Politecnico di Torino

Master's Degree Course in Computer Engineering

Master's Degree Thesis

Augmented Pick-to-light

Relatore:
Prof. Luca Ardito

Candidato:
Salvatore Zaccaria

Anno accademico 2019/20
Torino

Sommario

I recenti cambiamenti negli ordini dei clienti, che da un lato stanno diventando più frequenti e dall'altro richiedono quantità minori di più prodotti, incidono inevitabilmente sul lavoro dei gestori del magazzino, che si trovano nella posizione necessaria per soddisfare gli elevati volumi in una breve finestra di tempo. Inoltre, il prelievo manuale di magazzino è tradizionalmente caratterizzato da un elevato impatto del fattore umano. Il miglioramento di tale sistema porterà a una riduzione sia dei tempi di elaborazione degli ordini sia dei possibili errori umani. In questo senso, una possibile strategia può essere lo sviluppo e l'impiego di sistemi tecnologici in grado di supportare gli operatori durante i loro tour di picking. "Pick To Light systems" sono sistemi comodi e intuitivi per l'operatore, che ha le mani libere e non deve fare uso di alcun tipo di apparecchiatura fastidiosa che permette un apprendimento immediato. I sistemi Pick To Light sono modelli adatti a qualunque tipo di esigenza. I più abituali illuminano la posizione di prelievo e indicano la quantità richiesta, con una rapida conferma da parte dell'operatore, avvertendo l'operatore in caso di errore.

Lo scopo di questo studio è di presentare una nuova soluzione di progettazione pick-to-light in grado di guidare diversi operatori attraverso le loro attività, prevenire o ridurre gli errori da un nuovo sistema di controllo e allarme in tempo reale, basato sulle principali potenzialità della computer vision.

Dopo la descrizione delle caratteristiche tecniche e del funzionamento della nuova soluzione, viene proposto anche un confronto qualitativo con altri sistemi di prelievo senza carta esistenti.

Indice

Elenco delle figure 4

Elenco delle tabelle 6

1 Background e opere correlate 7
- 1.1 Introduzione . . . 7
- 1.2 Le sfide delle applicazioni mobili . . . 10
- 1.3 Computer Vision . . . 12
- 1.4 Obiettivi . . . 17
- 1.5 Stato dell'arte . . . 18
  - 1.5.1 Pick to voice . . . 19
  - 1.5.2 Pick to light . . . 20
  - 1.5.3 Pick to light gloves . . . 22
  - 1.5.4 Pick to vision . . . 24
- 1.6 Errori e tempo di raccolta . . . 26
- 1.7 Riguardo questa App . . . 26
  - 1.7.1 Tool usati in questa App . . . 27

2 Architettura e design 39
- 2.1 Analizzare il progetto Android . . . 39
- 2.2 Sviluppo . . . 40
  - 2.2.1 Schermata di login . . . 40
  - 2.2.2 Scheramata di benvenuto . . . 41
  - 2.2.3 Scansione bar code . . . 42
  - 2.2.4 Lista prodotti . . . 43
  - 2.2.5 Schermata informazioni . . . 46
  - 2.2.6 Rilevamento . . . 47

3 Certificare la validità 55
3.1 Inserimento oggetto a sistema 57
3.2 Prelievo oggetto 59
3.3 Ricarica oggetto (da parte di operatore) 62
3.4 Spostamento oggetto da un cassetto all'altro 64

4 Conclusioni 67
4.1 Punti di attenzione 67
4.1.1 Sviluppi futuri 68

Bibliografia 71

Elenco delle figure

1.1 Supply chain management. 7
1.2 Time line 9
1.3 Esempio di matrice 2D. 13
1.4 Esempio di matrice 3D. 14
1.5 Flusso di esecuzione di un pick-to-voice. 19
1.6 Esempio di attrezzatura per il pick-to-voice. 20
1.7 Flusso di un sistema pick-to-light cablato. 21
1.8 Flusso di un sistema pick-to-light wireless. 22
1.9 Esempio di un sistema con gloves RFID 24
1.10 Esempio di un sistema con smart glasses 25
1.11 Logo Android Studio 28
1.12 OpenCV time-line 29
1.13 Logo OpenCV 30
1.14 QRcode 33
1.15 Logo Zxing 33
1.16 Diagramma MTV Django 35
1.17 Django e postgres 35
1.18 Logo PostgreSQL 37

2.1 Logo App. 39
2.2 Screenshot schermata login. 41
2.3 Screenshot schermata benvenuto. 42
2.4 Screenshot scansione con ZXING. 43
2.5 Screenshot lista prodotti. 44
2.6 Adapter. 45
2.7 Screenshot informazioni prodotti. 46

2.8 Rilevamento scaffale. 48
2.9 Prodotto da prelevare. 50
2.10 Snippet della funzione removeAverage. 52
2.11 Snippet delle funzioni printRectangles e removeBiggest 53

3.1 Flussi logistici 55
3.2 UML caso 1 57
3.3 UML caso 2 60
3.4 UML caso 3 63
3.5 UML caso 4 65

Elenco delle tabelle

1.1 Confronto sistemi pick to light 23
1.2 Tecnologie a confronto 25
1.3 Differenze tra Android Studio e ADT 28

Capitolo 1

Background e opere correlate

1.1 Introduzione

Nell' ultimo decennio la globalizzazione ha introdotto una crescente evoluzione del lavoro. Infatti oggi l'obiettivo di ogni azienda è l'ottimizzazione dei processi al fine di ottenere soddisfazione lato cliente, in termine di qualità prodotto e tempestività di consegna (livello di servizio al 100%) e riduzione costi in termine di stock, manodopera e trasporti. Capiamo bene che l'automatizzazione aziendale svolge un ruolo fondamentale per raggiungere questi due obiettivi primari.
La logistica è il processo di pianificazione, organizzazione e controllo di tutte le attività di movimentazione e stoccaggio dei beni e delle informazioni dai punti di acquisizione delle materie prime, attraverso il processo produttivo dell'azienda, fino al cliente finale sotto forma di prodotti finiti come mostrato in figura 1.1.

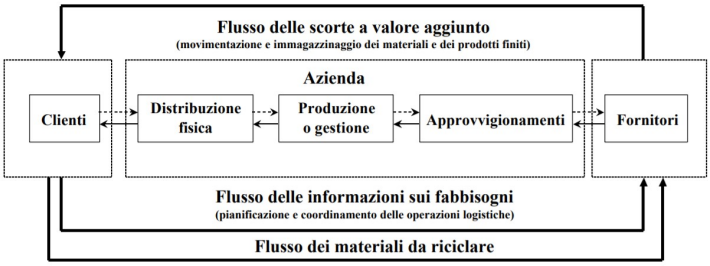

Figura 1.1. Supply chain management.

La logistica aziendale affonda le sue radici nella seconda guerra mondiale, in relazione personale e punti di approvvigionamento materiale durante la guerra. A loro volta, i primi corsi universitari e materiali sono stati negli Stati Uniti negli anni sessanta. Il termine "business logistico" caratterizza i processi che determinano sistema di gestione e coordinare tutto il flusso di materiale dalla fabbrica agli utenti finali (Cirulis & Ginters, 2013).
La distribuzione fisica consiste nel trasferimento del prodotto al cliente contribuendo alla generazione del prodotto assicurando al cliente il servizio al costo più basso possibile. Si basa sulla ricezione degli ordini, la gestione e la movimentazione delle scorte e il trasporto.
Il supporto alla produzione fornisce il controllo delle scorte dei materiali in lavorazione nelle varie fasi di produzione.
Gli approvvigionamenti supportano la produzione e la vendita di materiali, prodotti finiti dal fornitore e ai magazzini.
Le componenti del sistema logistico sono interconnesse e non possono essere ottimizzate singolarmente ma solo attraverso un approccio di tipo sistematico. La qualità e la tempestività delle informazioni conferiscono stabilità al sistema e se viene passata un'informazione errata può dar luogo a gravi inconvenienti nel funzionamento del sistema come scorte in eccesso o insufficienti.
La capacità di pianificare e gestire adeguatamente le attività logistiche costituisce uno degli aspetti maggiormente decisivi per il successo dell'impresa. In particolare l'evasione ordini si innesta nel processo logistico come una delle migliori carte per migliorare la competitività delle imprese. Il mercato spinge nella direzione di maggior frazionamento (piccole quantità) e maggior frequenza negli ordini obbligando ad ottenere un elevato livello di servizio e di sicurezza (assenza di errori nei prelievi e garanzia di consegna nei tempi stabiliti). Questa tendenza, se non inserita in un preciso contesto di gestione, provoca un forte incremento dei costi, spesso accompagnato da tassi di errore inaccettabili. Non esiste una soluzione univoca per tutte le realtà aziendali, ma bisogna focalizzarsi su una metodologia più consona valutando:

1. quantità degli articoli gestiti

2. dimensione dei singoli articoli

3. numero di ordini/giorno e stagionalità

4. tempo di consegna da garantire

Le aziende adesso sono sempre più digitali e interconnesse. L'industria 4.0 sta direzionando ad avere una produzione industriale del tutto automatizzata. L'espressione *Industrie 4.0* è stata usata per la prima volta alla Fiera di Hannover nel 2011, in Germania.
Come possiamo vedere dalla figura 1.2 finora le rivoluzioni industriali del mondo occidentale sono state tre: nel 1784 con la nascita della macchina a vapore e di conseguenza con lo sfruttamento della potenza di acqua e vapore per meccanizzare la produzione; nel 1870 con il via alla produzione di massa attraverso l'uso sempre più diffuso dell'elettricità, l'avvento del motore a scoppio e l'aumento dell'utilizzo del petrolio come nuova fonte energetica; nel 1970 con la nascita dell'informatica, dalla quale è scaturita l'era digitale destinata ad incrementare i livelli di automazione avvalendosi di sistemi elettronici e dell'IT (Information Technology). La data d'inizio della quarta rivoluzione industriale non è ancora stabilita, probabilmente perché è tuttora in corso e solo a posteriori sarà possibile indicarne l'atto fondante.

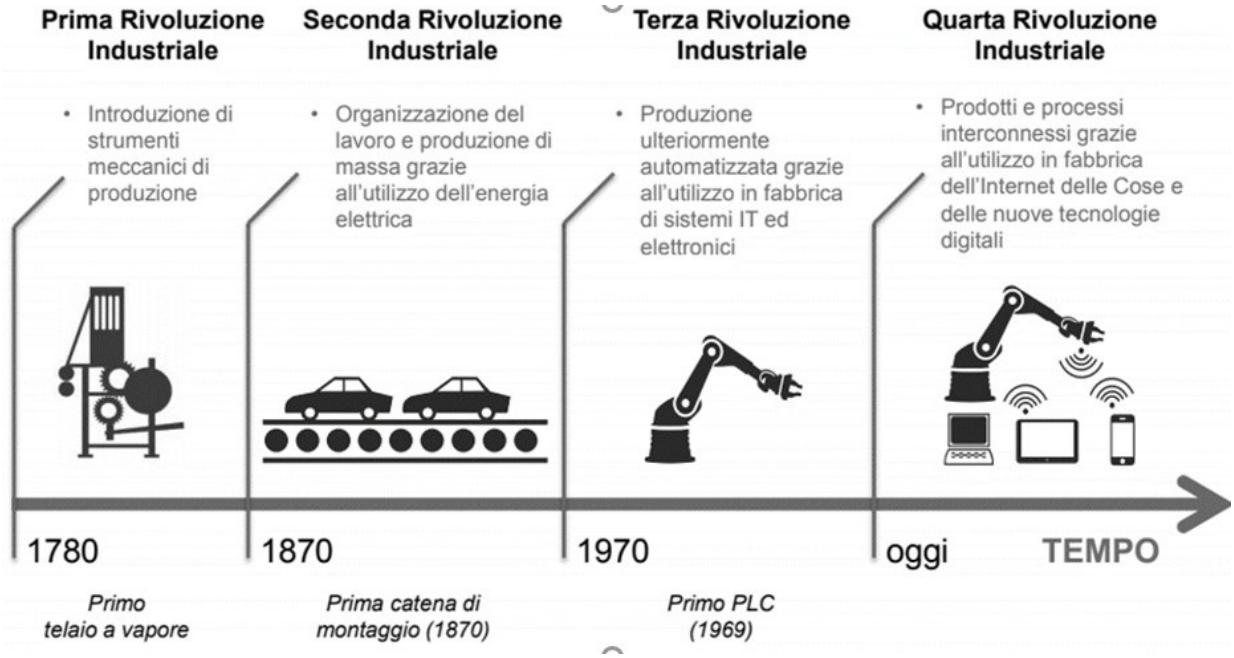

Figura 1.2. Time line

L'industria 4.0 e l'introduzione dell' e-commerce hanno eliminato nel consumatore

la percezione di spazio e tempo, aumentando l'esigenza di una disponibilità immediata di prodotto personalizzato. I cinque punti chiave che il magazzino deve rispettare sono velocità, flessibilità, precisione, efficienza e ordine. In questo contesto le macchine non possono sostituire l'essere umano con la flessibilità e capacità motorie (Reif & Günthner, 2009). La flessibilità è necessaria perché la gamma di prodotti e quindi la varietà di articoli aumenta. Dunque gli esseri umani sono spesso la migliore soluzione per la raccolta nei depositi.
Tuttavia, un'automazione efficace con strumenti che interagiscono attivamente con l'operato umano porterà sempre e comunque l'uomo al centro di ogni processo industriale, esonerandolo dalle attività più tediose e ripetitive che non apportano valore aggiunto. Questo porterà una valorizzazione delle risorse umane impiegate, limitando e/o eliminando gli errori commessi dall'uomo.
Il prelevamento degli articoli rappresenta un'attività sempre più critica da gestire, per snellirla ed ottimizzarla molte aziende stanno ricorrendo alla strategia del *PICKING* nelle sue varie forme specifiche. L' *order picking* è la raccolta di una gamma di articoli a seguito di ordini da parte dei clienti (Beuth, 1994).

In ambito distributivo e di produzione un errore di picking non comporterà solo una serie di operazioni correttive e quindi un aumento dei costi per l'azienda, ma anche incongruenza inventariali, un abbattimento delle scorte, rallentamenti e fermi della catena produttiva con conseguenti ritardi di consegna al cliente (ulteriori onerosi costi).
Ecco quindi che subentra l'esigenza di tecnologie "smart" che vanno a supportare l'operatore come il pick-to-light.

1.2 Le sfide delle applicazioni mobili

Il picking, la preparazione ordini e quella delle spedizioni sono critiche proporzionalmente alla grandezza dell'azienda e ai volumi trattati. Uno degli aspetti focali per il successo aziendale è quello di pianificare e gestire le attività logistiche.
L'attività di picking è composta da una serie di fasi consecutive che sono utili per scegliere quale organizzazione adottare. Queste fasi, indipendentemente da come viene gestito il magazzino sono:

1. Tempo presa in carico della bolla di prelievo.

2. Tempo di trasferimento per raggiungere la posizione di picking.
3. Tempo di identificazione merce leggendo dalla lista e consecutiva spunta sulla lista.
4. Tempo di prelievo effettivo.

Possiamo quindi distinguere due metodologie di prelievo dal magazzino. La prima più tradizionale prevede un utilizzo cartaceo della lista dei prodotti; la seconda, quella ormai più utilizzata, che prevede diversi tipi di sistemi informatizzati e di una gestione del database. I problemi dell' *order picking* senza tecnologia si posso raccogliere nei seguenti punti:

1. **Non trovare gli articoli**. Quando un articolo viene spostato senza che il picker venga informato significa che viene impiegato del tempo per cercarlo. Ciò ritarda il compimento dell'ordine, il che può portare una recensione negativa da parte di un cliente insoddisfatto. Inoltre, si va verso un magazzino altamente inefficiente.
2. **Errore da parte di un picker**. Vengono raccolti articoli o quantità sbagliate. Se l'errore passa tutti i controlli verranno spediti degli articoli errati che in futuro verranno rimborsati o sostituiti. Questo errore porterà un costo e una cattiva immagina all'azienda.
3. **Prelievo cartaceo**. Di per se il prelievo cartaceo è un problema di tempistiche e di inefficacia lasciando un margine di errore a causa di ordini non completati o persi.
4. **Nessuna priorità al momento del prelievo**. Nel momento in cui gli ordini non hanno una priorità vengono scelti gli ordini più recenti e ciò significa che ordini più vecchi o più importanti vengono trascurati o ritardati.
5. **Posizioni fisse**. La presenza di posizioni fisse elimina il potenziale di flessibilità per allocare gli articoli più richiesti vicino al banco pacchi quando necessario. Comporta anche uno spazio del magazzino non ottimizzato portando un costo di denaro e risorse.
6. **Rintracciare personale che esegue un ordine**. Non si è in grado di valutare quanto sia efficace il proprio personale e ad ogni ordine ne risponde il responsabile del magazzino.

Avere un sistema di gestione del magazzino (WMS) può aiutare a risolvere i precedenti problemi visto che si può indicare la posizione esatta dell'oggetto ricercato, può essere notificato un errore che il picker sta commettendo, elimina la carta dal magazzino aumentando la precisione e il tempo dell'ordine oppure può anche consentire posizioni dinamiche utilizzando molto meglio lo spazio del magazzino. Lo sviluppo di applicazioni mobili è già un compito piuttosto impegnativo da solo, per i seguenti motivi:

- le applicazioni mobili sono guidate dagli eventi, quindi ottengono il loro input non solo dall'utente ma anche da contesti esterni (ad es. sensori di posizione, dispositivi NFC, cambi di orientamento dello schermo, ecc.)
- energia, memoria e larghezza di banda limitate del dispositivo host
- interruzioni costanti causate da eventi di sistema e di comunicazione (notifica, messaggi o chiamate in arrivo, ecc.)
- la necessità che l'interfaccia si adatti alle diverse dimensioni e risoluzioni dello schermo

Le applicazioni mobili si interfacciano direttamente con il software gestionale del magazzino, e attraverso il display virtuali vengono visualizzate le istruzioni di picking. Utilizzandole, quindi, gli addetti alla logistica di magazzino, ricevono in modo corretto tutte le informazioni su dove si trovano gli oggetti, liberandosi così da scanner manuali e liste cartacee di prelievo. I picker con le applicazioni mobili trovano il percorso più veloce per raggiungere i prodotti e possono leggere direttamente i codici a barre o i QR code. Questa tecnologia con l'ausilio della realtà aumentata verrà usata in un futuro non molto lontano da tutti e trovare proficua applicazione nella gestione logistica di magazzino, soprattutto nel settore dell'e-commerce dove è richiesta sempre maggiore velocità ed efficienza per l'accorciamento del leadtime.

1.3 Computer Vision

Comprendere come un computer sia in grado di vedere un' immagine o una sequenza di immagini è uno dei processi più complessi nella scienza computazionale. L'evoluzione della tecnologia e lo sviluppo di nuovi algoritmi sofisticati hanno permesso di raggiungere risultati importanti nella computer vision.

A differenza del processo visivo umano, i computer sono stati progettati per vedere un' immagine come un insieme di numeri come mostrato in figura 1.3. Qualsiasi immagine è composta da un insieme di pixel, ognuno dei quali rappresenta la componente elementare di un'immagine. Affermare che la dimensione dell'immagine è di 1300 x 700 significa avere 1300 pixel in orizzontale e 700 in verticale, che equivale ad avere 910.000 (1300 x 700) pixel totali. Un computer capisce che l'immagine ha dimensione 1300 x 700 tramite la matrice composta da 1300 righe e 700 colonne.
L'insieme dei pixel rappresentano l'insieme di numeri necessari a un computer per vedere l'immagine e vengono organizzati secondo una matrice 2D o 3D di numeri compresi tra 0 e 255, a seconda che essa sia in scala di grigi o a colori.

```
[[  9   1  29  70 114  76   0   8   4   5   5   0 111 162   9   8  62  62]
 [  3   0  33  61 102 106  34   0   0   0   0  49 182 150   1  12  65  62]
 [  1   0  40  54 123  90  72  77  52  51  49 121 205  98   0  15  67  59]
 [  3   1  41  57  74  54  96 181 220 170  90 149 208  56   0  16  69  59]
 [  6   1  32  36  47  81  85  90 176 206 140 171 186  22   3  15  72  63]
 [  4   1  31  39  66  71  71  97 147 214 203 190 198  22   6  17  73  65]
 [  2   3  15  30  52  57  68 123 161 197 207 200 179   8   8  18  73  66]
 [  2   2  17  37  34  40  78 103 148 187 205 225 165   1   8  19  76  68]
 [  2   3  20  44  37  34  35  26  78 156 214 145 200  38   2  21  78  69]
 [  2   2  20  34  21  43  70  21  43 139 205  93 211  70   0  23  78  72]
 [  3   4  16  24  14  21 102 175 120 130 226 212 236  75   0  25  78  72]
 [  6   5  13  21  28  28  97 216 184  90 196 255 255  84   4  24  79  74]
 [  6   5  15  25  30  39  63 105 140  66 113 252 251  74   4  28  79  75]
 [  5   5  16  32  38  57  69  85  93 120 128 251 255 154  19  26  80  76]
 [  6   5  20  42  55  62  66  76  86 104 148 242 254 241  83  26  80  77]
 [  2   3  20  38  55  64  69  80  78 109 195 247 252 255 172  40  78  77]
 [ 10   8  23  34  44  64  88 104 119 173 234 247 253 254 227  66  74  74]
 [ 32   6  24  37  45  63  85 114 154 196 226 245 251 252 250 112  66  71]]
```

Figura 1.3. Esempio di matrice 2D.

Nelle immagini in scala di grigi, ciascun pixel rappresenta l'intensità di un solo colore. In altre parole, si dice che la matrice ha un canale. La matrice 3D dell'immagine a colori, invece, include 3 canali (Fig. 1.4): quello rosso, verde e blu (dall'inglese RGB, ovvero Red, Green e Blue).
Ogni canale è composto da una matrice 2D, simile a quella della scala di grigi, e perciò avremo una matrice per il canale rosso, una per il canale verde e una per quello blu, che poi sono impilate una sopra l'altra per ottenere la matrice 3D che mostra l'immagine a colori.
Tipicamente la computer vision è basata su reti neurali (deep learning) o altri algoritmi di AI. Quindi l'obiettivo principale della computer vision non è solo "vedere", ma anche elaborare e fornire risultati su quanto osservato.

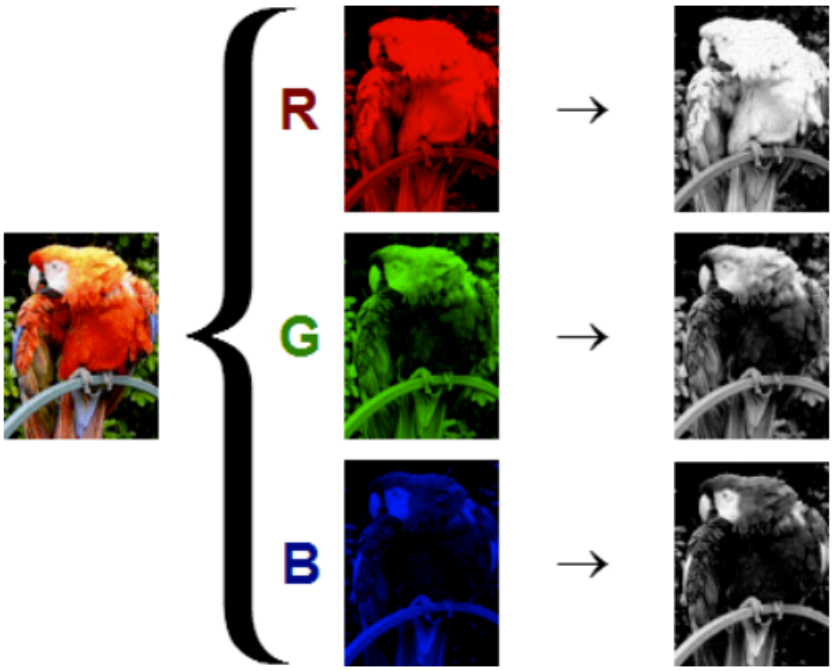

Figura 1.4. Esempio di matrice 3D.

Un sistema di computer vision dipende dallo specifico ambito di applicazione. In ogni caso, esiste sempre un insieme di funzioni tipiche:

- Acquisizione di immagini.
- Pre-processing. Prima di eseguire il training di qualsiasi algoritmo di computer vision bisogna eseguire alcune operazioni sulle immagini raccolte, come ad esempio l'eliminazione del rumore.
- Segmentation. Il processo decisionale per selezionale le aree di interesse all'interno delle immagini.
- Training. La fase di apprendimento e validazione del modello utilizzando data set composti da immagini.

In parole semplici, queste metodologie vengono utilizzate dagli algoritmi di computer vision in fase iniziale di studio dell'immagine al fine di cercare le linee che si

incontrano ad angolo e comprendendone una parte specifica dell'immagine con una sfumatura di colore. Questi angoli e caratteristiche sono gli elementi costitutivi che aiutano a trovare informazioni più dettagliate contenute nell'immagine.
Dopo aver compreso maggiormente come un computer vede e identifica un oggetto; in computer vision ci sono cinque tecniche che vengono utilizzate:

1. Classificazioni di immagini (Image Classification).Dato un insieme di immagini che sono tutte etichettate con una singola categoria, si cerca di prevedere se questa categoria appare in un nuovo set di immagini di prova misurandone l'accuratezza delle previsioni. Ad esempio, se sto studiando la categoria "gatto", cercherò di identificare in immagini differenti se esse rappresentano un "gatto" oppure no.

   Solitamente vengono studiate più di una categoria alla volta. Per risolvere questo problema, i ricercatori di visione artificiale "insegnano" all'algoritmo a riconoscere un dataset di immagini di prova, in modo che diventino abbastanza abili da prevedere con elevata probabilità di cosa si tratti, e solo dopo fanno elaborare le immagini nuove al pc.

2. Rilevazione degli oggetti (Object Detection). rappresenta la definizione degli oggetti all'interno delle immagini di solito tramite l'emissione di box ed etichette per singoli oggetti.

   Esistono problemi che classificano e localizzano molti oggetti e quelli che si focalizzano solamente su un oggetto dominante.

   Nel primo caso si cercano di identificare tutti gli oggetti rilevanti allo studio, mentre nel secondo, ad esempio se il problema tratta il rilevamento auto, l'algoritmo si concentrerà sul rilevare tutte le auto in una determinata immagine con i loro riquadri di delimitazione.

3. Tracciamento oggetti (Object Tracking).si riferisce al processo di seguire uno specifico oggetto di interesse, o più oggetti, in una determinata scena.

   Tradizionalmente ha applicazioni nelle interazioni video e nel mondo reale dove le osservazioni sono fatte dopo un rilevamento iniziale dell'oggetto.

4. Segmentazione semantica. Processo fondamentale della Computer Vision è il processo di segmentazione, che divide intere immagini in gruppi di pixel che possono quindi essere etichettati e classificati.

In particolare, la segmentazione semantica cerca di capire il ruolo di ciascun pixel nell'immagine.

5. Segmentazione dell' istanza. La segmentazione dell'istanza segmenta diverse istanze di classi, come l'etichettatura di 10 persone con 10 colori diversi. Nella classificazione, generalmente è presente un'immagine con un singolo oggetto come focus e il compito è dire cosa rappresenta l'immagine.

   Per segmentare le istanze, solitamente occorre svolgere compiti molto più complessi. Si vedono viste complicate con più oggetti sovrapposti e sfondi diversi, e non solo vengono classificati questi diversi oggetti, ma si identificano anche i loro confini, differenze e relazioni tra loro!

Quando si parla di computer vision tipicamente si tende a pensare solo ad applicazioni di riconoscimento/classificazione di oggetti o di videosorveglianza. In campo industriale la visione artificiale ha riscosso un notevole successo in quanto ha permesso l'abbattimento dei costi e la completa automatizzazione del controllo di qualità, con migliori risultati. I principali ambiti di applicazione riguardano:

- **Manutenzione predittiva**.
- **Controllo di processo**. La computer vision gioca un ruolo fondamentale nella guida dei robot in quanto permette di controllarne l'orientamento e la posizione nel campo operativo, nonché di rilevare eventi e anomalie per una completa analisi del flusso produttivo.
- **Metrologia**. Sfruttando la computer vision si possono effettuare misure automatiche in assenza di contatto tra oggetto e strumento di misura, garantendo accuratezza, ripetibilità e riproducibilità e un vantaggio in termine di riduzione dei tempi di processo.
- **Controllo di qualità**. L'uso di sistemi di computer vision consente il rilevamento ed il riconoscimento di difetti e la verifica del rispetto di tolleranze. Inoltre facilita il conteggio e la classificazione di prodotti in base a predeterminati parametri di qualità (misura, numero e tipologia di difetti).
- **Riduzione di difetti**. Qualsiasi azienda ha come obiettivo primario la creazione di componenti o prodotti privi di difetti. La tecnologia di sicuro aiuta in questa direzione, quantomeno riducendo drasticamente la probabilità di difetti.

- **Ispezione di package**. In alcuni settori, come l'industria farmaceutica per esempio, l'ispezione di pillole o capsule. Gran parte delle soluzioni per questo tipo di controlli è attualmente basata su computer vision. In questo modo è possibile il rigetto automatico delle confezioni difettose.
- **Letture di codici a barre e QR codes**.
- **Lettura di testi**.

Queste ed altre applicazioni vengono raggiunte grazie allo svolgimento di specifiche operazioni che mettono in gioco la capacità visiva del sistema; tra le più comuni si possono ricordare la ricostruzione di scene, il video tracking, l'inseguimento di un contorno, il rilevamento di eventi.

1.4 Obiettivi

Lo scopo di questo lavoro di tesi è di fornire una soluzione per il picking usando un'interfaccia uomo-macchina. Si propone di progettare un prototipo di sistema per le attività di order picking di magazzino, ovvero la guida di un operatore verso un determinato prodotto.
Il focus è la guida dell'operatore nella ricerca di articoli da prelevare sulla base di una lista, tanto più impegnativa quanto è maggiore il numero di prodotti gestiti, e deve essere considerata dal punto di vista del costo, tempo di risposta e accuratezza. Le attività di sviluppo si dovranno concentrare sulla realizzazione di una app in grado di rintracciare un determinato scaffale e visualizzarlo per mezzo della computer vision evidenziando il ripiano da cui prelevare il prodotto. L'app, attraverso una lettura QRcode, si occuperà inoltre di fare una serie di verifiche sull'articolo prelevato (a beneficio delle operazioni di scarico di magazzino, ecc.).
L'obiettivo di questa tesi inoltre è trovare una soluzione per superare gli inconvenienti dei tools in circolazione. Questi hanno come svantaggi la flessibilità in caso di riorganizzazione del magazzino oppure causa all'operatore un maggiore sforzo cognitivo. La soluzione punta a elementi d'innovazione quali:

- smartphone/smartwatch/tablet per la localizzazione, analisi dell'immagine per il riconoscimento del ripiano in real-time e, in futuro, per il controllo ottico dei prodotti

- marcatori QRCode per identificare in maniera univoca lo scaffale

- tag rfid passivi da posizionare su ogni ripiano (il costo di un tag rfid passivo è oggi nell'ordine dei centesimi di euro, il che permette il loro uso massivo ed eventualmente a perdere). Per una maggiore accuratezza, si potrebbe anche valutare la possibilità di associare un tag rfid anche al singolo prodotto

- sistema informativo centrale (SIC): contiene le informazioni necessarie: prodotti, posizione nel supermercato (coordinate in grado associare al prodotto una determinata posizione data da corsia/scaffale/ripiano)

1.5 Stato dell'arte

I lavoratori tradizionali eseguono gli ordini attraverso elenchi cartacei che sono intuitivi per gli esseri umani ma laboriosi da gestire. Le tecniche moderne, che non usano il lavoro d'ufficio, includono dispositivi mobili per l'immissione di dati, che hanno un elevato sforzo di gestione,ma sono collegati online al sistema di elaborazione dei dati del magazzino. Adesso i sistemi moderni hanno sostituito gli elenchi cartacei con sistemi *Pick-by-Voice* (PbV), *Agumented Reality* (AR) o *Pick-by-Light* (PbL) .

- PbV supporta il lavoratore dandogli tutte le istruzioni attraverso l'output di un computer ma sfortunatamente questo sistema presenta difficoltà in ambienti industriali a causa del rumore. In aggiunta è discutibile che un operatore sia comandato da una voce monotona e robotica.

- AR è una tecnologia che può supportare il senso visivo umano attraverso la combinazione del mondo reale e virtuale con l'interazione 3D in tempo reale. Questa soluzione richiede un sistema di tracciamento per il posizionamento. Si è dimostrato che tramite l'uso di AR i soggetti erano più veloci e hanno commesso meno errori(Walch, 2007).

- PbL offre all'utente un aiuto visivo installando piccole lampade su ciascun vano porta oggetti. I sistemi PbL però hanno il problema che i display devono essere integrati nella loro installazione e quindi non sono flessibili quando si deve cambiare disposizione del magazzino. I sistemi electronic paperless

pick-to-light, noti anche come computer aided picking system (CAPS) e migliorano la produttività (anche più del 50%), riducono gli errori dei picker e ne semplificano l'addestramento, riducendo molto i costi operativi.

1.5.1 Pick to voice

Il picking vocale è un sistema paperless che impiega istruzione vocali di facile comprensione per dirigere gli operatori di un magazzino. I sistemi di selezione vocale sono progettati per mantenere le mani e gli occhi degli operatori liberi, senza premere pulsanti o completare altre attività di sistema che potrebbero distrarli dalla raccolta. Ogni operatore è dotato di un dispositivo di selezione vocale come cuffie, microfono e un terminale per la voce come mostrato in figura 1.5.

Figura 1.5. Flusso di esecuzione di un pick-to-voice.

Spesso gli operatori portano uno scanner di codici a barre per una maggiore efficienza. Il dispositivo di selezione vocale dell'operatore pronuncerà le istruzioni su quali operazioni di selezioni devono essere eseguite e dove. Quando il sistema di selezione vocale dirige verbalmente un operatore in una posizione specifica e successivamente deve confermare che si trova nella corsia e davanti il cestino corretto. Il sistema di selezione vocale confermerà la posizione e fornirà quindi all'operatore la quantità di prelievo. Una volta completata la raccolta il sistema manderà l'operatore alla posizione di raccolta successiva. In particolare il sistema ottimizzerà il percorso di prelievo durante il processo per massimizzare l'efficienza. I vantaggi dei sistemi di selezione vocale andranno dal miglioramento dal ritmo di lavoro alla

riduzione degli errori. Viene dimostrato, inoltre, che i sistemi vocali richiedono una formazione minima per poter familiarizzare con la tecnologia. Con questo tipo di strumentazione venivano risolti diversi problemi:

- Bassa precisione di scelta
- Bassa produttività
- Operazioni limitate nel magazzino
- Layout del magazzino
- Elevato turnover del personale

Figura 1.6. Esempio di attrezzatura per il pick-to-voice.

1.5.2 Pick to light

Il pick to light è una delle tecniche che fa parte del tipo di picking denominato *picker-to-part* dove è l'operatore ad andare verso l'oggetto, esegue il picking e completa l'ordine prelevando un oggetto dopo l'altro. In questo caso un pulsante con

luce è associato ad ogni contenitore di oggetti uguali. La luce viene accesa dal sistema informativo per segnalare che l'oggetto deve essere prelevato. L'operatore segue la luce per localizzare l'oggetto, lo preleva e spinge il pulsante associato alla luce per segnalare l'avvenuto picking. Lo svantaggio più consistente del pick to light è la mancanza di flessibilità in caso di riorganizzazione fisica del magazzino: lo spostamento di un contenitore di oggetti da una posizione ad un'altra nel magazzino comporta lo spostamento dei pulsanti, e relativi cablaggi.

Pick to light cablati

Sono i primi sistemi apparsi sul mercato. Il sistema è facile, dato che ogni scaffale ha un dispositivo che si illumina in modo da indicare dove prelevare (Fig. 1.7). Nonostante offrono tempi di aggiornamento dati e accensione luce molto veloci e nessun problema di batteria dato che sono alimentati via cavo. Dall'altro lato hanno un significato costo di infrastruttura a causa del cablaggio degli scaffali.

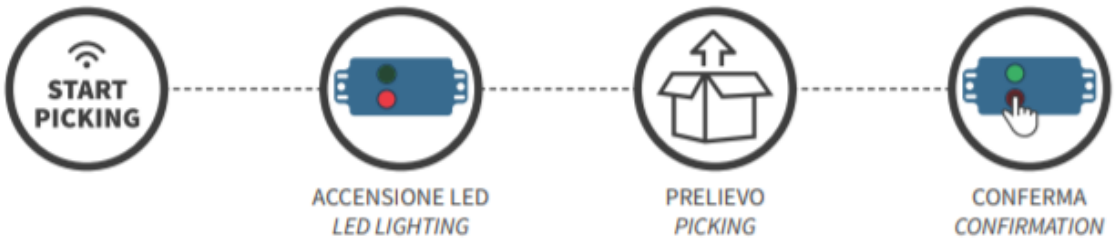

Figura 1.7. Flusso di un sistema pick-to-light cablato.

Pick-to-light wireless

La versione wireless presenta dei vantaggi rispetto alla soluzione cablata grazie alla comunicazione bidirezionale in radio frequenza che consente di azzerare i costi di cablaggio. Questa soluzione è particolarmente indicata negli e-commerce. Una volta scannerizzato il codice a barre viene demandato al server che contiene la posizione dei vari oggetti. Tramite l'antenna viene avvisato quale luce deve accendersi indicando di conseguenza al picker dove andare a prelevare (Fig. 1.8).

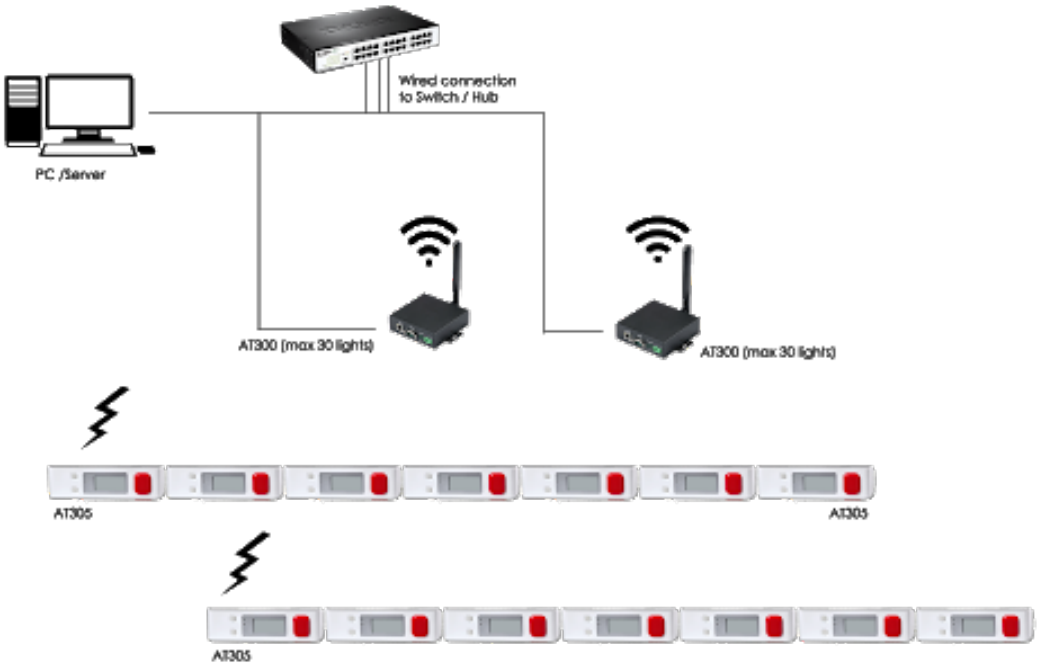

Figura 1.8. Flusso di un sistema pick-to-light wireless.

Pick-to-light con etichette elettroniche Wi-Fi

Grazie ai display consentono di visualizzare qualsiasi tipo di informazione (quantità, barcode, QRcode, immagini e ubicazioni). La conferma delle operazioni di picking può essere effettuata in varie modalità: lettura di barcode, comando vocale, sensoristica. La giacenza viene ogni volta aggiornata e compare sull' etichetta elettronica. Il sistema è composto da una antenna master che gestisce i pulsanti luminosi tramite Wi-Fi.

Riassiumiamo i punti salienti del confronto tra questi sistemi

1.5.3 Pick to light gloves

I dispositivi sono sono particolarmente vantaggiosi per i processi di assemblaggio nella produzione manuale o per i processi di prelievo senza carta. Attraverso il guanto con un' antenna integrata a livello del polso comunica automaticamente

Sistema	**Cablato**	**Wi-Fi**	**Etichette elettroniche**
Display LCD	Si	Si	No
Display grafico	Si/No	Si/No	Si
Formati di display grafici disponibili	Non competitivo	Non competitivo	Ottimo
Luce singola	Si	Si	Si
Luce RGB	Si/No	Si/No	Si
Gestione multioperatore	Si/No	Si/No	Si/No
Conferma picking	Pulsante	Pulsante	Barcode
Durata batterie	n.d.	Poco competitivo	Ottimo
Flessibilità layout scaffale	Non competitivo	Ottimo	Ottimo
Velocità accensione luci	Ottimo	Medio	Medio

Tabella 1.1. Confronto sistemi pick to light

con il transponder sulla merce e trasmette le merci e le informazioni di processo a un sistema centrale in tempo reale. Ciò significa che i prelievi parziali durante il montaggio e nel processo di approvvigionamento possono essere registrati automaticamente. Un sistema UHF RFID è integrato nella custodia del guanto RFID (banda di frequenza tra 865 e 869 MHz). Il guanto ha anche interfacce radio standardizzate come WLAN per la trasmissione di dati e la fornitura di energia. Tutti gli elementi sono fissati in modo che il polso possa muoversi in modo flessibile e l'utente non venga ostacolato durante il suo lavoro come mostrato in figura 1.9. Il sistema RFID è quindi costituito da un transponder RFID (con bobina d'antenna, circuiti e memoria) con un numero di identificazione o un codice di prodotto elettronico (EPC), un lettore che può essere utilizzato per leggere le informazioni e un'unità di trasmissione per la trasmissione dei dati.

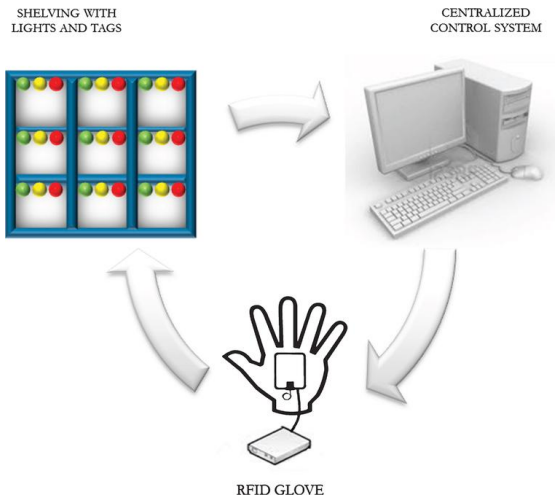

Figura 1.9. Esempio di un sistema con gloves RFID

1.5.4 Pick to vision

Uso di HMD

L'HMD (Head-Mounted Displays) è l'hardware più importante per il *Pick-by-Vision* perché è l'interfaccia tra il sistema umano e il sistema tecnico. Il suo compito è quello di visualizzare le informazioni necessarie per l'order picker. Un aspetto è l'ergonomia del dispositivo per questo motivo dovrebbe essere leggero ma anche robusto e con un funzionamento della batteria di otto ore. La maggior parte dei lavoratori possono immaginare di lavorare con un HMD per un giorno o anche di più per ulteriori valutazioni. Ma ci sono anche i lavoratori che generalmente non amano l'HMD.
Il sistema di localizzazione è il componente hardware più problematico di un'applicazione AR mobile. Molti fattori diversi come gradi di libertà, precisione, risoluzione, frequenza di aggiornamento e portata caratterizzano i sistemi di tracciamento e ci sono molti principi funzionali diversi come i sistemi elettromagnetici, inerziali, meccanici, ottici, radio o ultrasonici (Rolland et al., 2001). Lo scopo è quello di arrivare allo scaffale giusto (coarse navigation) per poi evidenziare la casella effettiva da cui prelevare (fine navigation) con l'aiuto di una visualizzazione chiamata meta visualization. Questa visualizzazione serve per indicare l'attenzione dell'utente su oggetti che al momento non rientrano nel campo visivo dell 'HMD. Questo problema di guidare un utente verso oggetti in posti arbitrari è stato affrontato

inizialmente da Feiner, Macintyre, and Seligmann con una linea tratteggiata combinata con l'evidenza dell' oggetto e successivamente Höllerer, Pavlik, and Feiner visualizzavando un puntatore 2D a forma di cono in un HMD tracciato per puntare a una vista nascosta dedicata punti di riferimento nell'ambiente dell'utente. Funziona come una bussola che è fissata al fondo dell'HMD.

Smart glasses

Gli occhiali smart glasses si interfacciano con il software gestionale e visualizzano le istruzioni di picking attraverso i loro display virtuali. In questo modo un operatore riceve in modo corretto le informazioni su dove si trovano gli oggetti e dove devono essere posizionati sul carrello (Fig. 1.10). I picker vengono aiutati a trovare il percorso più veloce per raggiungere i prodotti scansionando direttamente i codici a barre o i QR code.

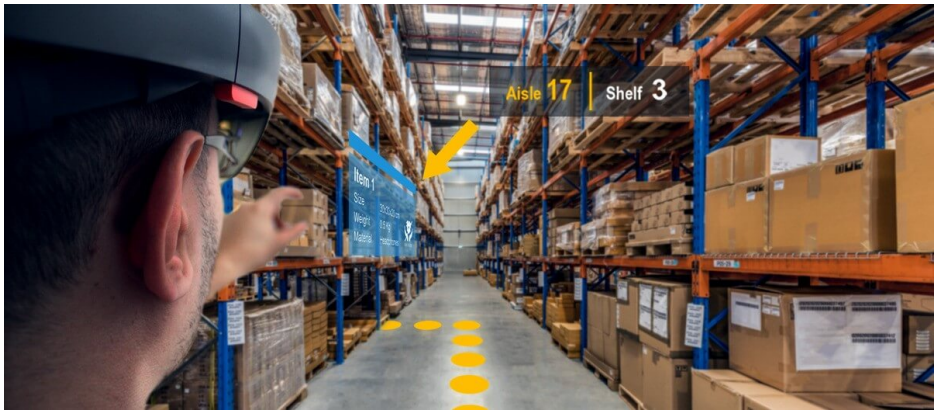

Figura 1.10. Esempio di un sistema con smart glasses

	Paper	**Pick-by-Light**	**Pick-by-Voice**	**Pick-by-Vision**
Senza carta	No	Si	Si	Si
RealTime information	No	Si	Si	Si
Hands free	No	Si	Si	Si
Prossimo pezzo	Si	Si	No	No
Precisione	No	Si	Si	Si
Mature technology	Si	Si	Si	No

Tabella 1.2. Tecnologie a confronto

1.6 Errori e tempo di raccolta

DHL Supply Chain ha già sperimentato soluzioni di questo tipo nei Paesi Bassi dove i dipendenti del magazzino di Bergen della Ricoh, azienda produttrice di stampanti e fotocopiatrici, sono stati in grado di ridurre del 25% il tempo necessario per prelevare i prodotti dagli scaffali e metterli in spedizione, dice Jan-Willem De Jong, Business Unit Director Technology presso DHL Supply Chain Benelux. Il tasso di errore dipende dalla tecnologia di picking.
Per un elenco cartaceo normalmente si aggira intorno lo 0,35%. Mentre con Il PbL e il PbV è rispettivamente 0,40% e 0,08%(Ten Hompel & Schmidt, 2008). Con il Pick-by-Vision è stato commesso un solo errore ogn i1904 righe di ordini prelevati. I tassi di errore sono soggetti alla distribuzione di Poisson perché il conteggio degli errori può assumere solo valori interi, non negativi ed errori di prelievo sono un evento raro.
Con i sistemi Pick-by-Vision i soggetti riuscivano ad essere il 4% più veloci rispetto all'elenco cartaceo.

1.7 Riguardo questa App

L'app sviluppata vuole supportare l'operatore a velocizzare l'operazione di picking in modo paperless ma con l'ausilio di un tablet/smartphone. Gli errori hanno una forte influenza sulla qualità della consegna e sul rapporto tra clienti e fornitori. La raccolta a *zero-defect* è un obiettivo importante e per fare che ciò avvenga si vuole automatizzare il processo dato che questa è l'ultima fase del processo prima che la merce venga consegnata ai clienti. Analizzando il pick to light cablato, si è detto, che il problema più grande è la mancanza di dinamicità nel layout del magazzino che ormai è fondamentale nell' industria 4.0. Se invece si analizzano gli altri due tipi di pick to light si vorrebbe eliminare del tutto la sensoristica e i display per minimizzare al massimo i costi di apparecchi per ogni scaffale. Lo stesso vale per il pick to voice che non solo l'operatore deve fare uno sforzo cognitivo in più ma può essere infastidito di interfacciarsi tutti i giorni con una voce metallica e soprattutto ricevere ordini da un computer vocale. La realtà virtuale è uno strumento innovativo molto coinvolgente ed è usato in molti campi. Purtroppo non tutti sono in grado di sopportare per tutte le ore di lavoro l' HMD o i smart glasses perchè l'operatore potrebbe provare una sensazione di disorientamento a

causa della distanza presente tra la videocamera e i suoi occhi. In alcuni studi è stato rilevato che vengono riconosciuti problemi definiti *cybersickness*, cioè sintomi di cinetosi dovuti all'immersione. Questo problema porta ad avere sensazioni di mal di testa, problemi alla vista e nausea.
Preso spunto da ognuno delle tecniche in circolazione, si è cercato di eliminare anche questi problemi. con l'app sviluppata si può avere un riscontro della persona che la sta usando e quindi la responsabilità non è concentrata sul responsabile del magazzino ma è distribuita a ogni operatore. Abbiamo anche voluto avere un server per gestire il magazzino in modo più semplice. In questo modo abbiamo un database di riferimento che ci indica la posizione dell' oggetto da voler recuperare. La dinamicità del layout non è compromessa, infatti basata aggiornare il database per la posizione dell'oggetto in modo che possiamo spostare in avanti gli oggetti più richiesti quando vogliamo. L'uso di uno smartphone o tablet è anche meno intrusivo e comunque pratico e facile da imparare. Infatti sarà il device a fungere da scanner per il QRcode e sul display comparirà la lista di oggetti che necessitano per concludere l'ordine. In più è stato pensata la duttilità che un operatore deve avere e quindi se per caso non riuscisse a finire l'ordine per qualsiasi motivo può riprendere tutto da dove aveva lasciato. Questo accade grazie al fatto che ogni volta che un prelievo è stato fatto bisogna notificarlo. Di solito venivano usati dei pulsanti o dei comandi vocali; ma si è voluto sfruttare la tecnologia già presente sui device e quindi grazie all' NFC possiamo notificare al sistema l'avvenuto prelievo.

1.7.1 Tool usati in questa App

Per sviluppare l' applicazione mobile ho usato diversi tools per svolgere le varie attività presenti in app. Innanzitutto, ho usato Android studio (Fig.1.11) come sistema di sviluppo avendo usato smartphone con sistema operativo Android.
Android è sicuramente uno dei sistemi operativi più versatili infatti è possibile sviluppare applicazioni in maniera semplice con costi ridotti o addirittura assenti. Tra i software open source più apprezzati c'è sicuramente Android Studio. Android studio è un ambiente di sviluppo integrato (IDE, integrated development enviroment) adibito per la creazione di applicazioni Android. È disponibile il download su Windows, Mac OS X e Linux e sostituisce gli Android Development Tools (ADT) di Eclipse, diventando l'IDE primario di Google per lo sviluppo nativo di applicazioni Android.

Tra le sue particolarità, oltre ad essere efficiente, si può riscontrare una maggiore

Feature	Android Studio	ADT
Build system	Gradle	Apache ANT
Maven-based build dependecies	Yes	No
Build variants and multiple-APK generation	Yes	No
Advanced Android code completion and refactoring	Yes	No
Graphical layout editor	Yes	Yes
APK signing and keystore management	Yes	Yes
NDK support	Coming soon	Yes

Tabella 1.3. Differenze tra Android Studio e ADT

facilità nel sviluppare applicativi. Basato sull'editor IntelliJ IDEA, molto popolare nel mondo Java, in maniera del tutto gratuita. Le ultime versioni, tra l'altro, hanno integrato il pacchetto JDK cosicché non si debba scaricare e installarlo. Android Studio permette di provare le app che si sviluppano su un emulatore software, utilizzando i virtual device (AVD), oppure direttamente su un device reale. Android Studio da la possibilità di programmare in un moderno linguaggio orientato a oggetti come Java, C#, C++, VB.

Figura 1.11. Logo Android Studio

Questo tool offre:

- Un sistema di compilazione basato su Gradle.
- Possibilità di costruire APK varianti e multipli.
- Ambiente di layout con editing.
- Supporto ai servizi Google e a vari tipi di dispositivi

- Funzionalità ProGuard e app-signing
- Strumenti per la cattura di prestazioni, usabilità, compatibilità di versione, e altri problemi.

Su Android Studio sono state aggiunte diverse librerie per svolgere delle attività. Una di queste è OpenCV (Fig.1.13), molto utile ai fini dell'applicazione. Infatti l'intelligenza artificiale (AI) è molto richiesta in questo periodo tecnologico.
Come è possibile vedere in figura 1.12 OpenCV nasce nel 1999 nei laboratori di Intel con l'aiuto di Gary Brandski, come strumento di test per l'analisi delle CPU in fase di sviluppo. L'elaborazione di immagini è bene noto essere una delle applicazioni che sfrutta molto la potenza di calcolo a causa dell'enorme quantità di dati da elaborare in tempi brevi. Nel 2000 è stata rilasciata al pubblico la prima visione durante la conferenza internazionale "Conference on Computer Vision and Pattern Recognition" organizzata dalla IEEE. Inizialmente OpenCV aveva tre scopi:

1. Ricerca avanzata sulla Visione Artificiale grazie ad un codice ottimizzato.
2. Diffondere conoscenza sulla Visione Artificiale fornendo un'infrastruttura comune agli sviluppatori in modo da poter sviluppare codice leggibile.
3. Creazione di applicazioni avanzate grazie a codice portabile.

Tra il 2001 e il 2005 sono seguite diverse versioni fino ad arrivare alla prima Release nel 2006.

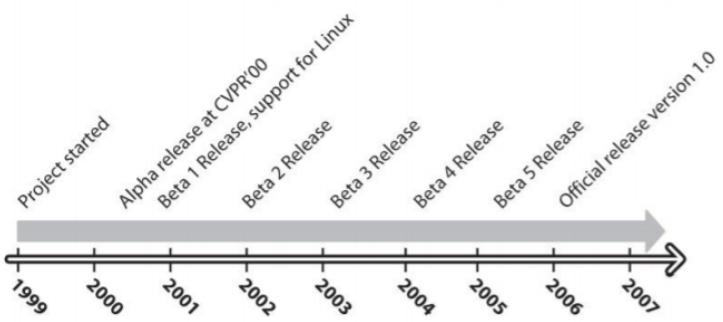

Figura 1.12. OpenCV time-line

Vadim Pisarevsky si è unito a Gary Bradsky nei laboratori di Intel per portare

avanti lo sviluppo della libreria. Nel susseguirsi degli anni e delle versioni il Team di OpenCV lasciò Intel e si spostò dalla compagnia in centri di ricerca, fino ad arrivare a Willow Garage, attuale "manutentore". Nel 2009 ci fu l'uscita della Release 2.0 dove venne aggiunto una gestione della memoria dinamica grazie alla gestione simile alla *Garbage Collection* tipica del linguaggio Java.
OpenCV è una libreria open source usata per la computer vision e il machine learning. Questo tool è stato creato per fornire un' infrastruttura alle applicazioni di computer vision e per accelerarne l'uso nei prodotti commerciali. La libreria ha più di 2500 algoritmi ottimizzati, usati per rilevare e riconoscere volti, identificare oggetti, classificare azioni umane nei video, tracciare oggetti in movimento, ecc. OpenCV ha interfacce C++, Python, Java e MATLAB ed è suportata da Mac, Android, Windows e Linux. Molte delle tecniche di image processing sono impiegate nel miglioramento delle immagini, nell'analisi dell'immagine e più in generale come passo di pre-elaborazione nei procedimenti di computer vision. OpenCv è una libreria scritta in C e C++, costituita da oltre 500 funzioni utili nel campo dell'image processing. OpenCV costituisce quindi una infrastruttura aperta e gratuita, compatibile con la Intel Image Processing Library (IPL). Quest'ultima è una libreria non open source che esegue solo operazioni di image processing e dalla quale OpenCV ha ereditato originariamente alcune strutture dati.

Figura 1.13. Logo OpenCV

La computer vision è la trasformazione dei dati prelevati da un' immagine o da un video in un nuovo tipo di rappresentazione. Queste trasformazioni sono eseguite per arrivare a particolari obbiettivi. La computer vision non è affatto un meccanismo semplice da far funzionare infatti l'elaboratore riceve una griglia di numeri dalla fotocamera. Avvincente è quanto riportato da Microsoft sulla sezione del sito

dedicato alla ricerca sulle tecnologie visive: "Lo scopo della ricerca sulla computer vision è di dotare il computer dell'abilità di comprendere le immagini ferme e in movimento. Anche se noi, come esseri umani, possiamo dare un senso alle fotografie e ai video, per un computer esse sono solo una matrice di numeri rappresentante la luminanza e il colore di un pixel. Come possiamo ottenere da questa matrice di numeri la comprensione che c'è una ragazza che sta giocando a pallone davanti all'edificio? Quanto alto sta gettando la palla? Qual è la struttura di fondo della casa? Questi sono gli argomenti che ci interessano!".
OpenCV è volto a fornire gli strumenti di base necessari per risolvere i problemi della computer vision.
L'accesso alla rete è un' attività comune tra le app Android. Siccome questo tipo di operazione e le necessità di svolgerle correttamente, è stata predisposta una libreria chiamata Volley. Questa libreria cura l'accesso alla rete:

- Gestione autonoma delle richieste e connessioni multiple.
- Varie classi per il supporto dei tipi più comuni di richieste.
- Caching delle risposte sia in memoria che su disco.
- Gestione delle priorità.
- strumenti di log, debug e tracciamento delle attività.

Per usare Volley bisogna impostare il permesso a internet nel file *AndroidManifest.xml*. Prima di tutto, visto che viene usato Volley, bisogna confrontarsi con due concetti fondamentali: *Request* e *RequestQueue*. Il protoccolo Http è basato sull' interazione tra client e server in termini di richiesta-risposta. Con un derivato della classe Request verrà formulata la richiesta e verrò accodata nella RequestQueue. Quindi non verrà eseguita la richiesta immediatamente ma verrà inserita in una coda. Sarà poi Volley ad eseguirla appena possibile secondo le sue politiche e le condizioni del sistema. Per ogni Request ci saranno due tipi di listener; il primo per la richiesta avvenuta con successo e il secondo per l'errore. Per l' ErrorListener verrà utilizzato un oggetto di tipo VolleyError che non è altro che un eccezione che maschera l'errore HTTP. Da ogni richiesta Volley si prende l'url per accedere direttamente alla response, un Json mandato dal server.
Il JSON (JavaScript Object Notation) è un formato semplice per lo scambio di dati. Diventa molto facile da scrive e leggere per le persone ed è altrettanto facile

da generare e analizzare la sintassi. JSON è un formato di testo completamente indipendente dal linguaggio di programmazione. JSON è basato su due strutture:

1. Un insieme di coppie nome-valore (oggetto).
2. Un elenco ordinato di valori (array).

Per fare il parsing viene usata la libreria Gson sviluppata da Google, che rende semplice e veloce la conversione tra gli oggetti Java e la rappresentazione JSON. Originariamente la libreria fu creata per essere ustata solo nei progetti Google, in seguito è stata resa disponibile agli sviluppatori. Ormai è diventata un esigenza quella di serializzare e deserializzare oggetti in formato JSON. Le caratteristiche principali di GSON sono le seguenti:

- Semplice e veloce da utilizzare.
- È possibile creare degli adapter personalizzati per definire la modalità di serializzazione degli oggetti.
- Ampio supporto di Java Generics
- Indipendente dai file sorgenti. Quasi tutte le librerie analoghe, invece, richiedono l'inserimento di alcune annotazioni nei bean che occorre serializzare.

Sempre lato app, viene caricata un'altra libreria importante per il funzionamento. Infatti, come già detto, si vuole integrare anche lo scan del QRcode. Il Qrcode è un codice a barre quadrato dove si possono essere scritte diverse informazioni come indirizzo URL, coordinate geografiche, informazioni e tanto altro; potendo memorizzare fino a un massimo di 4.296 caratteri alfanumerici o 7.089 caratteri numerici. I codici sono fatti di quadratini bianchi e neri, ognuno di questi è un modulo. Alcuni moduli non devono essere coperti o modificati, altrimenti il codice non verrà letto.

Sono stati evidenziati, nella figura 1.14, diversi colori:

- I tre grandi quadrati evidenziati in rosso indicano allo scanner i margini del QRcode.
- Il quadrato piccolo evidenziato in rosso è di allineamento.

Figura 1.14. QRcode

- Le strisce rosse che evidenziano quadratini neri e bianchi definiscono le posizioni di righe e colonne.
- Le sezioni verdi determinano il formato.
- I moduli evidenziati in blu rappresentano il numero della versione.

Normalmente vengono letti con dei lettori di codice a barre ma vengono usati sempre di più i cellulari per leggere queste informazioni.
Zxing (Fig.1.15) permette la lettura dei codici a barre tradizionali o a due dimensioni, i quali vengono interpretati attraverso la fotocamera integrata negli smartphone. Nel caso di questo lavoro, l'applicazione richiama Barcode Scanner che restituirà un numero che sarà la prima volta il codice dell' ordine e la seconda volta il numero dello scaffale.

Figura 1.15. Logo Zxing

Quanto visto fino ad ora riguardava la parte di front-end; adesso parleremo di come è stato architettato il back-end.

Nel contesto dello sviluppo Web, di solito quando parliamo di un'API RESTful ci riferiamo a servizi Web (o API Web). È un modo comune per esporre parti della tua applicazione a terze parti (applicazioni esterne e siti Web). Può essere orientato ai dati, nel senso che il tuo servizio Web (l'API RESTful) rende semplicemente disponibili le informazioni archiviate nei tuoi database utilizzando un formato comune, come XML o JSON. In questo modo, un'applicazione esterna può interagire con l'applicazione e i dati dell'utente, senza la necessità di connettersi direttamente al database.
Django è un server-side web framework scritto in Python per la creazione e la gestione di API web. Django si basa sul paradigma MTV (Model-Template-View) (Fig. 1.16):

- **Model**: è il livello del framework che rappresenta i dati, come accedervi, le loro validazioni e le realzioni tra entità differenti. E' una interfaccia che estrae sul contenuto delle tabelle del database della web application.
- **Template**: è il livello di presentazione, definisce cme i dati devono essere mostrati.
- **View**: è il livello che contiene la logica ed è il ponte tra modello e template. Riceve richieste HTTP e restituisce risposte.

Come tutte le architetture client / server, Django utilizza oggetti richiesta e risposta per comunicare tra il client e il server. Poiché Django è un framework Web, stiamo parlando di oggetti HTTP richiesta e risposta. Pertanto la vista recupera i dati del database tramite il modello, li formatta e li raggruppa in un oggetto response HTTP inviandoli al client come è possibile vedere in figura 1.17.

Abbiamo detto che i dati vengono recuperati attraverso i Serializers e le View. I serializzatori permettono di mappare dati complessi (come queryset e istanze di modelli) in tipi nativi del Python, i quali sono facilmente traducibili in JSON, XML o altri content types. Inoltre è possibile effettuare l'operazione inversa.
All'interno di una view è necessario soltanto definire i permessi, il modello su cui agire e il serializzatore che deve essere utilizzato per impacchettare i dati da tornare al client.
Il modello con cui Django si interfaccia alla base di dati segue il principio del Object Relational Mapping (ORM). Questo modello fornisce attraverso un'interfaccia

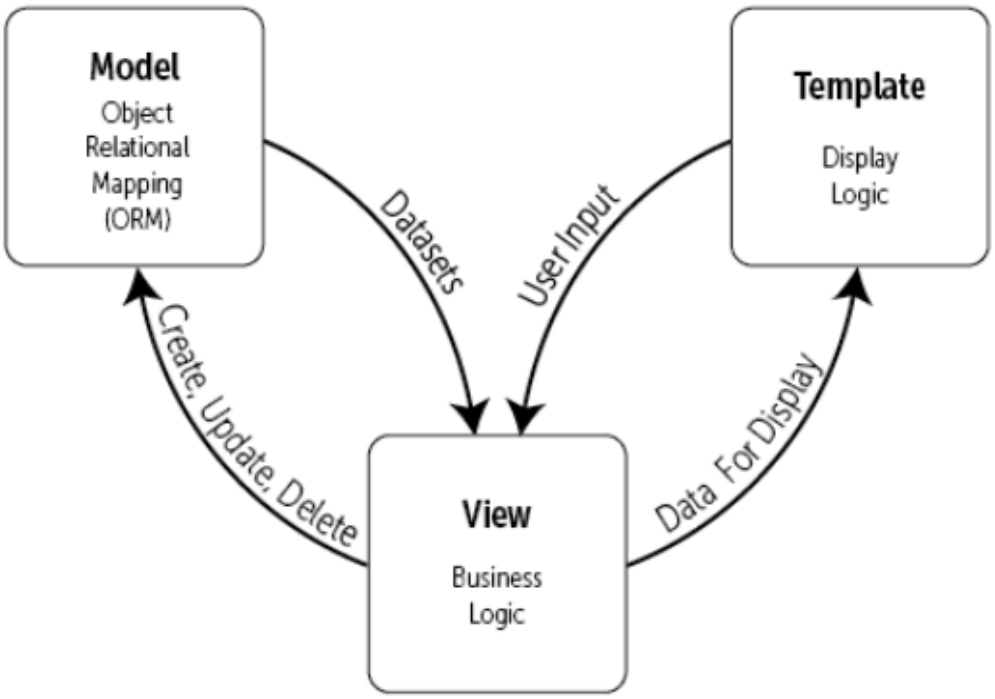

Figura 1.16. Diagramma MTV Django

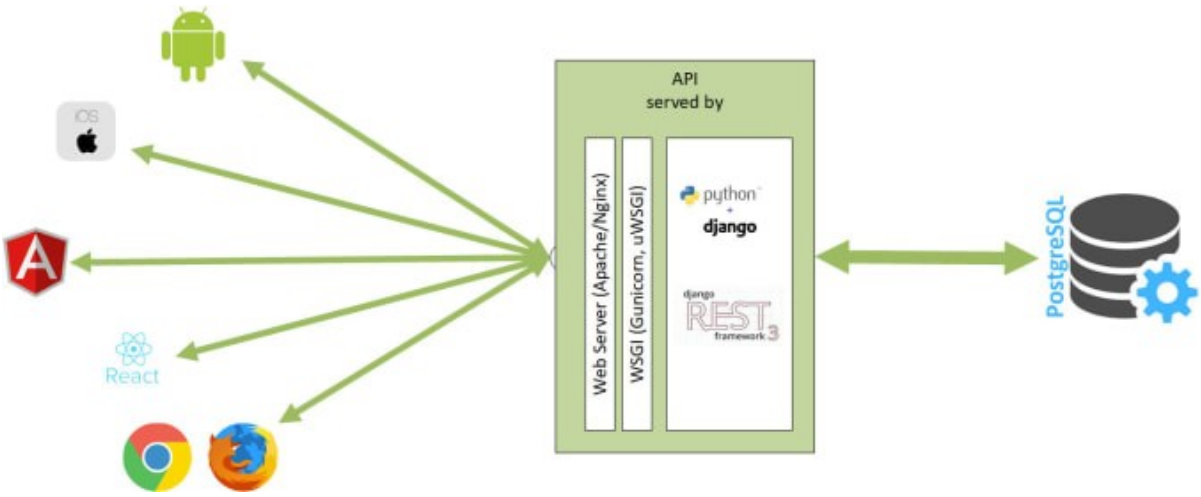

Figura 1.17. Django e postgres

orientata agli oggetti, tutti i servizi inerenti alla persistenza dei dati, astraendo nel contempo le caratteristiche implementative dello specifico database utilizzato. I principali vantaggi forniti da questa tecnica di programmazione sono:

- Permette un notevole riduzione della quantità di codice sorgente da redigere. L'ORM maschera dietro semplici operazioni le complesse attivit'a di creazione, prelievo, aggiornamento ed eliminazione dei dati dal database. Tali attività occupano di solito una buona percentuale del tempo di stesura, testing e manutenzione complessivo.

- Permette di vincolare a livello di classe i tipi degli oggetti che verranno poi creati nel database senza preoccuparsi delle conversioni di tipo.

- Permette di gestire le transizioni mediante l'uso del design pattern Unit of Work, che ritarda tutte le azioni di aggiornamento dei dati al momento della chiusura della conversazione; in questo modo le richieste inviate al database sono quelle strettamente indispensabili.

Per quanto riguarda il database è stato usato PostgreSQL (Fig. 1.18), il quale utilizzo è stato fondamentale e di rilievo per l'uso dei dati. PostgreSQL è il più completo database server open source. L'antenato di PostgreSQL è stato Ingres, sviluppato alla Università di Berkeley in California. Due studenti post-laure, Jolly Chen e Andrew Yu, aggiundero più tardi un parser SQL a Postgres. Nell'estate del 1996, dopo l'esigenza di un database SQL open-source, si formò un gruppo per lo sviluppo di PostgreSQL. Un rapido esame di PostgreSQL potrebbe sembrare che sia simile ad altri database. sso usa il linguaggio SQL per eseguire le query sui dati. Questi sono conservati come una serie di tabelle con chiavi esterne che servono a collegare i dati correlati. PostgreSQL rende più semplice costruire applicazioni per il mondo reale ,grazie alla sua programmabilità, utilizzando i dati prelevati dal database. I database SQL conservano i dati semplici dentro "flat table", richiedendo che sia l'utente a prelevare le informazioni usando le query. Convertire le informazioni dal mondo SQL a quello della programmazione a oggetti, presenta difficoltà dovute al fatto che i due mondi usano modelli di organizzazione dei dati molto differenti.

Figura 1.18. Logo PostgreSQL

Capitolo 2

Architettura e design

2.1 Analizzare il progetto Android

Il progetto in questione supera i problemi dati con gli apparecchi elettronici attraverso l'uso di un app android il cui logo è visibile in figura 2.1. L'applicazione mobile , chiamata 'PickToLight', con l'uso della computer vision riesce ad indicare all'operatore su quale scaffale si trova la merce che deve essere prelevata.

Figura 2.1. Logo App.

Inizialmente si presenta con una schermata di login, che tramite un web service opportunamente collegato, ci fa accedere alla schermata principale se user e password sono corrette. In questo modo si viene a conoscenza dell'User che esamina ed esegue l'ordine, mostrando anche una piccola descrizione dell'utente come ad esempio nome, cognome e matricola.
Se l'utente è riuscito ad accedere, appare un activity di benvenuto con un pulsante che ha come scopo di scansionare il codice EAN, che si trova sulla bolla contenente

la lista dei prodotti da prelevare. Una volta scansionata la bolla, in un'altra activity, compaiono tutti i prodotti che devono essere prelevati e in cima si può vedere il codice EAN associato. Per ogni merce si possono acquisire diverse informazioni quali la quantità, il corridoio e la fila dove andare a prelevare, il nome e la sigla del materiale.
Attraverso l'interazione con un pulsante si apre la fotocamera con il quale si scansiona il QRCode che si trova sullo scaffale per identificare che sia lo scaffale giusto e in caso di risultato positivo possiamo visualizzare lo scaffale dove andare a prelevare la merce.

2.2 Sviluppo

2.2.1 Schermata di login

Quando l'applicazione si apre compare la 'main activity' cioè la schermata principale, figura 2.2, dove l'operatore può fare il login. Prima di tutto, siccome devono essere fatte delle richieste a un server, bisogna mettere nel manifest del progetto il permesso per internet. Sono visibili due EditText al centro con lo scopo di inserire UserName e Password dell'operatore.
A fianco della password viene posizionato un ImageButton usato per nascondere o meno la password scritta dall'utente per aiutarlo a visualizzare ciò che ha scritto se è giusta o sbagliata. Una volta scritto UserName e Password bisogna premere il bottone login.
Volendo rispettare i requisiti di scalabilità e di servizio RESTful si è scelto di gestire l'autenticazione tramite JSON Web Token (JWT), invece che con le tradizionali sessioni o tramite cookie. Infatti con i JWT non si pone il problema di dover memorizzare e ritrovare i dati di sessione all'interno del database, snellendo in maniera significativa la comunicazione con esso. Lo scenario d'esecuzione è il seguente:

- L'utente chiama il servizio di autenticazione fornendo username e password.
- Il servizio di autenticazione verifica le credenziali ed in caso di successo restituisce un JWT firmato.
- Ad ogni nuova richiesta l'utente invia il token ed un servizio middleware ne verifica autenticità e validità.

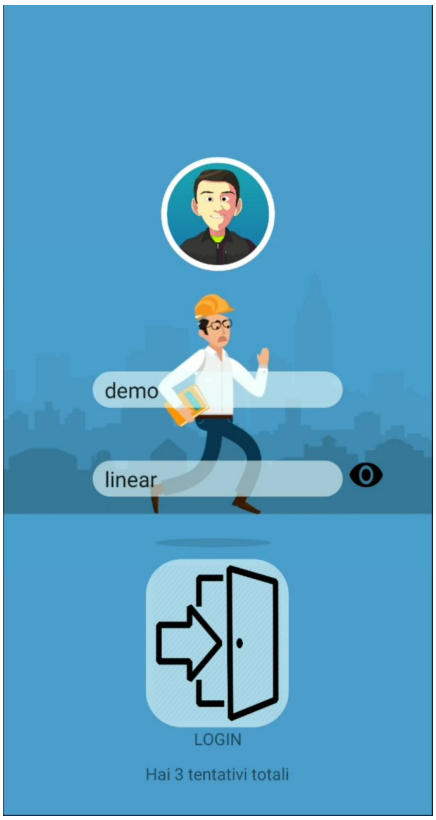

Figura 2.2. Screenshot schermata login.

Durante la pressione del pulsante viene richiamata una funzione che ha il compito di validare l'autorizzazione tramite un token utilizzando la libreria Volley e Gson. Queste due librerie servono rispettivamente per i servizi REST in android e per poter compilare la classe Utente direttamente dal json ricevuto in modo tale da prendere le informazioni dell'utente per le activity successive.
Nel caso ci fosse un errore di connessione con il server compaiono dei Toast che indicano il tipo di errore invece se l'errore dipende dalla errata digitazione della password compare un Toast che specifica la password sbagliata e decrementa il numero dei tentativi. Quando i tentativi arrivano a zero si chiude l'applicazione.

2.2.2 Scheramata di benvenuto

Una volta fatto il login e aver verificato l'autenticazione dell'utente si manda un intent alla nuova activity passando il token e tutto l'oggetto utente con le relative informazioni (Fig. 2.3).

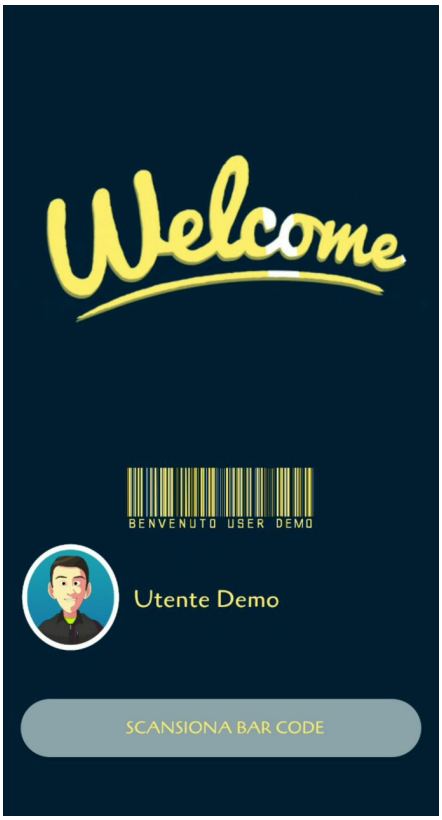

Figura 2.3. Screenshot schermata benvenuto.

Questa activity non fa molto in realtà. Infatti oltre a dare delle informazioni sull'operatore come username e nome per esteso ha un pulsante 'SCANSIONA BAR CODE'. Premendo questo tasto parte un nuovo intent per scansionare il bar code della bolla dove c'è la lista delle merci da prelevare.
In sostanza questa activity intermedia è più che altro grafica avendo lavorato prevalentemente sul layout. Infatti per la scritta 'welcome' si è pensato di mettere una GIF e di cambiare il font della scritta 'benvenuto user demo' in modo tale di avere l'effetto di un codice a barre sopra. Vengono in oltre settati le due TextView scrivendo il nome per esteso e l' UserName tramite la classe utente che è stata passata all'activity corrente e

2.2.3 Scansione bar code

Nel progetto android viene importata la libreria ZXING ("zebra crossing") è una libreria di elaborazione di immagini di codici a barre 1D/2D multiformato open

source implementata in Java.
Per poter usufruire però della libreria ed elaborare quindi il codice a barre bisogna innanzitutto mettere la dipendenza nel gradle scripts (build.gradle) e i permessi per utilizzare la fotocamera nel file manifest (AndroidManifest.xml).
Se il risultato è diverso da 'null' viene inizializzato un nuovo intent che mostra la lista dei prodotti, passando anche il codiceEAN e il token, uguale a quello iniziale, per l'autorizzazione a svolgere il compito.

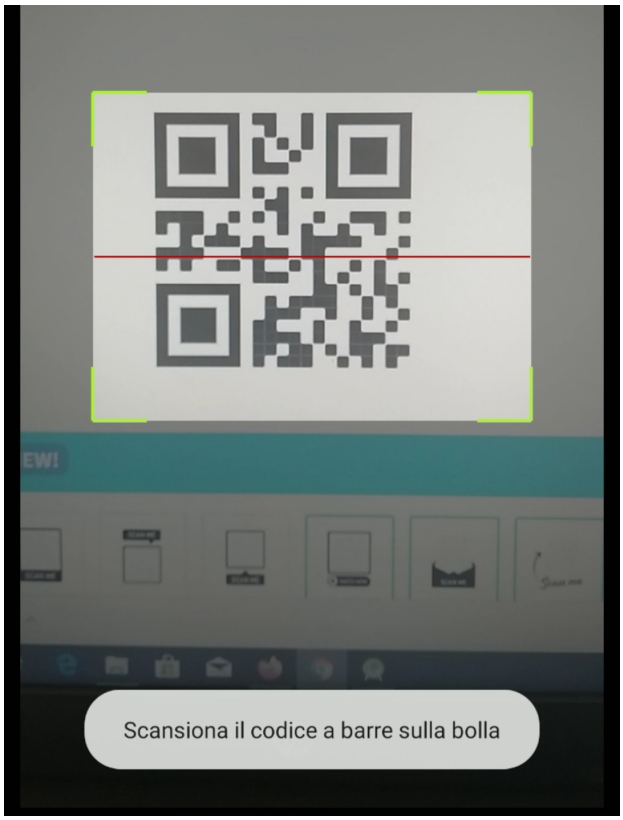

Figura 2.4. Screenshot scansione con ZXING.

2.2.4 Lista prodotti

A questo punto in questa activity vengono prese con il metodo 'getIntent().getStringExtra()' il codiceEan e il token. Il lavoro in questa activity è sia a livello di rete che a livello di layout.

Questa activity ha lo scopo di far vedere a video la lista degli oggetti che devono essere prelevati. Acquisendo il codiceEAN si compone l'url specifico per fare la chiamata di rete concatenando una stringa costante relativa all'API desiderata con il codiceEAN appena scansionato.
La risposta della chiamata alla rete viene usata per mappare nella classe 'Bulkmovement' i vari campi del json. La classe 'Bulkmovement' ha al proprio interno una lista di dettagli movimenti che vengono ispezionati dentro un costrutto for, passando ogni dettaglio movimento al costruttore di product. Il prodotto risultante, dopo aver settato i vari campi con il costruttore, viene aggiunto ad un ArrayList di 'Product'.
Dalla lista si passa alla parte grafica dove i soggetti principali sono la View e l'adapter. Nel layout dell'activity viene usato un oggetto che prende il nome di ListView; questo conterrà la lista dei vari oggetti che vogliamo prelevare. Dentro la lista ci sarà una riga per ogni prodotto quindi la prima cosa da fare è costruire con un file XML l'item custom della lista.

Figura 2.5. Screenshot lista prodotti.

Adesso c'è da inserire le righe nella ListView. Per fare questo passaggio si usa un adapter che è una classe apposita ('ProductListAdapter') per poter renderizzare dentro la lista i vari item. L'adapter di cui stiamo parlando è un componente che si occupa della rappresentazione grafica dei dati e dell'interazione con essi, per ogni elemento della ListView. La classe che implementa l'adapter deve estendere BaseAdapter che è la classe base comune di implementazione comune per un adapter. Dentro la classe ProductListAdapter viene usato l' override del metodo getView al quale viene passato la posizione dell'articolo all'interno del set di dati, la view e il ViewGroup; in questo modo si ottiene una vista che visualizza i dati nella posizione specificata. Per produrre un componente personalizzato si procede proprio nella ridefinizione di questo metodo. Adesso dobbiamo definire il nostro custom adapter. Tutto avviene attraverso un'operazione di inflating. L'inflating ci permette di istanziare un'oggetto da una risorsa XML usando View.inflate(Context context, int resource, ViewGroup root). Dunque, abbiamo ottenuto un riferimento al LayoutInflater di sistema che ci permette di assegnare al convertView il layout che abbiamo definito in XML.
Nell'activity in cui si vuole mostrare la lista si avrà un istanza dell'adapter usando il costruttore della classe ProductListAdapter in cui viene passato il contesto e la lista dei prodotti che devono essere visualizzati. La ListView visualizza le View ottenute da una implementazione dell'interfaccia ListAdapter che gli viene passata attraverso il metodo 'setAdapter' (visualizzato all'ultima riga nell'immagine).

```
mProductList.add(product);
adapter=new ProductListAdapter(getApplicationContext(),mProductList);
lvProduct.setAdapter(adapter);
```

Figura 2.6. Adapter.

Dentro la chiamata di inflate c'è corrispettivamente l'oggetto context per l'attività, l'ID della risorsa da inserire dentro la ViewList e un gruppo di viste che sarà il genitore (null).
Quindi, ridefiniamo il metodo getView() della classe ProductListAdapter e all'interno del metodo componiamo la visualizzazione dell'elemento della nostra lista. Sempre nel metodo getView si implementa la logica per il pulsante delle informazioni relative all'oggetto selezionato usando il metodo 'setOnClickListener'. Durante

il richiamo di questo metodo verrà eseguita una funzione di callback 'onClick(View v)' in cui si può implementare il codice. Infatti in questa callback si da vita a un intent che porta all'activity di descrizione del materiale.
Per ogni riga custom viene fatto il controllo di un campo della classe Product (il campo find) per rendere visibile la spunta di avvenuto prelevamento in modo da ricordare al picker quali merci sono state già prelevate.

2.2.5 Schermata informazioni

Come detto in precedenza una volta deciso quale prodotto prelevare e premendo l' ImageButton delle informazioni viene lanciato un intent per una nuova activity con tutte le informazioni sul prodotto.

Figura 2.7. Screenshot informazioni prodotti.

Le informazioni che vengono visualizzate sono la quantità di pezzi da prelevare, con una determinata sigla, in quello scaffale che si trova in una determinata posizione

e per questo vengono esposte anche il corridoio e il numero dello scaffale. Queste ultime sono rispettivamente cordinata X e Y.
Ulteriori informazioni sono la sigla e il nome del prodotto.
La descrizione deriva dall'uso appropriato di tre API:

- *Position:* indica la posizione del prodotto desiderato all'interno del magazzino in modo tale da aiutare l'operatore ad arrivare davanti lo scaffale giusto.

- *Material:* serve per la descrizione del materiale infatti al suo interno è possibile sapere il codice EAN, nome e sigla del materiale e la Position.

- *DetailMovement:* Viene usata per ricavare la quantità attraverso il metodo getQuantity() in modo tale da indicare all'operatore quanti di quei prodotti prelevare. Il dettaglio del movimento viene passato dall'intent e ripreso nella nuova activity tramite il metodo 'getSerializableExtra()'. Come campo pubblico dell'API si trova anche Material.

Una volta che l'operatore vuole iniziare a effettuare il prelievo si pigia l'ImageButton con la fotocamera in modo da far scattare un nuovo intent con lo scopo di riconoscere lo scaffale. Capita spesso di avere la necessità di recuperare un dato gestito dalla sub-activity, usando startActivityForResult(). Tutte le volte che usiamo questo metodo dobbiamo anche implementare un override, nell'activity chiamante, del metodo onActivityResult(), il quale dovrà contenere tutto il codice per gestire il dato da recuperare.

2.2.6 Rilevamento

Questa activity è la più importante tra tutte. Qui si usa la logica per rilevare i quadrati e prelevare l'articolo desiderato.
Prima di tutto si vuole verificare che l'operatore sia davanti lo scaffale giusto e quindi parte immediatamente un IntentIntegrator. Questo è un modo semplice per invocare la scansione del codice a barre e ricevere il risultato, senza la necessità di integrare, modificare o apprendere il codice sorgente del progetto. Per integrare, crea un'istanza di IntentIntegratore e chiama initiateScan() e attendi il risultato nella tua app.
L'activity deve implementare il metodo onActivityResult(int, int, Intent) dove si

gestisce il risultato della scansione. Quando l'utente ha terminato l'attività avviata e ritorna, il sistema chiama il metodo onActivityResult(). Se il risultato del qrCode è diverso della coordinata Y del materiale allora compare un Toast con scritto "Scaffale errato" altrimenti si è di fronte a quello giusto e si continua a operare.

Figura 2.8. Rilevamento scaffale.

Una volta che il result è giusto può partire l'activity per indicare il ripiano dove si trova l'oggetto per il materiale. La classe estende AppCompatActivity come ogni activity ma questa classe implementa anche l'interfaccia 'CameraBridgeViewBase.CvCameraViewListener'. Questa è una interfaccia di base, implementa l'interazione con la fotocamera e la libreria OpenCV. La sua responsabilità principale è quella di controllare quando è possibile abilitare la fotocamera, elaborare il frame, chiamare un ascoltatore esterno per apportare eventuali modifiche al frame

e quindi disegnare il frame risultante sullo schermo.
La funzione più importante è onCameraFrame. È la funzione di callback e viene richiamata al momento del recupero del frame dalla fotocamera. L'input della callback è un oggetto dela classe CvCameraViewFrame che rappresenta il frame dalla telecamera. Si aspetta che la funzione onCameraFrame restituisca il frame RGBA che verrà disegnato sullo schermo.
Una volta preso il frame di input si deve cambiare il formato il RGB perché OpenCV utilizza il formato BGR in modo da ottenere la matrice con i colori messi in modo giusto.
Come si nota dalla figura il ripiano dove andare a prelevare viene indicato da una frecciaverso il basso. Questa freccia non è altro che una bitmap trasformata in matrice e messa dentro la ROI (*Region Of Interest*). Abbiamo un'immagine RGB (*Red-Green-Blue*) ed è allettante tentare semplicemente di soglia il canale R e ottenere la nostra maschera. Si scopre che questo non funzionerà efficacemente poiché i valori RGB sono altamente sensibili all'illuminazione. Quindi, anche se la struttura dello scaffale è di colore rosso, potrebbero esserci alcune aree in cui, a causa dell'ombra, i valori del canale rosso dei pixel corrispondenti sono piuttosto bassi. L'approccio giusto è quello di trasformare lo spazio colore della nostra immagine da RGB a HSV (*Hue–Saturation–Value*). A differenza di RGB, definito in relazione ai colori primari, HSV è definito in modo simile al modo in cui gli umani percepiscono il colore. Quindi viene catturato un fotogramma live, convertiamo l'immagine da RGB a spazio colore HSV e quindi definiamo un intervallo specifico di valori HSV per rilevare il colore rosso.
Con la funzione inRange si preparano due maschere con range di rosso diverso e si faranno delle operazioni bit a bit tra le due maschere avendo poi una maschera finale con il metodo bitwise_or(Mat src1, Mat src2, Mat dest).

Vengono applicate le trasformazioni morfologiche le quali sono alcune semplici operazioni basate sulla forma dell'immagine. Normalmente viene eseguito su immagini binarie. Ha bisogno di due input, uno è la nostra immagine originale, il secondo è chiamato elemento strutturante o kernel che decide la natura dell'operazione. Due operatori morfologici di base sono Erosione e Dilatazione. Il kernel può essere creato usando getStructuringElement che restituisce un elemento strutturante della dimensione e della forma specificate per le operazioni morfologiche. La funzione costruisce e restituisce l'elemento di strutturazione che può essere ulteriormente passato per erodere o dilatare e come parametri vengono passati la forma

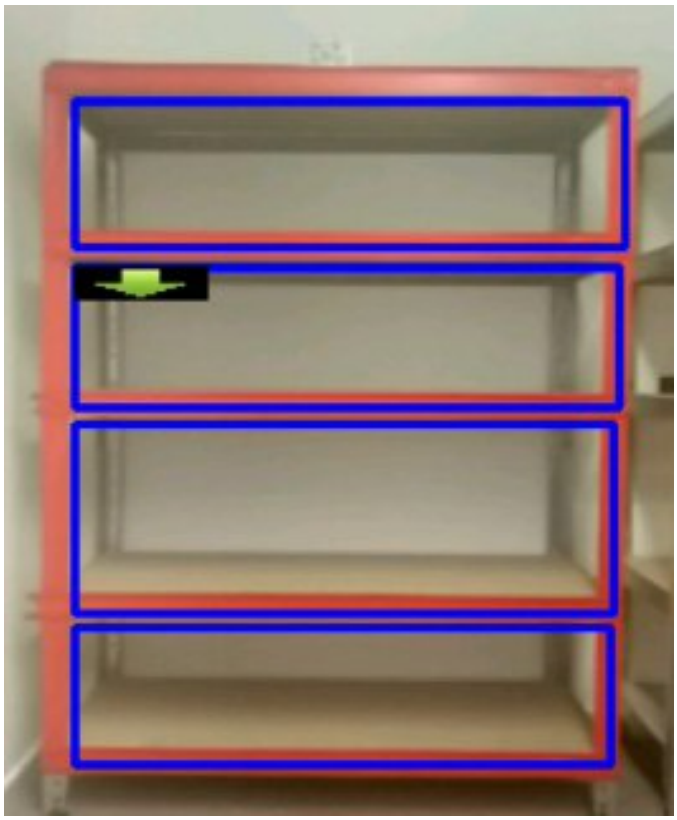

Figura 2.9. Prodotto da prelevare.

dell'elemento che potrebbe essere una di MorphShapes e la dimensione dell'elemento strutturante.
L' **erode** ha come effetto che quando l'elemento strutturante B viene traslato vicino ai bordi esso non è completamente contenuto in A. L'effetto dell'operatore su una immagine binaria è di erodere i contorni delle regioni dei pixel foreground (tipicamente i pixel bianchi) quindi l'area dei pixel di foreground si restringe in dimensioni e i buchi si allargano. L'effetto dell'operatore **dilate** su una immagine binaria è di allargare gradualmente i contorni delle regioni di pixel non di background. In questo modo l'area dei pixel foreground cresce in dimensione, mentre i vuoti in queste regioni diventano più piccoli.
Le operazioni morfologiche sulla maschera totale servivano per trovare in modo più efficace i contorni presenti nella maschera. Il metodo *findContours* serve per trovare i contorni e ogni contorno è memorizzato come un vettore di punti. Infatti il metodo prende come parametri l'immagine sorgente cioè la maschera e inserisce in

una lista di *MatOfPoint* i punti dei contorni. Trovati tutti i punti, con un ciclo for si converte il contorno i-esimo da *MtOfPoint* a *MatOfPoint2f* e consecutivamente l'approssimazione con *approxPolyDP()* definsce una curva continua e chiusa.
La funzione *Rect rect = boundingRect(MatOfPoint points)* calcola e restituisce il rettangolo di delimitazione utilizzando le opzioni e le caratteristiche di visualizzazione fornite, all'interno del rettangolo specificato. Il ciclo for si chiude aggiungendo in un ArrayList il rettangolo rect.
Se l' ArrayList non è vuoto viene fatto il sort mettendo i rettangoli dal più grande al più piccolo e successivamente vengono rimossi i rettangoli più grandi e quelli più piccoli. Questo passo viene fatto grazie alla funzione *removeAverage* (Fig. 2.10) la quale viene passato come parametro la lista di tutti i rettangoli.
La funzione prende tutti i rettangoli e calcola la media dell'area dei rettangoli. Nel momento in cui un rettangolo è più piccolo della media viene messo in un arrayList e alla fine viene rimosso dalla lista dei rettangoli totali.
Si era creato un caso in cui veniva preso il rettangolo più esterno che rappresentava la struttura dello scaffale. Questo problema è stato risolto rimuovendo il rettangolo più grande dalla lista (Fig. 2.11).
Dopo aver tolto i punti superflui si disegnano i rettangoli che sono rimasti in contrapposizione dei bordi indicando i ripiani con contorni blu.
Per concludere, una volta che si ha la lista completa dei rettangoli finali, si deve indicare quale sarà il ripiano in cui prelevare tramite una freccia indicatrice; infatti nella funzione verrà passato anche la coordinata del ripiano. Per prima cosa, si calcola la ROI dove andare a mettere la freccia indicatrice e viene fatta una *submat* con il frame scegliendo le coordinate della ROI come sezione. Successivamente vengono fatte delle operazioni sulla maschera da inserire nella ROI. Queste operazioni sono lo split dell'immagine nel quattro canali rosso (R), verde (G), blu (B) e alpha (A) inserendo ogni canale in un matrice e facendo l'AND bit a bit per ogni matrice di ogni canale e facendo il merge del risultato. Una volta ottenuto il merge bisogna fare la sovrapposizione con la funzione addWeighted in cui viene passata la ROI da mettere nel frame in alto a sinistra. Una volta che l'operatore ha concluso l'operazione conclude battendo con un oggetto NFC sul ripiano dello scaffale che ha posizionato un RFID passivo e nella lista compare una spunta e viene mandato anche un json al server per far si che venga cambiato anche lato server che l'operazione è conclusa. Questo serve nel momento in cui l'operatore non conclude tutto l'ordine ha bisogno di ricordarsi i materiali prelevati di quella lista. Una volta che

```
private List<Rect> removeAverage(List<Rect> rectAvg) {
    Rect maxRect = new Rect();
    Rect maxRect2 = new Rect();
    double diff_min = 0;
    double area_tot = 0;
    double avgArea = 0;
    for (Iterator<Rect> iterator = rectAvg.iterator(); iterator.hasNext(); ) {
        Rect rect = iterator.next();
        maxRect2 = area_tot == 0 ? rect : maxRect2;
        area_tot += rect.area();
        if (maxRect.area() < rect.area()) {
            maxRect = rect;
        }
        maxRect2 = maxRect.area() > maxRect2.area() ? maxRect : maxRect2;
    }
    double maxDifArea;
    maxDifArea = maxRect.area() - maxRect2.area();
    if (maxDifArea > area_tot / rectAvg.size()) {
        rectAvg.remove(maxRect);
        avgArea = (area_tot - maxRect.area()) / rectAvg.size();
        diff_min = maxRect2.area() - avgArea;
    } else {
        diff_min = maxRect.area() - avgArea;
    }
    List<Rect> removeObj = new ArrayList<>();
    for (Iterator<Rect> iterator = rectAvg.iterator(); iterator.hasNext(); ) {
        Rect rect = iterator.next();
        double diff_cur;
        diff_cur = (rect.area() - avgArea);
        if (diff_cur < diff_min/6) {
            removeObj.add(rect);
        }
    }
    rectAvg.removeAll(removeObj);
    return rectAvg; }
```

Figura 2.10. Snippet della funzione removeAverage.

ha prelevato tutti gli elementi quella bolla è stata conclusa.

```
private void printRectangles(List<Rect> rect_pnt, Mat img ) {
    removeBiggest(rect_pnt);
    for (Rect rect: rect_pnt) {
        Imgproc.rectangle(img, rect.tl(), rect.br(), new Scalar(0, 0, 255), thickness: 2);
    }
}

private List<Rect> removeBiggest(List<Rect> rect_pnt) {
    Rect maxRect = new Rect();
    Double diff = new Double( value: 0);
    Double area_tot = new Double( value: 0);

    for (Rect rect : rect_pnt){
        area_tot += rect.area();
        if (maxRect.area() < rect.area()){
            diff = rect.area() - maxRect.area();
            maxRect = rect;
        }
    }

    if(diff > (area_tot - maxRect.area())){

        rect_pnt.remove(maxRect);
    }

    return rect_pnt;
}
```

Figura 2.11. Snippet delle funzioni printRectangles e removeBiggest .

Capitolo 3

Certificare la validità

Il layout di un punto vendita è il risultato di una serie di considerazioni strategiche volte a produrre un'organizzazione efficace dello spazio a disposizione. La scelta del "layout" deve consentire lo sfruttamento migliore del magazzino nel suo complesso, ma anche di ogni sua area funzionale a seconda della destinazione del magazzino stesso. In particolare deve consentire il raggiungimento di specifiche prestazioni come ad esempio, prontezza di risposta alle esigenze del cliente per poter minimizzare i costi e massimizzare lo sfruttamento degli spazi e delle condizioni di lavoro. Quindi l'area che viene individuata nel layout di magazzino è quella adibita al posizionamento della merce. Forse l'area più importante di tutto il magazzino, infatti deve prevedere sempre l'uso degli spazi nei periodi di alta intensità. La logistica concerne l'andamento dei flussi fisici dei materiali (materie prime, semilavorati, componenti e prodotti finiti) ed i relativi flussi informativi, dal punto di origine a quello di destinazione, concretizzandosi soprattutto nell'area produttiva, dove il prodotto nasce e viene lavorato.

Figura 3.1. Flussi logistici

Il flusso fisico può essere suddiviso in tre fasi strettamente correlate, riguardanti l'approvvigionamento, la trasformazione e la distribuzione, che l'impresa svolge rispettivamente con i suoi fornitori, al proprio interno e con i clienti da servire, in modo che ad ognuna di esse il materiale acquisti un valore utile maggiore in funzione della disponibilità nel momento e nel luogo desiderato. L'approvvigionamento (logistica in entrata) consente all'impresa industriale di acquisire dai fornitori materie prime, parti e componenti, al fine di averne l'effettiva disponibilità in stabilimento nei modi e nei tempi richiesti dal processo produttivo e al minimo costo possibile. Durante il processo di trasformazione, la logistica segue il flusso dei materiali in lavorazione, assicurandone la tempestiva ed economica utilizzazione nelle varie fasi produttive, fino alla collocazione dei prodotti finiti nel relativo deposito. La distribuzione fisica (logistica in uscita) assicura il trasferimento materiale dei beni al cliente. Nell' impresa industriale, pertanto, essa prende avvio dal magazzino dei prodotti finiti e termina con la loro consegna, a seconda dei casi, al distributore o all'utilizzatore finale. Le tre fasi costituiscono il complessivo processo logistico il cui ruolo è quello di assicurare, come richiesto dal cliente, l'esatto tipo di materiale, nell'esatta quantità, al tempo e nel luogo esatto; il tutto deve essere fatto minimizzando i costi necessari a raggiungere un determinato livello di servizio, ovvero massimizzare quest'ultimo entro un prefissato limite di costi. Si generano così diverse aree di sovrapposizione, in cui si accentuano i problemi di coordinamento tra le diverse funzioni aziendali, i cui rapporti si caratterizzano sempre dalla contemporanea presenza di elementi cooperativi e conflittuali.

Quindi, i flussi di prodotto e flussi di informazioni devono trovare il modo di integrarsi al fine di garantire affidabilità (reliability) e agilità (agility) della supply chain per gestire opportunamente domanda e offerta. L'affidabilità può essere definita come la capacità di inviare il giusto prodotto, nella giusta quantità, nel giusto tempo, al giusto posto al minore costo (David et al., 2000). L'agilità può essere definita come la capacità di rispondere velocemente ai cambiamenti delle condizioni di mercato (Christopher, 2000). Il grafico di seguito illustra cinque fasi fondamentali che si collocano in momenti temporali sequenziali, una volta ricevuto l'ordine cliente:

1. Approviggionamento materie prime da fornitori.
2. Creazione di stock di materie prime a magazzino.
3. Produzione del prodotto finito.

4. Creazione di stock di prodotto finito a magazzino

5. Preparazione dell'ordine per consegnare al cliente.

Una volta definita la fase fondamentale da svolgere, in base all'esigenza richiesta, si andrà ad utilizzare l'app. Ogni operazione sotto descritta avrà una funzione propedeutica al raggiungimento e all'esecuzione di una delle fasi fondamentali.

3.1 Inserimento oggetto a sistema

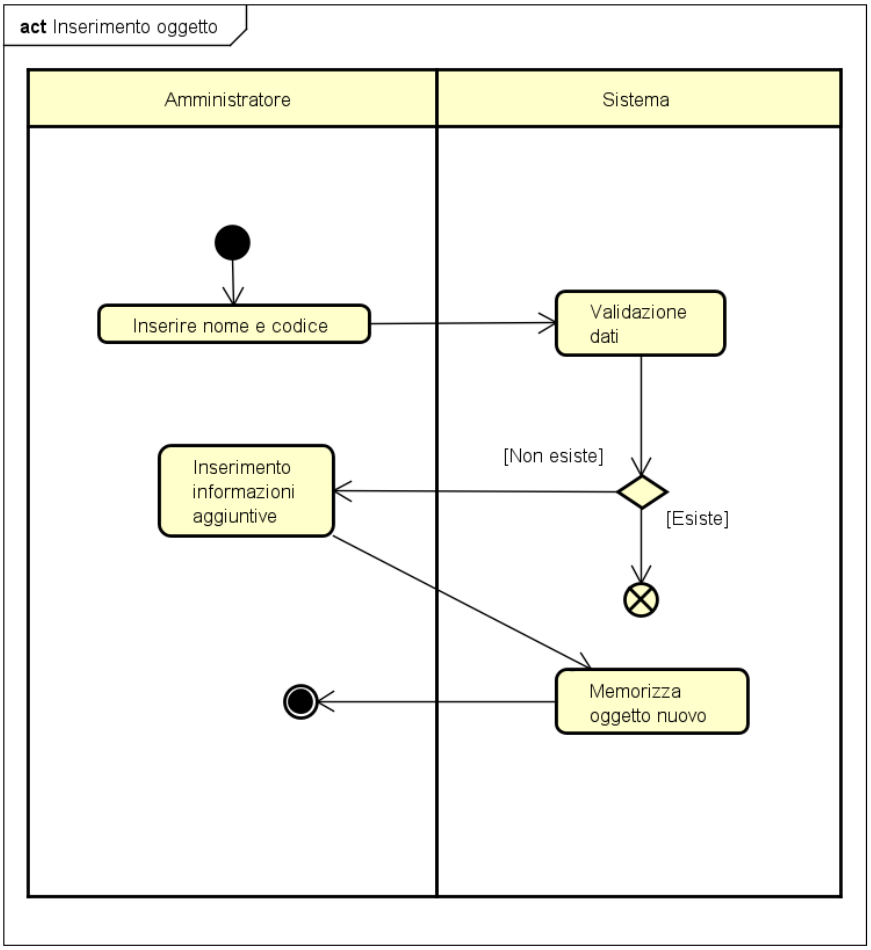

Figura 3.2. UML caso 1

L'amministratore dello stabilimento vuole inserire nel sistema un nuovo materiale. Ogni materiale ha un nome e un codice del materiale su cui fare riferimento.

L'amministratore dunque deve mettere nel database il codice, l'abbreviazione, la quantità, la descrizione e la posizione in cui si andrà a trovare il materiale. Un oggetto di quel tipo avrà sempre lo stesso codice così sarà facile da rintracciare.
La posizione serve a determinare la locazione (es.lo scaffale e il ripiano) in cui verrà depositato quel materiale. Nel momento in cui si inseriscono i vari dati il sistema deve verificare che il codice esista o meno nel sistema. Le opzioni possono essere tre:

- Il prodotto esiste.
- Il prodotto esiste nel sistema ma è presente un errore.
    1. c'è un nuovo prodotto all'interno delle distinte basi (nuovo prodotto finito, obsolescenza del prodotto precedente.
    2. il prodotto è lo stesso ma il codice viene aggiornato (es. errore durante la nascita delle distinte basi del raw material).
- Il prodotto non esiste quindi bisogna aggiungere le informazioni sopra elencate oltre al codice univoco e successivamente aggiornare la distinte basi.

Se vengono passati tutti i controlli il sistema convalida i dati inseriti e li memorizza nel database. Se, invece, avviene un errore il sistema notifica e richiede la rettifica.
Nome del caso d'uso: Inserimento oggetti a sistemi
Contesto: Sistema gestione prodotti
Livello: User-goal
Intenzione dell'attore: L' amministratore vuole inserire le informazioni riguardanti un nuovo oggetto
Attore primario: Amministratore di magazzino
Causa scatenante: Un nuovo oggetto deve essere messo nel database
Scenario di successo principale:

1. L'amministratore vuole inserire un nuovo oggetto
2. L'amministratore inserisce un nuovo oggetto
3. Il sistema verifica che non esiste già l'oggetto
4. L'amministratore inserisce il resto dei dati

5. Il sistema convalida e memorizza il nuovo oggetto

Estensione:

2.a L'amministratore annulla: il caso d'uso termina con fallimento.

3.b L'oggetto è già presente con un altro codice EAN:

3.b.1 Il sistema notifica l'errore e chiede di modificare il caso d'uso prosegue al passo 2

3.c I dati sono errati o incompleti:

3.c.1 Il sistema notifica l'errore chiedendo di rettificare il caso d'uso prosegue al passo 4

3.2 Prelievo oggetto

L'amministratore riceve un ordine e controlla se in magazzino sono presenti i vari oggetti in quantità favorevoli per soddisfare la richiesta. Nel momento in cui l'esito è positivo allora emette una bolla per l'operatore con un codice EAN che identifica la bolla stessa. L'operatore, munito di dispositivo android, apre la fotocamera dall'applicazione dopo aver fatto l'accesso e punta verso il codice a barre della bolla. L'accesso serve per tenere conto al sistema chi sta facendo l'operazione, infatti se una volta inserite le credenziali il server trova il match con username e password allora si può proseguire altrimenti si richiede l'inserimento corretto di entrambe i campi. Ciò è indispensabile per tenere lo storico delle operazioni eseguite degli operatori nel caso in cui si dovesse verificare mancanti o eccessi tra fisico e contabile. Una volta che sia andato a buon fine l'accesso si riesce a visualizzare la lista sul device e quindi si può cominciare a prelevare un oggetto senza seguire l'ordine visualizzato sullo schermo. Appena deciso cosa prelevare il sistema manda un json con le informazioni del prodotto e la quantità da prelevare in modo che l'operatore può svolgere l'azione celermente recandosi nella posizione indicata per prelevare. L'operatore posizionatosi davanti allo scaffale indicato ed esegue due step sequenziali:

- Confronto tra identificativo dello scaffale passato dal sistema e identificativo dello scaffale presente sullo scaffale (QRcode).

- Una volta riscontrato il match, verrà indicato il ripiano dal quale prelevare.

Finito il prelevamento l'operatore passa con l'NFC sul RFID passivo e questo indica la fine dell'azione. Il sistema segnala la fine dell'operazione e aggiorna lo stato della bolla. Quando tutti i prodotti sono stati prelevati allora il sistema cambia lo stato totale della bolla in modo che venga indicata la terminazione.

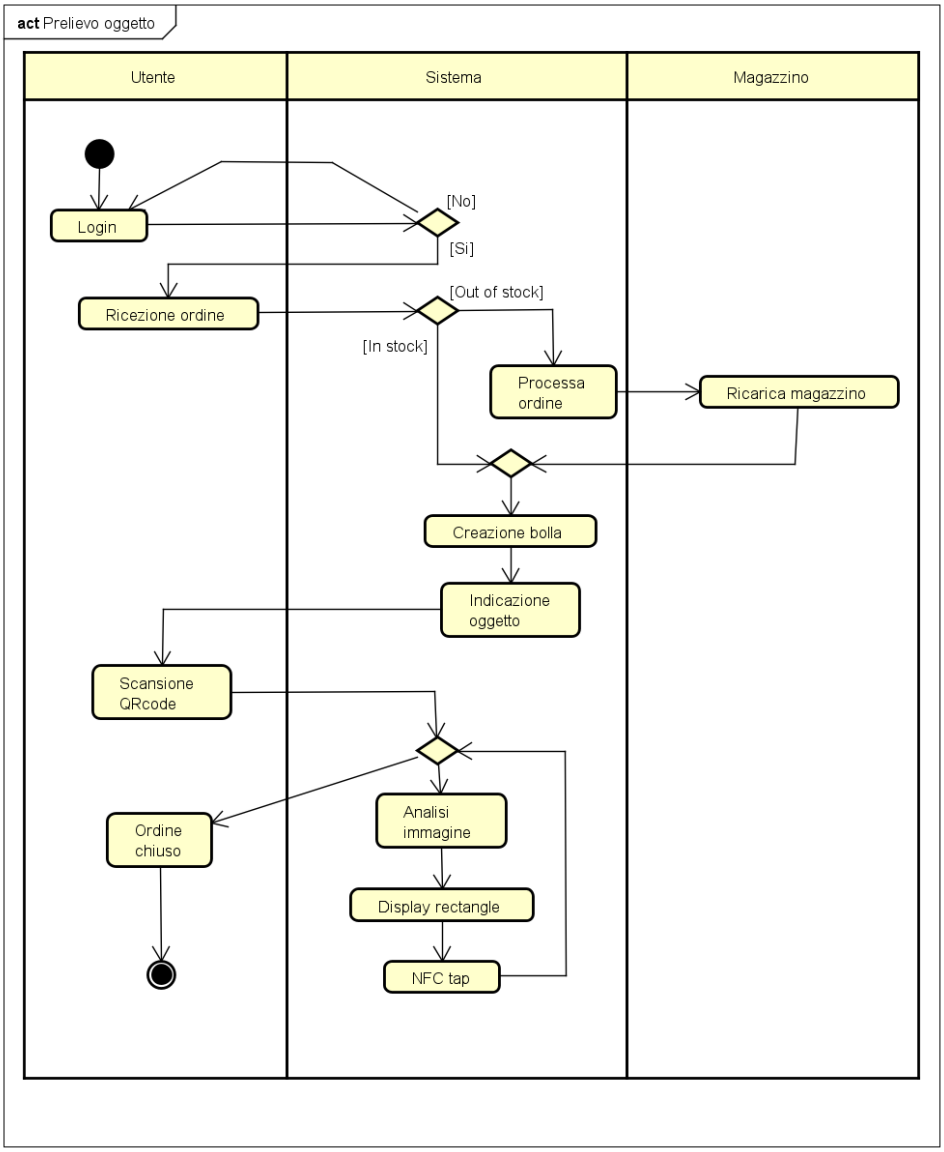

Figura 3.3. UML caso 2

Nome del caso d'uso: Prelievo oggetti
Contesto: Sistema gestione prodotti
Livello: User-goal
Intenzione dell'attore: Il picker deve prelevare uno o più oggetti dallo scaffale
Attore primario: Picker
Causa scatenante: Un nuovo oggetto deve essere prelevato dagli scaffali
Scenario di successo principale:

1. L'amministratore ordina il prelevamento di un nuovo oggetto
2. Il picker riceve una bolla da scansionare
3. Il picker si logga con username e password
4. Scansionamento del codice EAN
5. Visualizzazione della lista dei prodotti
6. Viene trovato l'oggetto e prelevato
7. Il sistema convalida e notifica il prelevamento del nuovo oggetto

Estensione:

3.a Picker non riconosciuto:

3.a.1 Il caso d'uso termina con un fallimento

6.a Quantità insufficiente:

6.a.1 Il sistema notifica l'errore terminando con un fallimento

7.a Non tutti gli oggetti sono stati prelevati:

7.a.1 Il sistema non può chiudere la lista e quindi non viene cambiato lo stato finale

3.3 Ricarica oggetto (da parte di operatore)

Dopo aver stilato un preventivo dei volumi da produrre l'amministratore ordina dai fornitori i prodotti da inserire a stock. Una volta ricevuta la merce l'operatore avrà il compito di allocarla nella postazione prestabilita. L' amministratore inserisce a sistema il numero di pezzi da aggiungere nello scaffale per quell'oggetto che possiede un proprio codice e compila una bolla per l'operatore con il titolo *Ricarica* in modo che non si confonda con il prelievo di un oggetto. Successivamente l'operatore che ha preso in carica il compito di ricaricare, scansiona il codice sulla bolla e riceverà dal sistema la quantità da inserire per quel prodotto specifico, il codice del prodotto, la locazione e l' ID dello scaffale. Il procedimento di carico è uguale e inverso al procedimento di prelievo. Infatti l'operatore scansionerà il QRcode e lo confronterà con l' ID dello scaffale con l'ausilio del device android. In questo modo si testerà la correttezza della ubicazione e una volta trovato il match vengono inserite le quantità stabilite in precedenza nel ripiano indicato. Finito di ricaricare il ripiano l'operatore passa con l'NFC sul RFID in modo da indicare al sistema che si è terminata il compito assegnato e lo stato della query passa a finito.
Nome del caso d'uso: Ricarica oggetto
Contesto: Sistema gestione prodotti
Livello: User-goal
Intenzione dell'attore: Il picker deve ricaricare uno o più ripiani dello scaffale
Attore primario: Picker
Causa scatenante: Un nuovo ripiano deve essere ricaricato
Scenario di successo principale:

1. L'amministratore ordina di ricaricare un ripiano
2. Il picker riceve una bolla da scansionare
3. Il picker si logga con username e password
4. Scansionamento del codice EAN
5. Visualizzazione della lista dei prodotti da caricare
6. Viene indicato il ripiano e ricaricato con oggetti
7. Il sistema convalida e notifica che il ripiano è stato ricaricato

Estensione:

3.a Picker non riconosciuto:

3.a.1 Il caso d'uso termina con un fallimento

6.a Ripiano già carico:

6.a.1 Il sistema notifica l'errore terminando con un fallimento

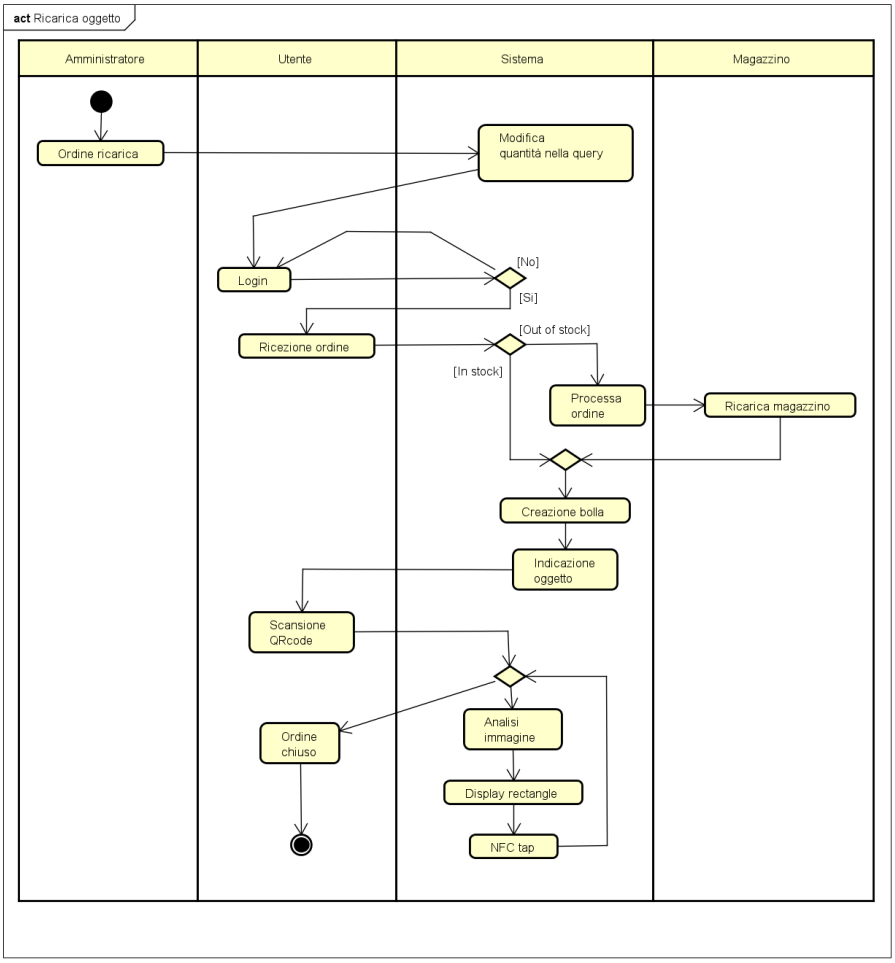

Figura 3.4. UML caso 3

3.4 Spostamento oggetto da un cassetto all'altro

A differenza del tradizionale sistema pick to light in cui si prevede l'utilizzo di dispositivi hardware quali cavi, sensori luminosi, ecc., il nuovo sistema è libero da qualsiasi vincolo strutturale del layout del magazzino. Nel momento in cui l'amministratore decide di voler cambiare la disposizione del magazzino è facilmente fattibile perché l'operatore verrà guidato da ciò che compare a schermo del dispositivo. Fondamentale è la funzione dell'amministratore che gestisce il back-end e quindi anche la modifica delle query nel database. L'operatore svolgerà l'operatore di scarico e carico descritte precedentemente per lo stesso prodotto in un arco temporale ridotto.
Nome del caso d'uso: Cambio layout magazzino
Contesto: Sistema gestione prodotti
Livello: User-goal
Intenzione dell'attore: Il picker deve spostare un oggetto da uno scaffale all'altro
Attore primario: Amministratore e picker
Causa scatenante: Un oggetto è più richiesto e viene messo nella parte iniziale del magazzino
Scenario di successo principale:

1. Il picker riceve una bolla da scansionare
2. L'amministratore cambia il posizionamento dell' oggetto nel database
3. Il picker si logga con username e password
4. Scansionamento del primo codice EAN
5. Visualizzazione della lista dei prodotti da prelevare
6. Viene trovato l'oggetto e prelevato
7. Il sistema convalida e notifica il prelevamento del nuovo oggetto
8. Viene indicato il ripiano e ricaricato con oggetti
9. Il sistema convalida e notifica che il ripiano è stato ricaricato

Estensione:

3.a Picker non riconosciuto:

3.a.1 Il caso d'uso termina con un fallimento

6.a Ripiano già carico:

6.a.1 Il sistema notifica l'errore terminando con un fallimento

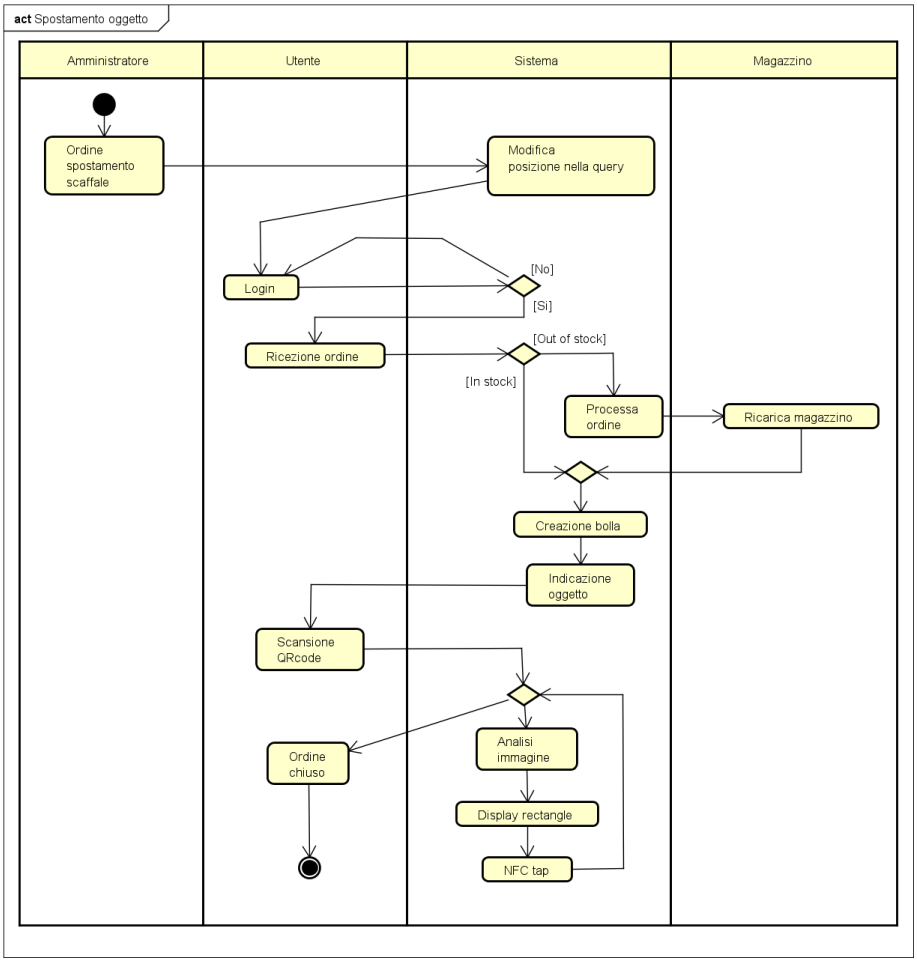

Figura 3.5. UML caso 4

Capitolo 4

Conclusioni

Le nuove tecnologie stanno influenzando sempre più le società industriali. L'impatto di queste in generale nella nostra vita quotidiana è più grande ogni volta.
L'applicazione è stata costruita per il sistema operativo Android, essendo quest'ultimo gratuito, oltre che semplice per lo sviluppo, grazie alla sua natura open source e all'ampia documentazione disponibile sull'API. I servizi disponibili in rete si sono resi fondamentali per la realizzazione delle funzionalità di base dell'applicazione.
Nonostante si pensi che l'implementazione del pick to light sia più favorevole in ambiti prettamente automatizzati, abbiamo constatato che anche in un magazzino senza automatizzazione può bastare un dispositivo android In collaborazione con l'azienda Linear System s.r.l si voleva sviluppare in modo semplice e più intuitivo possibile un progetto in grado di semplificare il picking anche in un contesto quotidiano. Infatti questo progetto di tesi si può applicare anche in altri ambiti come quello del supermercato con lo scopo di aiutare persone ipovedenti, per prelevare prodotti dagli scaffali.

4.1 Punti di attenzione

Sono stati riscontrati diversi problemi durante la progettazione. Un problema proviene dal riconoscimento dei ripiani perché bisogna avere la giusta luminosità e i frame presi dalla fotocamera più fluida cioè con pochi disturbi. In più si è dovuto fare un sort, dall'alto verso il basso, della lista di quadrati (ogni quadrato equivale a un ripiano) dato che inizialmente veniva preso random (per contrastare questo problema si potrebbero usare le reti neurali). Sempre per quanto riguarda i ripiani,

per migliorare il risultato finale, abbiamo ricoperto lo scaffale con dello scotch colorato (rosso nel nostro caso) perché altrimenti venivano riconosciuti più quadrati. Delimitando lo scaffale di rosso possiamo prendere un range consono e usarlo come maschera per il riconoscimento dei quadrati. Abbiamo avuto un problema anche con il posizionamento della fotocamera essendo che quella standard di openCV è messa in modo orizzontale ma a noi serve in verticale per prendere un solo scaffale alla volta e quindi si è dovuto prendere ogni frame in tempo reale girandolo e facendone una trasposta. Un punto su cui soffermarsi è anche la stampa della freccia per indicare il ripiano su cui prelevare. In questo caso il problema è stato riuscire a farlo stampare in modo dinamico in modo che si rimpicciolisse o si ingrandisse in base alla distanza dell'operatore. Un altro problema può essere la carica della batteria del dispositivo a causa dell'uso costante di internet per fare le chiamate al web service.

4.1.1 Sviluppi futuri

L'idea è quella di usare questo sistema in un magazzino integrato con un sistema di navigazione interno allo stabilimento per indicare, tramite una mappa interna, dove si trova lo scaffale su cui si deve prelevare.

Bibliografia

Ambrosino, F. (2018, 07 17). *Visione artificiale: dalla libreria opencv al machine learning.* Retrieved from `https://it.emcelettronica.com/visione-artificiale-dalla-libreria-opencv-al-machine-learning`

Beuth. (1994). Vdi: Vdi guideline 3590..

Christopher, M. (2000, 01). The agile supply chain. *Industrial Marketing Management*, *29*, 37-44. doi: 10.1016/S0019-8501(99)00110-8

Cirulis, A., & Ginters, E. (2013, 12). Augmented reality in logistics. *Procedia Computer Science*, *26*, 14–20. doi: 10.1016/j.procs.2013.12.003

David, S.-L., Frank, C., & Zvi, D. (2000, April). Quantifying the bullwhip effect in a simple supply chain: The impact of forecasting, lead times, and information. doi: https://doi.org/10.1287/mnsc.46.3.436.12069

Feiner, S., Macintyre, B., & Seligmann, D. (1993, July). Knowledge-based augmented reality. *Commun. ACM*, *36*(7), 53–62. Retrieved from `https://doi.org/10.1145/159544.159587` doi: 10.1145/159544.159587

Govoni, L. (n.d.). *Computer vision: come una macchina riconosce gli oggetti.* Retrieved from `https://lorenzogovoni.com/computer-vision/`

Höllerer, T., Pavlik, J. V., & Feiner, S. K. (1999). Situated documentaries: embedding multimedia presentations in the real world. *Digest of Papers. Third International Symposium on Wearable Computers*, 79-86.

Maci, L. (2019, 11 27). *Che cos'è l'industria 4.0 e perché è importante saperla affrontare.* Retrieved from `https://www.economyup.it/innovazione/cos-e-l-industria-40-e-perche-e-importante-saperla-affrontare/`

Reif, R., & Günthner, W. (2009, 05). Pick-by-vision: augmented reality supported order picking. *The Visual Computer*, *25*, 461-467.

Rolland, J. P., Davis, L. D., & Baillot, Y. (2001). A survey of tracking technologies for virtual environments. In *Fundamentals of wearable computers and augmented reality* (pp. 83–128). CRC Press.

Schwerdtfeger, B. (2010, 01). Pick-by-vision: Bringing hmd-based augmented reality into the warehouse.

Soluzioni pick to light per l'ecommerce. (n.d.). Retrieved from `https://www.logisticaefficiente.it/picktolightsystems/magazzino/automazione/soluzioni-pick-to-light-per-ecommerce.html`

Ten Hompel, M., & Schmidt, T. (2008). *Warehouse management.* Springer.

Walch, D. (2007). Augmented and virtual reality based training in the field of logistics. In *Proceedings of the 10th iaested international conference and advanced technology in education, cate 2007.*

Zühlke, K. (2013, 04 13). *Mit dem rfid-armband fertigungsobjekte identifizieren.* Retrieved from `https://www.elektroniknet.de/markt-technik/elektronikfertigung/mit-dem-rfid-armband-fertigungsobjekte-identifizieren-96515.html`