della scena. Unendo le equazioni 2.10 e 2.9, otteniamo:

$$\tilde{\mathbf{u}} = \mathbf{A}\mathbf{T}\tilde{\mathbf{P}}' = \mathbf{C}\tilde{\mathbf{P}}' \tag{2.11}$$

La matrice 3×4 $\mathbf{C}$ è detta *matrice di calibrazione della telecamera*. Possiamo scrivere l'equazione 2.11 anche come:

$$\tilde{\mathbf{u}} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\tilde{\mathbf{P}}' \tag{2.12}$$

dove $[\mathbf{R}|\mathbf{t}]$ è una matrice 3×4 e $\mathbf{K}$ è una matrice 3×3 che contiene i parametri intrinseci:

$$\mathbf{K} = \begin{bmatrix} \alpha_u & 0 & u_c \\ 0 & \alpha_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

In molte situazioni, il modello a pin-hole non è sufficiente per descrivere correttamente la relazione tra posizione nella scena dell'oggetto e posizione nel piano dell'immagine. Le lenti della telecamera introducono difatti una distorsione non trascurabile. In questi casi, il modello a pin-hole può essere esteso introducendo dei fattori correttivi che compensano la distorsione sistematica introdotta dalle lenti utilizzate [17]. La metodologia di correzione delle lenti che viene più frequentemente utilizzata è atta a compensare le deformazioni che causano una traslazione radiale nel piano dell'immagine. Un modello più preciso può essere ottenuto aggiungendo anche una componente tangente alla componente radiale introdotta precedentemente. Date quindi le coordinate distorte (u, v) fornite dal modello a pin-hole è possibile modellare la distorsione attraverso le seguenti equazioni al fine di ottenere la posizione del relativo punto non distorto (u_{undist}, v_{undist}):

$$\begin{cases} u_{undist} = u_d + (u_d - u_c)(K_1 r^2 + K_2 r^4 + K_3 r^6 + ...) + \\ \qquad (P_1(r^2 + 2(u_d - u_c)^2) + 2P_2(u_d - u_c)(y_d - y_c))(1 + P_3 r^2 + ...) \\ v_{undist} = v_d + (v_d - v_c)(K_1 r^2 + K_2 r^4 + K_3 r^6 + ...) + \\ \qquad (P_1(r^2 + 2(v_d - v_c)^2) + 2P_2(v_d - v_c)(y_d - y_c))(1 + P_3 r^2 + ...) \end{cases} \tag{2.14}$$

dove $K_1, K_2, ...$ sono i coefficienti di distorsione radiale, mentre $P_1, P_2, ...$ sono i coefficienti di distorsione tangenti. Le equazioni possono essere utilizzate per rettificare l'immagine in modo da eliminare l'effetto dato dalla distorsione. I parametri K_n e P_m vengono aggiunti ai parametri intrinseci della telecamera. Molti tests [26] [12] [62] hanno dimostrato che già considerando solamente la distorsione radiale del primo ordine (K_1), si ottengono risultati con una precisione di circa 0.1 pixel.

Il processo di trovare i parametri intrinseci ed estrinseci è chiamato *calibrazione della telecamera*. Nella sezione 2.4.1 tratteremo nel dettaglio questa procedura.

Valutazione comparativa

Nella valutazione sulla tipologia di sensore da utilizzare sono stati presi in considerazione diversi parametri. In primo luogo è stata considerata la robustezza del sistema, la quale garantisce la continuità di operazione, importante requisito per ogni sistema industriale (cfr. sezione 2.2.2). Uno dei principali fattori di robustezza per un sistema di bin picking è dato dalla tolleranza alla variazione delle condizioni di luce presenti nell'ambiente industriale. Dalla trattazione svolta nelle sezioni precedenti risulta chiaro che il sistema basato su telecamere è maggiormente sensibile all'illuminazione dell'ambiente. La scelta di un opportuno sistema di illuminazione risulta quindi critico nella progettazione di un sistema basato su telecamere, fermo restando sul fatto che, per qualsiasi sistema di illuminazione scelto, difficilmente si otterrà una configurazione nella quale le caratteristiche dell'immagini sono tali per cui gli algoritmi di visione non dovranno essere sviluppati in modo adattivo alle variazioni di luminosità nella scena.

Il secondo parametro fondamentale per la comparazione dei due sistemi è dato dal loro costo. I sistemi basati su telecamere sono omnipresenti non solo in ambiente industriale ma nei più disparati settori di applicazione, dalla videosorveglianza alla robotica autonoma, dalle applicazioni multimediali al cinema, etc. Questa diffusione ha fatto si che il costo di tali sistemi scendesse nel tempo, risultando al momento molto più competitivi rispetto ai sistemi basati su luce strutturata. E' possibile affermare che il costo di un sistema basato su luce strutturata sia di un ordine di grandezza superiore rispetto ad un sistema basato su telecamere.

Considerando il requisito di ridurre i costi di sistema (si veda 2.2.2), si è optato per l'utilizzo di sistemi basati sul telecamere standard.

2.3.2 Scelta del sistema di telecamere

Al fine di utilizzare le telecamera all'interno del sistema di bin-picking per la determinazione della posizione dei prodotti all'interno del contenitore, è necessario considerare che, in generale, analizzando una sola immagine non è possibile determinare la profondità degli oggetti. Difatti, un singolo pixel è l'immagine di un'intera linea nello spazio tridimensionale. Avendo a disposizione la posizione

di un punto sull'immagine possiamo quindi individuare la sua posizione spaziale al meno di un fattore di scala.

Per determinare la posizione tridimensionale abbiamo due possibilità. Avendo a disposizione il modello del prodotto, possiamo determinare quale sia la trasformazione prospettica che porti l'oggetto dal sistema di riferimento della scena al sistema di riferimento dell'immagine. La trasformazione può essere stimata avendo a disposizione solamente la posizione dell'oggetto nell'immagine e i parametri della telecamera. Con questa configurazione è possibile ottenere una buona precisione nel piano X-Y del contenitore, ma non si può dire altrettanto della stima della profondità del prodotto. Alternativamente è possibile utilizzare più telecamere per la determinazione della posizione dell'oggetto, ottenendo le informazioni sulla profondità mediante un processo di triangolazione simile a quello introdotto per i sensori basati su luce strutturata. In questo caso la stima sulla profondità dell'oggetto può migliorare sensibilmente, a discapito però del costo globale del sistema.

Nelle seguenti sezioni tratteremo nel dettaglio le caratteristiche dei sistemi mono telecamera e multi telecamera. La trattazione formerà le basi per la seguente valutazione comparativa.

Sistema mono telecamera

Utilizzando una sola telecamera precisamente calibrata è possibile stimare la posizione tridimensionale del prodotto all'interno del contenitore avendo a disposizione il modello dello stesso. Presupponendo di porre la telecamera con asse ottico parallelo all'asse Z del contenitore, la posizione nel piano X-Y può essere determinata con accuratezza. La profondità può d'altra parte essere stimata considerando i parametri della trasformazione prospettica che porti il prodotto dal sistema di riferimento della scena al sistema di riferimento dell'immagine. Intuitivamente questo corrisponde a stimare la distanza valutando la dimensione del prodotto nell'immagine e a valutare l'inclinazione dalle deformazioni a cui il prodotto è soggetto all'interno dell'immagine. E' immediato vedere come si abbia un'ampia propagazione dell'errore nella valutazione delle distanze, in quando la variazione di dimensione del prodotto nell'immagine di un singolo pixel fa variare sensibilmente la dimensione in unità di misura reali. Nel seguito della sezione descriveremo quantitativamente l'errore introdotto.

In figura 2.9 è riportato lo schema del sistema con una sola telecamera. La telecamera è posizionata in modo che l'asse ottico sia parallelo all'asse Z del contenitore. I parametri della telecamera e la sua posizione devono essere calcolati in modo tale da inquadrate l'intero contenitore. Per garantire questo è necessario determinare come i parametri intrinseci ed estrinseci siano legati alla dimensione del campo visivo. Consideriamo come incognite le dimensioni del sensore ($s_w \times s_h$) e la lunghezza focale f, che possono essere ottenuti dalle specifiche della telecamera e dell'ottica utilizzata, e l'altezza della telecamera dal piano del contenitore (h). Si noti che il rapporto tra larghezza e altezza dei sensori disponibili in commercio è fissa e pari a 4/3. E' quindi possibile utilizzare un solo parametro per la descrizione del sensore, ovvero la sua diagonale ($s_d = \sqrt{s_w^2 + s_h^2}$).

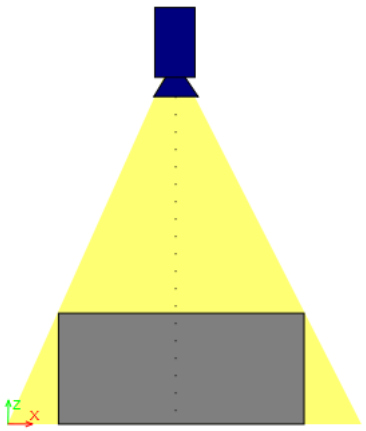


Figura 2.9: Sistema di bin picking con una sola telecamera.

Le dimensioni del contenitore (C_x, C_y, C_z) determinano la scelta dei parametri della telecamera in base all'altezza scelta. Per fare in modo che i parametri scelti soddisfino i requisiti richiesti si deve avere che (si veda 2.10.a):

$$\frac{s_d}{2f} \geq \frac{C_d}{2(h - C_z)} \tag{2.15}$$

Dove $C_d = \sqrt{C_x^2 + C_y^2}$ è la dimensione della diagonale del contenitore.

Al fine di definire l'errore nella scena in base all'errore in pixel nella valutazione della posizione dell'oggetto nell'immagine (e_{det}), è necessario valutare la dimensione nel mondo reale di un pixel dell'immagine. Si noti che e_{det} dipende dalla precisione consentita dall'algoritmo di model matching scelto. Detta

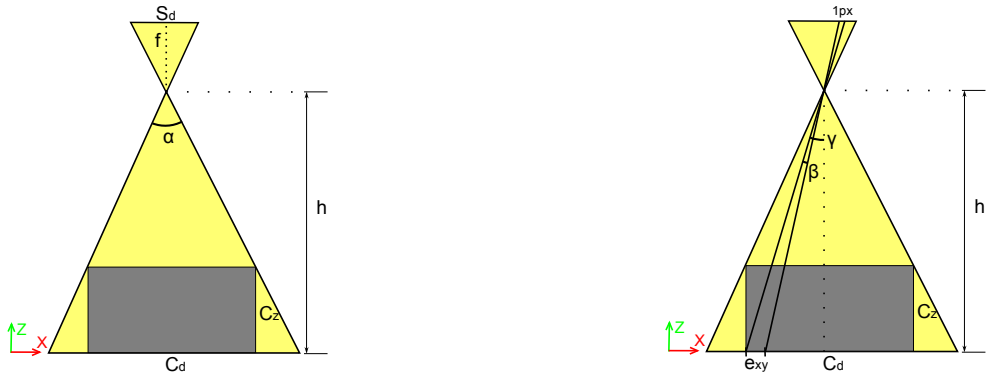


Figura 2.10: Calcolo dell'errore nei sistemi monotelecamera. La figura (a) riporta le dimensioni principali del sistema, mentre nella figura (b) sono schematizzate le informazioni necessarie al calcolo dell'errore nel piano X-Y.

$i_w \times i_h$ la dimensione in pixel dell'immagine, possiamo ottenere la dimensione in pixel della diagonale dell'immagine i_d. Notando che la dimensione di un pixel nel mondo reale è tanto più grande quanto più siamo distanti dalla telecamera, è necessario valutare la dimensione di e_{det} pixel in fondo all'angolo del contenitore. Innanzitutto, l'angolo sotteso del punto più distante in fondo del contenitore è dato da (si veda 2.10.b):

$$\gamma = atan\left(\frac{C_d}{2h}\right) \tag{2.16}$$

L'angolo sotteso da e_{det} pixel dell'immagine è dato da:

$$\beta = atan\left(\frac{1}{f}\frac{s_d}{i_d}\ e_{det}\right) \tag{2.17}$$

Dalle due equazioni precedenti possiamo quindi calcolare l'errore massimo nel piano X-Y del contenitore:

$$e_{xy} = \frac{C_d}{2} - h\ tan\left(\gamma - \beta\right) \tag{2.18}$$

Per la valutazione dell'errore nel calcolo della profondità, bisogna considerare che il calcolo della profondità per i sistemi mono telecamera è effettuato in base alla dimensione in pixel dell'oggetto nell'immagine. E' immediato vedere che il caso peggiore si ha quando l'oggetto è di dimensioni inferiori ed è posto nel punto più distante della telecamera. Possiamo considerare quindi, senza perdita di generalità, la misura di un segmento di dimensione pari alla diagonale del prodotto di dimensioni minime $P_{min} = \sqrt{(P_x^{min})^2 + (P_y^{min})^2}$ posto a

distanza $C_{max} = \sqrt{\frac{C_d}{2}^2 + h^2}$. Procedendo in modo simile a quanto fatto per la determinazione dell'errore nel piano X-Y, p_{min} è data da:

$$p_{min} = \frac{f}{C_{max}} P_{min} \tag{2.19}$$

In pixel otteniamo:

$$i_{min} = \frac{i_d}{s_d}\, p_{min} \tag{2.20}$$

Ora, incrementando i_{min} di $2e_{det}$ pixel, abbiamo che la distanza dal sensore diventa $C_{max} - e_z$, da cui possiamo calcolare l'errore nella profondità dato dal errore di individuazione del modello nell'immagine:

$$e_z = C_{max} - f\, \frac{i_d}{s_d} \frac{P_{min}}{i_{min} + 2e_{det}} \tag{2.21}$$

Le equazioni 2.15, 2.18 e 2.21 determinano completamente i vincoli per il calcolo della telecamera e dell'ottica necessaria per soddisfare i requisiti richiesti.

Sistema multi telecamera

I sistemi multi telecamera possono essere utilizzati per calcolare più precisamente la posizione nello spazio 3D del prodotto. In questi sistemi, la profondità viene valutata utilizzando una procedura di triangolazione del tutto simile a quella introdotta per i sistemi basati su luce strutturata (cfr. sezione 2.3.1).

Nella trattazione dei sistemi multi telecamera, consideriamo in primo luogo il caso dei sistemi stereo, ovvero basati su due telecamere. Per descrivere la relazione tra i punti 3D nella scena e le loro proiezioni sulle immagini delle due telecamere, è necessario ricorrere alla geometria epipolare. Supponiamo di avere due telecamere che osservano lo stesso punto della scena X. I punti O_L e O_R in figura 2.11 sono l'origine del sistema di coordinate delle telecamere sinistra e destra, rispettivamente. I punti e_L e e_R, ovvero la proiezione del punti focali di una telecamera nell'altra, vengono detto *epipoli*, mentre la linea che passa tra i due epipoli (e quindi anche per O_R e O_L) viene detta *baseline* del sistema stereo. La lunghezza della baseline è definita come la distanza tra O_L e O_R. Un punto x_L nel piano dell'immagine può essere la proiezione del punto X, ma anche la proiezione dei punti X_1, X_2, ... che giacciono sulla linea che congiunge O_L a X. Tutti questi punti vengono proiettati nell'immagine della telecamera destra in punti che giacciono sulla linea che congiunge e_R e X_R. Questa linea viene detta *linea epipolare*. Determiniamo ora matematicamente questo vincolo.

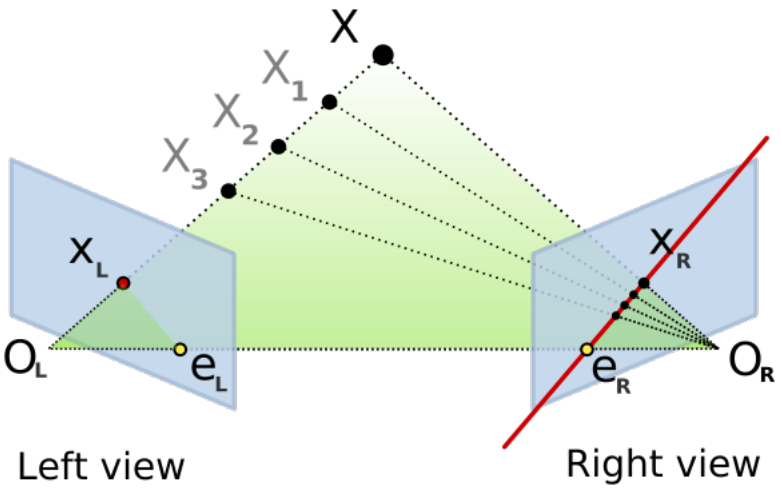


Figura 2.11: Due camere che osservano lo stesso punto della scena. Il piano dell'immagine è posto davanti ad una distanza f: questa è un modo equivalente ed alternativo di rappresentare il modello di telecamera a pin-hole.

Date due matrici delle telecamere C_L e C_R, dall'equazione 2.11 abbiamo:

$$x_L = C_L X \tag{2.22}$$

$$x_R = C_R X \tag{2.23}$$

Dato il punto x_L possiamo quindi calcolare l'insieme di punti nello spazio 3D di cui x_L può essere l'immagine. Questi punti giacciono tutti in una linea che passa per O_L e il punto $C_L^+ X_L$, dove C_L^+ è la pseudoinversa di C_L, $C_L^+ = C_L^T (C_L C_L^T)^{-1}$, per cui $C_L C_L^+ = I$. La linea può essere quindi espressa dalla seguente equazione:

$$X(\lambda) = C_L^+ \, x_L + \lambda \, O_L \tag{2.24}$$

parametrizzata attraverso lo scalare λ. Con $\lambda = 0$ otteniamo il punto $C_L^+ \, x_L$, mentre con $\lambda = \infty$ otteniamo O_L (si noti che i punti sono espressi in coordinate omogenee). Possiamo quindi proiettare questi punti nel piano dell'immagine della telecamera a destra, ottenendo $C_R \, C_L^+ \, x_L$ e $C_R \, O_L$. La linea epipolare è la linea che unisce questi due punti, la quale può essere ottenuta dal prodotto vettoriale tra i due punti:

$$l_R = C_R \, C_L^+ \, x_L \times C_R \, O_L \tag{2.25}$$

Detto $v = [v_x v_y v_z]^T$, definiamo la matrice simmetrica $[\mathbf{v}]_\times$:

$$[\mathbf{v}]_\times = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix} \tag{2.26}$$

Sapendo che $[v]_x \; x = v \times x$ possiamo scrivere l'equazione 2.25 in forma matriciale come:

$$l_R = [e_R]_x \; C_R C_L^+ x_L \tag{2.27}$$

Dato che il punto x_R giace sulla linea definita dall'equazione 2.24, abbiamo che $x_R \; l_R = 0$, da cui:

$$x_R^T \; [e_R]_x \; C_R C_L^+ x_L = x_R^T \; F \; x_L = 0 \tag{2.28}$$

dove

$$F = [e_R]_x \; C_R C_L^+ \tag{2.29}$$

è detta matrice fondamentale del sistema epipolare. Dato che $[e_R]_x$ ha rango 2, e $C_R C_L^+$ ha rango 3, la matrice F ha rango 2. La matrice F può essere calcolata a partire dai soli parametri delle due telecamere, a meno di un fattore di scala.

Se imponiamo che il sistema di riferimento della telecamera sinistra sia il nostro sistema di riferimento per la scena, abbiamo:

$$x_L = C_L \; X = K_L[I|0] \; X \tag{2.30}$$

$$x_R = C_R \; X = K_R[R_{LR}|t_{LR}] \; X \tag{2.31}$$

dove, R_{LR} e t_{LR} definiscono la rototraslazione tra il sistema di riferimento della telecamera sinistra e il sistema di riferimento della telecamera destra, e K_L e K_R sono le matrici contenenti i parametri intrinseci delle due telecamere. Possiamo ora scrivere F nella forma semplificata:

$$F = K_L^T [t_{RL}]_x R_{RL} K_R^{-1} \tag{2.32}$$

Esprimendo i punti x_L e x_R nelle coordinate normalizzate, otteniamo:

$$x'_L = K_L^{-1} \; x_L x'_R = K_R^{-1} \; x_R \tag{2.33}$$

Sostituendo nell'equazione 2.32, abbiamo:

$$x'_R \; [t_{RL}]_x \; R_{RL} \; x'_L = x'_R \; E \; x'_L = 0 \tag{2.34}$$

dove:

$$E = [t_{RL}]_x \; R_{RL} \tag{2.35}$$

viene detta *matrice essenziale* del sistema stereo. La matrice R_{RL} e t_{RL} possono essere calcolate dalla matrice essenziale utilizzando la decomposizione a valori singolari. Difatti, decomponendo $E = Udiag(1, 1, 0)V^T$, dove U e V sono scelte in modo tale da aver i determinanti maggiori di zero, allora $t_{RL} = [u_{13} \; u_{23} \; u_{33}]$ e $R_{RL} = UDV^T$ oppure $R_{RL} = UD^TV^T$. La matrice essenziale e la matrice fondamentale possono essere calcolate nel sistema reale utilizzando un metodo di calibrazione stereo. Si rimanda alla sezione 2.4.1 per maggiori dettagli sulla procedura di calibrazione.

Volendo ottenere la posizione 3D di un punto rilevato in entrambe le telecamere, dobbiamo ricorrere al processo di triangolazione. Dato che le misurazioni sono affette da errore, il vincolo epipolare descritto dall'equazione $x_R^T \; F \; x_L = 0$ non è soddisfatto precisamente. Dobbiamo quindi ricorrere ad un'approssimazione. Date le due equazioni $x_L = C_L \; X$ e $x_R = C_R \; X$ che descrivono il legame tra il punto tridimensionale e la sua proiezione nelle due immagini, possiamo combinare le due equazioni in forma lineare ad X. In primo luogo, il fattore di scale viene eliminato tramite il prodotto vettoriale, il quale fornisce tre equazioni per ogni punto nell'immagine. Ad esempio, per la prima immagine possiamo considerare $x_L \times (P \; X) = 0$, il quale fornisce il seguente sistema di equazioni:

$$x_L(C_L^{3T}X) - (C_L^{1T}X) = 0 \tag{2.36}$$

$$y_L(C_L^{3T}X) - (C_L^{2T}X) = 0 \tag{2.37}$$

$$x_L(C_L^{2T}X) - y_L(C_L^{1T}X) = 0 \tag{2.38}$$

$$\tag{2.39}$$

dove C_L^{iT} indica l'i-esima riga di C_L. Prendendo due delle equazioni in 2.39 per ognuna delle immagini otteniamo un sistema di quattro equazioni in quattro incognite, $AX = 0$, dove:

$$\mathbf{A} = \begin{bmatrix} x \; C_L^{3T} - p^{1T} \\ y \; C_L^{3T} - p^{2T} \\ x \; C_R^{3T} - p^{1T} \\ y \; C_R^{3T} - p^{2T} \end{bmatrix} \tag{2.40}$$

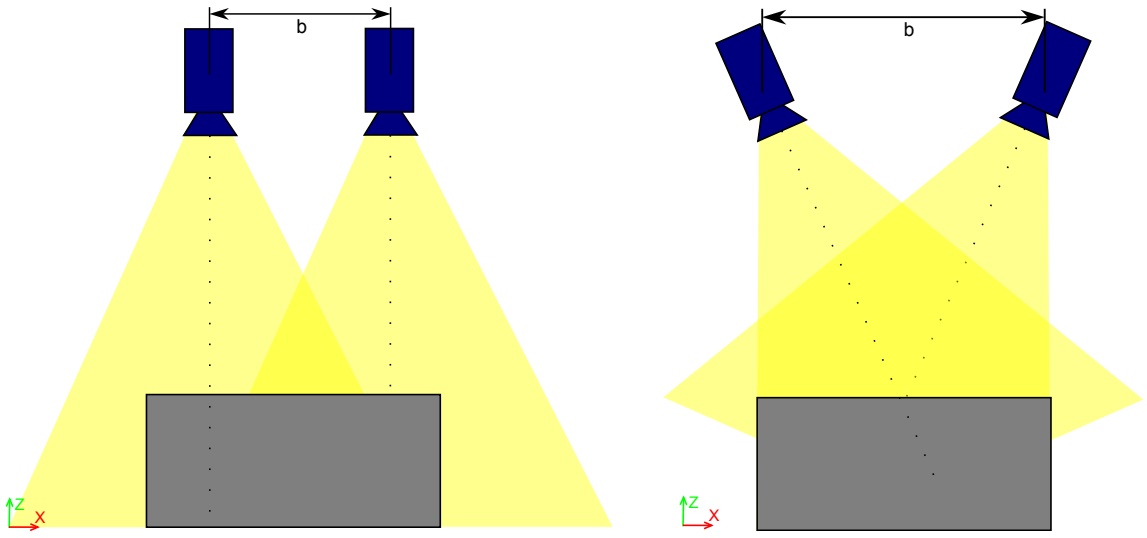
La soluzione data dal sistema $AX = 0$ non è la soluzione ottima in termine di minimizzazione della distanza del punto trovate dalle due rette epipolari. Proce-

dure più complesse, iterative o in forma chiusa, possono portare a soluzioni più precise. Si rimanda a [36] per una trattazione più dettagliata.

Nel posizionare le telecamere nel sistema stereo sono stati considerate due possibilità: entrambe le telecamere parallele all'asse Z del cassone, oppure telecamere poste ai lati con rotazione sull'asse Y verso l'interno del cassone. Nel caso di telecamere parallele (figura 2.12.a), si ha il vantaggio che le due linee epipolari sono parallele alla linea che unisce i due punti focali, e questa può essere allineata con l'asse orizzontale delle due immagini. Questo significa che per ogni punto in un'immagine, il corrispondente punto nell'altra immagine può essere ricercato guardando solo nella direzione orizzontale. Questa semplificazione è molto utile per il calcolo della disparità nell'immagine, semplificando quindi i problemi come il calcolo della mappa di profondità per ogni punto. D'altra parte, ponendo le telecamere parallelamente all'asse Z, una parte dell'immagine non inquadrerà il contenitore, riducendo in questo modo la risoluzione effettiva delle due telecamere. Considerando la baseline b del sistema stereo, si ha difatti che la scelta dell'ottica e della posizione delle telecamere è governato dalla seguente equazione:

$$\frac{s_d}{2f} \geq \frac{C_d + \frac{b}{2}}{2(h - C_z)} \tag{2.41}$$

Dove $C_d = \sqrt{C_x^2 + C_y^2}$ è la dimensione della diagonale del contenitore.



(a) (b)

Figura 2.12: Sistema di bin picking stereo nelle due configurazioni considerate.

Per valutare l'errore nella configurazione in cui le telecamere sono parallele all'asse Z del contenitore, calcoliamo innanzitutto la profondità data la distanza tra i due punti rilevati nelle due immagini, detta disparità. Le dimensioni dei due segmenti b_1 e b_2, dove $b = b_1 + b_2$ (si veda 2.13) è data da:

$$b_1 = -z\frac{x_1}{f} \tag{2.42}$$

$$b_2 = -z\frac{x_2}{f} \tag{2.43}$$

da cui possiamo ottenere b. Risolvendo in z otteniamo quindi:

$$z = f\ \frac{b}{d} \tag{2.44}$$

dove $d = x_2 - x_1$ è la disparità nell'immagine. Calcoliamo ora la baseline quando sia x_1 che x_2 vengono traslati verso l'interno nell'immagine di e_{det} pixel. Di conseguenza, la profondità aumenterà di e_z. Abbiamo che:

$$x_1 - e_{det}\frac{i_d}{s_d} = \frac{b_1\ f}{z} - e_{det}\frac{i_d}{s_d} = \frac{f\ b_1}{z + e_z} \tag{2.45}$$

Risolvendo in b_1 otteniamo:

$$b_1 = \frac{e_{det}\frac{i_d}{s_d}}{\frac{f}{z} - \frac{f}{z+e_z}} \tag{2.46}$$

Con calcoli simili si ottiene che $b_1 = b_2$, quindi $b = 2b_1$. Risolvendo ora l'equazione 2.46 nell'incognita e_z otteniamo

$$e_z = z\left(\frac{fb}{fb - 2\ z\ e_{det}\frac{s_d}{i_d}} - 1\right) \tag{2.47}$$

la quale rappresenta l'errore nella valutazione della profondità data da un errore di rilevazione di e_{det} nell'immagine. Considerando $z = \sqrt{\frac{C_d+b}{2}^2 + h^2}$, otteniamo l'errore massimo per la configurazione del sistema. Per quanto riguarda la precisione nel piano XY del cassone, si rimanda ai calcoli effettuati nella sezione precedente per il sistema mono telecamera.

In alternativa al posizionamento parallelo delle telecamere, è possibile porre le telecamere al bordo del contenitore con inclinazione sull asse Y verso l'interno dello stesso (vedi figura 2.12.b). Questa configurazione ha il vantaggio di poter sfruttare l'intero campo visivo di entrambe le telecamere. Si noti infatti che la scelta dell'ottica e del campo visivo della telecamera è governata dalla stessa equazione del caso mono telecamera (si veda equazione 2.15).

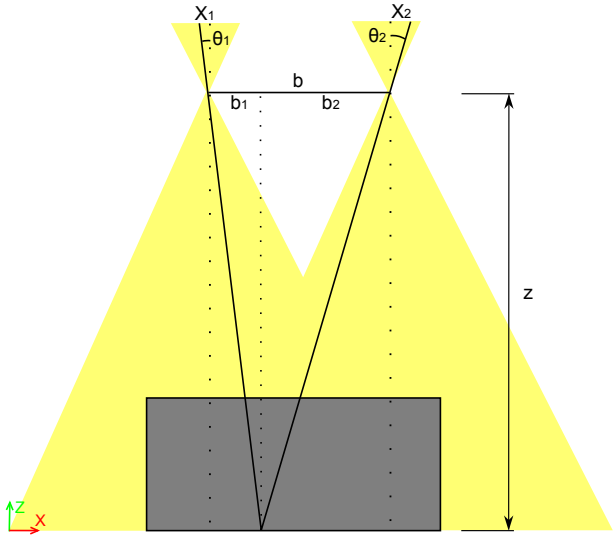


Figura 2.13: Errore nel calcolo della profondità in un sistema stereo.

Al fine di valutare l'errore nella valutazione della profondità, calcoliamo innanzitutto la profondità di un punto. Procedendo con ragionamenti simili a quanto fatto per il caso di telecamere parallele all'asse Z, calcoliamo in primo luogo la baseline a partire dalla posizione x_1 e x_2 dei pixel nell'immagine, ottenendo:

$$b = z\left(\frac{x_2 - x_1}{f} + 2\ tan\phi\right) \tag{2.48}$$

Risolvendo in z otteniamo:

$$z = \frac{bf}{x_2 - x_1 + 2\ f\ tg\phi} \tag{2.49}$$

Calcoliamo ora la baseline nel caso che il punto nella scena sia spostato in profondità di e_z e quindi si sposti nell'immagine di e_{pix}. Si noti che lo spostamento è identico nelle due immagini solo nel momento in cui il punto è nell'asse mediano tra le due telecamere. Senza perdita di generalità ci possiamo porre in questo caso. Per la baseline otteniamo:

$$b = (z + e_z)\left(\frac{x_2 - x_1 + 2e_{det}\frac{s_d}{s_i}}{f} + 2\ tan\phi\right) \tag{2.50}$$

sostituendo ora $x_2 - x_1$ ottenuto dall'equazione 2.49, possiamo quindi calcolare l'errore in profondità:

$$e_z = z\left(\frac{fb}{fb - 2\ z\ e_{det}\frac{s_d}{i_d}} - 1\right) \tag{2.51}$$

Tabella 2.2: Riassunto funzioni per il calcolo dell'errore.

Tipo Sistema	Errore piano X-Y (e_{xy})	Errore in Z (e_z)
Mono telecamera	$\frac{C_d}{2} - h\ tan\left(\gamma - \beta\right)$	$C_{max} - f\ \frac{i_d}{s_d}\frac{P_{min}}{i_{min}+e_{det}}$
Stereo parallelo	$\frac{C_d}{2} - h\ tan\left(\gamma - \beta\right)$	$h\left(\frac{fb}{fb-2\ h\ e_{det}\frac{s_d}{i_d}} - 1\right)$
Stereo inclinata	$h\left(tan(2\phi) - tan\left(2\phi\left(1 - \frac{1}{i_d}\right)\right)\right)$	$h\left(\frac{fb}{fb-2\ h\ e_{det}\frac{s_d}{i_d}} - 1\right)$

Si noti che l'errore nell'asse Z è identico al caso del sistema con telecamere parallele. L'errore sul piano X-Y è invece leggermente maggiore a causa dell'inclinazione delle telecamere. Per vedere questo, calcoliamo il punto di incontro x_0 tra il campo visivo e l'asse orizzontale:

$$x_0 = h\ tan(2\phi) \tag{2.52}$$

Considerando ora lo spostamento di un pixel, questo implica uno spostamento di $\frac{2\phi}{i_d}$ gradi, da cui è possibile calcolare:

$$tan\left(2\phi - \frac{2\phi}{i_d}\right) = \frac{h\ tg(2\phi) - e_{xy}}{h} \tag{2.53}$$

Risolvendo in e_{xy} otteniamo:

$$e_{xy} = h\left(tan(2\phi) - tan\left(2\phi\left(1 - \frac{1}{i_d}\right)\right)\right) \tag{2.54}$$

Nella tabella 2.2 sono state riportate le formule per il calcolo del errore nei tre casi considerati.

Valutazione comparativa

L'obiettivo della progettazione è di contenere i costi del sistema rispettando d'altro canto i requisiti sulle tolleranze specificati nella sezione 2.2.4. In quest'ottica, il sistema mono-telecamera è sicuramente preferibile, per lo meno a parità di caratteristiche del sensore utilizzato. Valutiamo allora l'errore nei tre casi descritti precedentemente nella sezione considerando l'utilizzo della stessa telecamera sia per il sistema mono-telecamera che per il sistema stereo.

Per riferimento consideriamo la telecamera *Sony XCD-SX910*, la quale possiede le seguenti specifiche:

Tabella 2.3: Parametri per il calcolo dell'errore

Dimensione Cassone ($C_x \times C_y \times C_z$)	$600 \times 400 \times 300$ mm
Diagonale Cassone (C_d)	721.11 mm
Focale	8 mm
Diagonale sensore	8 mm[1]
Altezza telecamera (h)	1050 mm
Dimensione minima prodotto (P_{min})	150 mm
Risoluzione immagine ($i_x \times i_y$)	1280×960 pixel
Diagonale immagine (i_d)	1600 pixel
Numero pixel di errore (e_{det})	1.5 pixel

Tabella 2.4: Valutazione errori per il sistema in esame.

Tipo Sistema	e_{xy}	e_z
Mono telecamera ($h = 850$)	1.25 mm	14 mm
Stereo parallelo ($h = 975, b = 300$)	1.25 mm	10.18 mm
Stereo inclinata ($h = 850, \phi = \pi/6$)	2.2 mm	4.11 mm

- Sensore: CCD da 1/2 B/W.
- Risoluzione massima: 1280 x 960.
- Frame rate: 15 fps.
- C mount

L'ottica selezionata possiede lunghezza focale di 6 mm.

Utilizzando i valori riportati in tabella 2.3 otteniamo i risultati in tabella 2.4. Dai valori riportati si può immediatamente notare che il sistema con due telecamere parallele non è conveniente, in quanto gli stessi valori per l'errore nel calcolo della profondità potrebbe essere garantiti dall'utilizzo di un sistema con una sola telecamera con risoluzione a 3 megapixel, la quale ha un costo inferiore rispetto a due telecamere ad 1.3 megapixel come quella considerata. Questo è dovuto al fatto che l'incremento della baseline, la quale fa guadagnare in precisione sul calcolo della profondità, è vincolata dal fatto che aumentando la baseline una

parte sempre minore dell'immagine inquadra il contenitore. Per risolvere questa problematica avremo potuto diminuire la lunghezza focale dell'obiettivo. Gli obiettivi con lunghezza focale ridotta introducono però una distorsione accentuata all'interno dell'immagine, la quale può essere solo parzialmente compensata dall'algoritmo di calibrazione.

Dalla tabella 2.4 notiamo che l'utilizzo di telecamere stereo inclinate migliora la rilevazione della profondità da 14 mm del sistema mono telecamera a 4.11 mm. Per ottenere gli stessi risultati con un sistema mono telecamera di risoluzione di circa 15 megapixel. Queste caratteristiche non sono attualmente disponibili nel mercato, se non in particolari prodotti di nicchia molto costosi.

Confrontando i valori ottenuti con quelli richiesti, si può notare che è possibile costruire il sistema mono telecamera rispettando comunque i vincoli imposti. Nel caso concreto in studio quindi, il sistema mono-telecamera è sufficiente. Nel seguito della trattazione ci concentreremo maggiormente su questo tipo di sistema. Ove opportuno, tratteremo comunque anche il caso di sistema stereo, in modo da fornire una panoramica sulle problematiche che si dovranno affrontare nel caso che i requisiti del sistema diventino più stringenti.

2.3.3 Scelta del sistema di illuminazione

La scelta del sistema di illuminazione del contenitore è di fondamentale importanza per il buon funzionamento degli algoritmi di riconoscimento, soprattutto per raggiungere le elevate precisioni richieste sulla posizione degli oggetti. Sono stati individuati due requisiti fondamentali che il sistema di illuminazione deve possedere: da un lato il sistema di illuminazione deve essere in grado di rendere il sistema il meno possibile sensibile alle variazioni di luminosità dell'ambiente circostante; dall'altro il sistema di illuminazione deve essere studiato in maniera tale da evitare il più possibile la formazione di ombre, le quali causano incertezza nella localizzazione dei bordi dei pezzi. Il primo requisito consente di ottenere un sistema più stabile, garantendo un maggior grado di continuità di operazione del sistema (si veda la sezione 2.1), mentre il secondo garantisce che l'errore in pixel degli algoritmi di model matching sia il minore possibile, permettendo quindi di utilizzare telecamere con risoluzione inferiore a parità di tolleranza desiderata (si veda 2.3.2 per una trattazione sulla dipendenza tra errore in millimetri di rileva-

zione dell'oggetto nel mondo reale ed errore in pixel di rilevazione nell'immagine, chiamato nella sezione e_{pix}).

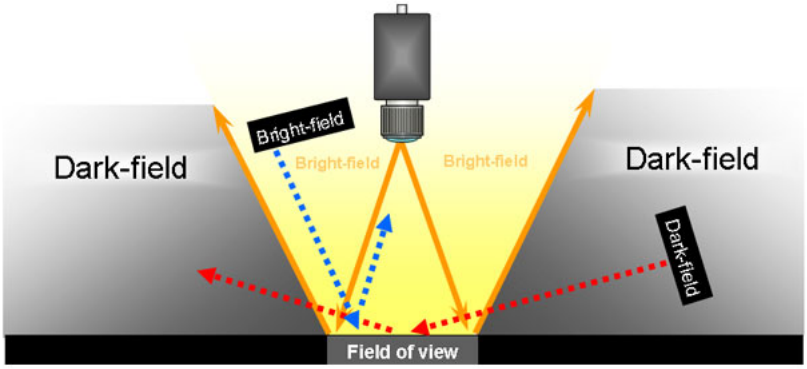


Figura 2.14: Differenza tra sistemi dark-field e bright-field.

E' possibile individuare diverse tecniche di illuminazione ormai consolidate le cui caratteristiche formeranno le basi per le successive valutazioni. In primo luogo consideriamo la distinzione tra illuminazione bright-field ed illuminazione dark-field, le due tipologie più diffuse. L'illuminazione bright-field si ha quando i raggi emessi dalla sorgente luminosa vengono riflessi dagli oggetti contenuti nella scena all'interno del campo visivo della telecamera, mentre l'illuminazione dark-field si ha quando i raggi vengono riflessi al di fuori del campo visivo della telecamera (si veda figura 2.14). L'illuminazione a dark field viene utilizzata soprattutto per la rilevazione dei difetti superficiali, in quanto è semplice con questi sistemi determinare eventuali discontinuità nel materiale. Esse non sono quindi di interesse per la nostra applicazione. Per l'illuminazione bright-field è possibile effettuare un'ulteriore distinzione tra illuminazione parziale, detta directional lighting, nella quale l'illuminazione è ottenuta da singoli illuminatori e quindi l'angolo solido di illuminazione risultante è ridotto, e l'illuminazione completa, ottenuta facendo uso di un numero maggiore di illuminatori i quali vanno a formare un angolo solido totale risultante maggiore. La prima soluzione ha il vantaggio di essere di semplice implementazione, in quanto richiede l'utilizzo e il posizionamento di un solo illuminatore. D'altra parte ha lo svantaggio di generare delle ombre non volute e di creare un'illuminazione non regolare. Nel caso in esame, entrambe le problematiche hanno un impatto negativo sul sistema finale. D'altra parte, l'illuminazione a campo completo, detta anche *dome*, si ottiene utilizzando un

illuminatore a forma di cupola di dimensioni tali da isolare completamente il sistema di visione dall'ambiente circostante (si veda 2.15). In quest'ultimo caso, dato che la luce viene prima riflessa dalla superficie del dome, si ha un illuminazione uniforme con luce diffusa, la quale garantisce l'eliminazione dei riflessi abbaglianti che potrebbero presentarsi particolarmente in oggetti riflettenti, come alcuni dei prodotti del caso in esame (si veda figura 2.2.a). Inoltre, sempre grazie al particolare posizionamento della fonte luminosa, si ha una eliminazione delle ombre, fattore molto importante per la qualità della rilevazione dell'oggetto all'interno dell'immagine. D'altra parte, l'utilizzo di un dome nasconde anche degli svantaggi non trascurabili. In primo luogo il dome deve racchiudere l'intero componente che deve essere analizzato. Nel nostro caso, è necessario che il dome sia di dimensioni maggiori di cassone, quindi in generale particolarmente elevate. Questo ha un impatto diretto sui costi del sistema di illuminazione. Inoltre, gli ingombri del sistema sono rilevanti, ed è necessario prevedere un sistema meccanico che permetta di rimuovere il dome quando il robot deve prelevare il pezzo del cassone. L'introduzione del sistema meccanico presenta due inconvenienti: in primo luogo è necessario introdurre un motore per la sua movimentazione. Fortunatamente è sufficiente utilizzare un pistone che porti il dome nelle due posizioni di interesse, ovvero vicino al cassone oppure fuori ingombro. In secondo luogo, i tempi di movimentazione del dome devono essere sommati al tempo di elaborazione delle immagini, caratteristica che può impattare negativamente sul tempo ciclo se i tempi di produzione sono particolarmente ridotti.

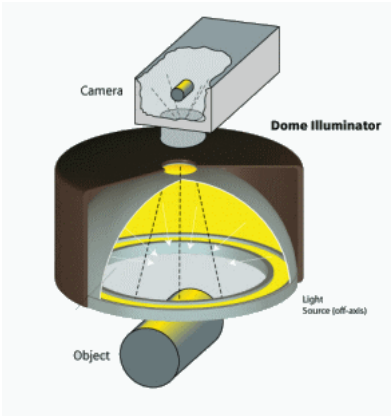


Figura 2.15: Illuminatore dome.

In alternativa ai bright-field e dark-field, è stato considerato anche l'illumina-

zione back light, ovvero di utilizzare una sorgente luminosa diffusa posta dietro la scena. In questo caso si ha una semplice implementazione del sistema, e si ottengono degli ottimi risultati per la valutazione dei profili degli oggetti ripresi. Purtroppo però, le caratteristiche del sistema non ci permettono l'installazione di una luce back-light, in quando essa sarebbe completamente oscurata quando il contenitore è pieno. Questo tipo di soluzione, di particolare economicità e stabilità, sarebbe la scelta migliore solo nel caso si utilizzi un nastro trasportatore con una superficie vibrante che sia in grado di separare i pezzi da prelevare.

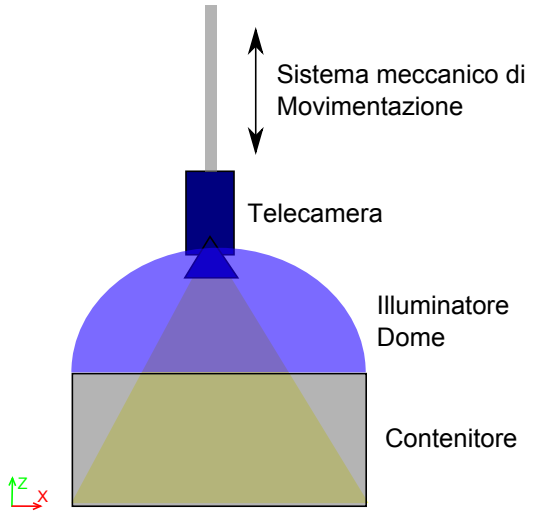
In ultima istanza, è stato preso in considerazione anche l'illuminazione stroboscopica. In questo caso viene utilizzato un illuminatore a LED particolarmente potente che viene attivato solo per un piccolo intervallo di tempo durante il quale viene acquisita l'immagine, garantendo un tempo di vita particolarmente lungo. D'altro canto, i sistemi di illuminazione con luce stroboscopica sono decisamente più costosi rispetto alle sorgenti luminose tradizionali, ed è richiesta un'accurata sincronizzazione tra il controller e il tempo di acquisizione della telecamera. Infine, necessitano di scudi protettivi per la sicurezza del personale. Queste problematiche hanno fatto abbandonare la selezione di questa tipologia di illuminazione.

Per quanto riguarda la tipologia di sorgente di illuminazione, sono state considerate due ipotesi: le sorgenti fluorescenti oppure sorgenti a LED. E' noto che le sorgenti a LED normalmente soddisfano maggiormente i requisiti applicativi dei sistemi di visione, in quanto hanno un tempo di vita e una stabilità luminosa del tempo nettamente superiore alle lampade fluorescenti. D'altra parte il costo dei LED è decisamente superiori rispetto alle sorgenti a fluorescenza. Questo è particolarmente rilevante nel caso dell'utilizzo del dome nella nostra applicazione, dato che le dimensioni elevate del contenitore.

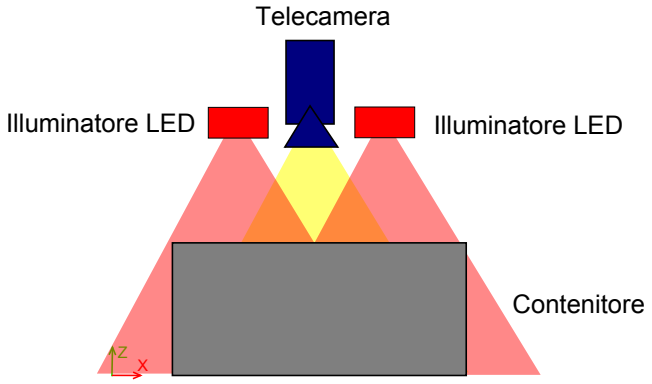
Infine nella scelta dei componenti dei sistemi di illuminazione sono stati presi in considerazione i filtri da utilizzarsi per rendere il sistema meno sensibile alla luce ambientale. I filtri agiscono sulla lunghezza d'onda della radiazione luminosa, permettendo di selezionare quali siano le frequenze di interesse. Ad esempio, è possibile utilizzare un filtro passa-banda rosso a 660nm insieme ad una luce LED monocromatica rossa, il quale consente la trasmissione alla telecamera del 95% della radiazione con lunghezza d'onda vicina al rosso, diminuendo fino a 35 volte i contributi di sorgenti a fluorescenza e fino a 4 volte i contributi derivanti dalla

luce solare.

2.4 Realizzazione del sistema di bin-picking



(a)



(b)

Figura 2.16: Schematizzazione del sistema completo nelle due realizzazioni ipotizzate: (a) soluzione con dome, (b) soluzione con LED.

Per la realizzazione del sistema sono state prese in considerazione due ipotesi. Nella prima, riportata in figura 2.16.a, si fa utilizzo di un illuminatore dome con il duplice scopo di illuminare correttamente la scena e isolare il sistema dalla

luce ambientale. Questa rappresenta sicuramente la soluzione ottima, in quanto garantisce la creazione di un sistema molto robusto. D'altra parte, il costo di un dome di tali dimensioni è elevato ed è inoltre necessario introdurre un sistema di movimentazione meccanico per portare il dome fuori ingombro quando il robot deve prelevare il prodotto. La seconda ipotesi prevede invece l'utilizzo di un illuminatore LED rosso e di equipaggiare l'ottica della telecamera con un filtro passa banda in modo da diminuire l'influenza della luce ambientale sul sistema. Si noti che questa soluzione è decisamente più economica, ma richiede che gli algoritmi di visione siano tolleranti ai cambi di illuminazione dell'ambiente. Sarà obiettivo degli esperimenti riportati nel capitolo 5 determinare se l'algoritmo di model matching proposto sia in grado di tollerare opportunamente la variazione delle condizioni di illuminazione.

Una volta costruito il sistema, è necessario eseguire una procedura di calibrazione della telecamera in modo da poter riportare le dimensioni in pixel nell'immagine in dimensioni nell'ambiente reale. La calibrazione della telecamera verrà trattata nella sezione 2.4.1 per i sistemi mono-telecamera. Nella sezione tratteremo anche brevemente le estensioni necessarie per la calibrazione di una coppia di telecamere stereo. Ottenuta la posizione dell'oggetto nel sistema di riferimento della telecamera, è quindi necessario che anche il robot sia in grado di agire utilizzando lo stesso sistema di riferimento. La procedura di calibrazione che permette alla telecamera ed al robot di condividere un unico sistema di riferimento sarà trattata nella sezione 2.4.2.

2.4.1 Calibrazione della telecamera

Le procedure di calibrazione delle telecamere sono atte alla stima dei parametri estrinseci ed intrinseci mediante misurazioni dirette sugli oggetti contenuti nella scena. Esistono diverse procedure di calibrazione, differenziate in base alle modalità di operazione e ai requisiti dell'applicazione. Nel nostro caso, è di fondamentale importanza ottenere una calibrazione la più precisa possibile. Una delle procedure di calibrazione più diffuse e che permettono di ottenere risultati con precisione maggiore è la procedura proposta da Zhang [67] [37] [62]. La procedura richiede solamente che la telecamera inquadri un pattern planare, solitamente una scacchiera, in almeno due posizioni distinte. La geometria del pattern è conosciuta, così come le distanze tra le feature di interesse (gli angoli tra i quadrati

nel caso della scacchiera). Il metodo consiste nella valutazione di una prima stima dei parametri utilizzando una formula chiusa, seguita da una raffinamento basato su una procedura iterativa di minimizzazione. I vantaggi principali della procedura utilizzata sono la sua semplicità di utilizzo e la grande precisione che essa riesce a raggiungere.

Nel capitolo 5 verranno quantificate le precisioni descritte dalla procedura descritta.

Calibrazione stereo

Nonostante che le procedure per la calibrazione potrebbero essere eseguite indipendentemente sulle due telecamere, ottenendo quindi implicitamente anche la trasformazione relativa tra le due telecamere, considerando entrambe le immagini contemporaneamente si ottengono dei risultati migliori. Questo viene fatto eseguendo, a monte della calibrazione singola delle due telecamere, un passo di raffinamento nel quale i parametri delle due telecamere vengono stimati congiuntamente in base alla posizione delle features in entrambe le immagini. Il raffinamento consente di ridurre l'errore sia dei parametri intrinseci che di quelli estrinseci.

2.4.2 Calibrazione robot-telecamera

Dato che il sistema di visione restituisce una posizione nel sistema di riferimento della scena, è necessario poter valutare la trasformazione necessario per il passaggio al sistema di riferimento del robot. Solo in questo modo il robot può partecipare attivamente al processo interpretando correttamente le informazioni provenienti dal sistema di visione.

La procedura per stimare questa trasformazione è semplice e può essere integrata all'interno della procedura di calibrazione della telecamera. Come abbiamo visto nella sezione precedente, la procedura di calibrazione della telecamera si basa sul riconoscimento della posizione nello spazio tridimensionale di un determinato pattern visualizzato nelle immagini. Come uscita della procedura si ha quindi, oltre che i parametri intrinseci ed estrinseci della telecamera, anche le informazioni sulla posizione nel sistema di riferimento della scena dei pattern utilizzati. Quindi, se il pattern viene fissato sul TCP del robot in una posizione nota (ad esempio centrando correttamente il TCP con il centro della scacchiera),

avendo a disposizione la posizione del TCP, calcolato dalle informazioni sulla posizione dei giunti date dagli encoder del robot, risulta molto semplice ottenere la trasformazione tra i due sistemi di coordinate. Dato che la procedura di calibrazione della telecamera si basa su diverse posizioni del pattern, si otterranno diverse matrici di trasformazione. E' allora opportuno svolgere una procedura di minimizzazione dell'errore in modo da valutare la trasformazione risultate.

Capitolo 3

Model Matching

Nel presente capitolo definiremo la problematica e gli algoritmi software per la rilevazione delle istanze dei prodotti all'interno dell'immagine e la stima della loro posizione nello spazio tridimensionale.

In primo luogo verrà introdotto nella sezione 3.1 il problema del model matching formalmente, per poi passare ad una rassegna dei principali algoritmi disponibili in letteratura nella sezione 3.2. I limiti di tali approcci e le caratteristiche del sistema in esame formeranno le basi per l'introduzione all'algoritmo proposto nella presente tesi, descritto nel dettaglio nella sezione 3.3. Infine, nella sezione 3.4 verrà trattato come sia possibile estendere l'algoritmo per poter valutare la posizione del prodotto utilizzando una coppia di telecamere stereo oppure per determinare la posizione di un oggetto non planare.

3.1 Il problema del model matching

L'obiettivo principale di un sistema di visione per il bin-picking è la rilevazione all'interno dell'immagini delle istanze dei prodotti e la stima della loro posizione nel sistema di riferimento della scena. Questo problema, noto con il nome di *model matching* o *model-to-image registration problem* è stato studiato in modo estensivo sia in ambito accademico che industriale, in quanto presenta applicazioni in molti settori, dall'automazione industriale all'analisi delle immagini per diagnosi medica, dalla navigazione robot alla video-sorveglianza.

Ogni algoritmo di model matching è composto da due parti fondamentali: l'estrazione delle feature di interesse, la quale governa la descrizione del modello

e l'interpretazione dell'immagine, e il matching tra l'immagine e il modello. La prima decisione progettuale riguarda quali siano le caratteristiche dell'oggetto che si vogliono estrarre in modo da fornire una descrizione accurata e distintiva dell'oggetto. Le caratteristiche possono essere di basso livello, come edge o corners o la loro posizione relativa, oppure di alto livello, come le linee o archi da cui è formato l'oggetto. Nell'individuazione delle features è importante considerare la presenza di features che siano invarianti rispetto ad una classe di trasformazioni, in quanto questa caratteristica, come vedremo, semplifica il processo di matching successivo. Ad esempio, la descrizione del modello attraverso la sua area, è invariante rispetto alle trasformazioni di roto-traslazione, in quanto la misura dell'area non dipende dalla posizione dell'oggetto nello spazio 2D. D'altra parte, l'area non è una caratteristica invariante alle trasformazioni di similitudine. Una volta selezionate le features di interesse, è necessario disporre degli opportuni algoritmi per l'estrazione delle stesse dal modello e dall'immagine. Normalmente queste due fasi vengono condotte con algoritmi differenti. Ad esempio, se è disponibile il modello CAD 2D di un prodotto, è possibile determinare la posizione dei corner valutando la descrizione analitica del modello, mentre per l'estrazione della stessa feature all'interno dell'immagine è necessario utilizzare un algoritmo di corner detection. Si noti come la scelta delle feature influisca sulla qualità di estrazione, in particolare all'interno dell'immagine. Solitamente la scelta di feature di più alto livello fornisce un processo di estrazione meno sensibile al rumore presente nell'immagine. D'altra parte, introduce un'ulteriore complicazione nel sistema di visione, in quando la qualità dell'algoritmo di estrazione influisce direttamente sulla qualità dell'intero sistema di model matching. Scegliendo feature troppo complicate si ha quindi una probabile minore stabilità globale del sistema di visione.

Una volta individuate le features nel modello e nell'immagine, si pone il problema di stimare la posa del prodotto. Per fare questo è necessario in primo luogo valutare la corrispondenza delle features del modello con le features trovate nell'immagine. Si consideri che il processo di estrazione nell'immagine può aver generato un gran numero di features, appartenenti ad una delle istanze dei modelli a cui siamo interessati oppure non appartenenti a nessun modello[1]. Per il

[1]Le feature che non appartengono a nessun modello ma sono generate da un errore del processo di estrazione vengono dette *clutter*

matching è importante da un lato che le feature siano il più possibile distintive, in modo da ridurre lo spazio di ricerca, dall'altro optare per la strategia di ricerca migliore che massimizzi le probabilità di trovare la soluzione nel minor numero di tentativi possibili. Il numero di corrispondenze da vagliare va anche considerato in base al tempo necessario per la valutazione: verificare molte ipotesi con un processo semplice può essere meno costoso che verificare poche ipotesi con un processo complicato.

3.1.1 Applicazione al bin-picking

Nei processi di bin-picking industriali, solitamente si ha a disposizione il modello geometrico dei prodotti che possono essere presenti nella scena. Il modello può essere fornito attraverso un disegno CAD oppure ricavato da una fase di training del sistema durante la quale si acquisiscono delle immagini del prodotto in posizioni note. I modelli possiedono solitamente delle caratteristiche distintive che possono essere utilizzate durante il processo di riconoscimento, come segmenti rettilinei, archi o fori. Inoltre, al contrario di quanto avviene in alcune applicazioni di analisi di immagini mediche, l'oggetto non è deformabile: possiamo considerare quindi che esso è soggetto ad una trasformazione rigida nello spazio tridimensionale. Come ulteriore semplificazione, rispetto ad esempio ai sistemi di navigazione robot o alla video-sorveglianza, il numero di oggetti che è necessario riconoscere è normalmente limitato a poche decine, il che permette una ricerca esaustiva di tutti i modelli nella scena.

Una delle complicazioni maggiori del problema del bin-picking rispetto a sistemi basati su part-feeder e nastro trasportatore è dato dal fatto che si possono verificare diverse configurazione degli oggetti nella scena: gli oggetti possono toccarsi, possono sovrapporsi ed assumere un'orientazione arbitraria rispetto alla telecamera. Queste problematiche rendono il problema di model matching molto difficoltoso nel caso generale. È possibile d'altra parte porre dei vincoli che consentano di semplificare l'approccio alla soluzione. In primo luogo, è possibile considerare il problema del model matching limitato al riconoscimento di oggetti planari, come ad esempio lamiere appena uscite dal processo di taglio. Questo pone una notevole semplificazione negli algoritmi, in quanto si passa dal riconoscimento di un oggetto tridimensionale di forma arbitraria alla ricerca di un modello bidimensionale all'interno dell'immagine. Si può inoltre supporre che gli

oggetti a cui siamo interessati per il riconoscimento siano posti in prima approssimazione parallelamente al piano del contenitore. Questa caratteristica si traduce nel cercare in prima istanza un prodotto soggetto ad una sola trasformazione di similitudine rispetto al modello 2D fornito. Si noti che questa ipotesi è verificata solo per prodotti che sono inclinati di pochi gradi rispetto al piano del contenitore. D'altra parte, i prodotti molto inclinati sono più difficili da prelevare: il parallelismo rispetto al piano del contenitore è quindi una caratteristica desiderata per la scelta dell'oggetto da prelevare tra i diversi oggetti riconosciuti. Si pone quindi un vincolo che, se da una parte non permette la rilevazione di tutte le istanze di prodotto, dall'altro consente di effettuare una preselezione dei prodotti migliori per il prelievo. Successivamente alla determinazione grossolana della posizione del prodotto mediante stima della similitudine, è quindi possibile introdurre un passo di raffinamento che sia in grado di valutare la posizione precisa nello spazio 3D del prodotto.

Un'ulteriore semplificazione viene fornita dalla presenza nella scena di una sola tipologia di oggetto. Questa caratteristica riduce il tempo di elaborazione nell'immagine, in quando è necessario effettuare la ricerca di un singolo modello.

La trattazione si baserà quindi inizialmente sulla condizione semplificata di riconoscimento di un singolo oggetto piano all'interno della scena. Quando opportuno, verrà trattato come le estensioni a generici modelli tridimensionali (si veda sezione 3.4.2) o al riconoscimento di più oggetti avranno effetti sugli algoritmi presentati.

3.2 Approcci precedenti

Grazie agli svariati campi di applicazione che necessitano di algoritmi di model matching, è presente in letteratura una vasta rassegna di algoritmi che permettono di risolvere il problema nelle diverse ipotesi di semplificazione [21] [7] [47].

Nel seguito della sezione verranno introdotti alcuni dei più comuni metodi per la risoluzione del problema, con particolare riferimento alla ricerca di un singolo modello di un oggetto planare.

3.2.1 Template matching

L'approccio più semplice per il rilevamento di un oggetto nella scena è dato dal *Template Matching*. Il template matching utilizza una maschera di convoluzione basata sul modello che vogliamo trovare all'interno dell'immagine. L'output della convoluzione sarà massimo quando la struttura dell'immagine sarà simile alla struttura del template utilizzato.

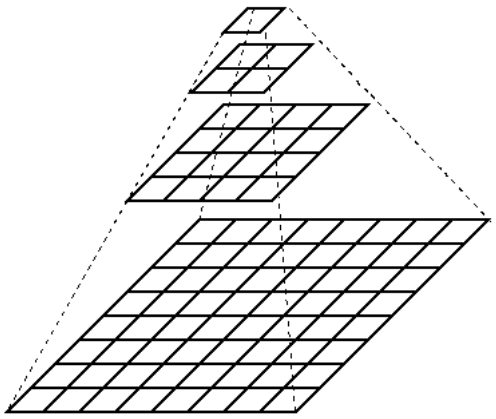


Figura 3.1: Piramide di immagini.

Per la creazione del template si parte dal modello e si crea un'immagine binaria dove i pixel con valore 1 indicano la presenta del modello in quel punto. Otteniamo quindi l'immagine $I_T(x_t, y_t)$. L'immagine di ingresso viene a sua volta binarizzata, ad esempio identificando i bordi presenti nell'immagine e in seguito eseguendo un operazione di sogliatura, ottenendo in questo modo l'immagine $I_S(x_s, y_s)$. Utilizzando come metrica la somma dei valori assoluti delle differenze (SAD, Sum of absolute differences) di ogni singolo pixel otteniamo:

$$SAD(x, y) = \sum_{i=0}^{T_{col}} \sum_{j=0}^{T_{row}} |I_S(x_s + i, y_s + j) - I_T(i, j)| \tag{3.1}$$

dove $T_{row} \times T_{col}$ è la dimensione del template. Otteniamo in questo modo una matrice di dimensione pari all'immagine dal quale è possibile determinare il punto di maggiore affinità con il modello, il quale corrisponde alla presunta posizione del prodotto nell'immagine.

Si noti che il template matching è in grado di determinare solo la traslazione a cui è soggetto il modello. Per includere trasformazioni più complesse è necessario

agire sul template in modo da includere l'effetto di queste trasformazioni. Ad esempio, volendo individuare la similitudine nello spazio 2D, è necessario considerare diversi template per tutte le possibili combinazioni di rotazioni e scala del modello. Per ogni template viene quindi valutata la risposta separatamente. I risultati possono essere successivamente raggruppati in una matrice a quattro dimensione, dove ogni dimensione rappresenta un parametro della trasformazione. Per trovare il candidato migliore è infine necessario eseguire un processo di clustering.

Grazie alla sua semplicità, il template matching offre diversi vantaggi implementativi. Da un lato è semplice definire una procedura efficiente per il suo calcolo, da eseguire eventualmente su hardware dedicato. Dall'altro, è possibile definire dei criteri di terminazione dell'algoritmo che rendono il tempo di elaborazione nel caso medio molto ridotto. Ad esempio è possibile scartare un ipotesi di posizione quando la somma parziale dopo la valutazione di n punti è inferiore ad una soglia determinata. Inoltre, l'intero algoritmo può terminare nel momento in cui si ottiene un valore maggiore di un determinato limite impostato per il quale il riconoscimento è ritenuto concluso con successo. Un altro modo per ridurre il tempo di elaborazione è quello di eseguire l'algoritmo su una piramide di immagini, dove ogni immagine viene dimezzata in dimensioni rispetto al livello precedente (si veda figura 3.1). Si parte quindi dal livello superiore e, per ogni ipotesi individuata, si effettua una ricerca limitata ad un intorno del valore trovato nell'immagine precedente, fino ad arrivare al livello inferiore della piramide contenente l'immagine originale.

La misura di similarità data dalla somma dei valori assoluti delle differenze eseguita su immagini binarizzate, nonostante la sua semplicità, presenta svantaggi sia per quanto riguarda la tolleranza al rumore presente nell'immagine che riguardo alle variazioni delle condizioni di illuminazione della scena. In [60] è stato introdotto un metodo di ricerca basato sul template matching che consente di superare queste limitazioni. L'algoritmo base viene esteso su due fronti distinti: in primo luogo viene considerata l'immagine in scala di grigi data dall'applicazione di un algoritmo di edge detection senza sogliatura (come ad esempio l'operatore di Sobel oppure il metodo di Canny interrotto prima della non-maximum suppression). Questo consente di incrementare il valore della similitudine anche in caso che il template non sia precisamente sovrapposto con il prodotto presente

nell'immagine. In secondo luogo viene introdotta una nuova misura per la valutazione della similitudine tra il template e l'immagine, la quale è stata studiata per essere robusta alle variazioni di luminosità ambientale.

Per vedere i dettagli del metodo, è necessario riformulare il problema del template matching. Il modello, invece di essere riportato nell'immagine, viene dato da un insieme di punti $p_i = (x_i, y_i)^T$ ai quali sono associati i rispettivi vettori di direzione $d_i = (t_i, u_i)$, $i = 1, ..., n$. L'immagine in ingresso viene quindi elaborata per determinare quale sia la direzione del gradiente in ogni suo punto, $e_{x,y} = (v_{x,y}, w_{x,y})$. Al fine di valutare la similarità tra l'immagine e il modello posizionato nel punto $q = (x, y)^T$, viene calcolata la somma su tutti i punti del modello del prodotto vettoriale tra la direzione del gradiente nel modello e la direzione del gradiente nell'immagine:

$$s = \frac{1}{n} \sum_{i=0}^{n} \langle d_i, e_{q+p_i} \rangle \tag{3.2}$$

Se una parte del modello è mancante in quanto occlusa, non vi sono bordi nella corrispondente posizione dell'immagine, quindi i vettori di direzione avranno una lunghezza limitata influendo poco nella somma. Allo stesso modo, se ci sono punti di bordi rilevati erroneamente nell'immagine, è probabile che non vi sia il rispettivo punto del modello in quella posizione. La misura appena introdotta non è però realmente invariante ai cambiamenti di luminosità, in quanto la lunghezza dei vettori del gradiente sarà maggiore dove la variazione di luminosità è più accentuata. La soluzione è quella di normalizzare i vettori in modo che abbiamo lunghezza unitaria, ottenendo:

$$s = \frac{1}{n} \sum_{i=0}^{n} \frac{\langle d_i, e_{q+p_i} \rangle}{\|d_i\| \|e_{q+p_i}\|} \tag{3.3}$$

Questa misura è inoltre più tollerante al rumore nell'immagine, in quanto quest'ultimo introdurrà dei risultati che, mediamente, verranno eliminati tra loro. L'equazione 3.3 restituisce un valore più elevato quando le direzioni dei vettori nel modello e nell'immagine coincidono. In alcune applicazioni è importante ottenere un alto valore di similarità anche se il contrasto è invertito. Questo può essere ottenuto applicando il valore assoluto di s. In altre circostanze non si dispone di informazioni sul contrasto per il modello. Questo è particolarmente importante nei casi in cui il modello sia stato costruito utilizzando una descrizione analitica,

come quella fornita da un disegno CAD. Avendo solo queste informazioni, non è possibile determinare il verso del gradiente. Possiamo quindi ignorare il verso applicando il valore assoluto ad ogni elemento della sommatoria:

$$s = \frac{1}{n} \sum_{i=0}^{n} \left| \frac{\langle d_i, e_{q+p_i} \rangle}{\|d_i\| \|e_{q+p_i}\|} \right| \tag{3.4}$$

Si noti che, a causa della normalizzazione dei vettori del gradiente, anche i vettori di piccola entità, dovuti ad esempio a rumore, influiscono nella sommatoria. Al fine di migliorare la qualità della rilevazione in presenza di rumore all'interno dell'immagine, è importante effettuare una sogliatura dei vettori di direzione rilevati.

3.2.2 Generalized Hough Transform

La *Generalized Hough Transform* (GHT), introdotta da Ballard nel 1980 [9], è uno dei metodi più diffusi per la ricerca di un modello all'interno dell'immagine. Essa rappresenta una generalizzazione della trasformata di Hough [38], la quale consente di trovare le istanze di semplici modelli geometrici all'interno dell'immagine: linee, cerchi, archi o in genere ogni forma che possa essere descritta attraverso un'equazione analitica chiusa.

Nel descrivere il funzionamento della GHT, introduciamo innanzitutto la trasformata di Hough per la ricerca di linee all'interno di un'immagine. Se l'immagine in ingresso non è binaria, il primo passo effettuato solitamente è la ricerca dei bordi all'interno dell'immagine in ingresso, il quale può essere effettuato tramite il noto Canny edge detection [19] oppure attraverso l'operatore di Sobel [58] seguito da un processo di sogliatura. A causa delle imperfezioni nell'immagine o nell'operazione di ricerca dei bordi, potrebbero mancare dei punti nell'immagine oppure i punti potrebbero non essere perfettamente allineati. C'è quindi una differenza tra la posizione ideale dei punti all'interno dell'immagine e quello che realmente è presente in essa, rendendo il task di ottenere la posizione del segmento non banale. L'obiettivo della trasformata di Hough è quello di risolvere questo problema raggruppando i pixel che si suppone appartenere ad una linea dell'immagine mediante un meccanismo di voto. Una linea può essere descritta in forma

esplicita utilizzando l'equazione [2] $y = mx + q$. L'idea chiave della trasformata di Hough è di descrivere la linea non in base ai suoi punti nell'immagine, ma in base ai suoi parametri m e q. Lo spazio dei parametri viene chiamato *spazio di Hough*. Lo spazio viene quantizzato in base alla precisione richiesta ed ha quindi inizio il processo di voto. Per ogni pixel dell'immagine, vengono incrementate le celle (m_i, q_i) per cui il punto appartiene all'equazione $y = m_i x + q_i$. Si noti che per ogni pixel dell'immagine viene incrementata una linea nello spazio di Hough. Alla fine del processo, i parametri della linea vengono individuati valutando quale sia la cella che ha ricevuto più voti. Si noti che, in generale, all'interno dell'immagine possono essere presenti più linee. In questo caso, invece di selezionare semplicemente la cella con più voti, è necessario effettuare un processo di clusering che riesca ad individuare quali sono le celle nello spazio di Hough che rappresentano linee distinte.

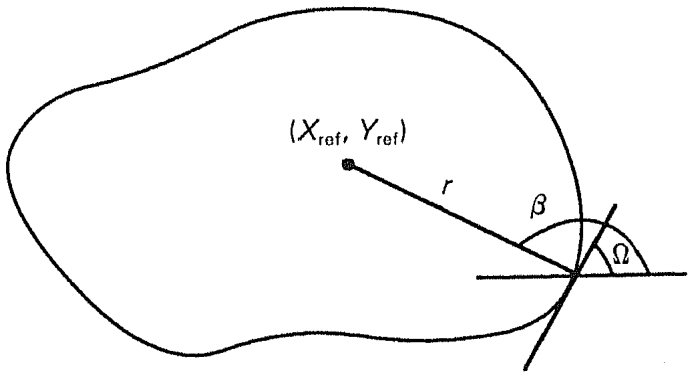


Figura 3.2: Rappresentazione del modello nella Generalized Hough Transform.

Mentre la trasformata di Hough è in grado di trovare semplici forme analitiche come linee, archi e cerchi, non è possibile con essa parametrizzare oggetti di forma arbitraria, a meno di aumentare drasticamente la dimensione dello spazio di Hough risultante. Per risolvere questo problema, Ballard [9] ha proposto un metodo alternativo per l'utilizzo della trasformata di Hough che la rende di fatto applicabile anche per problemi di matching di oggetti per forma arbitraria. Invece di rappresentare il modello con una formula analitica, viene costruita una

[2] Si noti che con l'equazione $y = mx + q$ non è possibile descrivere linee parallele all'asse delle Y dal piano cartesiano, per le quali è necessario descrivere la linea in coordinate polari. L'equazione viene qui utilizzata per semplicità di trattazione.

rappresentazione basata sulla distanza e l'inclinazione dei segmenti che congiungono i bordi con un punto scelto arbitrariamente come punto di riferimento (si veda 3.2). La rappresentazione viene inserita in una look-up table, detta *R-Table*, indicizzata attraverso l'orientazione che il punto di bordo possiede. Per ogni iterazione dell'algoritmo, si considera solamente la valutazione della traslazione a cui il modello è soggetto. La domanda a cui si risponde diventa quindi: dato un punto nell'immagine e un punto nel modello, a quale traslazione deve essere soggetto il modello in modo che i due punti coincidano? Il matching tra i punti viene filtrato valutando l'inclinazione dei punto di bordo nell'immagine e quindi vengono confrontati solo i punti con orientazione simile. La valutazione delle celle nello spazio di Hough con più voti fornisce la traslazione del modello. Se il modello può essere soggetto a trasformazioni di grado maggiore della semplice traslazione all'interno dell'immagine, la soluzione è di valutare la traslazione per ogni possibile configurazione come fatto per l'algoritmo di template matching (cfr. sezione 3.2.1). Ad esempio, nel caso di trasformazione di similitudine questa operazione produce uno spazio a quattro dimensioni sul quale è necessario effettuare successivamente il clustering dei risultati. Nel qual caso si voglia considerare la posizione di un oggetto planare nello spazio tridimensionale, si ottengono sei gradi di libertà, mentre considerando un oggetto tridimensionale di forma arbitraria i gradi di libertà diventano otto. Come si vede, l'algoritmo GHT non è applicabili in questi casi, in quanto il numero di ipotesi da verificare è esponenziale sulla dimensione dello spazio di Hough.

Sia la trasformata di Hough che la GHT producono dei risultati che hanno una precisione legata alla dimensione delle celle nello spazio di Hough. Per incrementare la precisione, è necessario ridurre la dimensione delle celle e quindi aumentare il loro numero nello spazio di Hough. D'altra parte, aumentando il numero di celle aumenta sia la memoria necessaria per il salvataggio della tabella, sia il tempo di esecuzione dell'algoritmo. Per risolvere questi problemi, sono stati introdotti degli approcci gerarchici che vanno sotto il nome di *Adaptive Hough Transform* [40] [28]. L'idea di fondo è di analizzare prima lo spazio di Hough con una quantizzazione elevata, in modo da individuare le zone di interesse, per poi analizzare più nel dettaglio le zone trovate. Un'altra tecnica simile per ridurre la richiesta di memoria e il tempo di elaborazione è di utilizzare una piramide di immagini.

Al fine di ridurre ulteriormente il tempo di elaborazione, data la semplicità dell'algoritmo è possibile sfruttare le accelerazioni hardware delle schede grafiche, come fatto in [61]. L'implementazione su scheda grafica dell'algoritmo di Hough ha permesso di ridurre di un ordine di grandezza il tempo necessario alla valutazione della posizione dell'oggetto.

Durante lo studio degli algoritmi basati su GHT per la soluzione del problema si è anche valutato l'utilizzo di segmenti del modello come feature al posto dei punti come trattato precedentemente. Il numero di feature del modello e nell'immagine risultano così ridotte, scendendo da diverse centinaia a poche decine, riducendo in questo modo il tempo di verifica di una singola ipotesi. I risultati ottenuti non sono stati però soddisfacenti, in quanto la capacità della GHT di accumulare correttamente i voti risultava eccessivamente dipendente dalla posizione rilevata dei segmenti all'interno dell'immagine, generando nelle prove effettuate diversi falsi positivi.

Gli approcci basati su GHT sono in genere robusti alle occlusioni o alla presenza di clutter. Sfortunatamente però, la GHT richiede una stima molto accurata della posizione dei bordi nell'immagine o un complesso e costoso metodo per il clustering dei risultati. Questo problema è particolarmente accentuato per modelli complessi. L'accuratezza richiesta non è in questi casi raggiungibile, anche senza considerare il rumore nell'immagine, in quanto la discretizzazione dell'immagine porta a errori nella valutazione della direzione del bordo che sono troppo elevati perché la GHT possa funzionare correttamente. Infine, la GHT si basa sulla binarizzazione dell'immagine, la quale rende l'algoritmo sensibile alla variazione di luminosità all'interno della scena.

3.2.3 Geometric Hashing

Il geometric hashing [66] [8] [56] è un metodo efficiente per la rilevazione di oggetti 2D o 3D all'interno della scena. Gli oggetti vengono modellati utilizzando un insieme di feature locali, come possono essere i punti del contorno oppure i corner nell'immagine. Nella fase di elaborazione off-line, vengono scelte delle feature che formano una base per la rappresentazione del modello in numero necessario da ottenere l'invarianza rispetto alla trasformazione a cui può essere soggetto il modello nell'immagine. Ad esempio, considerando una similitude nello spazio 2D, è sufficiente definire tre punti come base. Una volta scelti i punti, tutte le

rimanenti features vengono descritte nella relativa base. Nel caso di similitudine, è facile vedere come la base definisca un sistema di riferimento cartesiano nel quale è possibile definire la posizione delle feature utilizzando due valori. La coppia di valori vengono quindi utilizzati per indicizzare una hash table bidimensionale dove vengono inseriti il modello e la base utilizzata per il loro calcolo. Il procedimento è ripetuto per ogni base possibile.

Durante la fase di riconoscimento, vengono scelti casualmente tre features all'interno dell'immagine per formare una base. Per ogni altra feature viene quindi calcolata la sua posizione nella base scelta e utilizzato questo valore per indicizzare la look-up table. Ogni coppia di modello e base contenuta nella cella della look-up table viene quindi votata come possibile candidata. Al termine della votazione, si determina quale sia la coppia di modello e base che abbia ottenuto più voti. Se i voti raggiungono una determinata soglia, allora il riconoscimento ha successo e la posa è determinata utilizzando i punti che descrivono le basi.

Il geometric hashing ha il vantaggio di poter trattare facilmente il riconoscimento di molti oggetti contemporaneamente, in quanto questo implica un aumento di tempo di creazione della hash table e una maggiore dimensione della stessa, ma non influisce negativamente nella fase di riconoscimento. Esso è inoltre resistente alla presenza di occlusioni o clutter. D'altra parte, la sua applicabilità è subordinata alla scelta di feature distintive che si trovino in basso numero all'interno del modello e nell'immagine. Difatti, avendo n feature nel modello, si ha che il numero di basi è pari a $\binom{n}{3}$. Già per poche decine di feature si trova un numero particolarmente elevato di basi possibili. La situazione peggiora nel caso in cui si voglia trovare la posizione di un oggetto tridimensionale nello spazio tridimensionale, per il quale è necessario considerare quattro punti per ogni base, ottenendo un numero di basi pari a $\binom{n}{4}$. I punti rilevati nell'immagine sono in numero solitamente maggiore, in quanto vi sono dei punti appartenenti a diverse istanze del modello o clutter nella scena. In questa condizione, detto m il numero di punti trovati nella scena, la probabilità di scegliere una base i cui punti appartengano tutti ad una stessa istanza è pari a:

$$p = \frac{\binom{n}{3}}{\binom{m}{3}} \tag{3.5}$$

Considerando una situazione media in cui un modello è descritto da 10 features e nella scena se ne trovano 100, si ottiene che la probabilità è pari a circa il 0.074%.

Nel caso medio è quindi necessario effettuare diverse selezioni di base prima di ottenere il riconoscimento. Fortunatamente, l'algoritmo di verifica dell'ipotesi data la base scelta è molto veloce.

3.2.4 Approcci basati sui descrittori

Gli approcci descritti nelle sotto sezioni precedenti sono atti a velocizzare la rilevazione del modello introducendo metodologie specifiche per la selezione dei candidati nello spazio di ricerca. Un alternativa a questi approcci è quella di ridurre le dimensioni dello spazio di ricerca rendendo le features estratte più distintive. Si ottiene in questo modo che una feature nell'immagine può corrispondere solo a una o comunque poche feature nel modello, riducendo di molto il numero di ipotesi da vagliare.

Negli ultimi 20 anni sono stati introdotti diversi descrittori con l'obiettivo di creare feature molto distintive. Il più conosciuto ed utilizzato da questi metodi è stato introdotto da Lowe sotto il nome di SIFT (Scale Invariant Feature Transform) [43], il quale ha il vantaggio ulteriore di offrire una caratterizzazione della feature che sia indipendente dalla dimensione della stessa nell'immagine. Il calcolo della similitudine viene condotto dal matching di due features. Per ogni feature nell'immagine, viene scelta la feature del modello il cui descrittore ha distanza inferiore. Grazie al fatto che le feature sono particolarmente distintive, è possibile considerare solamente le prime features, riducendo in questo modo lo spazio di ricerca sensibilmente. I descrittori delle SIFT si basano sulla caratterizzazione dell'intorno del punto di interesse estratto. Perché si possano ottenere delle feature distintive, è necessario che il modello stesso contenga delle caratteristiche distintive, come ad esempio delle texture o delle forme particolari. Questo non è applicabile al caso in esame, in quanto si hanno dei prodotti geometricamente molto semplici e privi di textures. Inoltre, durante l'estrazione delle features nell'immagine ripresa dal sistema, è molto più probabile che vengano individuati dei punti di interesse dati dalla particolare configurazione dei prodotti nel contenitore, piuttosto che nel prodotto stesso. Vengono quindi create molto features che non possono essere utilizzate per la localizzazione di un singolo prodotto. Infine, le SIFT sono solo parzialmente invarianti ai cambiamenti di illuminazione non soddisfacendo quindi un importante requisito per il nostro sistema.

In letteratura sono presenti diversi approcci similari alle SIFT, tra le quali possiamo citare SURF [10] (Speeded Up Robust Feature), GLOH [45] (Gradient Location and Orientation Histogram) e LESH [53] (Local Energy based Shape Histogram). Tutti i metodi indicati soffrono però delle stesse problematiche elencate in precedenza per il caso in esame.

Alcuni studi si basano sui corner trovati all'interno dell'immagine. Nelle Patch-Duplets [41] [30] viene utilizzato l'Harris corner detector con precisione sub-pixel per l'estrazione dei punti di interesse. I descrittori sono quindi formati utilizzando delle coppie di punti di interesse. Nel dettaglio, il descrittore viene calcolato dalla rappresentazione a doppio angolo [33] delle orientazioni locali in un'area a forma rettangolare nell'intorno dei punti di interesse trovati. La dimensione e l'orientazione dell'area considerata è determinata in base al segmento di connessione tra i due punti. In fase di riconoscimento, si utilizzano la distanza tra i due punti trovati per ottenere la scala, la differenza di inclinazione per ottenere la rotazione e la distanza tra i punti mediani dei due segmenti per ottenere la traslazione. E' quindi sufficiente il matching di una sola feature per ottenere la stima della similitudine. Lo studio condotto in [64] ha dimostrato l'applicabilità delle Patch-Duplets al bin picking di oggetti tridimensionali. Per il training del modello è d'altra parte necessario acquisire diverse immagini dell'oggetto nelle sue possibili orientazioni, operazione tediosa e che pone rallentamenti ai cambiamenti del processo di produzione. Non è possibile ottenere queste informazioni solo dal modello CAD dell'oggetto. Questo è dovuto ancora una volta al fatto che i descrittori dipendono dalle caratteristiche ottiche dell'immagine nelle zone nell'intorno dei punti di interesse piuttosto che dalle sole caratteristiche geometriche del modello.

3.3 L'algoritmo proposto

I contributi del presente studio rispetto a quanto disponibile in letteratura sono tre. In primo luogo, lo spazio di ricerca è limitato dall'introduzione di feature distintive per la rappresentazione del modello, il che garantisce una velocità di elaborazione nettamente superiore se confrontata con gli approcci di template matching, GHT, oppure geometric hashing descritti nella sezione 3.2. In secondo luogo, il processo di estrazione delle features si basa solo su caratteristiche

geometriche che possano essere estratte anche da un modello fornito attraverso disegno CAD. In questo modo viene risolto il problema dato dall'assenza di textures rilevanti all'interno del prodotto (si veda la sezione 3.2.4). Terzo, il processo di valutazione della bontà della stima durante il matching non viene effettuato dal confronto dei descrittori, come avviene per le SIFT o gli altri metodi basati su descrittori, ma dal calcolo di una metrica direttamente sull'immagine acquisita la quale risulta essere particolarmente robusta alle variazioni di luminosità e all'occlusione.

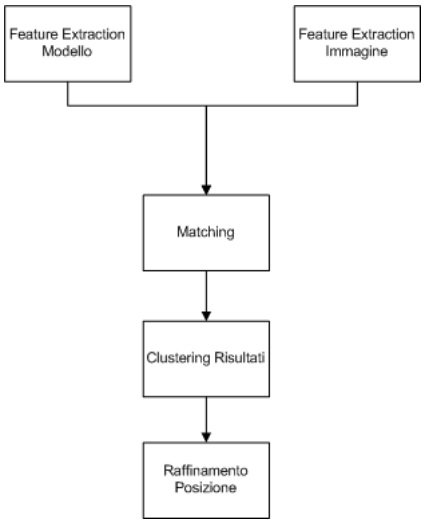


Figura 3.3: Passi dell'algoritmo.

L'algoritmo proposto è composto da quattro passi fondamentali rappresentati nel diagramma di immagine 3.3. In primo luogo vengono estratte le features di interesse nel modello e nell'immagine. Queste vengono quindi fatte corrispondere tra loro, in modo da ottenere come risultato le diverse pose possibili per il modello nell'immagine. Per ogni ipotesi verrà calcolato un valore di similitudine basata sulle caratteristiche dell'immagine. Le coppie posa-similitudine subiranno quindi un processo di clustering per determinare le pose candidate. Per ognuna di essere sarà infine eseguito un processo di minimizzazione dell'errore che consenta di migliorare la stima della posizione dell'oggetto nel sistema di riferimento della scena, ottenendo in questo modo il risultato dell'algoritmo. Nel seguito della sezione analizzeremo i quattro passi individuati separatamente.

3.3.1 Estrazione delle features

La prima scelta progettuale di un algoritmo di model matching è relativa alla scelta della feature da utilizzare. Queste possono essere di tre tipi: locali, globali o di relazione. Le features globali rappresentano delle caratteristiche dell'intero oggetto all'interno dell'immagine, come ad esempio la sua area o il suo centroide. Nei sistemi di bin picking, a causa della presenza di più oggetti posizionati in diverse e non prevedibili configurazioni, da un lato è complesso estrarre queste informazioni, in quanto presupporrebbero l'utilizzo di una procedura di segmentazione dell'immagine, dall'altro la scelta di questi tipi di feature renderebbe il sistema sensibile alle occlusioni. Le features globali sono quindi state scartate. Per quanto riguarda la feature locali, sono state prese in considerazione due tipologie di feature: punti di interesse oppure elementi geometrici contenuti nel modello (nel caso semplificato linee o archi). Scegliendo quest'ultimi, a causa dell'occlusione o del processo di estrazione stesso, non è possibile presupporre che gli elementi geometrici trovati nell'immagine siano completi. Questa situazione complica l'algoritmo di matching. Inoltre, gli oggetti da rilevare possono avere forma arbitraria, composta da elementi diversi dai linee o archi, come ellissi, spline o bordi di forma arbitraria. In questo caso non possiamo far altro che o complicare complicare il processo di estrazione introducendo anche queste tipologie di elementi oppure ignorare parte dei bordi nell'immagine. Entrambe le scelte peggiorano la qualità del sistema. La scelta migliore è risultata essere l'estrazione di punti di interesse all'interno dell'immagine. Il matching di un singolo punto non permette d'altra parte la stima della similitudine tra modello e immagine. Si è allora deciso di concentrare l'attenzione su feature che fossero composte da una coppia di punti di interesse, così come fatto dalle patch-duplet [41], utilizzando quindi delle feature di relazione. Come punto di interesse si è deciso di utilizzare i corner, la cui posizione può essere facilmente determinata sia dal modello CAD, dato che corrispondono agli incroci tra le diverse entità da cui è composto, sia nell'immagine, grazie ai numerosi algoritmi di corner extraction presenti in letteratura [35] [57] [44] [32] [42].

Il descrittore viene calcolato dalle caratteristiche dell'immagini o del modello in un intorno del punto di interesse. Al contrario di quanto scelto nei metodi presenti in letteratura (si veda la sezione 3.2.4), si vuole ottenere un descrittore che sia utilizzabile anche per modelli senza texture o altre informazioni distin-

tive, come quelli considerati nel caso in esame. Si è allora deciso di costruire il descrittore in base agli angoli formati dei lati incidenti ai due punti di interessi considerati. Aggiungendo la distanza tra i due punti, si ottiene un descrittore di tre elementi che consente di determinare la trasformazione di similitudine tra il modello e l'istanza dello stesso nell'immagine effettuando il matching di una sola feature.

I descrittori definiti non sono particolarmente distintivi se confrontati con quelli utilizzati per le SIFT, i quali hanno 128 elementi. Fortunatamente, come vedremo, lo spazio di ricerca verrà ulteriormente ridotto considerando delle euristiche date dal caso in esame. Ad ogni modo, la procedura descritta risulta molto più veloce rispetto agli approcci descritti nella sezione 3.2 dove è necessario vagliare tutte le possibili posizioni dei modelli.

La procedura di estrazione delle feature nel modello e nell'immagine sono differenti e verranno trattate nelle due sottosezioni successive separatamente.

Estrazione delle features nel modello CAD 2D

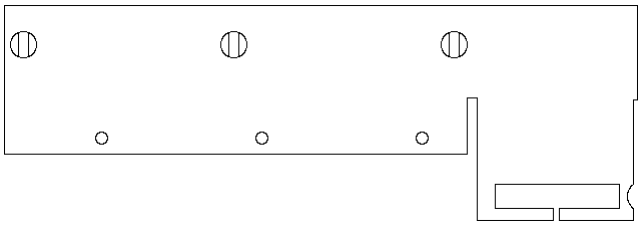


Figura 3.4: Disegno CAD contenente il modello del prodotto.

In figura 3.4 è riportato la rappresentazione CAD 2D del prodotto utilizzato per i test. Esso è formato da due tipi di entità, segmenti ed archi circolari. Nel seguito verranno considerate solo questi due tipi di entità, che rappresentano le entità più comuni presenti normalmente nei disegni CAD 2D. E' d'altra parte banale estendere i ragionamenti effettuati anche nel caso di entità di altro tipo (e.g. archi di ellisse, spline o altro).

Le entità nel modello vengono prese a coppie e valutata la loro eventuale intersezione. Il punto di intersezione è la posizione del punto di interesse. E' quindi necessario valutare gli angoli incidenti. Nel caso che le due entità siano segmenti, l'angolo viene determinato banalmente dalla differenza di inclinazione dei due

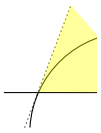


Figura 3.5: Calcolo intersezione tra segmento ed arco.

segmenti. Nel caso che i segmenti si incontrino nei loro estremi, viene aggiunto un solo punto di interesse con angolo dato dalla differenza di inclinazione. Nel caso invece il punto di incontro sia interno ad uno dei due segmenti, è necessario aggiungere due punti di interesse, l'uno con angolo uguale alla differenza di inclinazione, l'altro con il suo supplementare. Nel caso che una delle entità sia un arco di circonferenza, viene considerata la tangente all'arco passante per il punto di incontro (si veda figura 3.5), riconducendosi quindi al caso precedente. Si noti che in questo caso possiamo avere che le entità si intersechino in due punti distinti, portando ad un massimo di quattro i possibili punti di interesse.

Si noti che entità isolate, come potrebbero essere dei semplici fori all'interno dell'immagine, non hanno nessuna intersezione con altre entità e quindi non generano punti di interesse. Se il modello è composto solo da entità di questo tipo, come potrebbe essere un disco o una corona circolare, l'algoritmo non è in grado di procedere. La limitazione in queste situazioni degenere può essere risolta affrontando questi casi particolari mediante l'utilizzo di altri tipi di algoritmi, come potrebbero essere la trasformata di Hough per la ricerca di cerchi all'interno dell'immagine oppure il template matching.

Le features vengono quindi create considerando tutte le coppie di punti di interesse trovati e aggiungendovi la distanza euclidea tra gli stessi. Se nel modello sono presenti n entità, ognuna delle $n(n-1)$ coppie crea fino a quattro punti di interesse, in base alle caratteristiche dell'intersezione. Abbiamo quindi $4n(n-1)$ punti di interesse. Considerando che le feature sono date dalla scelta di una coppia di punti di interesse, nel caso pessimo si hanno $(4n(n-1))(4n(n-1)-1)$ possibili feature, ovvero il numero di feature è nel caso pessimo dell'ordine di n^4. Nei casi reali però, ogni entità si interseca con una sola altra entità, o in certi casi eccezionali con due o tre. Possiamo quindi affermare che nel caso medio il numero di punti di interesse sono dell'ordine di n, portando quindi il numero di feature nell'ordine di n^2. Nel modello considerato come esempio, abbiamo che il

prodotto è descritto da 32 entità, ottenendo circa un migliaio di features. Una ricerca esaustiva, per quanto possibile, richiederebbe molto tempo di elaborazione, dovendo effettuare il matching tra ogni feature del modello con ogni feature dell'immagine. Per risolvere questa problematica, si è deciso di operare a monte una selezione delle feature nel modello seguendo delle euristiche che si basano sull'analisi del caso in esame.

La selezione delle feature migliori per il modello si basa sulle seguenti considerazioni:

- Quando più sono distanti i punti di interesse, tanto è migliore la stima della trasformazione utilizzando il matching tra solo una coppia di feature.

- Quanto più sono distanti i punti di interessa, tanto più è probabile che il prodotto trovato nell'immagine sia privo di occlusioni. Considerando due punti vicini difatti, non vengono date informazioni sull'occlusione delle altre parti del modello. Questa è solo una condizione necessaria, non sufficiente, per determinare che l'istanza del modello trovata nell'immagine è priva di occlusioni, in quando non ci vengono date da questo vincolo informazioni sul modello nei punti compresi tra i punti di interesse scelti.

- Dopo aver selezionato una feature all'interno dell'immagine, le feature con punti di interesse molto vicini tra loro non aggiungono informazioni per la rilevazione del modello. Deve quindi essere inclusa un empirica per la valutazione di quali feature sono troppo simili tra loro, in modo da effettuare un partizionamento dell'insieme delle feature e considerare solo un elemento come rappresentativo dello stesso.

- Le entità di dimensione ridotta non vengono rilevate in modo stabile all'interno dell'immagine. E' quindi consigliabile non utilizzare i punti di interesse dati da tali entità.

Dalla discussione precedente risulta chiaro che l'algoritmo di filtraggio deve eliminare le feature con distanza inferiore a d_{min} tra i punti di interesse. Al fine di introdurre una soglia adattativa al modello utilizzato, si è deciso di porre $d_{min} = \alpha P_{max}$, dove P_{max} è la dimensione della diagonale del bounding box del prodotto. Dalle prove effettuate, un valore di $\alpha = 0.5$ consente di rimuovere più di metà delle features, ottenendo il risultato desiderato. Le feature rimanenti vengono quindi

ordinate per distanza dei punti di interesse decrescente. Questo ordinamento permette di ottenere risultati migliori già nelle prime iterazioni dell'algoritmo, aprendo la strada ad eventuali euristiche per la terminazione dell'algoritmo di matching prima che l'intero spazio delle soluzioni venga visitato. Per evitare di avere feature i cui punti di interesse sono troppo vicini tra loro, viene eseguito un algoritmo quadratico in cui, presa una feature, vengono eliminate le feature che hanno somma della distanza dei due punti di interesse interiore a c_{min}. Si noti che le feature sono state precedentemente ordinate per distanza dei punti di interesse, quindi l'algoritmo mantiene le feature con distanza maggiore. Ancora una volta, il parametro c_{min} viene dato relativamente a P_{max}, $c_{min} = \beta P_{max}$. Valutazioni empiriche hanno portato a fissare il valore di β a 0.1. Dato che le entità di dimensione ridotta difficilmente vengono correttamente individuate all'interno dell'immagine, sono state preliminarmente rimosse tutte le entità con dimensione inferiore a $\gamma = 0.15$.

Con la procedura descritta si ottiene una sostanziale riduzione del numero di feature, portando il numero da migliaia a poche decine, garantendo in questo modo un notevole incremento delle prestazioni dell'algoritmo.

Estrazione delle features nell'immagine

Come abbiamo visto, il punto di interesse nell'immagine è rappresentato da un corner. Sono stati vagliati diversi algoritmi per l'individuazione dei corner all'interno dell'immagine: corner detector di Harris [35] [48] [55], SUSAN (Smallest Univalue Segment Assimilating Nucleus) [57], l'operatore di Harris-Laplace [44], il corner detector di Gilles [32] e il laplaciano della gaussiana [42]. Dalle prove effettuate con le immagini di test, il corner detector di Harris si è rivelato essere il migliore per quanto riguarda il trade-off tra stabilità dell'algoritmo ed efficienza computazionale. Data la criticità nella rilevazione dei punti di interesse per la successiva stima della similitudine, in seguito all'individuazione di un corner la sua posizione è stata eventualmente raffinata utilizzando una stima con precisione sub-pixel. Questa è stata ottenuta effettuando l'interpolazione lineare dell'immagine nell'intorno del punto di interesse ed eseguendo nuovamente l'algoritmo di Harris nella nuova immagine creata.

Una volta determinata la posizione dei punti di interesse, ci si è quindi posti il problema di valutare se quel determinato corner potesse rappresentare l'interse-

zione di due segmenti incidenti con angolo θ. Questa condizione viene soddisfatta se il gradiente in un intorno del punto di interesse assume direzioni la cui differenza è $\pi - \theta$ analizzando i pixel in direzioni radiali al punto di interesse che formano un angolo di θ (si veda figura 3.6). Definiamo il valore numerico che rappresenta la probabilità che un determinato corner possa rappresentare l'intersezione di due segmenti incidenti con angolo θ con il nome di *risposta all'angolo* θ. Si noti che, data la casuale disposizione degli oggetti nel contenitore, un corner può rappresentare contemporaneamente l'intersezione di più segmenti appartenenti a modelli differenti o al contenitore stesso. Per ogni corner è quindi necessario valutare la risposta ad ognuno degli angoli di interesse, dati dai possibili angoli formati delle entità di cui è composto il modello.

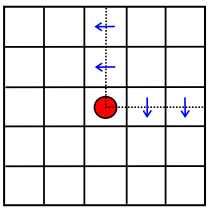


Figura 3.6: Risposta attesa per l'angolo di 90°. In figura è riportato un corner i cui lati uscenti sono disposti a 90°. L'angolo formato tra i gradienti dei due lati è ancora di 90°. Questa rappresenta la condizione ideale, per cui la risposta all'angolo deve essere massima.

Vediamo ora come calcolare le risposte all'angolo θ. Il primo passo è quello di calcolare il gradiente nell'immagine. I passi per il calcolo del gradiente sono semplici: l'immagine I viene filtrata con un opportuno kernel che sia in grado di evidenziare gli edge nelle due direzioni X e Y, ottenendo in questo modo due immagini I_x e I_y. Il modulo del gradiente del pixel (i, j)-esimo viene quindi calcolato da $G_m(i,j) = \sqrt{I_x(i,j)^2 + I_y(i,j)^2}$, mentre la sua direzione è data da $G_d(i,j) = atan\left(\frac{I_y(i,j)}{I_x(i,j)}\right)$. Nella scelta del kernel per il filtraggio, sono state considerate due opzioni: il filtro di Sobel [58] oppure la derivata della funzione gaussiana utilizza anche dall'algoritmo di Canny per l'edge detection [19]. Mentre il primo kernel ha prestazioni più elevate, in quanto di dimensioni più inferiori, il secondo ottiene dei risultati migliori nel caso di immagini con molto rumore. Dato che il peggioramento in prestazioni della derivata della gaussiana non è eccessivo

con i moderni calcolatori, si è scelta la seconda opzione, riservando la prima per applicazioni con stringenti vincoli real-time.

Al fine di eliminare l'effetto dato dal rumore, dato che il seguito della procedura per la creazione dei descrittori non considera il modulo del gradiente, viene effettuata una sogliatura in modo da considerare solo i punti che abbiano $G_m(i,j) > m_{thres}$. Dato che comunque il rumore non incide eccessivamente nei successivi calcoli, si è scelto un valore di soglia molto permissivo, pari a $m_{thres} = 0.1$.

Il passo successivo è la creazione, per ogni θ, delle maschere dei pixel che verranno utilizzate per il calcolo della risposta. La maschera è costruita considerando una patch di dimensione $m_{size} \times m_{size}$ dove, partendo dalla cella centrale, si prosegue in direzione radiale in direzione dell'asse X, escludendo il pixel centrale. Le maschere rimanenti vengono quindi ottenute ruotando di m_{step} gradi la direzione del segmento (si veda figura 3.7). Si noti che, per i passi successivi, m_{step} deve essere un divisore di θ. Nei test effettuati si sono ottenuti risultati soddisfacenti per $m_{size} = 11$ e $m_{step} = \frac{\theta}{4}$.

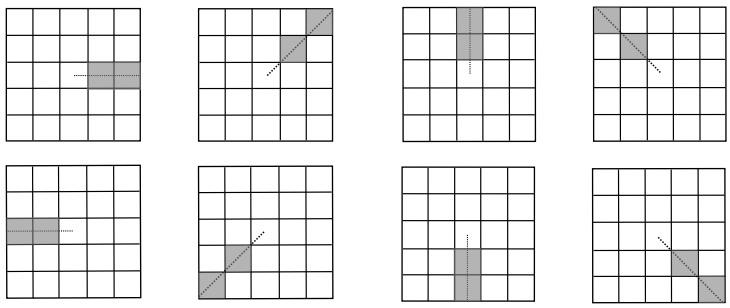


Figura 3.7: In figura sono riportate le maschere per la valutazione della risposta all'angolo $\theta = 90°$, con parametri $m_{size} = 5$ e $m_{step} = \frac{\theta}{2}$.

Ogni maschera così creata viene quindi sovrapposta centrata nella posizione del corner all'immagine contenente la direzione del gradiente, permettendo in questo modo di individuare i pixel di interesse. I valori di gradiente così ottenuti vengono inseriti in un istogramma dove gli angoli vengono raggruppati con passo m_{step}. Chiamiamo h_i questi istogrammi, dove $i = 1, ..., \frac{2\pi}{m_{step}}$, ognuno dei quali possiede $\frac{2\pi}{m_{step}}$ valori.

Avendo a disposizione gli istogrammi, il passo finale del calcolo della risposta è dato dal confronto tra gli istogrammi che rappresentino segmenti che forma-

no un angolo θ. Questa condizione viene data dalla valutazione dell'indice degli istogrammi, garantendo che la differenza sia $m_{diff} = \frac{\theta}{m_{step}}$. Per ogni coppia di istogrammi, h_i e $h_{i+m_{diff}}$, vengono valutati gli elementi che rappresentino quantizzazione degli angoli con differenza $\pi - \theta$. Questo viene ottenuto ponendo che la differenza tra i due indici sia $a_{diff} = \frac{\pi-\theta}{m_{step}}$. Ogni coppia di valori viene quindi moltiplicata tra loro e la risposta all'angolo viene calcolata dal massimo di questi valori. Formalmente otteniamo:

$$r(\theta) = \max_{i,j=1,\ldots,\frac{2\pi}{m_{step}}} \left(h_i(j) \cdot h_{mod(i+m_{diff})}(mod(j + a_{diff}))\right) \tag{3.6}$$

dove la funzione $mod(i)$ è definita come:

$$mod(i) = \left[(i-1)mod\frac{2\pi}{m_{step}}\right] + 1 \tag{3.7}$$

Si noti che la moltiplicazione tra i due valori tende a privilegiare le situazioni in cui, considerando direzioni che formano un angolo θ, si ottengono valori di gradiente nei due segmenti che hanno differenza di $\pi - \theta$, come desiderato. Il valore massimo della funzione è $\left(\frac{m_{size}-1}{2}\right)^2$.

Con la risposta agli angoli calcolata per ogni punti di interesse, possiamo ora passare alla creazione dei descrittori, ricordando che, come fatto per l'estrazione delle features nel modello, questi sono dati da una coppia di punti di interesse e della lunghezza del segmento che li congiunge. Il primo passo è la valutazione di tutte le possibili coppie di angoli nei descrittori del modello. Per ognuno dei due angoli nella coppia vengono quindi presi tutti i punti di interesse nell'immagine che abbiano una risposta all'angolo maggiore del valore imposto come soglia r_{thres}. Questi vengono infine combinati tra di loro in modo da formare i descrittori.

3.3.2 Matching

L'algoritmo di matching è composta da due passi: in primo luogo vengono accoppiate le feature del modello con le feature dell'immagine, consentendo in questo modo di creare un ipotesi sulle trasformazioni di similitudine a cui è soggetto il modello. L'ipotesi viene quindi vagliata utilizzando una funzione di similitudine basata sull'immagine acquisita.

Per l'accoppiamento delle feature, per ogni feature del modello vengono considerate tutte le features nell'immagine che abbiano gli angoli uguali. Da queste

vengono quindi rimosse le feature per cui il rapporto tra la dimensione dei segmenti abbia dei valori fuori dai limiti imposti dalla valutazione della scala massima e minima del modello all'interno dell'immagine ottenute grazie al processo di calibrazione.

Dalla coppia di feature vengono quindi calcolate le possibile similitudini, le quali forniscono la posa del modello all'interno dell'immagine. La traslazione $t = [t_x\ t_y]^T$ viene calcolata dalla differenza dei punti mediani dei due segmenti, la rotazione rot dalla differenza delle loro inclinazioni e la scala s dal rapporto tra le due lunghezze. La trasformazione di similitudine, espressa in forma matriciale, è allora data da:

$$sim = \begin{bmatrix} 1 & 0 & \frac{t_x}{s} \\ 1 & 0 & \frac{t_y}{s} \\ 0 & 0 & \frac{1}{s} \end{bmatrix} \cdot \begin{bmatrix} cos(rot) & sin(rot) & 0 \\ -sin(rot) & cos(rot) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -t_x \\ 1 & 0 & -t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} cos(rot) & sin(rot) & \frac{t_x}{s} - t_x\ cos(rot) - t_y\ sin(rot) \\ -sin(rot) & cos(rot) & \frac{t_y}{s} + t_x\ sin(rot) - t_y\ cos(rot) \\ 0 & 0 & \frac{1}{s} \end{bmatrix} \tag{3.8}$$

Si noti che, se i due angoli della feature sono uguali, è necessario considerare anche la situazione duale che corrisponde ad invertire l'associazione tra le due coppie di punti di interesse. Per come è stata definita la trasformazione, questo consiste semplicemente nel considerare anche la similitudine con stessi parametri di traslazione e scala e con rotazione aumentata di π. Infine, la stessa procedura va effettuata considerando che il modello si può trovare ribaltato all'interno dell'immagine. Per ogni matching tra due features si hanno quindi da due a quattro possibili pose da considerare.

A partire dalle pose determinate dalla corrispondenza tra le feature del modello e quelle dell'immagine, è ora necessario determinare una procedura per discriminare quali di esse possano effettivamente rappresentare un istanza del modello. Dato il gran numero di ipotesi da vagliare, è necessario che la valutazione sia eseguita con un algoritmo che abbia un tempo di esecuzione limitato. A tal scopo è stato considerata una variante della misura di similarità introdotta in [60] è descritta nella sezione 3.2.1, ottenendo in questo modo un indicazione sulla presenza del modello nella posizione ipotizzata che risulta essere robusta alla variazioni di luminosità, all'occlusione e al clutter presente nella scena. Le variazioni rispetto a

quanto descritto nella sezione 3.2.1 sono due: in primo luogo saranno considerate solo le direzioni dei gradienti, considerando quindi tutti i vettori di lunghezza unitaria. Questa scelta assicura una maggiore resistenza alla variazioni brusche di luminosità. Ricordando che l'immagine contenente la direzione dei gradienti è stata sogliata considerando solo i punti per cui il rispettivo modulo fosse maggiore di m_{thres}, buona parte del rumore presente nell'immagine è già stato rimosso. La parte rimanente contribuirà mediamente in modo uguale in tutti i calcoli di similitudine, non condizionando quindi eccessivamente la comparazione tra i diversi valori. In secondo luogo, ottenendo le caratteristiche del gradiente per il modello da un disegno CAD, non abbiamo modo per discriminare se la sua direzione sia di un angolo oppure del suo opposto. Per risolvere questa problematica consideriamo allora il valore assoluto del prodotto scalare tra il vettore gradiente nell'immagine e nel modello. Ricordando che $\langle a, b\rangle = \|a\| \ \|b\| \ cos(\angle a - b)$, possiamo riscrivere la funzione di similitudine utilizzata:

$$s = \frac{1}{n} \sum_{i=0}^{n} \sum_{j=0}^{n} |cos(G_d(i,j) - M_d(i,j)| \tag{3.9}$$

dove M_d è l'immagine di dimensioni pari alle dimensioni di G_d contenente la direzione del gradiente del modello nella posa da vagliare.

La misura di similitudine ha un valore da 0 a 1, dove 1 rappresenta il matching perfetto tra il modello e l'immagine. Il discostamento dal valore 1 è dato da un lato dall'errore nel calcolo della trasformazione di similitudine, dall'altro dalle occlusioni. L'errore nel calcolo della trasformazione è causato sia da un errore nella rilevazione dei corner sia dal fatto di aver considerato una similitudine invece che una più corretta omografia per descrivere la posa del modello. La situazione peggiora quanto più i prodotti sono inclinati rispetto al piano X-Y. Da questa discussione risulta chiaro che non ci possiamo affidare ad una semplice valutazione del valore s rilevato per un modello, accettando ad esempio i valori di s al di sopra di una certa soglia. E' altresi necessario utilizzare una procedura che consenta di rafforzare le diverse ipotesi tra di loro. Se più feature della stessa istanza del modello vengono trovate nell'immagine, si ha che diverse pose con parametri poco distanti tra loro avranno valori della funzione di similitudine molto elevati. Questa condizione deve essere evidenziata dall'algoritmo. Formalmente, quanto introdotto significa effettuare un clustering nello spazio 4-dimensionale delle possibili trasformazioni a cui l'elemento può essere soggetto.

3.3.3 Clustering risultati

Al fine di effettuare il clustering dei risultati è stato utilizzato l'algoritmo *mean-shift*, il quale rappresenta una procedura generica e non parametrica per la ricerca dei cluster contenuti nello spazio considerato. Al contrario del classico approccio al clustering dato dalla *K-means* [27], non vengono fatte ipotesi sulla distribuzione di probabilità a cui i punti sono soggetti ne al numero di cluster. Questa seconda condizione è di fondamentale importanza nel nostro problema, in quanto non siamo a conoscenza a priori di quante istanze dell'modello si trovino nell'immagine. L'algoritmo è stato proposto inizialmente in [31], per poi essere riadattato da Cheng [20] per gli algoritmi di visione artificiale. In particolare, Cheng ha permesso di considerare anche il peso dei punti nella stima, che nel nostro caso è dato dal valore delle funzione di similitudine associato alla posa. Recentemente, l'algoritmo è stato ulteriormente esteso per risolvere problemi di visione che includono la segmentazione [22], lo smoothing adattativo [23] e il tracking [24] [16].

Per illustrare l'algoritmo mean shift, consideriamo di avere un insieme S di punti nello spazio euclideo n-dimensionale. Per ogni $s \epsilon S$ è inoltre definita una funzione $w(s)$ che fornisce il peso di s. Sia $K(x)$ una funzione kernel che indica quando x contribuisce alla stima della media. Abbiamo che la media dei campioni m nell'intorno del punto x è data da:

$$m(x) = \frac{\sum_{i=1}^{n} K(x - x_i)\ w(x_i)\ x_i}{\sum_{i=1}^{n} K(x - x_i)\ w(x_i)} \tag{3.10}$$

La differenza $m(x) - x$ viene detta mean shift. L'algoritmo di mean shift è un algoritmo iterativo che sposta i punti verso la loro media, ovvero, ad ogni iterazione, si ha $x \leftarrow m(x)$. L'algoritmo termina quando la differenza $x - m(x)$ è al di sotto di una determinata soglia.

Tipicamente, la funzione K è simmetrica in direzione radiale, ovvero $K(x) = k(-\|x\|^2)$, dove k è detto profilo del kernel. La funzione più comunemente utilizzata come profilo del kernel è la gaussiana:

$$k(x) = \frac{1}{\sqrt{(2\pi)^d\ \|\sigma\|}} e^{-\frac{1}{2}\ \sigma\ x} \tag{3.11}$$

dove d è la dimensione dello spazio euclideo. Il vettore σ ha quattro elementi, i quali rappresentano le varianze nelle dimensioni dello spazio euclideo.

Per effettuare il clustering, S contiene inizialmente l'insieme delle pose date dall'algoritmo di matching. Durante l'esecuzione dell'algoritmo mean-shift, i punti variano la loro posizione spostandosi verso la media calcolata. La condizione in cui alcuni dei punti considerati vanno a coincidere corrisponde al fatto che i punti appartengono allo stesso cluster. Quando si verifica questa condizione, che nel problema in esame corrisponde ad aver ottenuto due stime della stessa istanza, i due punti vengono unificati in un unico punto il cui peso è dato dalla somma dei pesi dei due punti.

Terminata la procedura di clustering dei risultati, è ora necessario selezionare i candidati per la fase di raffinamento successiva. Per questa procedura possono essere scelte due strategie: selezionare il candidato con valore massimo, oppure tutti i candidati che abbiano valori superiori ad una soglia impostata. La scelta di quale strategia adottare si basa sul voler riconoscere un solo oggetto contenuto nel contenitore oppure tutti ed è quindi dettata dalle caratteristiche che si vogliono ottenere per l'applicazione.

3.3.4 Minimizzazione dell'errore

Al termine del clustering, le stime della posizione sono molto grossolane, in quando sono delle trasformazione di similitudine che si basano sulla corrispondenza di due soli punti. Nell'applicazione considerata è invece di fondamentale importanza ottenere una stima quanto più precisa possibile della posizione degli oggetti nella scena. E' quindi necessario effettuare una procedura di raffinamento che consenta di minimizzare l'errore tra la posa determinata e la posizione dei bordi nell'immagine.

Il primo passo della procedura è dato dall'ottenimento della posizione precisa dei bordi all'interno dell'immagine. Per fare questo è possibile utilizzare una variante dell'algoritmo di Canny dove venga inclusa la non-maximium suppression per individuare la posizione esatta dei bordi, ma al quale sia stato rimosso il passo successivo di sogliatura con isteresi. Si noti che la sogliatura con isteresi è stata rimossa per preservare la proprietà di invarianza ai cambiamenti arbitrari di luminosità dell'algoritmo fin qui descritto. Un miglioramento ulteriore al rilevamento dei bordi nell'immagine potrebbe essere ottenuto utilizzando una procedura con precisione sub-pixel, come quella riportata in [59].

Il problema di stima della trasformazione che si desidera affrontare in questa sezione può essere formalizzato nel seguente modo: dato Q l'insieme dei punti q_i estratti dall'immagine e E l'insieme delle entità geometriche e_i di cui è composto il modello, trovare la trasformazione Θ per cui si abbia:

$$arg\min_{\Theta} \sum_{i=1}^{n} \min_{j=1,\dots,m} d(M(\Theta, e_j), q_i) \tag{3.12}$$

dove, $M(\Theta, e_j)$ rappresenta l'applicazione della trasformazione Θ alle entità di E, e $d(\cdot)$ è la funzione per il calcolo della distanza punto-entità. Nel caso in esame, sono stati individuati due modi equivalenti per esprimere la funzione M. Da un lato è possibile considerare l'omografia nello spazio bidimensionale dell'immagine, dato che è noto essa rappresenti la trasformazione dei punti di un piano nello spazio 3D ai punti in un altro piano vista dalla telecamera. Essendo il nostro oggetto planare, l'omografia permette di descrivere correttamente la sua posa nello spazio 3D. Alternativamente, è possibile rappresentare con M direttamente la posa del prodotto nello spazio 3D e quindi proiettare tale posa nello spazio bidimensionale per determinare la posizione delle entità del modello nell'immagine. Questa seconda opzione possiede due vantaggi: in primo luogo, lavorando nello spazio tridimensionale si ottiene una stima che è indipendente dalla posizione e dalle caratteristiche della telecamera utilizzata. Grazie a questo è possibile integrare nello stesso processo di minimizzazione rilevazioni provenienti da più telecamere, semplificando quindi l'estensione stereo di cui parleremo nella sezione 3.4.1. In secondo luogo, si ha modo di vincolare semplicemente i valori di rotazione e traslazione nello spazio 3D. In questo modo si evita di raggiungere durante la minimizzazione configurazioni impossibili per il prodotto all'interno del contenitore. Per queste ragioni si è scelto di operare nello spazio tridimensionale, considerando con Θ la trasformazione di roto-traslazione in tale spazio. Per poter calcolare la distanza tra i punti nell'immagini e le entità del modello è necessario che la posa 3D venga proiettata all'interno dell'immagine. Per fare questo ci possiamo avvalere della matrice C contenente i parametri della telecamera ottenuta dalla procedura di calibrazione. Abbiamo che, detti $p'(X, Y, Z)$ i punti nello spazio 3D del modello, corrispondenti ad esempio agli estremi dei segmenti, e detta Θ la posizione del modello nel sistema di riferimento della scena, la posizione dei punti $p(X, Y)$ nel piano dell'immagine è data da:

$$p = C\ \Theta\ p'(i) \tag{3.13}$$

Dall'equazione 3.13 è semplice ricavare l'espressione analitica dell'entità nello spazio bidimensionale.

In fase di inizializzazione dell'algoritmo di minimizzazione è necessario determinare Θ a partire dalla similitudine h fornita dal processo di matching, rappresentata da una matrice 3×3 in coordinate omogenee. Per vedere come questo sia possibile, consideriamo la proiezione nel piano dell'immagine dei punti del modello centrato nel sistema di riferimento della scena. Abbiamo che la posizione del punto nel piano dell'immagine è dato da $x = CX$, dove X è il punto nello spazio 3D, mentre x è la sua proiezione nel piano dell'immagine. A partire da x, nel passo di matching abbiamo valutato la matrice h per cui $x' = hx$, dove x' sono i punti dell'istanza del modello rilevata nell'immagine. Considerando la rototraslazione che descrive la posizione del modello abbiamo inoltre che $X' = \Theta\ X$. Con semplici passi otteniamo che $hCX = C\Theta\ X$, da cui, rimuovendo X, abbiamo $hC = C\Theta$. Per risolvere in Θ dobbiamo ricorrere alla pseudo-inversa di C, ovvero alla matrice C^+ tale per cui $CC^+ = I$. Si ha che $C^+ = C^T(CC^T)^{-1}$, da cui otteniamo la seguente relazione:

$$\Theta = C^T(CC^T)^{-1}\ h\ C \tag{3.14}$$

Per effettuare la minimizzazione, è necessario calcolare la distanza tra ogni punto dell'immagine con le entità del modello. Il matching viene effettuato in modo da considerare anche la direzione del gradiente, in modo da ottenere implicitamente una procedura che rispecchi la funzione di similarità descritta.

La procedura di minimizzazione descritta presenta ancora una problematica: essa non è robusta alla presenza di outliers, ovvero di quei punti dell'immagine che non appartengono effettivamente all'instanza del modello considerata oppure sono generati a causa di errore. La procedura di minimizzazione tenterà comunque di fornire un valor di distanza a questi punti, impattando negativamente nella qualità della stima della posa. Per risolvere questo problema, è possibile riformulare la funzione di minimizzazione riportata nell'equazione 3.12 nel modo seguente:

$$arg\min_{\Theta} \sum_{i=1}^{n} \min_{j=1,\dots,m} \rho\left(\frac{d(M(\Theta, e_j), q_i)}{\sigma}\right) \tag{3.15}$$

dove ρ è una loss function robusta e σ rappresenta la scala dell'errore, ovvero la deviazione standard delle distanze calcolate. Per la definizione della funzione

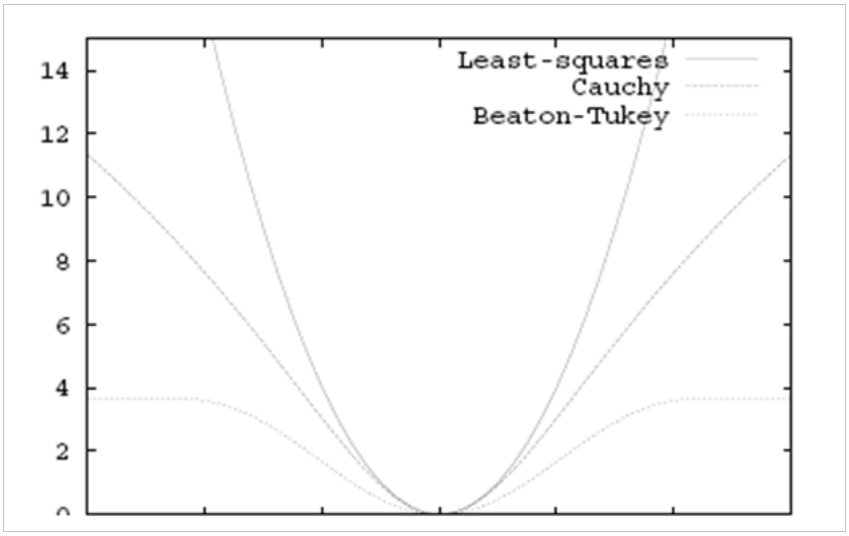


Figura 3.8: Funzione biweight di Beaton-Tukey in confronto alla stima a minimi quadrati.

ρ, possiamo utilizzare la Beaton-Tukey biweight function, la quale considera solo i punti che abbiano una distanza inferiore ad a volte la deviazione standard dell'errore:

$$\rho(u) = \begin{cases} \frac{a^2}{6}\left[1 - \left(1 - \left(\frac{u}{a}\right)^2\right)^3\right] & |u| \leqslant a \\ \frac{a^2}{6} & |u| > a \end{cases} \tag{3.16}$$

Si noti che ponendo $\rho(u) = u^2$ otteniamo una stima ai minimi quadrati. Si veda la figura 3.8.

Per il calcolo della scala dell'errore, possiamo ricorrere all'algoritmo MUSE (Minimum Unbiased Scale Estimator) [46]. L'algoritmo valuta quale sia la scala dell'errore avendo a disposizione tutte le distanze tra immagine e modello calcolate. Come passo preliminare, tutte le distanze vengono ordinate in modo crescente. L'idea chiave è di calcolare la deviazione standard considerando solo gli n valori più piccoli. Nel momento in cui si hanno solo inliers, al crescere di n la deviazione standard sarà decrescente, in quanto abbiamo delle stime via via migliori considerando più punti. Quando però si includeranno gli outliers, la stima della deviazione standard inizierà a crescere, in quando si stanno inserendo fonti di errore nella valutazione. Prendendo quindi il minimo dei valori ottenuti al variare di n si ottiene una stima della deviazione standard robusta alla presenza di outliers (si veda figura 3.9).

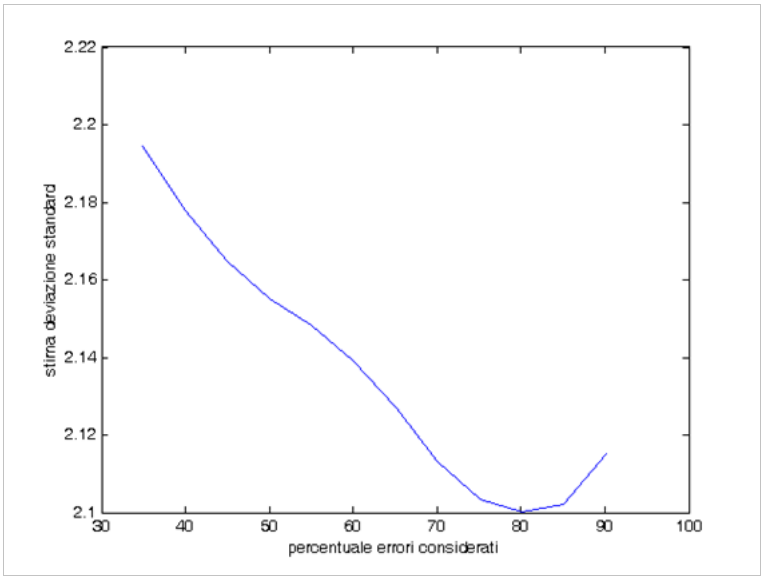


Figura 3.9: Valutazione della deviazione standard aumentando il numero di campioni. Solo l'80% dei punti considerati sono inliers.

La procedura di minimizzazione è stata effettuata utilizzando il metodo di Levenberg-Marquardt. Al fine di facilitare la convergenza dell'algoritmo, è possibile considerare che, nel caso in esame, l'inclinazione nel piano X-Y del contenitore è piccola. Possiamo quindi prima valutare la posa migliore facendo variare solamente i parametri di traslazione e di rotazione nell'asse Z. Ottenuta una posa stabile, possiamo quindi raffinare ulteriormente la posizione includendo anche i rimanenti parametri nel processo di minimizzazione.

3.4 Estensioni

Nella sezione corrente valuteremo le variazioni necessarie per l'algoritmo descritto nella sezione 3.3 in due situazioni distinte. In primo luogo, nella sezione 3.4.1, valuteremo come sia possibile utilizzare le immagini provenienti da due telecamere per aumentare la precisione nella stima della posizione del prodotto. Nella sezione 3.4.2 valuteremo invece come estendere l'algoritmo per permettere la rilevazione di oggetti non planari.

3.4.1 Visione stereo

Considerando di avere a disposizione l'immagine della scena proveniente da due telecamere in posizioni distinte, entrambe opportunamente calibrate in modo da condividere lo stesso sistema di riferimento per la scena, possiamo utilizzare le informazioni provenienti da entrambe le telecamere per rendere più precisa la stima della posizione del prodotto.

Il modo più semplice è di intervenire sul solo processo di minimizzazione (cfr. sezione 3.3.4). Per fare questo, è in primo luogo necessario determinare quali delle trasformazioni stimate dall'esecuzione dell'algoritmo di matching eseguito indipendentemente sulle due immagini rappresentino la stessa istanza del modello nella scena. Per poter confrontare le stime è necessario innanzitutto calcolare la roto-traslazione che corrisponde a ciascuna delle similitudini trovate mediante la formula 3.14. Effettuando il clustering descritto nella sezione 3.3.2 possiamo quindi ottenere le roto-traslazioni rigide a cui si suppone sia soggetto il modello, per poi eseguire il processo di minimizzazione sommando i contributi di errore provenienti da entrambe le immagini. Al fine di ridurre il numero di pose da vagliare, è possibile anche preventivamente proiettare le ipotesi nelle due immagini e quindi valutare la funzione di similitudine data dall'equazione 3.9. Se la funzione di similitudine non ottiene almeno un valore s_{min} in entrambe le immagini, la posa considerata può essere scartata.

3.4.2 Estensione al riconoscimento di oggetti non planari

L'idea di base per l'estensione dell'algoritmo per il riconoscimento di prodotto di forma arbitraria consiste nella generazione di una serie di viste 2D dell'oggetto per le quali viene utilizzato l'algoritmo proposto per la stima della posizione nell'immagine. Una volta ottenuta la stima migliore ed la rispettiva vista utilizzata, è possibile determinare la posizione nello spazio 3D dell'oggetto. In letteratura sono presenti diversi lavori che descrivono come sia possibile utilizzare algoritmo per la determinazione della posa di oggetti planari per oggetti non planari [63] [34] [18] [15] [25].

Il primo problema da affrontare risulta essere la creazione delle viste 2D del prodotto. Per fare questo, è necessario considerare la proiezione dell'oggetto nel campo visivo della telecamera in ogni sua possibile orientazione. In [65], il nu-

mero di possibili viste viene ridotto considerando che l'oggetto dispone di solo alcune posizioni stabili, in quanto appoggiato su un nastro trasportatore. Questo non è possibile nel nostro caso, in quanto l'oggetto può assumere orientazione arbitraria all'interno del contenitore. Nel caso più generale, è necessario considerare la posizione dell'oggetto per ogni rotazione rispetto al suo centro e per ogni sua posizione all'interno del contenitore. In prima istanza è d'altra parte possibile ignorare la posizione all'interno del contenitore, assumendo che la deformazione prospettica data dallo spostamento del prodotto dall'asse ottico della telecamera sia trascurabile. Questo è vero quanto più la telecamera è distante dal contenitore. Possiamo quindi considerare solamente la rotazione dell'oggetto rispetto al suo centro, ottenendo quindi delle viste dipendenti solo dai parametri di rotazione (r_x, r_y, r_z). I valori di rotazione vengono quindi discretizzati in p elementi ciascuno. Si ottengono in questo modo p^3 possibili viste per l'oggetto. Considerando che nel caso in esame l'algoritmo proposto permette il rilevamento dell'oggetto pur non considerando inizialmente la rotazione del prodotto di 30° nel piano X-Y, possiamo discretizzare la rotazione in 12 posizioni per ogni asse di rotazione, ottenendo in questo modo 1728 viste 2D dell'oggetto. Dato il numero elevato di viste, eseguire l'intero algoritmo considerando ogni vista singolarmente conduce al un incremento sostanziale del tempo di esecuzione. Per ridurre il numero di viste è possibile effettuare un raggruppamento gerarchico ad albero delle stesse. Le viste rappresentano inizialmente le foglie dell'albero. Quando due nodi contengono viste che hanno modelli simili tra loro, questi vengono raggruppati da un unico nodo padre, il quale contiene una delle viste scelta per rappresentare la coppia. Il processo può continuare su più livelli. Nella fase di riconoscimento, si parte della radice dell'albero. Quando un modello associato ad una determinata vista viene trovato nell'immagine, vengono vagliate anche le viste figlie, fino ad arrivare alle foglie dell'albero [63].

I passi dell'algoritmo di riconoscimento sono:

- Estrazioni delle features all'interno dell'immagine. Si noti che il numero di possibili angoli tra i segmenti è ora molto maggiore che nel caso planare, dato che l'incrocio tra due lati nello spazio tridimensionale può formare un angolo arbitrario nella vista bidimensionale.

- Per ogni vista 2D:

  - Matching tra il modello dato dalla vista considerata con l'immagine.
  - Clustering dei risultati nello spazio 4-dimensionale della trasformazione di similitudine
  - Calcolo delle rispettive pose 3D dell'oggetto nel sistema di riferimento della scena.

- Clustering nello spazio 6-dimensionale delle roto-traslazione 3D per la determinazione delle pose candidate.

- Esecuzione dell'algoritmo di minimizzazione per le pose candidate raccolte nel passo precedente.

Dal lato algoritmico, l'ultimo problema da risolvere consiste nella creazione delle viste 2D a partire dal prodotto. Per fare questo è necessario ricorrere ad un algoritmo che consenta l'eliminazione delle entità non visibili. In letteratura sono presenti diversi approcci per la risoluzione del problema, noto come *hidden line removal* [49]. Seppur utilizzando questi tipi di algoritmi, si pone ancora il problema di determinare se un determina lato presente nel modello 3D può generare un edge all'interno dell'immagine. Perché questo si possa verificare, è necessario che le facce incidenti al lato nel modello 3D abbiamo un'orientazione relativa maggiore di una determinata soglia impostata. All'aumentare di questa soglia viene ridotto il numero di entità considerate nel modello associato alla vista 2D, diminuendo in questo modo la complessità computazionale dell'algoritmo [63].

Capitolo 4

Framework software

Ponendosi l'obiettivo di creare un framework che possa essere utilizzato nei diversi contesti della visione computazionale, nel presente capitolo verrà descritto come sia possibile ottenere uno schema di sviluppo software che consenta di ottenere caratteristiche quali la riusabilità del codice, la modularità, la scalabilità e l'interoperabilità tra le diverse tecnologie, mantenendo d'altra parte enfasi sull'efficienza computazionale del sistema finale.

Nella sezione 4.1 descriveremo le motivazioni che hanno condotto allo sviluppo di un nuovo framework, mentre la più formale raccolta dei requisiti per i sistemi di visione considerati verrà descritta nella sezione 4.2. Passeremo quindi alla trattazione dell'architettura proposta nella sezione 4.3, per poi descrivere il framework nella sezione 4.4. Infine, nella sezione 4.5, vedremo alcuni esempi di utilizzo.

4.1 Motivazioni

La nascita di librerie software nel campo della visione computazionale è certamente indice del fatto che si sia considerato seriamente come affrontare le problematiche relative alla riusabilità e l'efficienza del codice, fornendo una serie di strutture dati ed algoritmi standard. In questo senso, l'introduzione della libreria open-source Intel OpenCV [4] rappresenta certamente un importante pietra miliare per la visione computazione in ambito accademico, così come le diverse alternative commerciali, tra le quali possiamo citare le MIL (Matrox Image Li-

brary) [5], sono i punti di riferimento per lo sviluppo dei sistemi di visione in ambito industriale.

In contrasto alla maturità di questi strumenti, pochi sforzi sono stati condotti per applicare i più classici principi dell'ingegneria del software nei sistemi di visione computazionale. L'architettura dei sistemi di visione è sempre passata in secondo piano rispetto allo sviluppo degli algoritmi in esso contenuti. D'altra parte, affrontando queste problematiche si ottengono tre benefici principali. In primo luogo, l'utilizzo di metodologie che consentano di standardizzare il processo di creazione del software permettono di risolvere il problema di riutilizzo delle soluzioni prodotte in contesti anche molto diversi tra loro. In secondo luogo, fornendo soluzioni standard a problemi come le modularità, la scalabilità della soluzione e l'interoperabilità tra le diverse tecnologie non si può che migliorare l'efficienza a lungo termine del processo di sviluppo. Terzo, l'utilizzo di una piattaforma di sviluppo condivisa permette di introdurre funzionalità avanzate quali la gestione della comunicazione interprocesso e l'esecuzione parallela degli algoritmi che possono essere semplicemente integrate nelle diverse applicazioni sviluppate. Grazie a questo, è possibile migliorare semplicemente l'efficienza computazionale del sistema finale senza gravare sullo sviluppo dell'applicazione specifica.

Al fine di ottenere i benefici sopra elencati, si è deciso di sviluppare un framework che consenta di migliorare il processo di sviluppo software. L'obiettivo finale è quello di ottenere un sistema di semplice utilizzo grazie al quale lo sviluppatore possa creare librerie di componenti software riutilizzabili. Grazie a questo approccio, l'effort necessario allo sviluppo di un nuovo sistema di riduce, in quanto consisterà nella composizione dei moduli già disponibili e testati per le attività più comuni, permettendo allo sviluppatore di concentrarsi sullo sviluppo dei nuovi algoritmi, che a loro volta verranno standardizzati come nuovi moduli al termine dello sviluppo.

Durante la progettazione ci si è posti l'obiettivo di creare un sistema minimale e non invasivo, che permetta un rapido apprendimento da parte dello sviluppatore e una semplicità di integrazione del codice già sviluppato all'interno del nuovo framework. Per ottenere questo, l'approccio scelto è stato quello di garantire l'integrazione delle applicazioni già sviluppate ponendo meno vincoli possibili. Al livello più semplice di utilizzo deve essere possibile eseguire le proprie appli-

cazioni all'interno del framework mediante la creazione di semplici wrapper o copia/incolla del codice. In questo modo non si sfruttano tutte le potenzialità offerte, ma d'altra parte si permette un immediato ingresso all'interno del nuovo sistema. Man mano che la conoscenza degli strumenti offerti aumenta, è quindi possibile modificare la propria applicazione per sfruttare le funzionalità offerte. Questo può significare l'inglobare parte delle funzionalità all'interno di moduli separati oppure sfruttare gli strumenti di temporizzazione o esecuzione parallela offerti. Parallelamente a questo, il framework deve proporre delle interfacce che nascondano le complessità intrinseche del sistema. Da un lato, dove possibile, è necessario privilegiare la configurazione dell'applicazione mediante interfaccia grafica o in prima istanza file di configurazione. Dall'altro, il processo di creazione dei componenti deve essere di semplice utilizzo e ben documentato, in maniera da indurre l'utente a standardizzare il codice prodotto in nuovi componenti che possano essere riutilizzabili all'interno di altre applicazioni.

4.2 Raccolta dei requisiti

Al fine di condurre una raccolta dei requisiti che comprendesse diversi ambiti applicativi della visione computazione, sono stati considerati i seguenti casi d'uso:

- Tracking mono telecamera. L'applicazione consente di tracciare il movimento degli oggetti all'interno dei frame consecutivi acquisiti da una singola telecamera.

- Tracking mediante OmniDome. L'OmniDome è un sistema composto dall'accoppiamento di una telecamera omnidirezionale ed una telecamera dome. Nelle applicazione di tracking, la telecamera omnidirezionale viene utilizzata per rilevare gli spostamenti degli oggetti nella scena. Queste informazioni vengono utilizzate per il controllo della telecamera dome, la quale si occupa di ottenere immagini ad alta risoluzione degli oggetti rilevati. Le immagini acquisite possono quindi essere utilizzate per l'esecuzione di altri algoritmi, come ad esempio la lettura di targhe o la classificazione degli oggetti, oppure più semplicemente per la visualizzazione del flusso video.

- Sistema di controllo qualità multi-telecamera. Nei sistemi di controllo qualità che impiegano più telecamere l'oggetto deve essere ripreso da più punti

di vista distinti. In questo caso è di fondamentale importanza la temporizzazione dell'acquisizione e sono solitamente forniti vincoli sulla durata dell'esecuzione degli algoritmi, in modo da assicurare che l'elaborazione termini prima che il prossimo oggetto da controllare sia disponibile.

- Sistema di bin picking. Rappresenta il sistema descritto nella presente tesi. L'esecuzione dell'algoritmo di visione viene innescata dalla ricezione di un evento fornito dalla cella di lavoro e il risultato ottenuto deve essere utilizzato per il controllo del robot manipolatore.

Grazie ad una prima analisi delle applicazioni descritte, possiamo determinare i seguenti requisiti funzionali che il framework deve soddisfare:

RF1. Temporizzazione dell'esecuzione. La temporizzazione delle applicazioni può essere gestita dal framework oppure attraverso la logica fornita dall'applicazione stessa. Sono state individuate le seguenti strategie di esecuzione:

- Esecuzione singola: la temporizzazione dell'esecuzione viene gestita dall'applicazione stessa. Il framework non deve far altro che avviare la prima esecuzione.
- Esecuzione continua: la temporizzazione di ogni singolo ciclo di esecuzione viene gestita dall'applicazione stessa. Il framework si limita a chiamare ripetutamente l'esecuzione dell'applicazione non appena il ciclo precedente termina.
- Esecuzione periodica: la temporizzazione dell'applicazione viene gestita dal framework, il quale si occupa di innescare l'esecuzione ad intervalli regolari. Si noti che in questo caso va definita la politica da adottare nel caso che l'esecuzione occupi più in un periodo.
- Esecuzione su richiesta: l'esecuzione viene innescata mediante richiesta di un agente esterno al framework. Questa modalità è particolarmente utile nei casi in cui l'esecuzione debba essere controllata da un dispositivo esterno, come nel caso del sistema di bin picking.

RF2. Esecuzione multi-tasking: il framework deve permettere l'esecuzione di più applicazioni contemporaneamente.

RF3. Esecuzione multi-threading: deve essere possibile specificare quali parti delle applicazioni debbano essere eseguite in parallelo.

RF4. Gestione politiche di esecuzione: devono essere previste diverse politiche di esecuzione delle applicazioni. Le politiche riguardano in particolare la possibilità di specificare tempi limite per l'esecuzione oppure di definire la priorità di esecuzione delle diverse parti dell'applicazione.

RF5. Gestione comunicazione: devono essere predisposte le opportune procedure per la comunicazione tra le diverse parti che compongono l'applicazione. Volendo porre l'attenzione sull'efficienza computazionale, al fine di evitare costose operazioni di copia si è scelta una strategia di comunicazione basata su memoria condivisa. Durante l'implementazione è necessario riservare particolare attenzione all'accesso concorrente ai dati.

RF6. Sistema distribuito: un architettura che consenta di effettuare chiamate remote potrebbe consentire l'utilizzo del framework anche in applicazioni distribuite. Questo requisito si traduce nella presenza di un server che accetti comandi provenienti da client remoti e nella definizione di procedure per la serializzazione e la deserializzazione dei parametri di configurazione e dei risultati.

Forte delle motivazioni descritte nella sezione 4.1, il framework deve soddisfare inoltre i seguenti requisiti non funzionali:

RNF1. Architettura basata su componenti: il framework deve consentire la creazione di componenti con interfaccia ben definita. Questo approccio consente di ottenere un buon grado di separazione tra le diverse funzionalità dell'applicazione, garantendo in questo modo una minore interdipendenza tra le stesse. Così facendo, la possibilità di utilizzo delle soluzioni sviluppate in ambiti anche molto differenti tra loro aumenta. La creazione delle applicazione si traduce quindi nella definizione di come i componenti vengano eseguiti e come essi comunichino tra loro.

RNF2. Caricamento dinamico: dato che si avrà nel tempo una crescita continua dei componenti sviluppati, una soluzione monolitica non è applicabile. E' necessario quindi predisporre le opportune funzionalità che consentano il caricamento dinamico dei componenti.

RNF3. Semplicità di utilizzo: le modalità di creazione di nuovi componenti o l'organizzazione degli stessi per la creazione delle applicazioni deve essere molto semplice e ben documentata al fine di facilitare l'ingresso ai nuovi utenti del framework.

RNF4. Semplicità di descrizione delle applicazioni: al fine di facilitare le fasi di analisi e documentazione delle applicazioni, deve essere previsto un linguaggio visuale basato su diagrammi che consenta di descrivere come i diversi componenti interagiscono tra di loro.

RNF5. Indipendenza dalla piattaforma: il codice prodotto deve poter essere eseguito sulle diverse piattaforme di riferimento, al momento attuale Windows e Linux.

RNF6. Interoperabilità: il framework deve garantire l'interoperabilità di componenti basati su tecnologie differenti. Ad esempio, potrebbe risultare necessario far comunicare componenti basati su OpenCV con componenti basati sulle MIL.

RNF7. Scalabilità. Il framework deve facilitare la creazione di applicazioni scalabili, in particolar modo riguardo al numero di processori utilizzati per l'elaborazione degli algoritmi. Questo si traduce nel porre enfasi sulla facilità di creazione di soluzioni multi-threading e multi-tasking, requisiti funzionali già individuati per il sistema.

4.3 Stile architetturale

Il più comune stile architetturale per la creazione di applicazioni che debbano elaborare uno stream di informazioni, come può essere un flusso video nelle applicazioni di visione, è dato dal modello detto *Pipes and Filters*. L'idea base del modello è quella di racchiudere i diversi passi di elaborazione all'interno di componenti, detti filtri. I connettori tra i diversi componenti sono detti *pipes*. Se un *pipe* collega due filtri, l'uscita del primo diventa l'ingresso del secondo. Seguendo lo stile descritto, i filtri sono costruiti in modo da essere indipendenti tra loro: essi non condividono informazioni di stato, ma l'unica comunicazione avviene attraverso i *pipes*.

Il modello *pipes and filters* permette di ottenere un elevato grado di indipendenza tra i componenti, consentendo in questo modo un semplice riutilizzo del codice prodotto e facilitando il mantenimento e l'evoluzione del software. Inoltre, esso permette implicitamente un esecuzione parallela e distribuita dei diversi componenti prodotti. Lo stile architetturale presenta d'altra parte alcune problematiche. In primo luogo, dato lo stile di esecuzione sequenziale in ingresso ai filtri, si possono creare dei colli di bottiglia nel sistema. Il filtro non è difatti in grado di gestire il nuovo flusso dati prima che l'elaborazione del precedente sia completata. Il tempo di esecuzione dell'intero sistema è quindi dato dalla velocità del filtro più lento. In secondo luogo, il modello prevede che i dati vengano copiati tra un filtro e il successivo. Questo comporta un decremento delle prestazioni del sistema a causa delle costose operazioni di copia. Inoltre, il modello non definisce nessuna modalità per mantenere delle corrispondenze tra i diversi flussi dati che seguono percorsi di esecuzione distinti, ad esempio per permettere l'unificazione delle informazioni al fine di creare un uscita combinata. Infine, i parametri dei diversi componenti non vengono modellati in modo coerente al flusso dei dati. Si devono quindi predisporre funzionalità esterne al sistema stesso per la configurazione degli algoritmi. D'altra parte, non vi è nessun modo per definire parametri per un algoritmo che dipendano dall'esecuzione di altri algoritmi, escludendo quindi la possibilità di includere cicli di feed-back all'interno del sistema.

Il modello architetturale proposto per il framework consente di risolvere le problematiche evidenziate per il modello *pipes and filters* mantenendo comunque i suoi vantaggi. I componenti, detti *moduli applicativi*, rappresentano i centri di elaborazione dell'applicazione. Ogni modulo applicativo è identificato attraverso un nome univoco e viene definito logicamente attraverso i suoi dati in ingresso, parametri e dati in uscita. I moduli vengono connessi tra loro in modo da definire il flusso di esecuzione dell'applicazione. Al termine dell'esecuzione di un modulo, viene innescata parallelamente l'esecuzione per tutti i moduli ad esso collegati. Si noti che, a differenza di quanto avviene per il modello *pipes and filters*, i connettori non definiscono un flusso di dati ma solo di esecuzione. Non viene posto nessun vincolo sul numero di connessioni entranti o uscenti per un modulo, ottenendo quindi un grafo orientato di esecuzione. I moduli che non hanno archi entranti nel grafo vengono detti moduli sorgente. Si noti che vi possono essere più nodi con questa caratteristica all'interno del sistema.

I diversi moduli applicativi devono poter leggere i dati in ingresso ed i parametri e scrivere i dati in uscita. Per evitare costose operazioni di copia, si è deciso di utilizzare un modello di comunicazione basato su memoria condivisa. Inoltre, si è deciso di suddividere i dati in due repository distinti in base alla durata del loro ciclo di vita:

- *Session Context*: contiene i dati persistenti che sono necessari per l'intera esecuzione dell'applicazione. Possono essere i dati di configurazione utilizzati dai moduli oppure informazioni che vengono raccolte durante le varie esecuzioni degli algoritmi.

- *Execution Context*: contiene i dati volatili che sono necessari ad un singolo ciclo di esecuzione dei moduli contenuti nell'applicazione. La creazione dell'Execution Context avviene contestualmente all'invocazione il rispettivo modulo sorgente, mentre la memoria allocata viene rilasciata nel momento in cui l'esecuzione dell'ultimo modulo della catena termina.

Il modello previsto permette di risolvere le varie problematiche del modello *pipes and filter*. In primo luogo, non viene posto nessun vincolo per l'esecuzione concorrente dello stesso modulo, in quanto i dati sono modellati in modo separato dalle connessioni. In questo modo si evita la creazione di colli di bottiglia. In secondo luogo, i dati non vengono copiati ma sono mantenuti in una memoria condivisa tra i vari componenti. Inoltre, è semplice definire corrispondenze tra i diversi flussi dati, in quanto essi sono contenuti in repository condivisi all'interno dello stesso ciclo di esecuzione. Infine, la presenza di una memoria condivisa permette da un lato di trattare uniformemente i dati di ingresso con i dati di configurazione, dall'altro permette di definire dei cicli di feed-back.

4.3.1 Rappresentazione grafica

Risulta molto utile, sia in fase di analisi che di documentazione di un'applicazione, ricorrere all'utilizzo di una rappresentazione grafica che descriva come i componenti interagiscano tra di loro. A tal fine, è stato definito un linguaggio visuale composto dai seguenti simboli grafici:

- Un modulo viene rappresentato attraverso un quadrato etichettato in basso con il nome del modulo stesso.

- Una freccia orientata tra due moduli indica che l'esecuzione di un modulo è innescata dall'altro.

- Un etichetta nel seguente formato indica una risorsa:
  $[sorgente] : [nome] < [tipo] >$
  dove $[sorgente]$ può essere SC per il *Session Context* o EC per l'*Execution Context*. Il tipo può essere omesso se risulta chiaro dal contesto.

- Una freccia orientata tra un modulo e una risorsa denota che quel modulo legge oppure scrive su quella risorsa, in base al verso della freccia.

- Una freccia entrante verso un modulo che non proviene da un altro modulo o da una risorsa denota che l'esecuzione del modulo viene gestita direttamente dall'applicazione. L'etichetta della freccia indica quale strategia di esecuzione venga utilizzata per quel determinato modulo sorgente (si veda 4.4.2 per un elenco delle modalità di esecuzione gestite).

- Un insieme di moduli interconnessi tra loro in modo da definire un sottosistema viene rappresentato attraverso una coppia di quadrati parzialmente sovrapposti tra loro. Si noti che questa descrizione non è completa in quanto non vengono fornite informazioni dettagliate su come gli altri moduli o risorse siano collegati agli elementi del sottosistema. Queste informazioni devono essere chiare dal contesto oppure fornite in modo separato.

Si noti che, per semplificare la rappresentazione, non è necessario visualizzare tutte le connessioni tra risorse e moduli che le utilizzano. Come regola generale, è sufficiente inserire nel diagramma solo la connessione tra la risorsa e il modulo che l'ha prodotta, mentre è possibile evitare di riportare tutti i suoi utilizzatori. Nella sezione 4.5 vedremo diversi esempi di utilizzo del diagramma descritto.

4.4 Il framework

L'esecuzione delle applicazioni all'interno del framework si basa sull'iterazione tra tre elementi principali: *Application Server*, *Application* e *Application Module*. L'*Application Server* si prende carico di gestire l'esecuzione delle applicazioni e di fornire un unica interfaccia per il controllo delle stesse dall'esterno del sistema. *Application* si occupa a sua volta di gestire l'esecuzione dei moduli in essa

contenuti, in particolare temporizzando l'esecuzione e fornendo gli strumenti di comunicazione. Infine, gli *Application Module* rappresentano l'unità di elaborazione del sistema. Ogni *Application Module* definisce i suoi input, i suoi output e i diversi parametri di configurazione. Infine, il framework definisce le modalità di comunicazione dei dati tra i vari componenti mediante l'utilizzo di due dizionari detti *Session Context* e *Execution Context*.

Nelle seguenti sotto sezioni tratteremo nel dettaglio i componenti principali del sistema.

4.4.1 Application Server

L'*Application Server* ha tre funzionalità principali: gestire l'esecuzione delle applicazioni, permettere la loro configurazione e consentire la lettura dei risultati. Al fine di consentire l'accesso remoto a queste funzionalità (si veda RF-6), l'*Application Server* deve essere implementato in modo da consentire la comunicazione attraverso rete, ad esempio utilizzando un socket TCP. Si noti che questo impone la definizione di un linguaggio per impartire i comandi di esecuzione e di configurazione, nonché la necessità di salvare su stream i dati di configurazione o i risultati. In entrambi i casi si è deciso di optare per il formato XML per lo scambio dei dati. L'overhead fornito dalla serializzazione in XML è giustificato da una maggiore semplicità nella definizione delle specifiche. E' necessario porre attenzione sul fatto che i dati che transitano dall'*Application Server* sono informazioni di controllo, non flussi complessi come potrebbero essere le immagini acquisite, per i quali è necessario prevedere altre metodologie di comunicazione.

L'*Application Server* dispone delle seguenti funzionalità:

- Aggiungere nuove applicazioni o rimuovere quelle esistenti.
- Avviare o fermare l'esecuzione delle applicazioni.
- Monitorare lo stato delle applicazioni presenti all'interno del server.
- Gestire la configurazione delle applicazioni presenti.
- Permettere la lettura e la scrittura da parte di un agente esterno delle informazioni presenti all'interno dei *Session Context* delle diverse applicazioni.

4.4.2 Application

Il componente *Application* rappresenta l'applicazione attualmente in esecuzione all'interno del server. *Application* fornisce tre funzionalità principali: gestione della comunicazione, della composizione e dell'esecuzione dei moduli contenuti all'interno del sistema.

La comunicazione tra i diversi *Application Module* avviene con due modalità. In primo luogo, durante la fase di configurazione dell'applicazione, vengono definite delle relazioni padre-figlio tra i diversi moduli. Quando l'esecuzione del padre termina, questa scatena l'esecuzione di tutti gli algoritmi definiti come figlio, ottenendo in questo modo uno schema di esecuzione asincrona tra i diversi componenti. L'esecuzione di ogni figlio avverrà su un thread distinto, garantendo in questo modo un implicita struttura multi-threading per l'intera applicazione. In secondo luogo, è prevista una metodologia per la comunicazione dei dati e dei parametri di configurazione tra i diversi moduli che compongono l'applicazione. Come detto, si è deciso di optare per una strategia di comunicazione basata su memoria condivisa. Si noti che questo implica la necessità di predisporre gli opportuni meccanismi di locking per evitare che più moduli modifichino contemporaneamente le stesse informazioni. I dati vengono organizzati in dizionari, ai quali è possibile accedere utilizzando come chiave una stringa. Sono presenti due dizionari, il *Session Context* e il *Execution Context*. Questa organizzazione consente di demandare al framework il rilascio delle risorse non più necessarie al termine di un ciclo di esecuzione dell'applicazione. Si noti che, in base alle politiche di esecuzione, possono essere presenti in memoria più istanze di *Execution Context*, mentre in ogni caso sarà presente solo un istanza di *Session Context*.

La gestione della composizione dei moduli e delle loro dipendenze può avvenire dall'esterno del sistema, interagendo con l'*Application Server* mediante l'utilizzo di un interfaccia grafica predisposta per lo scopo, oppure attraverso l'utilizzo di file di configurazione. In fase di configurazione, quando un modulo viene aggiunto all'interno del sistema, vengono definite due informazioni. In primo luogo viene specificato come collegare il modulo alle risorse ad esso necessarie. Per ogni dato deve essere specificata la sorgente (*Execution Context* o *Session Context*) e la relativa chiave del dizionario. Per semplificare la configurazione, i moduli definiscono i valori di default per questi parametri. Oltre alla definizione dei dati, dobbiamo configurare come gli algoritmi vengano eseguiti. Un modulo può

essere eseguito su richiesta dell'applicazione oppure la sua esecuzione può essere innescata dal termine dell'esecuzione di uno degli altri moduli. Se il modulo viene eseguito su richiesta dell'applicazione, abbiamo diverse politiche per la sua esecuzione:

- Extern: l'esecuzione dell'algoritmo viene determinata dalla ricezione di un comando da parte dell'applicazione dall'esterno, passando attraverso l'*Application Server*. Questa politica di esecuzione è utile nel caso l'esecuzione dell'applicazione dipenda da un evento esterno (ad esempio su segnale di I/O in applicazioni di automazione industriale).

- Run-Once: l'esecuzione dell'algoritmo avviene in fase di avviamento del server. Questa politica è utile nel caso sia il modulo stesso a gestire la temporizzazione.

- Loop: l'esecuzione del modulo viene posta all'interno di un ciclo, in modo tale che l'esecuzione successiva inizi non appena quella corrente è terminata. Questa politica può essere utile ad esempio per l'acquisizione da telecamera nel caso non si voglia perdere un singolo frame. Il modulo viene eseguito continuamente ed esso si mette in attesa del frame sul device della telecamera. Non appena riceve il frame, questo viene salvato nell'output e l'esecuzione del modulo termina. In modalità *loop*, una nuova esecuzione del modulo ha subito inizio, in modo da potersi mettere in attesa sul device per il frame successivo. E' prevista la possibilità di definire un timeout di esecuzione degli algoritmi, in modo da soddisfare eventuali vincoli soft real-time dell'applicazione.

- Loop-Wait: in modo simile alla politica *loop*, l'esecuzione è iterativa, ma in questo caso prima di eseguire nuovamente il modulo si attende che un altro modulo, tipicamente l'ultimo della catena, abbia terminato l'esecuzione. Questa politica è utile per far si che non vi siano esecuzioni parallele della stessa catena di algoritmi.

- Periodic: l'esecuzione del modulo è periodica, con periodo impostato dall'utente. Utilizzando questa politica è inoltre necessario specificare cosa accade quando l'esecuzione richiede più del periodo indicato. L'esecuzione

precedente può essere interrotta, si può ignorare l'esecuzione attuale oppure l'esecuzione attuale può essere eseguita in un altro thread. Si noti che quest'ultima condizione può portare ad un rapido esaurimento delle risorse computazionali.

Inoltre, per ogni modulo vengono definiti quali siano i moduli da eseguire al termine della sua esecuzione.

Ad ogni esecuzione di un modulo che venga eseguito direttamente dall'applicazione viene creato il rispettivo *Execution Context*, il quale viene successivamente passato ai moduli a valle. Al termine della catena di algoritmi, vengono rilasciate le risorse in esso contenute.

Si noti che è l'applicazione che definisce la struttura del grado di esecuzione, non il modulo stesso[1]. In questo modo viene garantito un maggior disaccoppiamento per i moduli, che si traduce in un maggior grado di riutilizzo degli stessi in contesti anche molto diversi tra loro.

4.4.3 Application Module

L'*Application Module* rappresenta l'unità di elaborazione dell'applicazione. Seguendo il paradigma di programmazione basato sui componenti, ogni modulo definisce esplicitamente i suoi input, output e parametri di configurazione. Inoltre, al fine di trattare uniformemente i flussi di controllo insieme ai flussi di elaborazione dati, ogni modulo può definire uno o più entry-point. Un entry-point effettua il collegamento tra un nome testuale ed una determinata funzione contenuta nell'*Application Module*. In questo modo è possibile richiedere, a partire dall'*Application Server*, l'esecuzione di particolari procedure definite nel modulo. Per vedere perché questo risulti necessario, consideriamo il caso di un modulo applicativo che gestisca l'acquisizione da web cam e la configurazione dei parametri di acquisizione. Il modulo ha quindi due flussi: il flusso di acquisizione e il flusso di controllo per la configurazione. Nonostante in questo semplice caso sia possibile integrare i due flussi, verificando ad esempio prima di ogni esecuzione se i parametri di configurazione siano cambiati, una soluzione più elegante potrebbe venir fornita del trattare distintamente i due flussi. Quando l'utente

[1]Un eccezione a questo è data dal modulo applicativo *List Dispatcher* che tratteremo nella sezione 4.5.1

vuole variare la configurazione mentre l'applicazione è in esecuzione, comunica con l'*Application Server* per invocare l'entry-point di configurazione, il quale scatenerà la chiamata alla funzione collegata. Si noti che l'esecuzione dell'elaborazione e della configurazione non può avvenire parallelamente. Sarà allora compito dell'*Application*, quando riceve la richiesta da parte dell'*Application Server*, di assicurarsi che l'esecuzione dell'acquisizione non sia in corso.

La definizione dell'interfaccia del modulo può avvenire in due modi: attraverso la definizione di un linguaggio IDL (Interface Definition Language), oppure attraverso l'utilizzo di opportune funzioni rese disponibili dal framework. La prima scelta è sicuramente la migliore, in quanto garantisce una maggiore separazione tra il modulo e la sua implementazione. D'altra parte, la definizione di funzioni rese disponibili allo sviluppatore è di più semplice apprendimento ed utilizzo.

Dal lato implementativo, l'*Application Module* è una classe astratta che deve essere implementata dallo sviluppatore. A tal fine, vengono resi disponibili nella classe base i seguenti metodi:

- `void Init()`: è il metodo chiamato dal framework durante l'inizializzazione del modulo.
- `void Process(ExecutionContext* ex)`: è l'entry-point principale del modulo.
- `void Finalize()`: è il metodo chiamato dal framework in fase di finalizzazione del modulo.

Sono inoltre rese disponibili le seguenti funzionalità che consentono la definizione dell'interfaccia del modulo:

- `void DefineInput(String name, String type)`: permette di definire un input specificando nome e tipo di dato. Dato che il framework verrà sviluppato in C++, il quale non dispone della *reflection*, l'associazione tra tipo di dato e classe che lo implementa deve essere fatto attraverso l'utilizzo del design pattern *Factory*. Deve essere lo sviluppatore stesso, in fase di sviluppo delle classi dati, a definire questa associazione mediante chiamata ad un'opportuna funzione.
- `void DefineOutput(String name, String type)`: permette di definire un output per l'algoritmo.

- `void DefineEntryPoint(String name, Function fref)`: permette di associare un determinato entry-point ad una funzione della classe.

4.4.4 Gestione dei dati condivisi

Il *Session Context* e *Execution Context* rappresentano le strutture dove vengano salvati i dati durante l'esecuzione degli algoritmi. I dati vengono organizzati in dizionari, ai quali è possibile utilizzando come chiava una stringa. Le classi devono consentire l'accesso ai dati in esse contenuti in modo concorrente da più thread. Per questa ragione è indispensabile predisporre gli opportuni meccanismi di locking delle risorse.

I dati contenuti nei dizionari devono essere accessibili anche da remoto. Per questo motivo, le strutture dati contenute devono avere le opportune funzioni di serializzazione e deserializzazione dei dati. A tal fine, si è deciso di definire una classe base, chiamata *StorableObject*, da cui tutti gli oggetti inseriti all'interno dei dizionari devono derivare, che definisce le funzioni di serializzazione e la deserializzazione su XML dei dati contenuti nell'oggetto. Inoltre, come abbiamo visto, la gestione del ciclo di vita degli oggetti contenuti nell'*Execution Context* viene gestito direttamente nel framework, rilasciando le risorse nel momento in cui un ciclo di esecuzione termina. Affinché questo sia possibile, è necessario che gli *StorableObject* si occupino anche di gestire la memoria per gli oggetti che inglobano, con modalità dipendenti dall'oggetto stesso.

4.5 Esempi applicativi

Nella sezione corrente vedremo come sia possibile strutturare le applicazioni utilizzando il framework proposto. In particolare, verranno presi come esempi i casi d'uso descritti nella sezione 4.2. L'obiettivo non è quello di trattare esaustivamente le applicazioni descritte, quanto più di evidenziare come problemi classici nel campo della visione possano essere risolti mediante l'utilizzo dell'approccio proposto.

4.5.1 Tracking mono-telecamera

L'obiettivo dell'applicazione è di tracciare il movimento degli oggetti rilevati all'interno delle immagini provenienti da una singola telecamera. In primo luogo è necessario determinare quali siano gli oggetti in movimento. Per fare questo possiamo ricorrere alle tecniche di *background subtraction*, le quali aggiornano nel tempo la rappresentazione del background della scena, ovvero delle parti della scena che rimangono immutate tra le diverse immagini, in modo da poter evidenziare quale sia il foreground, ovvero gli oggetti in movimento. Il foreground così ottenuto viene quindi segmentato, in modo da ottenere i diversi blob che rappresentano i diversi oggetti in movimento. Ognuno di essi viene quindi inseguito indipendentemente nei frame successivi in modo da identificare i suoi spostamenti.

Dalla breve discussione possiamo individuare i moduli che compongono il sistema:

- Image Acquisition: si occupa di acquisire l'immagine dalla telecamera.
- Background Subtraction: si occupa di mantenere il modello di background nel tempo e restituire l'immagine di foreground attuale.
- Blob Detection: segmenta l'immagine alla ricerca dei blob. Mantiene inoltre la lista dei blob su cui si sta eseguendo il tracking.
- Blob Tracker: traccia gli spostamenti di un determinato blob.

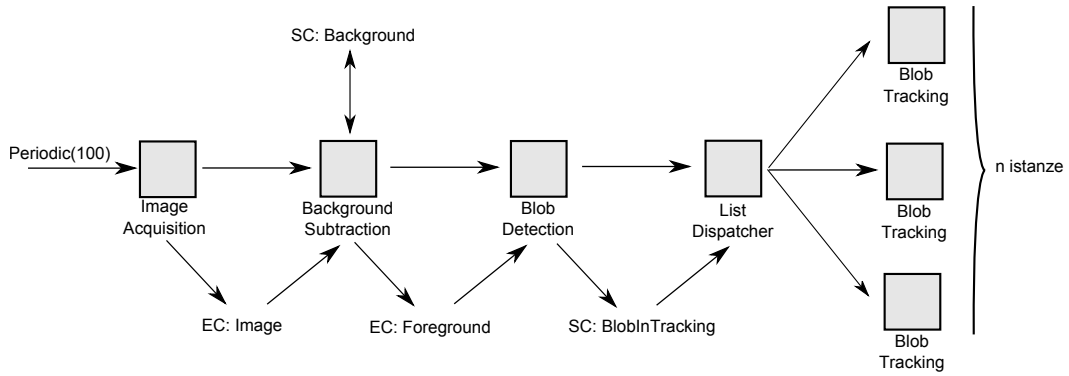


Figura 4.1: Diagramma dell'applicazione di tracking mono-telecamera.

In figura 4.1 è riportato il diagramma che rappresenta l'applicazione. L'esecuzione avviene ad una velocità di 10 fps all'interno di un thread periodico. Si

sarebbe potuto anche scegliere di demandare la velocità di esecuzione all'hardware, utilizzando la modalità di esecuzione *loop-wait*. Nel primo passo viene acquisita l'immagine che viene poi salvata all'interno dell'*Execution Context* con il nome di *Image*. Quindi, l'algoritmo di background subtraction viene eseguito per fornire come risultato l'immagine *Foreground* e nel contempo aggiornare il background contenuto nel *Session Context*. Si noti che è necessario che il background sia nel *Session Context*, altrimenti esso verrà eliminato al termine del ciclo di esecuzione. Si ha quindi il *Blob Detection*, il quale aggiorna la lista dei blob contenuti nel *Session Context*. A questo punto, dato che vogliamo che l'algoritmo di tracking venga eseguito in parallelo ma non conosciamo il numero di thread da eseguire, dobbiamo ricorrere al modulo di sistema *List Dispatcher*. Questo modulo si occupa di creare un pool di moduli che vengono richiamati per ogni elemento della lista fornita. Con questa strategia, il numero di thread su cui eseguire il tracking viene determinato in modo dinamico.

Supponiamo ora che il tracking venga effettuato con una procedura particolarmente efficiente che lavora sul foreground non segmentato, come potrebbe essere ad esempio un algoritmo di correlazione basato su SSD (Sum of Squared Differences). In questo caso, sarebbe opportuno che il tracking venga eseguito ad una velocità maggiore della più pesante procedura di blob detection. Questo può essere ottenuto riorganizzando i moduli come riportato in figura 4.2. Il blob detection viene ora eseguito ogni secondo, mentre il tracking è aggiornato ogni decimo. Si noti che il foreground, essendo necessario anche per il blob detection, è stato ora spostato nel *Session Context*. Solo configurando differentemente i moduli abbiamo ottenuto un applicazione che risponde a requisiti differenti.

4.5.2 Tracking attraverso OmniDome

L'OmniDome è un sistema composto dall'accoppiamento di una telecamera omnidirezionale ed una telecamera dome. Nelle applicazione di tracking, la telecamera omnidirezionale viene utilizzata per rilevare gli spostamenti degli oggetti nella scena. Queste informazioni vengono utilizzate per il controllo della telecamera dome, la quale si occupa di ottenere immagini ad alta risoluzione degli oggetti rilevati. Nell'esempio in esame, le immagini vengono utilizzate per determinare se l'oggetto in movimento è una persona utilizzando un algoritmo di face

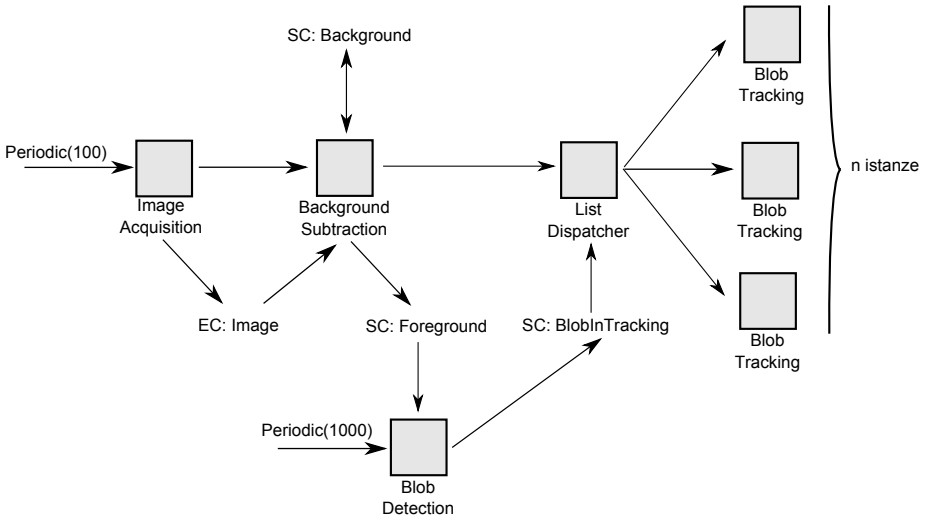


Figura 4.2: Nuova versione del diagramma dell'applicazione di tracking mono-telecamera.

detection. Il risultato così ottenuto viene riportato a schermo. In figura 4.3 è riportato il diagramma del sistema.

Anche in questo caso è stato possibile eseguire parallelamente l'elaborazione delle due immagini utilizzando periodi di esecuzione differenti. La telecamera PTZ nella configurazione scelta viene controllata ad ogni immagine acquisita dalla telecamera omnidirezionale. E' ad ogni modo possibile disaccoppiare i due algoritmi facendo gestire l'esecuzione del modulo *PTZ Commander* direttamente all'applicazione.

Durante l'utilizzo dell'applicazione può risultare utile variare la politica di controllo della telecamera dome. Ad esempio si può passare da una politica che suddivida il tempo equamente sui diversi blob individuati ad una politica che effettua il tracking del solo blob di dimensioni maggiori. Per garantire la corretta esecuzione di questo flusso di controllo, è stato previsto un ulteriore entry-point per il modulo *PTZ Commander* chiamato *ChangePolicy*. Quando invocato, la nuova politica di esecuzione viene letta dal *Session Context* ed applicata al modulo. Si noti che il framework stesso garantisce che il cambio di politica non avvenga in modo concorrente all'esecuzione dell'algoritmo di controllo del dome.

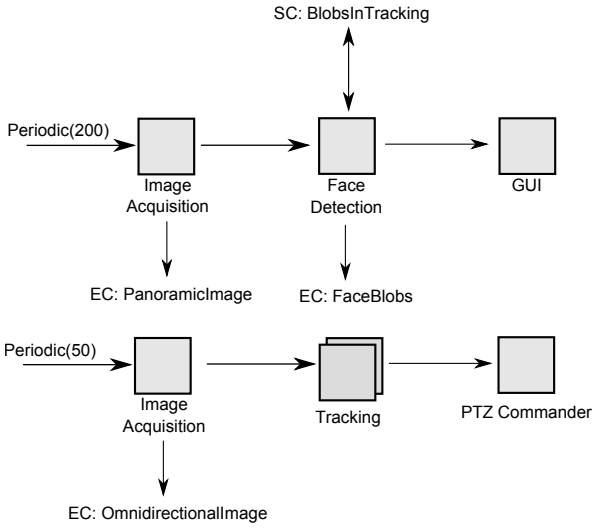


Figura 4.3: Diagramma dell'applicazione di tracking utilizzando la telecamera OmniDome.

4.5.3 Controllo qualità multi-telecamera

Nei sistemi di controllo qualità che impiegano più telecamere l'oggetto deve essere ripreso da più punti di vista distinti. In questo caso è di fondamentale importanza la temporizzazione dell'acquisizione e sono solitamente forniti vincoli soft real-time per gli algoritmi, in modo da assicurare che l'elaborazione termini prima che il prossimo oggetto da controllare sia disponibile.

Consideriamo il caso in cui stiamo osservando lo stesso oggetto da quattro punti di vista distinti. L'esecuzione dell'acquisizione è comandata da un trigger hardware che assicura che tutte le telecamere acquisiscano l'immagine nello stesso istante. Il segnale fornito viene utilizzato anche per comunicare all'applicazione di elaborare le immagini. E' necessario che l'intero processo termini prima di un secondo, ovvero prima che l'oggetto successivo sia presente nel campo visivo delle telecamere.

Al fine di sfruttare al meglio le risorse computazionali a disposizione, è auspicabile che l'esecuzione degli algoritmi di controllo qualità avvenga in parallelo. Il processo decisionale deve essere d'altra parte centralizzato, in quanto si suppone basarsi dal confronto delle elaborazioni eseguite. Con i vincoli descritti otteniamo il sistema rappresentato dal diagramma riportato in figura 4.4.

L'esecuzione su più thread viene garantita dal modulo di sistema *Dispatcher*,

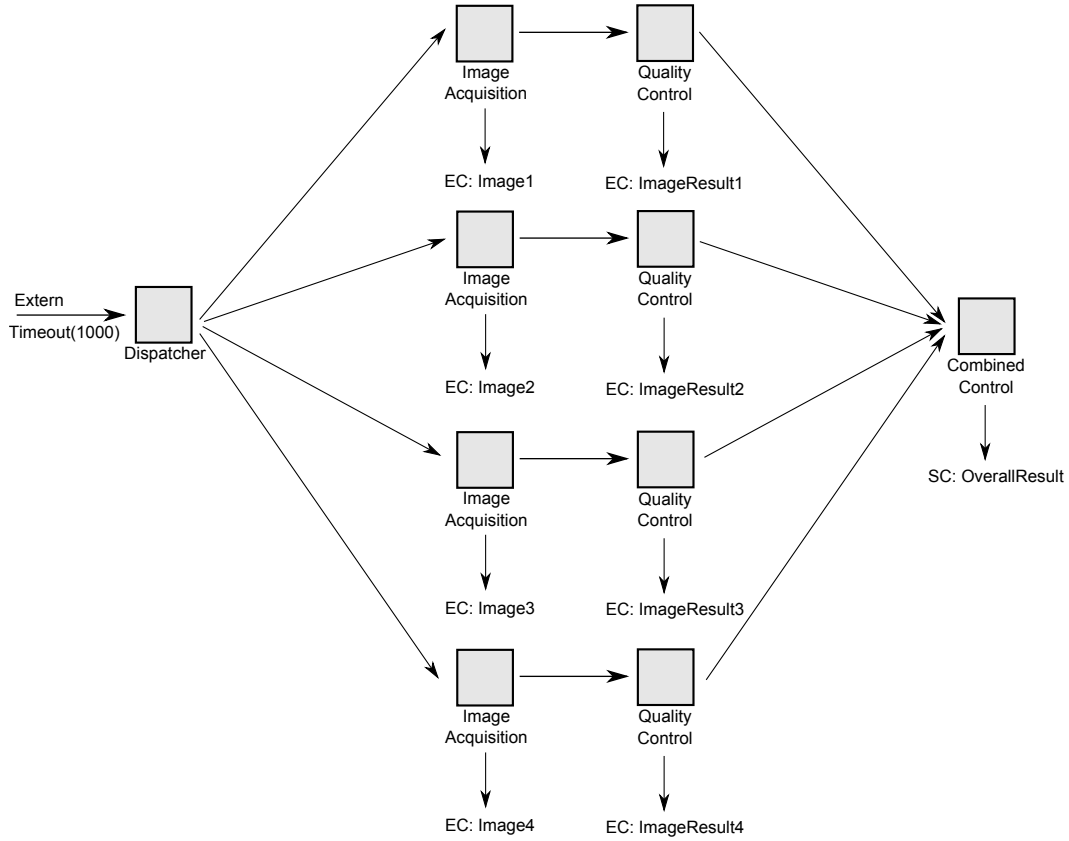


Figura 4.4: Diagramma dell'applicazione controllo qualità multi telecamera.

il quale si occupa di innescare i quattro algoritmi di acquisizione contemporaneamente. L'applicazione è composta da quattro blocchi identici di acquisizione delle immagini e controllo qualità. Ognuno dei blocchi è stato configurato in modo da operare su una telecamera distinta. Al termine di ognuno degli algoritmi di controllo qualità viene quindi eseguito un modulo di controllo combinato. Si noti che l'esecuzione del modulo viene scatenata quattro volte. Solo durante l'ultima chiamata si avranno però a disposizione i risultati di tutti e quattro gli algoritmi all'interno dell'*Execution Context*. L'esecuzione dell'algoritmo che effettua la combinazione dei diversi risultati può essere quindi posticipata a quando si hanno a disposizione tutte le informazioni necessarie.

L'esempio riportato ha evidenziato la situazione in cui l'esecuzione di un modulo venga innescata da più moduli differenti tra loro. In casi come questi l'unico modo che si ha per determinare quale sia il modulo chiamante si ha interrogando l'*Execution Context*, il quale fornisce appunto il contesto dell'esecuzione attuale.

4.5.4 Sistema di bin picking

Nel sistema di bin picking oggetto della presente tesi, come si è visto nella sezione 2.2.2, si desidera che il tempo di elaborazione delle immagini sia mascherato rispetto al movimento del robot. Questo si traduce in un disaccoppiamento tra l'algoritmo di elaborazione delle immagini e l'algoritmo che consente il controllo del robot. L'esecuzione viene gestita attraverso l'utilizzo di due segnali distinti generati dai dispositivi presenti nella cella di lavoro. Si ottiene in questo modo l'applicazione descritta dal diagramma riportato in figura 4.5.

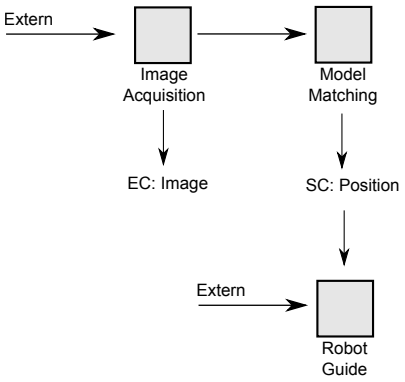


Figura 4.5: Diagramma dell'applicazione per il bin picking.

Nel caso si desideri utilizzare un sistema multi-telecamera, l'esecuzione degli algoritmi di model matching vanno eseguiti in parallelo sulle diverse immagini acquisite. In seguito i risultati vengono unificati per ottenere la stima della posizione del prodotto. Si ottiene in questo modo il sistema di figura 4.6, dove è stato riportato il solo sistema di visione.

Un altro vantaggio dell'approccio modulare proposto è dato dalla sostituibilità dei moduli applicativi. Nel momento in cui si hanno a disposizione più moduli che compiano la stessa mansione definendo quindi la stessa interfaccia è possibile sostituire a run-time i due moduli agendo solo sulla configurazione dell'applicazione.

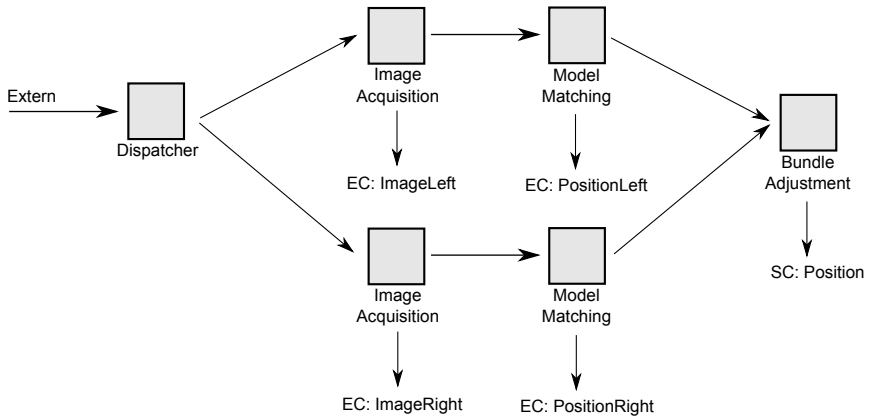


Figura 4.6: Diagramma dell'applicazione per il bin picking stereo.

Capitolo 5

Esperimenti

Durante lo studio sono stati condotti degli esperimenti atti a valutare le prestazioni dell'algoritmo esposto nel capitolo 3 per il caso in esame in confronto alle prestazioni fornite dagli algoritmi disponibili in letteratura. A tal fine sono stati implementati gli algoritmi Generalized Hough Transform (sezione 3.2.2) e l'algoritmo di Steger [60] basato sul Template Matching (sezione 3.2.1). Nella sezione 5.1 vedremo i dettagli implementativi di questi algoritmi. Il secondo passo per la valutazione delle prestazioni consiste nella raccolta delle immagini che formeranno il data set. La procedura sarà descritta nella sezione 5.2. In seguito, nella sezione 5.3 verrà dettagliata l'esecuzione dell'algoritmo proposto in una delle immagini di test, al fine di poter valutare i diversi passi dell'algoritmo in un caso concreto di utilizzo. Infine, nella sezione 5.4 saranno descritti quantitativamente e qualitativamente i risultati ottenuti.

5.1 Implementazione algoritmi

Per l'implementazione degli algoritmi è stato scelto Matlab. La scelta ha garantito una maggiore semplicità e velocità di programmazione e debugging degli algoritmi, a scapito però delle prestazioni computazionali. Quest'ultime sono risultate particolarmente degradate rispetto ad una ben più veloce implementazione in C/C++ soprattutto a causa delle struttura intrinseca degli algoritmi. In tutti e tre i casi considerati si ha difatti una struttura fortemente iterativa, situazione nella quale l'interprete di Matlab introduce un sensibile overhead computazionale.

Nel seguito della sezione tratteremo i dettagli implementativi degli algoritmi, ponendo particolare enfasi su quanto la struttura iterativa degli stessi influenzi il tempo di esecuzione, definendo quindi implicitamente i guadagni in performance che si potrebbero ottenere da un implementazione C/C++. Verrà inoltre posta enfasi su quali siano i passi dell'algoritmo dove sia possibile introdurre parallelismo, in maniera da evidenziare l'incremento di prestazioni che possa essere dato dall'utilizzo di computer con processori multi-core, attualmente i più presenti nel mercato.

5.1.1 Implementazione Geometric Hough Transform

Per l'implementazione della GHT si è seguito quanto definito nell'articolo originale di Ballard e descritto nella sezione 3.2.2. Per velocizzare l'esecuzione dell'algoritmo, esso è stato eseguito su una piramide di immagini (si veda sezione 3.2.1). La dimensione dell'intorno di interesse considerato durante la verifica delle ipotesi diminuisce scendendo nella piramide, mentre il numero di campioni nella discretizzazione rimane fisso, garantendo quindi di ottenere valori sempre più precisi quanto più è maggiore il dettaglio nell'immagine. Ad ogni livello i risultati sono stati clusterizzati in modo da unificare le ipotesi vicine tra loro.

Per le immagini acquisite, il passo di discretizzazione è stato impostato a 5° e 0.1 per rotazione e scala, rispettivamente, mentre il range dei valori per la scala è stato impostato a [1.4 2.2].

Si noti che ogni ipotesi di orientazione e scala del modello viene valutata indipendentemente dalle altre, consentendo quindi di avere un ottimo grado di parallelismo per l'algoritmo. Il tempo di esecuzione nell'implementazione in Matlab è gravemente affetto dalla presenza di 4 cicli for anninati per ogni livello della piramide, necessari per la valutazione della traslazione per ogni valore di ribaltamento, rotazione e scala. Il quarto ciclo for viene utilizzato per scorrere la lista dei punti presenti nell'immagine. D'altra parte, per la valutazione delle possibili traslazioni a cui può essere soggetto il modello viene utilizzata una singola moltiplicazione tra matrici.

5.1.2 Implementazione Template Matching

L'implementazione del template matching è stata effettuata seguendo le specifiche fornite dal paper di Steger sull'argomento [60]. Come nel caso della GHT, si è utilizzata una piramide di immagini, effettuando una ricerca completa solo al livello superiore della piramide e quindi portando le ipotesi trovate nei livelli inferiori. I parametri di discretizzazione utilizzati sono gli stessi descritti nella sezione precedente per la GHT.

Anche nel caso di template matching si ottiene un ottimo grado di parallelismo, in quanto la valutazione della funzione di similitudine può essere eseguita indipendentemente per le diverse orientazioni e scale dell'oggetto. L'algoritmo di template matching è però afflitto maggiormente rispetto alla GHT dalle prestazioni nell'esecuzione delle iterazioni di Matlab. Per ogni possibile valore di ribaltamento, scala e rotazione è difatti necessario valutare la funzione di similitudine per ogni valore di traslazione in X ed in Y, ottenendo in questo modo ben cinque cicli for annidati. Dato che il tempo necessario per l'esecuzione dell'algoritmo richiedeva oltre 45 minuti in queste condizioni, si è deciso di implementare in C la funzione per la determinazione della funzione di similitudine per i diversi valori di traslazione in X e Y, portando il tempo di esecuzione a 3 minuti circa. Oltre che aver permesso una più rapida esecuzione degli esperimenti, questo ha consentito di stimare grossolanamente l'incremento di prestazioni dato dall'implementazione efficiente degli algoritmi.

5.1.3 Implementazione algoritmo proposto

L'implementazione dell'algoritmo proposto segue direttamente dalla descrizione contenuta nella sezione 3.3. Come nei casi precedenti, si ha un ottimo grado di parallelismo sia nella fase di estrazione che di matching delle features. La struttura iterativa dell'algoritmo è decisamente più semplice, includendo un massimo di due cicli for nella fase di matching, uno per scorrere la lista di feature nel modello e l'altro per la lista di feature nell'immagine.

5.2 Raccolta data set

Per la valutazione delle prestazioni degli algoritmi implementati si è resa necessaria la raccolta di immagini utilizzando una configurazione che potesse rappresentare il più possibile la configurazione del sistema di bin picking presentata nella sezione 2.4. Il primo passo è stato quello di calibrare precisamente la telecamera utilizzata. In seguito sono state acquisite diverse immagini nell'intento di rappresentare un'ampio spettro delle casistiche che si potessero verificare anche nel sistema reale.

5.2.1 Configurazione del sistema

Ponendo maggiore attenzione sulle prestazioni dell'algoritmo in termini della sua capacità di trovare le istanze del modello all'interno dell'immagine quanto più della sua precisione nella rilevazione dei prodotti all'interno della scena, la configurazione del sistema prevede l'utilizzo di una sola telecamera. L'utilizzo di un sistema stereo in modo da incrementare la precisione assoluta nella determinazione della posa, come si è visto nella sezione 3.4.1, è una diretta generalizzazione di questo caso.

La telecamera utilizzata per il sistema è una *Basler SCA1000-30FM*, le cui caratteristiche rilevanti all'applicazione sono:

- Sensore: CCD 1/3 a colori.
- Risoluzione massima: 1024×768.
- Framerate: 30 fps.
- C-Mount.
- Interfaccia Firewire-1394B

L'ottica dispone di una lunghezza focale di 4.8mm. La telecamera è stata posta ad un altezza di 70cm, in modo da consentire di inquadrare completamente il contenitore. Si noti che questi valori non sono sufficienti per determinare la posa con la precisione richiesta. D'altra parte, come abbiamo già detto, l'obiettivo degli esperimenti non è quello di valutare la precisione di rilevazione, quanto più di confrontare le prestazioni dell'algoritmo proposto rispetto agli algoritmi disponibili in letteratura nell'identificazione dei prodotti nella scena.

Al fine di poter determinare la resistenza dell'algoritmo alle variazioni di luminosità, si è fatto utilizzo di illuminatori ad incandescenza posti ai lati del contenitore. Si ricordi che, come detto nella sezione 2.4, un ipotesi di sistema prevedeva l'utilizzo di un illuminatore dome per raggiungere l'invarianza alle condizioni di luce ambientale. Con gli esperimenti contenuti in questo capitolo, si vuole determinare anche se sia possibile fare a meno dell'utilizzo di tale illuminatore e del relativo sistema di movimentazione con pistone. In questo modo si potrebbe propendere per l'ipotesi con illuminatore a LED, decisamente più economica.

5.2.2 Calibrazione telecamere

Per la determinazione della calibrazione della telecamera si è eseguita la procedura descritta nella sezione 2.4.1 utilizzando 11 immagini che riprendono il pattern a scacchiera in posizioni distinte. Utilizzando l'implementazione Matlab disponibile in [1] si sono ottenuti i seguenti risultati:

Parametro	**Valore**
Lunghezza Focale	$[1009.02\ 1012.14] \pm [11.43\ 11.64]$
Centro immagine	$[510.50\ 402.89] \pm [12.34\ 12.26]$
Rapporto dimensione pixel	$[0.00] \pm [0.00]$
Distorsione radiale	$[-0.31\ 0.11\ -0.0013] \pm [0.025\ 0.068\ 0.0016]$
Distorsione tangente	$[-0.00039\ 0.00000] \pm [0.00166\ 0.00000]$
Errore di riproiezione	$[1.07\ 1.40]$

La matrice dei parametri estrinseci è stata calcolata in modo da ottenere che il sistema di riferimento della scena sia posto al centro del contenitore con asse Z parallelo all'asse della telecamera ma con direzione opposta.

5.2.3 Raccolta immagini

Le immagini sono state acquisite con l'obiettivo di individuare una casistica nella disposizione relativa dei prodotti più significativa possibile. Inoltre, sono state acquisite le immagini in due condizioni limite: in primo luogo considerando prodotti inclinati sensibilmente rispetto al piano X-Y del contenitore. In secondo luogo, i prodotti sono stati portati in tutto il campo visivo della telecamera, simulando quindi situazioni in cui il contenitore fosse pieno e casi in cui il con-

tenitore fosse quasi vuoto. Le immagini sono state acquisite con condizioni di illuminazione sensibilmente differenti tra loro.

Delle diverse immagini acquisite per il data set, sono state selezionate 300 immagini del prodotto in figura 2.2.b che contengono un sotto insieme rappresentativo delle diverse casistiche possibili per la disposizione dei prodotti nel contenitore ed per le condizioni di illuminazione. Inoltre, al fine di verificare la robustezza del sistema alla variazione delle caratteristiche ottiche del materiale del prodotto, sono state acquisite alcune immagine del prodotto riportato in figura 2.2.a, il quale presenta una superficie particolarmente riflettente.

5.3 Esecuzione dell'algoritmo proposto

Nella sezione corrente analizzeremo i risultati ottenuti nei diversi passi di esecuzione dell'algoritmo proposto, in modo da valutare qualitativamente le prestazioni in un caso concreto di utilizzo

5.3.1 Estrazione delle features

Preliminarmente all'elaborazione delle immagini si è resa necessaria l'estrazione features nel modello. La procedura descritta nella sezione 3.3.1, ha portato all'identificazione delle 8 feature nel modello riportate in figura 5.1. Nell'algoritmo sono stati impostati i parametri per il filtraggio α, β e γ a 0.5, 0.2 e 0.1, rispettivamente.

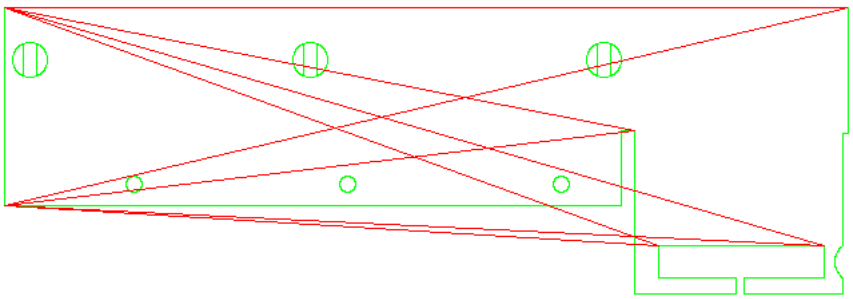


Figura 5.1: Feature rilevate per il modello. Il profilo del prodotto è riportato in verde, mentre i segmenti che uniscono i due punti di interesse che compongono le features sono visualizzati in rosso.

Il primo passo dall'algoritmo per l'elaborazione delle immagini consiste nel determinare le feature. In primo luogo vengono calcolati i corner utilizzando l'algoritmo di Harris. Il risultato nell'immagine di test ha portato all'individuazione dei 474 punti di interesse riportati in figura 5.2. Il passo successivo richiede la valutazione della risposta all'angolo. Dato che nel modello sono presenti solo punti di interesse con lati incidenti a 90°, è sufficiente valutare la risposta all'angolo $\theta = 90°$. Utilizzando i parametri $m_{size} = 11$ e $m_{step} = \frac{\theta}{4}$ e considerando solo i punti la cui risposta sia maggiore di $r_{thres} = 2$, otteniamo i 164 corner riportati in figura 5.3. Dai punti di interesse si ottengono quindi 13366 feature, le quali vengono filtrate imponendo che la distanza tra i punti sia compresa tra $s_{min}M_{min}$ e $s_{max}M_{max}$, dove s_{min} ed s_{max} rappresentano rispettivamente i valori per la scala minima e massima del modello nell'immagine ottenuti dal processo di calibrazione, mentre M_{min} e M_{max} sono la lunghezza minima e massima delle feature del modello. Si ottengono in questo modo 6547 feature valide. Si noti che questo passo non è fondamentale, in quanto le feature vengono filtrate anche successivamente nel passo di matching, ma è necessario solamente per ridurre i requisiti computazionali dell'algoritmo.

5.3.2 Matching

Il primo passo del matching consiste nel determinare, per ogni feature del modello, le possibili candidate tra le feature dell'immagine. Questo viene fatto confrontando i descrittori in modo da avere che gli angoli siano uguali e la distanza dei punti di interesse sia inclusa tra s_{min} ed s_{max} volte la distanza tra i rispettivi punti di interesse del modello. Nel nostro caso, abbiamo solo angoli di 90° quindi tutte le feature individuate sono possibili candidate. La limitazione data dalla dimensione dei segmenti porta invece ad una notevole riduzione delle features da considerare, mediamente pari al 70% delle possibili feature presenti nell'immagine. Avendo 8 features per il modello e 6547 features nell'immagine, solo il 70% delle quali valide, si ottengono circa 36000 match potenziali. Dato che il modello è composto da soli segmenti rettilinei, si hanno quindi 144000 trasformazioni di similitudine risultanti da vagliare attraverso la funzione di similitudine utilizzata.

In figura 5.4 è riportato il risultato dell'algoritmo per l'immagine di test.

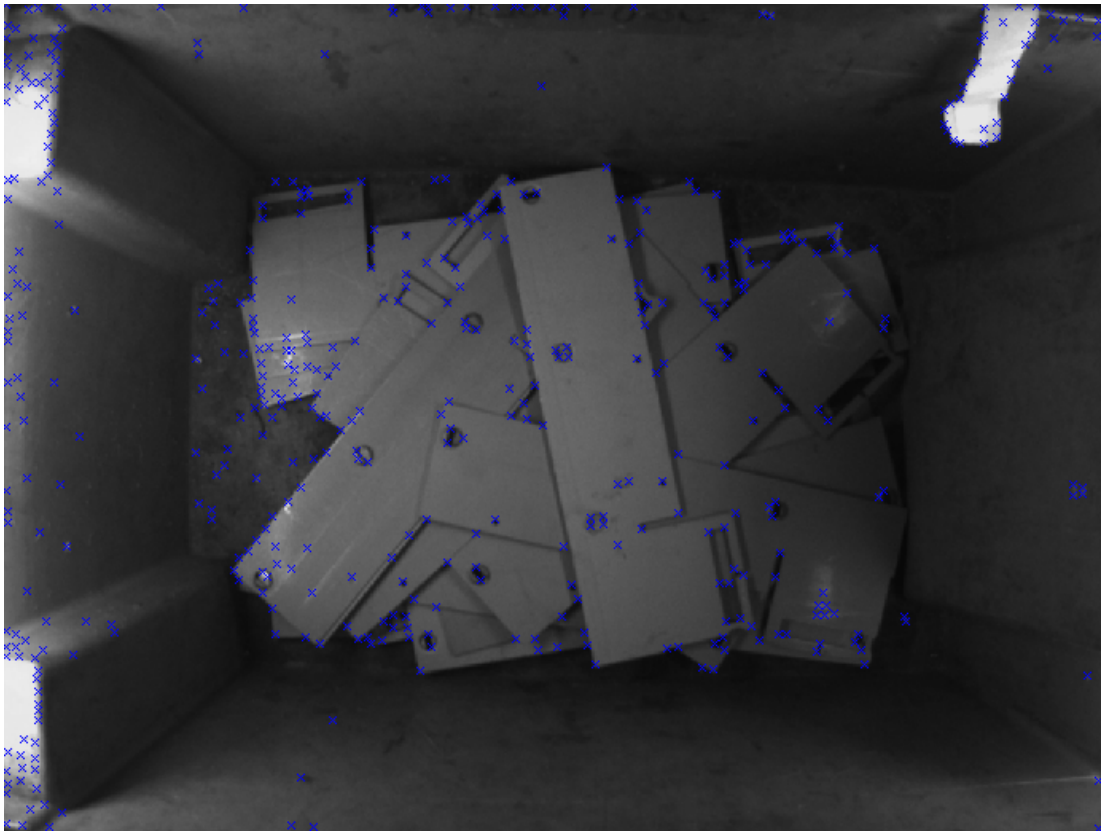


Figura 5.2: Punti di interesse trovati nell'immagine di test.

5.4 Valutazione risultati ottenuti

Gli algoritmi implementati sono stati utilizzati per l'elaborazione delle 300 immagini contenute all'interno del data set. I risultati sono stati valutati quantitativamente secondo i seguenti parametri:

- $p_{success}$: indica la percentuale delle immagini nelle quali uno o più prodotti sono stati individuati correttamente.

- $p_{correct}$: indica la percentuale delle immagini nelle quali sono stati rilevati uno o più dei prodotti che sono posizionati in modo tale da poter essere afferrati da parte del manipolatore.

- p_{error}: tra le immagini che hanno avuto successo, indica la presenza di almeno un oggetto non identificato correttamente.

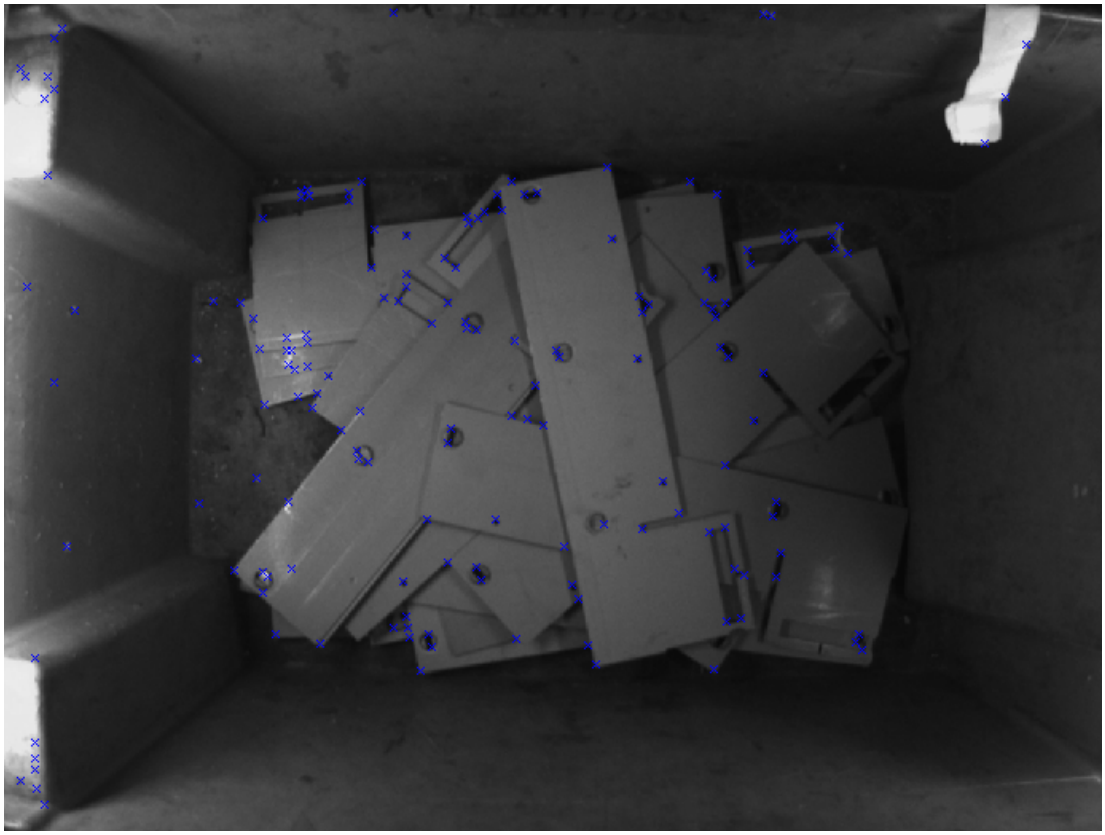


Figura 5.3: Punti di interesse rimanenti in seguito al filtraggio in base alla risposta all'angolo.

In tabella 5.1 sono riportati i risultati ottenuti per i tre algoritmi. Come si vede dalla tabella, l'algoritmo che presenta le prestazioni peggiori è Generalized Hough Transform. Queste è imputabile a due cause principali: in primo luogo la bontà dell'algoritmo GHT dipende fortemente dal valore di soglia utilizzato durante la rilevazione degli edge. Essendo il valore di soglia fisso, è impossibile determinare un valore che sia in grado di ottenere risultati soddisfacenti per le differenti condizioni di illuminazione. Inserendo un valore molto permissivo che consenta di ottenere valori per ogni immagine di test, i punti di edge creati non riescono ad essere clusterizzati correttamente dall'algoritmo. Possiamo quindi affermare che l'algoritmo GHT non è tollerante alle variazioni di luminosità. In secondo luogo, l'algoritmo è sensibile fortemente alla dimensione dei passi per la discretizzazione della rotazione e della scala. La discretizzazione fa si che l'algoritmo non utilizzi esattamente l'orientazione e scala delle istanze di prodotti

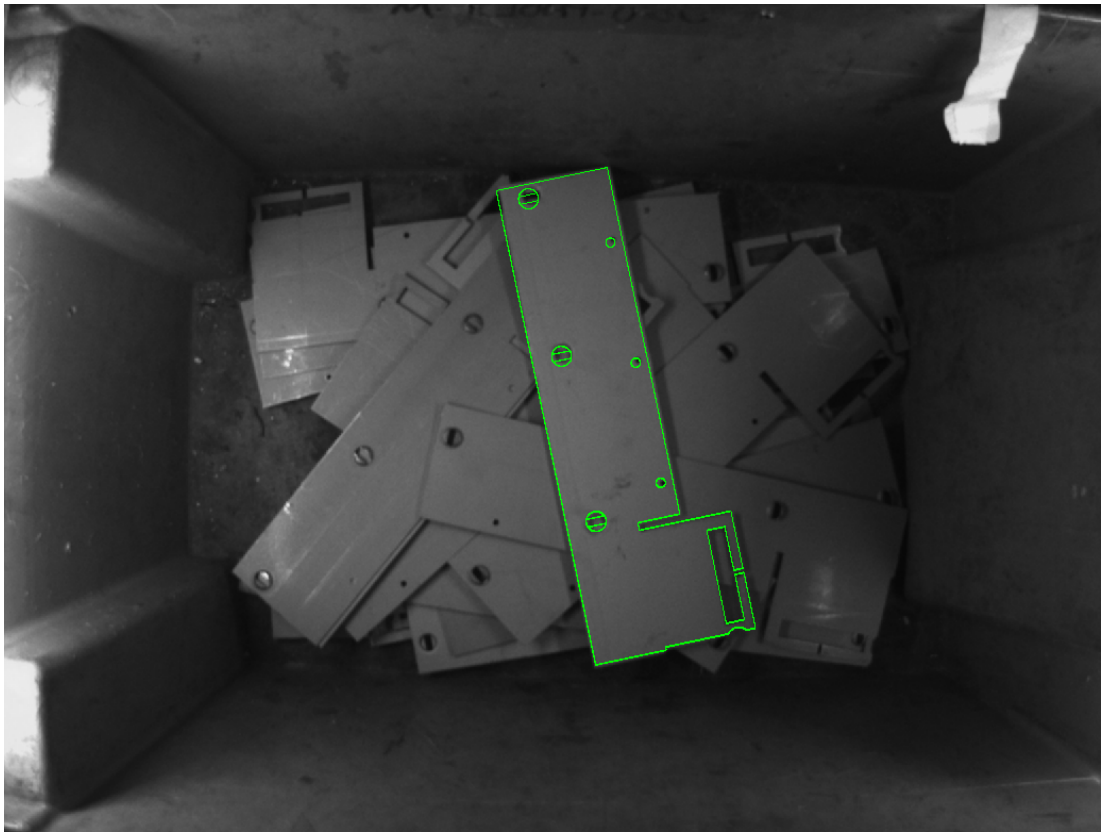


Figura 5.4: Risultato dell'esecuzione dell'algoritmo nell'immagine di test.

contenute nell'immagine, ma una sua approssimazione al valore più vicino. Aumentando i passi si ottengono risultati migliori, a scapito però di un aumento esponenziale del tempo di esecuzione dell'algoritmo.

Anche l'algoritmo basato sul template matching è sensibile alle problematiche della discretizzazione dei valori di rotazione e di scala che affligge la GHT. D'altra parte, la funzione di similitudine utilizzata è per costruzione robusta alle variazioni di luminosità dell'ambiente. Questo giustifica l'incremento di successo nella rilevazione dei prodotti per l'approccio basato su template matching.

Infine, l'algoritmo proposto, grazie al fatto che non è sensibile alla discretizzazione, in quanto calcola le ipotesi di similitudine utilizzando i punti di interesse rilevati nell'immagine stessa, riesce a raggiungere la più alta percentuale di successo, pari quasi alla totalità delle immagini analizzate. Nel dataset raccolto si è verificata una sola situazione in cui l'algoritmo non è stato in grado di determinare la presenza del prodotto nell'immagine. Analizzando l'immagine si è notato

che l'unico prodotto distinguibile all'interno del contenitore si stava muovendo durante l'acquisizione. Per questo l'immagine risulta sfuocata, impedendo la corretta rilevazione dei punti di interesse da parte del corner detector. Per evitare questa situazione nel sistema reale è sufficiente attendere alcuni secondi in seguito al prelievo del prodotto prima di acquisire l'immagine successiva.

Tabella 5.1: Risultati quantitativi dati dall'esecuzione degli algoritmi sulle immagini di test.

Algoritmo	$p_{success}$	$p_{correct}$	p_{error}
GHT	40.3%	15.6%	0%
Template Matching	61.3%	39.0%	0%
Algoritmo Proposto	99.6%	83.0%	10.3%

Oltre alla percentuale di successo, un parametro fondamentale per il bin picking è dato dalla percentuale di immagini dove sono stati individuati i prodotti che possono essere afferrati dal robot manipolatore. Anche per questo parametro, l'algoritmo proposto fornisce i risultati migliori sia in valore assoluto che in valore relativo al numero di immagini dove i prodotti sono stati rilevati con successo. Questo è dovuto al fatto che, al contrario di quanto avviene per gli altri algoritmi, si è riusciti ad includere esplicitamente un euristica che assicuri la selezione dei prodotti con minori occlussioni.

Nell'algoritmo proposto si è però rilevata la condizione singolare in cui si ottengono come risultato dell'algoritmo sia posizioni corrette che errate, condizione riportata in tabella 5.1 con il nome p_{error}. In figura 5.5 è riportato uno degli esempi dove questo si è verificato. Si può notare in figura che è stato rilevato sia il prodotto corretto che un istanza errata, centrata sul prodotto corretto ma ruotata di 180°. La condizione si verifica perché buona parte del contorno dell'oggetto è condivisa tra le due ipotesi. La soluzione può essere fornita dalla successiva fase di minimizzazione dell'errore nella stima della posa. Quando la posa è ottenuta in modo preciso considerando anche la deformazione prospettica si ha difatti che il prodotto privo di occlusioni deve presentare il valore della funzione di similitudine prossimo ad 1. Si può includere quindi una sogliatura dei valori della funzione di similitudine che permetta di eliminare le ipotesi errate. Si noti comunque che in tutti i casi in cui questa condizione si è verificata, gli

algoritmi GHT e di template matching non sono riusciti a rilevare la presenza di nessuno dei prodotti.

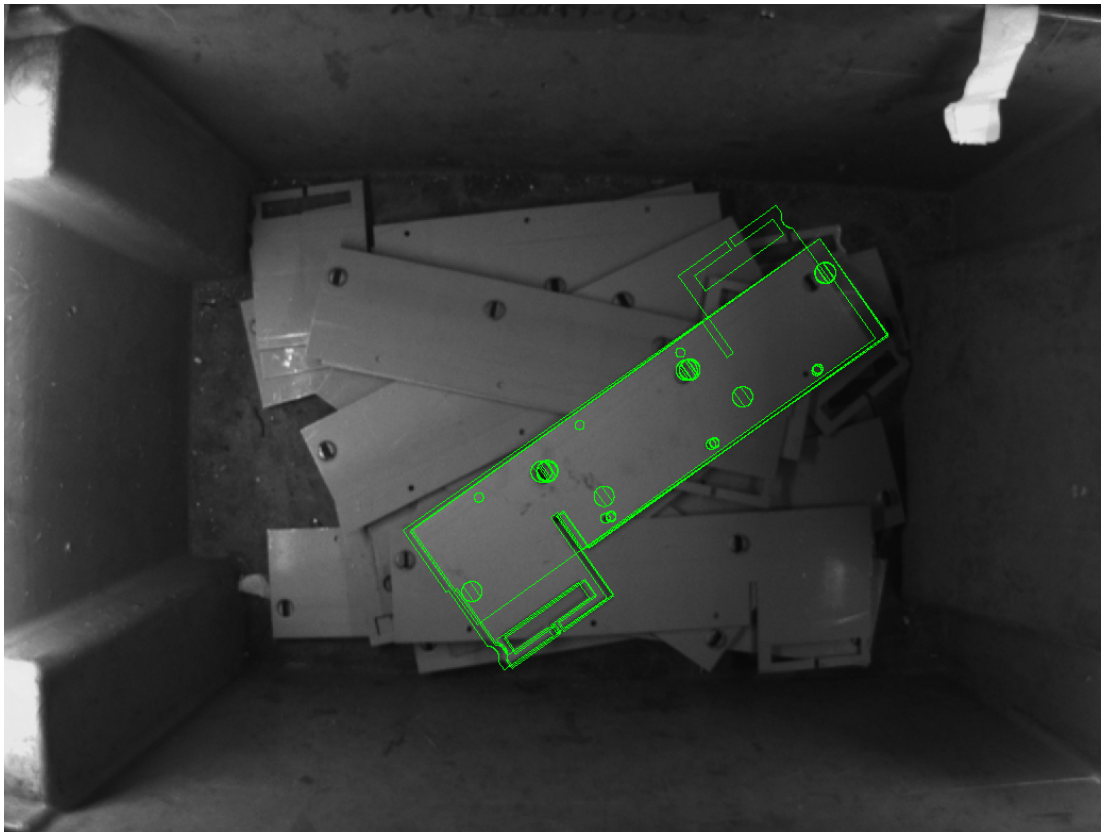


Figura 5.5: Risultato dell'esecuzione dell'algoritmo nel quale sono stati forniti sia valori corretti che errati per la stima della posa.

Nella tabella 5.2 è riportato il tempo di esecuzione dei diversi algoritmi ottenuti utilizzando un computer con processore Intel Core I5 a 2.4 MHZ su sistema Windows 7 a 64 bit. L'algoritmo GHT offre le prestazione computazioni peggiori tra gli algoritmi considerati. Nel valutare il tempo di esecuzione del template matching, è necessario ricordare che i dati solo stati ottenuti utilizzando una funzione in C per il calcolo della traslazione. Come abbiamo visto nella sezione 5.1, utilizzando un implementazione basata interamente su Matlab, si ha un incremento di un fattore 15 del tempo di esecuzione, ottenendo valori comparabili alla GHT. Considerando questo è possibile stimare che le prestazioni computazionali dell'algoritmo proposto siano tra le 25 e le 30 volte maggiori rispetto agli algoritmi considerati presenti in letteratura. Questo è dovuto al fatto che, mentre

gli algoritmi GHT e il template matching eseguono il calcolo per ogni possibile valore della posa del prodotto, nell'approccio proposto l'insieme delle pose viene fornito dalla procedura di matching tra le diverse feature individuate.

Tabella 5.2: Tempo di esecuzione degli algoritmi. Per il template matching, i dati sono stati ottenuti utilizzando l'implementazione in C di parte dell'algoritmo. Si veda la sezione 5.1.2

Algoritmo	t_{mean}	t_{max}	t_{min}
GHT	54 m 25 s	93 m 40 s	41 m 40 s
Template Matching	3 m 42 s	6 m 50 s	2 m 40 s
Algoritmo Proposto	2 m 00 s	2 m 50 s	0 m 56 s

Grazie all'ottimo grado di parallelismo posseduto dagli algoritmi GHT e template matching, dovuto al fatto che la quasi totalità del tempo di esecuzione è impegnato per la valutazione delle diverse ipotesi di orientazione e scala, l'implementazione multi-threading di questi algoritmi può garantire un fattore di guadagno prossimo ad S, dove S è il numero di processori disponibili. Per l'approccio proposto, abbiamo la suddivisione dei tempi medi nelle diverse fasi dell'algoritmo riportata in figura 5.6. Come si può vedere, il 79% del tempo di esecuzione è dominato dal matching. Gli algoritmi di elaborazione dell'immagine, ovvero il calcolo del gradiente e dei corner, occupano solo una piccola parte del tempo di esecuzione totale, pari al 4%. Questo è dovuto anche al fatto che questi algoritmi consistono nella chiamata di poche funzioni di Matlab altamente ottimizzate. Sia il matching che la fase di filtraggio delle features, i quali occupano il 97% del tempo di esecuzione, hanno una struttura intrinsecamente parallelizzabile. Utilizzando la nota legge di Amdhal che lega il miglioramento globale dell'algoritmo dato dalla parallelizzazione di una porzione P dello stesso,

$$speed - up = \frac{1}{(1-P) + \frac{P}{S}} \tag{5.1}$$

possiamo calcolare il fattore di incremento di prestazioni atteso dall'utilizzo di computer con S processori. Ad esempio, considerando un processore quad-core, attualmente il più diffuso nel mercato dei personal computer desktop, otteniamo un fattore di 3.57.

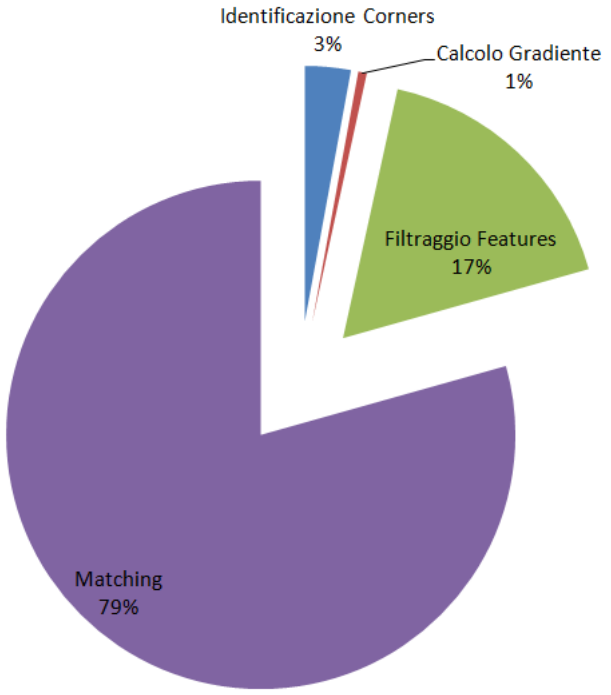


Figura 5.6: Suddivisione dei tempi di esecuzione nelle diverse fasi dell'algoritmo.

Forti anche delle prove condotte per l'algoritmo di template matching (si veda 5.1.2), si stima che un implementazione efficiente, la quale sfrutti le IPP (Intel Integrated Primitives [3]) per l'esecuzione degli algoritmi di visione come il corner detector di Harris o il filtraggio dell'immagine e consideri gli opportuni meccanismi di caching dei risultati per il calcolo della trasformazione e la valutazione della funzione di similitudine, possa ottenere un miglioramento nel tempo di esecuzione dell'algoritmo di un fattore 20. In queste condizioni, il tempo di esecuzione dell'algoritmo passa a circa 6 secondi nel caso medio. Inoltre, sfruttando l'ottimo grado di parallelismo, il tempo di esecuzione può scendere sotto i due secondi utilizzando un sistema basato su processore quad-core. Possiamo quindi affermare che l'algoritmo rispetta i vincoli dettati dall'esecuzione in modo mascherato all'interno di un normale tempo ciclo.

In figura 5.8 sono presenti due immagini acquisite con condizioni di illuminazione nettamente differenti tra loro. Come si può vedere, l'algoritmo è stato in grado di individuare correttamente la posizione dei prodotti all'interno di entrambe le immagini. Infine, con lo scopo di verificare la robustezza del sistema alla variazione delle caratteristiche ottiche del materiale del prodotto, sono state

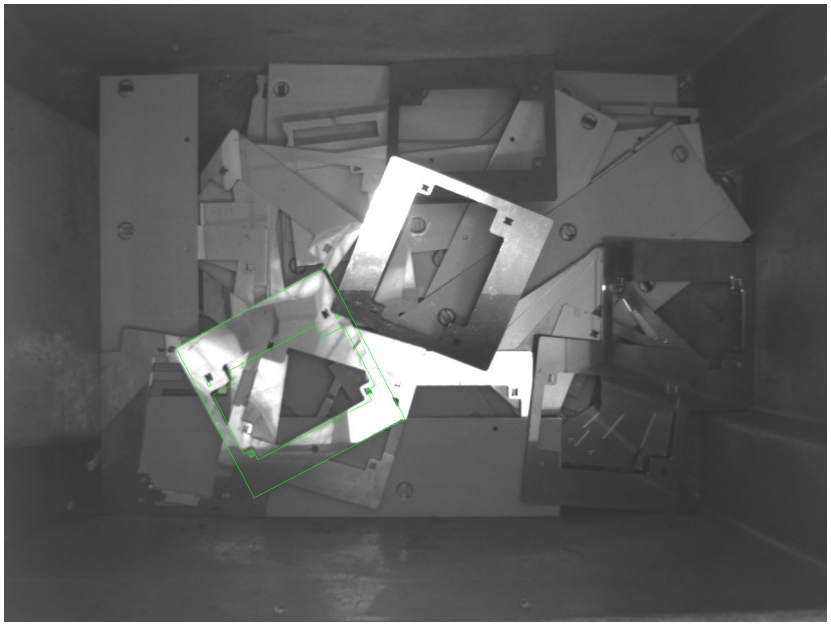
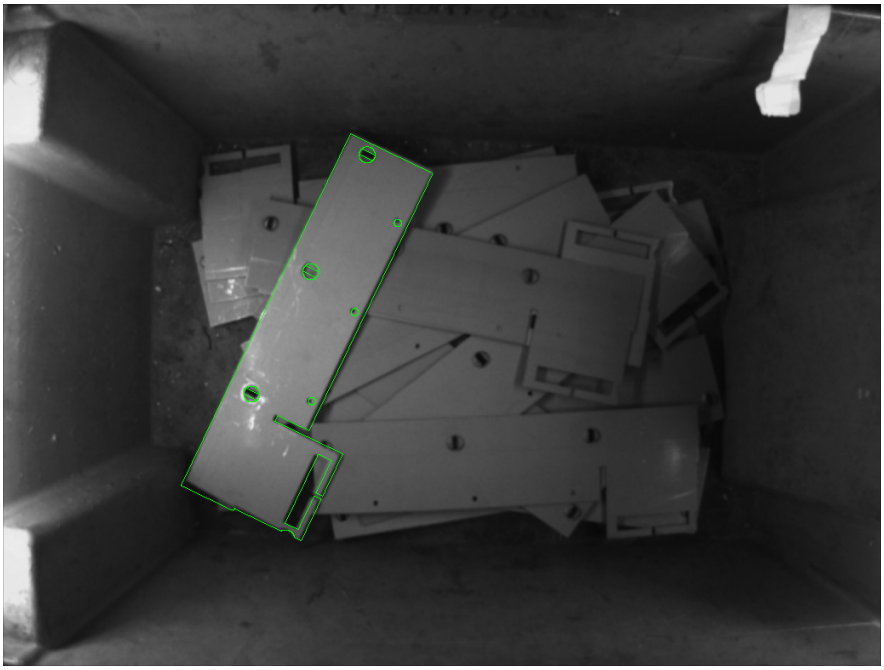
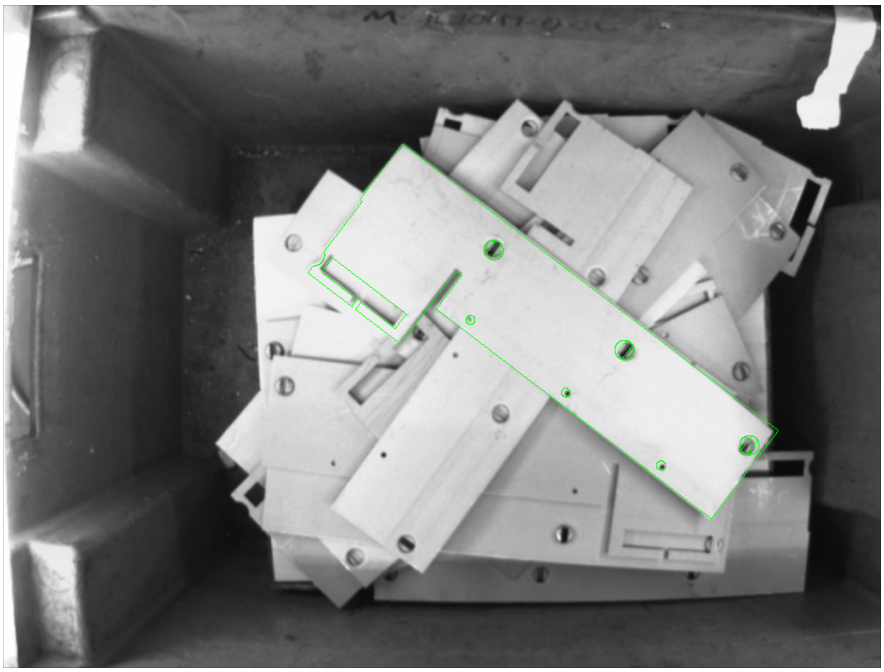


Figura 5.7: Risultato dell'esecuzione dell'algoritmo per una delle immagine di test del prodotto riportato in figura 2.2.a.

acquisite dieci immagini del prodotto riportato in figura 2.2.a, il quale presenta una superficie particolarmente riflettente. In figura 5.7 è riportata una delle immagine di test considerate. Si può immediatamente notare come siano presenti evidenti riflessi speculari causati dal materiale che generano delle forti variazioni di contrasto all'interno del singolo prodotto. Anche in queste condizioni l'algoritmo proposto ha consentito la rilevazione del prodotto nelle immagini acquisite, confermando l'applicabilità dello stesso per il problema del bin picking anche in presenza di riflessi speculari nel prodotto.



(a)



(b)

Figura 5.8: Risultati ottenuti con immagini acquisite in condizioni di illuminazione nettamente differenti tra loro.

Capitolo 6

Conclusioni

Nel presente studio abbiamo affrontato il problema del bin picking per il prelievo di oggetti planari all'interno di un contenitore da parte di un robot manipolatore. Durante la progettazione ci si è posti gli obiettivi di creare un sistema di visione che consenta di rispettare i requisiti di utilizzare il robot presente nella cella di lavoro, mascherare il tempo di esecuzione in modo da ridurre il tempo ciclo, permettere la continuità di operazione nel tempo ed essere di basso costo. Rispettando i vincoli forniti dalle specifiche descritte in fase di progettazione, si è quindi optato per un sistema basato su telecamera standard ed illuminatori a LED, soluzione decisamente più economica di alternative attualmente disponibili sul mercato basate sull'utilizzo di luce strutturata.

In seguito ci si è concentrati sugli algoritmi di visione necessari alla soluzione del problema dell'identificazione della posa dei prodotto posti all'interno del contenitore. A seguito di una valutazione degli algoritmi disponibili il letteratura, si è proposto un approccio che consente di unificare i vari punti di forza degli studi precedenti. In primo luogo, l'algoritmo riduce lo spazio di ricerca effettuando il matching tra descrittori basati sulle sole caratteristiche geometriche del modello, in modo da poter essere utilizzato anche in assenza di texture e accettando come ingresso un disegno CAD per la descrizione del prodotto. La valutazione della bontà delle ipotesi fornite dal matching si basa quindi ma su una funzione di similarità definita sulle caratteristiche dell'immagine stessa, la quale risulta essere particolmente robusta alla variazione delle condizioni di illuminazione, all'occlusione ed alla presenza di clutter all'interno della scena. Per la valutazione dell'approccio è stato quindi acquisito un data set in modo da rappresentare uno

spettro delle casistiche che si potessero presentare nell'ambiente reale. I risultati ottenuti sono stati molto incoraggianti, dato che l'algoritmo si è dimostrato in grado di riconoscere gli oggetti all'interno dell'immagine nel 99.6% dei casi ottenendo un tempo di esecuzione medio fino a 30 volte inferiore rispetto ai metodi disponibili in letteratura.

Parallelamente allo sviluppo del sistema di bin picking, si è voluto proporre con la presente tesi un nuovo framework software che possa essere utilizzato nei diversi contesti della visione computazionale. Il primo passo è stato la proposta di un nuovo stile architetturale per la creazione di applicazioni che elaborano stream di dati, come un flusso video, il quale propone dei miglioramenti al classico modello *pipes and filters* sia per quanto riguarda le prestazioni computazionale che l'uniformità nella trattazione delle informazioni, mantenendo d'altra parte le vantaggiose caratteristiche di modularità e semplicità di definizione che questo presenta. Con in mente obiettivi quali la riusabilità del codice, la modularità, la scalabilità e la semplicità di utilizzo, il nuovo stile architetturale è stato tradotto nella progettazione di un nuovo framework. In questo contesto sono stati definiti i componenti principali del sistema e come i requisiti individuati possano essere soddisfati con la nuova architettura. Infine, sono state presentati diversi esempi di utilizzo, tra i quali è presente anche il sistema di bin picking descritto nella presente tesi.

6.1 Sviluppi futuri

Per quanto riguarda il sistema di bin picking, il primo passo da effettuare riguarda la realizzazione di un prototipo completo basato su quanto descritto nel presente studio. Per fare questo è in primo luogo necessario implementare in modo efficiente gli algoritmi descritti, per quindi completare lo sviluppo software includendo anche le procedure per la corretta comunicazione con il robot manipolatore presente nella cella. Solo mediante l'utilizzo in un caso concreto si può avere la conferma definitiva che il nuovo algoritmo possa essere in grado di risolvere il problema nell'ambiente reale. In seguito l'algoritmo proposto può essere esteso in modo da includere l'utilizzo di più telecamere o il riconoscimento di oggetti di forma arbitraria. In questo modo si potrà ottenere un sistema che possa essere utilizzato nei più diversi contesti applicativi.

Contestualmente all'implementazione del sistema di bin picking è necessario sviluppare anche il framework software proposto. In questo modo, lo sviluppo procede di pari passo ad un suo primo utilizzo, consentendo di correggere eventuali problematiche non considerate durante il processo di analisi e progettazione non appena si presentino. Al termine dello sviluppo i moduli possono essere standardizzati per gettare le prime basi per la creazione di una libreria di componenti. In seguito ci si aspetta che anche le altre applicazioni possano venir integrate nel framework, garantendo in questo modo una rapida crescita degli utilizzatori. Solo con una partecipazione attiva da parte di tutti gli interessati si possono ottenere i benefici a lungo termine dati dal riutilizzo dei componenti sviluppati.

Bibliografia

[1] Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/.

[2] Echord - european clearing house from open robotics develpment. http://www.echord.info/.

[3] Intel integrated performance primitives. http://software.intel.com/en-us/intel-ipp/.

[4] Intel opencv. http://sourceforge.net/projects/opencvlibrary/.

[5] Mil - maxtor image library. http://www.matrox.com/imaging/.

[6] Sick. http://www.sick.se/.

[7] Juan Andrade-cetto and Avinash C. Kak. Object recognition. *Wiley Encyclopedia of Electrical and Electronics Engineering*, 1:2000, 2000.

[8] Stephane Baldo and Djamel Yahia Meddah. Subtractive primitives used in pattern matching, 2005. US Patent 7319791.

[9] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. pages 714–725, 1987.

[10] Herbert Bay, Tinne Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *9th European Conference on Computer Vision*, Graz Austria, May 2006.

[11] M. Benosman and G. Le Vey. Control of flexible manipulators: A survey. *Robotica*, 22(05):533–545, 2004.

[12] H.A. Beyer. Accurate calibration of ccd-cameras. *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR apos;92., 1992 IEEE Computer Society Conference on*, pages 96 – 101, 1992.

[13] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. pages 348–353, 2000.

[14] K. Boehnke. Object localization in range data for robotic bin picking. *IEEE International Conference on Automation Science and Engineering (CASE)*, 22-25:572 – 577, 2007.

[15] H. Borotschnig, L. Paletta, M. Prantl, and A. Pinz. Appearance-based active object recognition. *Image and Vision Computing*, 18:715–727, 2000.

[16] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, (Q2), 1998.

[17] D.C. Brown. Decentering distortion of lenses. 7:444–462, 1966.

[18] J.H.M. Byne and Anderson J.A.D.W. A cad-based computer vision system. *Image and Vision Computing*, 16:533–539, June 1998.

[19] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, November 1986.

[20] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(8):790–799, 1995.

[21] Roland T. Chin and Charles R. Dyer. Model-based recognition in robot vision. *ACM Computing Surveys*, 18:67–108, 1986.

[22] Dorin Comaniciu, Peter Meer, and Senior Member. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.

[23] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. The variable bandwidth mean shift and data-driven scale selection. In *in Proc. 8th Intl. Conf. on Computer Vision*, pages 438–445, 2001.

[24] Dorin Comaniciu, Visvanathan Ramesh, Peter Meer, Senior Member, and Senior Member. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:564–577, 2003.

[25] Christopher M. Cyr and Benjamin B. Kimia. A similarity-based aspect-graph approach to 3d object recognition. *Int. J. Comput. Vision*, 57(1):5–22, 2004.

[26] F. Devernay and O. D. Faugeras. Automatic calibration and removal of distortion from scenes of structured environments. pages 62–72, 1995.

[27] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. 2001.

[28] Olivier Ecabert and Jean-Philippe Thiran. Adaptive hough transform for the detection of natural shapes under weak affine transformations. *Pattern Recognition Letters*, 25(12):1411 – 1419, 2004.

[29] J. Forest, J.M. Teixidor, J. Salvi, and E. Cabruja. A proposal for laser scanners sub-pixel accuracy peak detector. pages 525–532, 2003.

[30] Per-Erik Forssen and Anders Moe. Autonomous learning of object appearances using colour contour frames. In *CRV '06: Proceedings of the The 3rd Canadian Conference on Computer and Robot Vision*, page 3, Washington, DC, USA, 2006. IEEE Computer Society.

[31] K. Fukunaga and L.D. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. 21(1):32–40, January 1975.

[32] S. Gilles. *Robust Description and Matching of Images*. 1988.

[33] G.H. Granlund and H. Knutsson. *Signal Processing for Computer Vision*. Kluwer, December 1995.

[34] W. Eric L. Grimson and Daniel P. Huttenlocher. On the verification of hypothesized matches in model-based recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(12):1201–1213, 1991.

[35] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[36] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[37] Janne Heikkila and Olli Silven. A four-step camera calibration procedure with implicit image correction. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1106, 1997.

[38] P. V. C. Hough. Machine analysis of bubble chamber pictures. pages 554–556, CERN, Geneva, Switzerland, 1959.

[39] K. Hsiao and T. Lozano-Perez. Imitation learning of whole-body grasps. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5657–5662, 2006.

[40] J. Illingworth and J. Kittler. The adaptive hough transform. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):690–698, 1987.

[41] Björn Johansson and Anders Moe. Patch-duplets for object recognition and pose estimation. In Ewert Bengtsson and Mats Eriksson, editors, *Proceedings SSBA'04 Symposium on Image Analysis*, pages 78–81, Uppsala, March 2004. SSBA.

[42] Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30:79–116, 1998.

[43] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[44] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *Int. J. Comput. Vision*, 60(1):63–86, 2004.

[45] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.

[46] James Miller and Charles V. Stewart. Muse: Robust surface fitting using unbiased scale estimates. In *In CVPR*, pages 300–306, 1996.

[47] Joseph L. Mundy. Object recognition in the geometric era: A retrospective. In *Toward CategoryLevel Object Recognition, volume 4170 of Lecture Notes in Computer Science*, pages 3–29. Springer, 2006.

[48] Alison Noble. *Descriptions of Image Surfaces*. 1989.

[49] Michael S. Paterson and F. Frances Yao. Efficient binary space partitions for hidden-surface removal and solid modeling. *Discrete Comput. Geom.*, 5(5):485–503, 1990.

[50] Rafael Pelossof, Andrew Miller, Peter Allen, and Tony Jebara. An svm learning approach to robotic grasping. In *Int. Conf. on Robotics and Automation*, pages 3512–3518, 2004.

[51] M. Ribo and M. Brandner. State of the art on vision-based structured light systems for 3d measurements. pages 2 – 6, 2005.

[52] C. Rocchini, P. Cignoni, C. Montani, P. Pingi, and R. Scopigno. A low cost 3d scanner based on structured light. *Computer Graphics Forum*, 20:299 – 308, 2001.

[53] M. Saquib Sarfraz and Olaf Hellwich. Head pose estimation in face recognition across pose scenarios. In *VISAPP (1)*, pages 235–242, 2008.

[54] Ashutosh Saxena, Justin Driemeyer, and Andrew Y. Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157–173, February 2008.

[55] Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

[56] Christian Simon and Djamel Meddah. Geometric hashing method for model-based recognition of an object, 2006. US Patent 7327888.

[57] S. M. Smith and J. M. Brady. Susan - a new approach to low level image processing. *International Journal of Computer Vision*, 23:45–78, 1995.

[58] I Sobel and G Feldman. A 3x3 isotropic gradient operator for image processing. 1968.

[59] C. Steger. Subpixel-precise extraction of lines and edges. pages 141–156, 2000.

[60] C.T. Steger. Occlusion, clutter, and illumination invariant object recognition. In *PCV02*, page A: 345, 2002.

[61] Robert Strzodka, Ivo Ihrke, and Marcus Magnor. A graphics hardware implementation of the Generalized Hough Transform for fast object recognition, scale, and 3d pose detection. In *Proceedings of IEEE International Conference on Image Analysis and Processing (ICIAP'03)*, pages 188–193, 2003.

[62] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. pages 221–244, 1992.

[63] Markus Ulrich, Christian Wiedemann, and Carsten Steger. Cad-based recognition of 3d objects in monocular images. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 2090–2097, Piscataway, NJ, USA, 2009. IEEE Press.

[64] Fredrik Viksten, Per-Erik Forssén, Björn Johansson, and Anders Moe. Comparison of local image descriptors for full 6 degree-of-freedom pose estimation. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 1139–1146, Piscataway, NJ, USA, 2009. IEEE Press.

[65] C. von Bank, D. M. Gavrila, and C. Wöhler. A visual quality inspection system based on a hierarchical 3d pose estimation algorithm. 2003.

[66] Haim J. Wolfson and Isidore Rigoutsos. Geometric hashing: An overview. *Computing in Science and Engineering*, 4:10–21, 1997.

[67] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. volume 1, pages 666–673 vol.1, 1999.