UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria Meccanica

Progettazione di un processo robotizzato per la rimozione di cuciture su calzature mediante sistemi di visione e controllo ibrido di posizione-forza

Design of a robotic process for removal of stitchings on footwear through vision systems and position-force hybrid control

Candidato:
Federico Gentilucci

Relatore:
Prof. Giacomo Palmieri

Correlatore:
Dott. Matteo Forlini

Anno Accademico 2023-2024

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria Meccanica

Progettazione di un processo robotizzato per la rimozione di cuciture su calzature mediante sistemi di visione e controllo ibrido di posizione-forza

Design of a robotic process for removal of stitchings on footwear through vision systems and position-force hybrid control

Candidato:
Federico Gentilucci

Relatore:
Prof. Giacomo Palmieri

Correlatore:
Dott. Matteo Forlini

Anno Accademico 2023-2024

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria Meccanica
Via Brecce Bianche – 60131 Ancona (AN), Italy

Indice

1 Introduzione **1**

1.1 Robotica collaborativa 1
1.1.1 Industria 4.0 1
1.1.2 Robotica industriale e collaborativa 2
1.2 Intelligenza artificiale 5
1.2.1 Machine learning 5
1.2.2 Rete Neurale 5
1.3 Automazione nel settore calzaturiero 7
1.4 CALZARE 4.0 8
1.4.1 Descrizione ed obiettivo del progetto 8
1.4.2 Enti coinvolti 9
1.5 Obiettivo del lavoro 10

2 Identificazione delle cuciture **13**

2.1 Obiettivo del sistema di visione 13
2.2 Algoritmi di matching per estrazione feature da immagine 13
2.2.1 Pattern matching 15
2.2.2 Applicazione del pattern matching 18
2.3 Intelligenza artificiale 26
2.3.1 Preparazione dei dati 27
2.3.2 Acquisizione delle immagini 28
2.3.3 Data augmentation 29
2.3.4 Annotazione 32
2.3.5 Organizzazione delle immagini 34
2.3.6 Nota su hardware usato 35
2.3.7 Nota su Google Colab 35
2.3.8 Training 35
2.3.9 Risultati 37
2.3.10 Studio risultati forniti dopo l'addestramento di un modello YOLO- metriche di valutazione 44
2.3.11 Studio degli iperparametri per l'ottimizzazione 50
2.4 Studio e scelta dei componenti 57
2.4.1 Profilometro 57
2.4.2 Test con profilometro 58
2.4.3 Camera con sensore laser 61

3 Studio e test del processo robot **63**
3.1 Obiettivo del processo 63
3.2 Tool per il test 63
3.3 Programmazione del robot 64
3.4 ROBOGUIDE 66
3.5 Preparazione della cella 67
3.6 Controllo di posizione 73
3.7 Controllo di forza 74

4 Risultati ottenuti **77**
4.1 Cella Robot 77
4.2 Sistema di visione 79

5 Conclusioni **81**

Elenco delle figure

1.1 Esempio di applicazione robot non collaborativa 3
1.2 Esempio di applicazione robot collaborativa 4
1.3 Esempio schematico della struttura di una rete neurale 6
1.4 Schema concettuale delle operazioni eseguite da una CNN. 7
1.5 Rappresentazione del ciclo di vita di una calzatura includendo CALZARE 4.0 9

2.1 Immagine di partenza per l'identificazione delle primitive 15
2.2 Rappresentazione di una binarizzazione per evidenziare dei punti su cui calcolare la trasformata 16
2.3 Sovrapposizione delle curve ottenute con Hough sull'immagine originale 16
2.4 Rappresentazione schematica del funzionamento dell'algoritmo di pattern matching: T=I(i,j): porzione di immagine considerata; Template t=t(u,v) con u,v dimensioni del template in pixel 17
2.5 Immagine di riferimento per il template matching 18
2.6 Ritaglio dell'immagine utilizzato come template 19
2.7 Risultati ottenuti dallo script per il pattern matching 24
2.8 Risultati ottenuti per il pattern matching con diversa immagine di riferimento 24
2.9 Risultati ottenuti per il pattern matching, caso con interpolazione di terzo grado 25
2.10 Rappresentazione grafica dell'andamento della curva di terzo grado rispetto una linea interpolante di primo grado 25
2.11 Immagine contenente le etichette riportate in tabella 2.1 28
2.12 Esempio di immagine acquisita con camera industriale, scarpa Santoni 29
2.13 Esempio di immagine acquisita con camera industriale, scarpa alternativa 29
2.14 Esempio di immagine senza etichette 30
2.15 Immagine annotata selezionando una singola cucitura per ogni etichetta 33
2.16 Immagine annotata selezionando due punti di cucitura per ogni etichetta 34
2.17 Immagine con errore di annotazione 34
2.18 Matrice di confusione semplice 38
2.19 Matrice di confusione con valori normalizzati 39
2.20 Rappresentazione grafica dell'andamento della funzione F1 39
2.21 Rappresentazione grafica dell'andamento della funzione di precisione (*precision*) 40

2.22 Rappresentazione grafica dell'andamento della funzione di richiamo (*recall*) 40
2.23 Rappresentazione grafica dell'andamento relativo tra precisione e richiamo 41
2.24 Grafici per l'andamento delle funzioni di costo, precisione, richiamo e la loro combinazione (mAP) 41
2.25 Esempio di uno dei batch di validazione usato nella rispettiva fase durante l'addestramento 42
2.26 Risultati della previsione effettuata sul batch di immagini in Figura 2.25 43
2.27 Rappresentazione grafica dell'equazione per ricavare la IOU 46
2.28 Grafici con comportamenti e caratteristiche delle labels utilizzate . . 49
2.29 Grafici con istogrammi di correlazione delle labels utilizzate 50
2.30 Matrice di confusione semplice, caso con etichette contenenti una doppia cucitura 53
2.31 Rappresentazione grafica dell'andamento della funzione F1, caso con etichette che includono due cuciture 53
2.32 Rappresentazione grafica dell'andamento della funzione di precisione (*precision*), caso con etichette che includono due cuciture 54
2.33 Rappresentazione grafica dell'andamento della funzione di richiamo (*recall*), caso con etichette che includono due cuciture 54
2.34 Grafici per l'andamento delle funzioni di costo, precisione, richiamo e la loro combinazione (mAP), caso con etichette che includono due cuciture 55
2.35 Esempio di uno dei batch di validazione usato nella rispettiva fase durante l'addestramento 55
2.36 Risultati della previsione effettuata sul batch di immagini in Figura 2.36 56
2.37 Esempio di applicazione industriale con uso di un profilometro . . . 58
2.38 Immagine della nuvola di punti 59
2.39 Immagine della superficie ottenuta dopo aver elaborato la nuvola di punti 59
2.40 Acquisizione eseguita dal profilometro ad area 60
2.41 Esempio di elaborazione per ricavare punto medio della cucitura . . 60
2.42 Camera scelta per l'applicazione 62

3.1 Modello CAD del tool sostitutivo utilizzato nelle prove 64
3.2 Controparte reale del tool utilizzato per i test di traiettoria e controllo di forza 65
3.3 Acquisizione della suola da parte dello scanner 68
3.4 Grafico della disposizione dei punti ricavati dall'immagine scannerizzata 68
3.5 Modello 3D ottenuto dai punti estratti in Figura 3.4 69
3.6 Sistema di fissaggio della scarpa 69

3.7 Risultato fornito dal codice Matlab: mostrati i sei punti misurati con l'insieme dei punti dello scanner dopo la trasformazione 73

4.1 Rappresentazione virtuale ottenuta in ROBOGUIDE della cella di lavoro 78
4.2 Controparte reale della cella di lavoro 78
4.3 Frame ricavato dalla ripresa video 79

Elenco delle tabelle

2.1 Tabella che rappresenta una possibile struttura di alcune etichette . 27
2.2 Tabella di rappresentazione della struttura di una matrice di confusione, caso con due classi di oggetti 48
2.3 Tabella con i valori impostati per gli iperparametri più comuni . . . 52

3.1 Tabella con i valori delle coordinate dei sei punti misurati tramite posizione del TCP del robot 71
3.2 Tabella con i valori della matrice di rototraslazione ottenuta dal codice Matlab 73

Sommario

Il lavoro svolto consiste nello studio e nella progettazione di un sistema robotizzato per applicazione industriale.

Esso si inserisce nel progetto CALZARE 4.0, guidato dall'azienda Santoni S.p.A., con sede centrale a Corridonia, e specializzata nella produzione di calzature prodotte artigianalmente per il mercato di lusso.

Il progetto nasce dall'idea di rientrare nelle attuali tendenze di economia circolare, portate avanti negli ultimi anni in sempre più settori industriali.

Recentemente molte aziende manifatturiere, per poter essere più competitive nel mercato globale, hanno bisogno di automatizzare i loro processi produttivi per ottenere migliori risultati per quanto riguarda qualità del prodotto e produttività.

Il recente interesse nell'ambito ecologico genera inoltre una spinta da parte delle aziende nel rivedere in generale la loro intera catena del valore: a partire dai fornitori e dal loro rapporto con essi, passando per i processi produttivi, fino al post-vendita e al servizio clienti.

Ogni aspetto della catena di produzione dell'industria viene riorganizzata e modificata per poter implementare processi che siano compatibili dal punto di vista della sostenibilità.

Va inoltre considerata la componente delle risorse umane: negli anni recenti si sta cercando di prendere in maggiore considerazione i temi della sicurezza sul lavoro e del benessere degli operatori nel contesto lavorativo.

Per quanto riguarda CALZARE 4.0, l'obiettivo del progetto, una volta completato, sarà quello di fornire ai clienti prodotti con elementi riciclati, oltre ad un servizio di sostituzione della suola, per permettere maggiore vita utile del prodotto.

Tale obiettivo richiede, come già accennato, la rivisitazione di tutta la catena del valore, a partire dai materiali forniti, per passare alle lavorazioni per la produzione della scarpa, per terminare nel post-vendita, informando i clienti delle scelte effettuate per il prodotto e includendo il nuovo servizio.

La ricerca eseguita si occupa,all'interno dell'intero progetto, del processo di rimozione della suola: il sistema studiato consiste in una cella robotizzata che si occupa della rimozione delle cuciture che collegano la suola alla tomaia, cioè la parte superiore della scarpa. Dato che essa verrà riciclata, deve restare intatta al termine delle operazioni di rimozione. Inoltre, è necessario che non restino sfridi di alcun tipo all'interno della scarpa: quindi, il processo consiste nello sfilare la cucitura senza operazioni che comportino la rottura della stessa.

Tenendo in considerazione tali vincoli, lo studio ha inizialmente analizzato gli strumenti hardware e software necessari. Tenendo in considerazione l'automazione del processo, sono stati studiati sensori di visione per l'acquisizione di informazioni sulla scarpa e sulle cuciture, per poi includere lo sviluppo di un software I.A. che permetta di riconoscere le cuciture da rimuovere, per terminare infine con lo studio della cella robotizzata e del suo movimento durante la lavorazione.

L'obiettivo finale del lavoro è valutare la fattibilità della cella di lavoro robotizzata ed esporre una prima proposta di come potrebbe essere ottenuto il sistema finale, in modo da avere una base di partenza per la produzione dell'apparato che poi andrà ad essere implementato realmente nel contesto industriale.

A seguito di un confronto con i vari enti in collaborazione nel progetto, si è ipotizzato un procedimento preliminare per la lavorazione:

- Viene inserita la scarpa da smontare nella cella;
- il robot si avvicina alla calzatura ed esegue una traiettoria attorno al profilo della suola;
- un sistema di acquisizione provvede a identificare una posizione precisa delle cuciture rispetto al terminale del robot;
- un apposito strumento montato sulla flangia robot provvede a sfilare le cuciture presenti lungo la traiettoria percorsa;
- al termine della lavorazione, si dovrebbe ottenere in uscita la scarpa senza il filo che connetteva la suola inferiore con il resto della calzatura, oltre al filo stesso come scarto di lavorazione.

Capitolo 1

Introduzione

1.1 Robotica collaborativa

1.1.1 Industria 4.0

La recente evoluzione della tecnologia ha portato le aziende a un costante aggiornamento della loro struttura produttiva e organizzativa, includendo maggiormente sistemi informatici e macchinari automatici.

Questa tendenza è riassunta nel diffuso concetto di Industria 4.0, il quale viene comunemente denominato come la quarta rivoluzione industriale: integrazione delle nuove tecnologie per migliorare produttività, qualità produttiva degli impianti, ma anche garantire migliori condizioni di lavoro degli operatori. [1]

Il paradigma Industria 4.0 si concentra sulla definizione di smart factory, ovvero una realtà industriale basata su tre concetti di base:

- Smart production: introduzione delle nuove tecnologie per migliorare l'ambiente produttivo della smart factory, includendo e favorendo collaborazione tra operatore e macchine;
- Smart energy: esecuzione e progettazione di tutta la catena del valore tenendo conto della gestione intelligente dell'energia impiegata, considerando l'uso di sistemi di produzione sostenibile e riducendo gli sprechi;
- Smart service: serie di infrastrutture che ottimizzano il contatto tra azienda e cliente finale, ma anche tra azienda ed altri enti esterni, quali fornitori e partner aziendali. Le infrastrutture considerate sono sostenute dalle nuove tecnologie informatiche.

Il tutto è basato sull'inserimento delle nuove tecnologie all'interno della realtà industriale. Le tecnologie principali su cui si basa Industria 4.0 sono [2]:

- Internet of things: uso di tecnologie che costituiscono una rete di elementi eterogenei;
- Simulazione: generazione di simulazioni dei processi produttivi e dei prodotti;

- Cybersecurity: l'introduzione di sistemi informatici richiede necessariamente l'attenzione per la sicurezza dei dati e delle informazioni salvate;
- Integrazione: stretta collaborazione dei sistemi interni ed esterni all'azienda, oltre che con i vari agenti che contribuiscono alla produzione e all'industria in generale;
- Cloud computing: utilizzo di database e sistemi di calcolo remoti. In base alle esigenze, potrebbe essere preferibile delegare l'elaborazione dei dati a server dedicati esterni all'azienda;
- Fabbricazione additiva: una nuova metodologia di produzione da materia prima caratterizzata dall'aggiunta di materiale da zero, in contrapposizione alla rimozione dello stesso nelle tecniche tradizionali;
- Big data analytics: generazione ed analisi di grandi quantità di dati provenienti da sensori e sistemi di acquisizione, ma anche da altri reparti dell'azienda o anche dall'esterno;
- Augmented reality: usate in relazione alle altre tecnologie per garantire maggiore facilità di comprensione e immersione nel mondo digitale;
- Robots e sistemi automatici: sistemi automatici in grado di eseguire operazioni ripetitive ad alta velocità e con elevata affidabilità di esecuzione, spesso in collaborazione o sotto supervisione della componente umana.

Più di recente si è iniziato a diffondere anche il concetto di industria 5.0, che aggiunge ai punti cardine già definiti nella 4.0 altre tendenze industriali quali: transizione digitale, transizione ecologica ed economia circolare, oltre che maggiore centralità dell'uomo nel mondo della produzione industriale. [3]

Queste tendenze portano quindi a un aumento della produttività e dell'organizzazione produttiva, sociale e finanziaria dell'azienda, aumentandone la competitività nel mercato. Ciò favorisce le industrie a aderire ai principi posti dall'industria 4.0 e 5.0.

1.1.2 Robotica industriale e collaborativa

L'introduzione di sistemi automatizzati lungo la linea di produzione rende più veloce e ripetibile il processo di produzione, garantendo un costante livello di qualità e massimizzando l'efficienza.

Ciò è possibile grazie, in particolare, all'implementazione di celle robotizzate: zone delimitate agli operatori umani nelle quali possono operare uno o più sistemi robotici.

La norma ISO 8373 definisce un robot come "meccanismo programmato e attuato, con un grado di autonomia, per effettuare locomozione, manipolazione o posizionamento." [4]

In generale i robot sono dispositivi mobili programmabili e in grado di muoversi autonomamente una volta avviati. In base alla loro programmazione, permettono una elevata varietà di operazioni eseguibili, anche considerando la possibilità di aggiungere appositi dispositivi (tool) atti a eseguire determinati lavori.

Le componenti principali di un sistema robotico richiamano la struttura di un sistema di controllo, in quanto caratterizzate da [5]:

- Una unità di governo, la quale controlla il sistema di azionamento che muove il robot ed elabora i dati ottenuti dai sensori provenienti dall'esterno o collegati al robot stesso.

- Sistema di azionamento: insieme delle interfacce di potenza e attuatori elettrici che permettono il movimento del robot.

- Manipolatore: struttura effettiva del robot, caratterizzata da un certo numero di bracci e dal tool che poi esegue la lavorazione;

- Sensori: dispositivi che raccolgono informazioni sullo stato del robot, come gli encoder per conoscere la posizione angolare dei vari motori sui bracci, oppure sensori esterni, che forniscono informazioni sull'ambiente in cui si trova il robot.

I sistemi robotici industriali sono caratterizzati da elevata ripetibilità, pur mantenendo velocità elevate e quindi tempi di produzione ridotti, grazie al loro controllo per quanto riguarda posizionamento, velocità e forze impiegate durante la lavorazione. Una limitazione tipica dei robot industriali, tuttavia, è la necessità di isolamento rispetto al mondo esterno: durante l'esecuzione dei programmi da parte della macchina, la cella di lavoro è chiusa e non è possibile interagire con ciò che si trova al suo interno(Figura 1.1).

Ciò è dovuto alla pericolosità intrinseca del sistema: una volta avviato il robot industriale, esso non tiene conto di eventuali ostacoli non previsti nella sua area di lavoro, con il conseguente rischio di urti e danni per oggetti o persone nelle vicinanze.

Figura 1.1: Esempio di applicazione robot non collaborativa

La presenza di una zona della catena di produzione non accessibile durante il processo limita la flessibilità di tutto il sistema;va inoltre considerato che la cella resta un pericolo in caso di incidenti, malfunzionamenti o errore umano.

Per questo motivo negli ultimi anni sono stati sviluppati robot che possano operare a stretto contatto con operatori umani senza rischi o pericoli: i robot collaborativi, o cobot (collaborative robot).

La loro struttura è ripresa dai tradizionali robot industriali, sebbene si siano diffusi principalmente cobot antropomorfi, la cui catena cinematica è caratterizzata da un insieme di bracci collegati in serie tra loro, al cui terminale viene poi collegato il tool.(Figura 1.2)

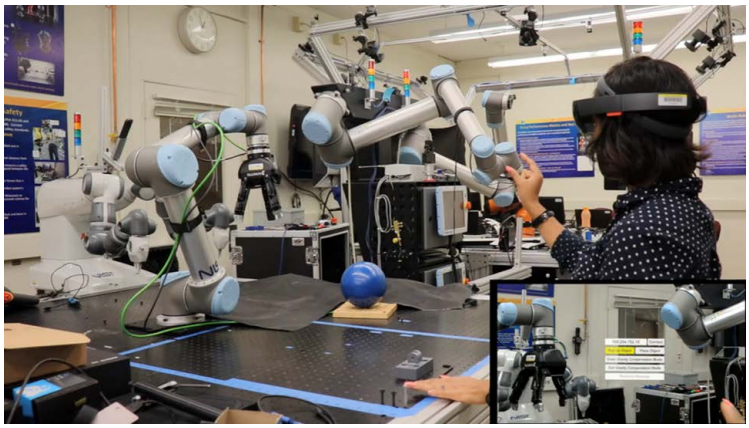

Figura 1.2: Esempio di applicazione robot collaborativa

I robot collaborativi sono progettati e costruiti per poter operare a velocità e dinamiche inferiori, oltre ad includere nella loro unità di controllo degli appositi dispositivi di sicurezza in caso di urti o comportamenti anomali, caratterizzati tipicamente da sensori di accelerazione e di forza.

Vi è inoltre l'uso di materiali plastici o gommosi per ricoprire la struttura interna, in modo da limitare i danni anche in caso di urti con persone.

Naturalmente le dimensioni tipiche sono più contenute rispetto ai robot tradizionali, dato che una maggiore massa in movimento, seppur con velocità limitate, è un maggiore pericolo per gli operatori.

Le limitazioni in dimensioni, capacità cinematiche e dinamiche li rendono mediamente meno performanti dei corrispettivi robot industriali. In tal senso, solitamente l'uso di un cobot rispetto un robot dipende molto dal tipo di lavoro svolto e dall'ambiente in cui viene inserito.

Queste accortezze rendono i cobot in grado di operare sia autonomamente che in collaborazione con utenti umani, con rischi minimizzati e con un inferiore spazio occupato dal sistema.

Le aziende produttrici dei cobot inoltre cercano di massimizzare la semplicità di programmazione e l'ergonomia dei loro prodotti, in modo da non necessitare di personale esperto per l'uso dei cobot ed assicurare facilità di manipolazione.

1.2 Intelligenza artificiale

Il concetto di Intelligenza artificiale (I.A.) assume definizioni varie, con multiple accezioni indicate nel corso del tempo, anche se il concetto di base resta comunque relativamente fissato, definendo l'I.A. come una disciplina che "studia i fondamenti teorici, le metodologie e le tecniche che consentono di progettare sistemi hardware e sistemi di programmi software atti a fornire all'elaboratore elettronico prestazioni che, a un osservatore comune, sembrerebbero essere di pertinenza esclusiva dell'intelligenza umana." [6]

In questo senso, l'obiettivo di un sistema di intelligenza artificiale consiste nell'emulare il comportamento e il ragionamento umano.

In generale, si fa coincidere la nascita del concetto di intelligenza artificiale con la nascita dei primi sistemi di calcolo automatici, sviluppati a partire dagli anni '40, e la cui evoluzione è avvenuta in parallelo allo sviluppo tecnologico in ambito informatico ed elettronico.

1.2.1 Machine learning

L'I.A. come disciplina comprende diversi sottocampi, il più diffuso dei quali è l'apprendimento automatico, detto anche Machine Learning (M.L.): ovvero algoritmi basati sull'apprendimento a partire dai dati, e in grado di migliorare le loro prestazioni senza essere direttamente programmati.

Questi algoritmi solitamente identificano un modello in grado di fare previsioni, prendere decisioni e adattare il suo comportamento in base alla situazione in cui si trova. [7]

Il machine learning può essere ulteriormente suddiviso in diverse categorie, le principali rappresentano il modo con cui l'algoritmo apprende dai dati di partenza:

- Apprendimento supervisionato: vengono fornite informazioni sui dati di partenza in modo da sapere subito quali informazioni devono essere apprese;
- Apprendimento non supervisionato: i dati vengono forniti senza ulteriori informazioni, l'algoritmo deve quindi apprendere in modo autonomo, basandosi esclusivamente sui dati forniti;
- Apprendimento semi-supervisionato: approccio intermedio tra i precedenti.
- Apprendimento con rinforzo: viene inserito un elemento di rinforzo che premia o punisce il modello in addestramento in base al suo comportamento.

1.2.2 Rete Neurale

Una ulteriore sottocategoria di apprendimento automatico è la rete neurale. Una rete neurale è un modello di calcolo la cui struttura stratificata assomiglia alla struttura della rete di neuroni nel cervello, con strati di nodi connessi. [8]

Il funzionamento è basato sulle cellule neurali umane, le quali si attivano in base a certi segnali elettrici o chimici che ricevono in ingresso, e il cui output è a sua volta un segnale diretto ad altri neuroni.

Una rete neurale imita questa struttura, basandosi su funzioni di attivazione, come ad esempio una somma pesata, il cui risultato viene fornito ad una ulteriore funzione, ovvero un neurone successivo.

I vari neuroni sono poi strutturati in livelli o layer, in ognuno dei quali i neuroni ricevono i risultati del livello precedente e forniscono gli output generati al livello successivo.

I livelli che ricevono i dati in ingresso e quelli che restituiscono il risultato in uscita sono detti rispettivamente input layer e output layer, mentre i livelli intermedi, di numero variabile in base al tipo di rete neurale, sono detti livelli nascosti o hidden layer.(Figura 1.3)

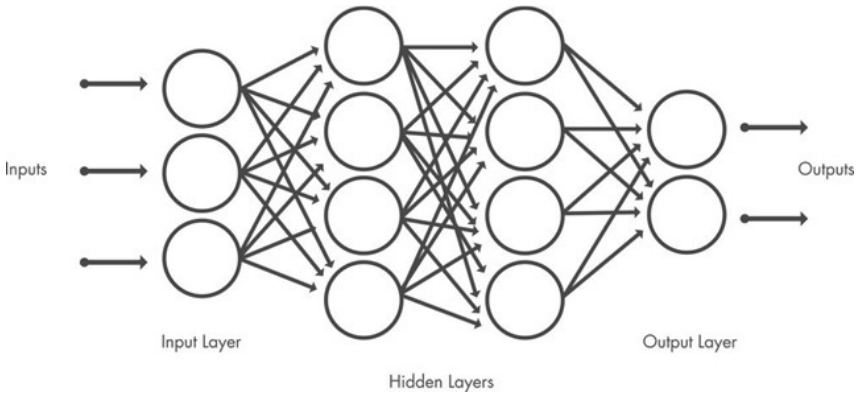

Figura 1.3: Esempio schematico della struttura di una rete neurale

Recentemente è stato introdotto anche il concetto di deep learning: basato sulla rete neurale, si contraddistingue per l'elevato numero di neuroni e livelli, oltre che per la grande mole di dati necessari per assicurare delle prestazioni di molto superiori alle reti neurali tradizionali.

Un particolare tipo di rete neurale "profonda" è la rete neurale convoluzionale, o CNN (convolutional neural network): consiste nell'applicazione di una rete di questo tipo nell'ambito della visione artificiale, in modo da apprendere e generare modelli a partire da dati visivi.

Gli algoritmi usati sono pensati per operare con delle matrici, ovvero la struttura di base con cui può essere interpretata un'immagine digitale. Nei casi comuni i vari livelli della rete eseguono operazioni sui pixel dell'immagine per ricavarne delle caratteristiche (features).(Figura 1.4)

Le operazioni più comuni effettuate dai livelli della CNN sono [9]:

- Convoluzione: applicazione di filtri convoluzionali per estrarre features dall'immagine;

- Unità lineare rettificata (ReLU): operazione che mappa i valori negativi a zero e mantiene intatti i valori positivi; è un caso particolare di funzione di attivazione.
- Pooling: esegue la riduzione del numero di dati in uscita, senza cancellare le informazioni necessarie per l'obiettivo del modello, in modo da semplificare l'output.

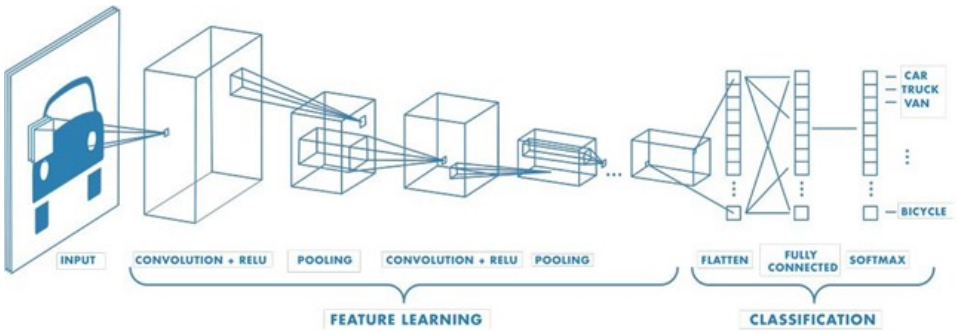

Figura 1.4: Schema concettuale delle operazioni eseguite da una CNN.

1.3 Automazione nel settore calzaturiero

L'industria calzaturiera in generale è caratterizzata da una rilevante presenza di personale: storicamente si sono diffusi i mestieri di artigiani calzolai, i quali, con l'avvento dell'era industriale, contribuiscono per la maggior parte nella realizzazione di calzature.

Naturalmente l'inserimento nel mercato causato dalla globalizzazione ha portato alla necessità di innovare il settore includendo macchinari automatici.

Tuttavia, il loro inserimento avviene principalmente in parti ancora limitate della catena di produzione, relegando l'uso di macchinari automatici e semiautomatici a operazioni non artigianali (come pick and place, incollaggio, manipolazione della suola). [10]

Recentemente, a causa della continua evoluzione del mercato, del settore industriale e manifatturiero e della società in generale, le aziende calzaturiere si stanno orientando verso l'installazione di sistemi automatici e robotici.

La natura dei processi di produzione, tuttavia, rallenta l'intervento di macchine e robot lungo la catena di produzione, poiché:

- Vi è necessità di una elevata manualità ed esperienza per garantire una buona qualità del prodotto finale, soprattutto se si tratta di oggetti di lusso;
- Le tecniche perfezionate dagli artigiani potrebbero non essere adatte per essere svolte da celle robot, a causa della continua variabilità del prodotto, la quale va oltre le tipiche e ripetitive mansioni di questi sistemi;

- Infine, è ancora molto diffusa, nei lavoratori in generale, il timore della sostituzione della risorsa umana con dei macchinari: sebbene ciò non sia tendenzialmente vero, continua ad essere presente una certa diffidenza nell'installazione di robot e sistemi simili nelle industrie manifatturiere. Per rimuovere questo ostacolo, si cerca di proporre una collaborazione tra uomo e macchina, piuttosto che una sostituzione, soprattutto in considerazione dei principi di industria 4.0.

Tali difficoltà, infatti, possono essere superate considerando l'inserimento dei robot collaborativi, in modo da garantire uno stretto contatto tra macchinario e operatore; inoltre l'aggiunta di sensori (es. sensori di forza o sensori di visione) per incrementare la capacità di adattamento a task non ripetitivi e variabili, incrementa la flessibilità offerta dai cobot.

Nel lavoro svolto, tenendo presenti le considerazioni espresse, è stata progettata una cella di lavoro fornita di un robot collaborativo.

Le fasi di progettazione e studio del sistema sono state agevolate poichè si è potuto interagire direttamente con il robot nell'esecuzione delle varie parti.

Allo stesso modo, in seguito alla sua installazione nel processo produttivo industriale, sarà dunque facilitato il lavoro per gli operatori adetti alla cella.

1.4 CALZARE 4.0

1.4.1 Descrizione ed obiettivo del progetto

Il progetto "CALZARE 4.0 - CALZatura circolARE 4.0" è un progetto di Ricerca Industriale e di Sviluppo Sperimentale, sviluppato dall'azienda Santoni S.p.A., in collaborazione con due PMI dello stesso settore: Robot System Automation S.r.l ed E Plast S.r.l, oltre alla partecipazione dell'Università Politecnica delle Marche, finanziato dal Ministero delle Imprese e del Made in Italy.

Il progetto prevede lo studio, lo sviluppo e l'implementazione di processi e servizi innovativi a supporto della realizzazione di una nuova calzatura ecosostenibile. Esso mira a seguire i paradigmi di industria 4.0 nell'ottica umanocentrica e della sostenibilità.

L'obiettivo di CALZARE 4.0 è di creare un ambiente industriale collaborativo, basato quindi sulla cooperazione uomo-macchina, sempre restando nell'ottica human centered.

Il progetto, inoltre, propone un approccio di economia circolare: è previsto infatti un servizio di circolarità che permette di dare nuova vita alle calzature usate.

Infine, l'azienda Santoni mira a sensibilizzare la propria clientela sui temi della sostenibilità e dell'economia circolare, oggi sempre più rilevanti per l'ambiente e la salute.

La finalità del progetto è un sistema di produzione di una scarpa ecosostenibile, con elementi ottenuti con materiali riciclati, incluso un servizio di recupero tramite sostituzione della suola inferiore.

In Figura 1.5 si riporta una rappresentazione schematica di come potrebbe essere un ciclo di vita di una calzatura.

Il lifecycle inizia a partire dalla vendita del prodotto; in seguito all'uso, la scarpa si deteriora, quindi il cliente ricontatta l'azienda in modo da eseguire il recupero della calzatura: la suola viene rimossa e sostituita, la parte superiore (tomaia) viene rammendata e connessa con la nuova suola.

A questo punto la scarpa viene restituita al cliente, che può riutilizzarla.

Una volta terminato il secondo ciclo di vita, l'intera scarpa viene riciclata, recuperando le materie prime.

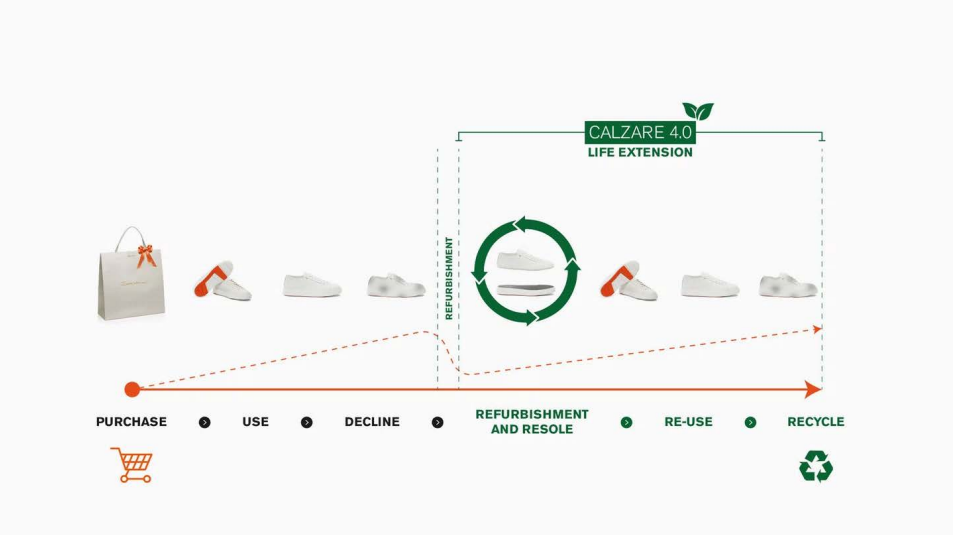

Figura 1.5: Rappresentazione del ciclo di vita di una calzatura includendo CALZARE 4.0

1.4.2 Enti coinvolti

- Santoni S.p.A.:è un'azienda manifatturiera specializzata nella produzione e vendita di calzature di lusso con sede centrale a Corridonia (MC).

  Nata nel 1975 con un primo laboratorio di calzature, si è evoluta in quella che oggi è un'icona del luxury footwear made in italy a livello internazionale, grazie anche alla combinazione di tradizione e innovazione: l'azienda ha infatti coniugato un sistema di progettazione ad alta tecnologia con un sistema di produzione preindustriale, basato esclusivamente sul lavoro artigianale.

  Negli ultimi anni l'azienda si è avvicinata alla nuova tendenza industriale globale di innovazione 4.0, con l'implementazione delle tecnologie digitali e di automazione, oltre che alla sostenibilità ambientale e sociale.

- Robot System Automation S.r.l.: Robot System Automation (RSA) è un'azienda, nata nel 1984, che opera nel settore dell'automazione industriale, in

particolare nel settore robotico e dell'automazione per la produzione di calzature e pelletteria.

RSA è in grado di fornire supporto alle aziende di questi ambienti, grazie alla sua esperienza come costruttore di macchine.

Allo stesso modo, RSA permette di progettare, assemblare e installare vari sistemi automatici, garantendo elevata flessibilità e specializzazione in base alla richiesta del cliente.

Il supporto alle aziende che collaborano con RSA va dallo studio di fattibilità fino all'assistenza post-vendita.

- E Plast S.r.l.: è una PMI, fondata nel 2020 e con sede a Loro Piceno (MC), che si occupa della produzione di fondi in materiali plastici.

  In particolare, l'azienda è specializzata nella produzione di suole, servendosi delle ultime tecnologie 4.0 presenti sul mercato.

  E plast si è sempre basata sul rispetto per l'ambiente, tanto da spingersi a investire in una lean production dal basso impatto ambientale, secondo il GLOBAL RECYCLE STANDARD (GRS).

- Università Politecnica delle Marche: Nel progetto CALZARE 4.0, l'Università Politecnica delle Marche è rappresentata dal Dipartimento di Ingegneria Industriale e Scienze Matematiche (DIISM).

  Il dipartimento si occupa di ricerca, didattica e trasferimento tecnologico nei vari ambiti dell'ingegneria industriale, dalla meccanica alla meccatronica passando per il design industriale.

  Il dipartimento include diversi gruppi di ricerca complementari e sinergici tra loro, i quali affrontano problematiche di simulazione virtuale di processi e prodotti, di tecnologie produttive e impianti di trasformazione, di sistemi energetici, human centered design, metodi di progettazione, robotica applicata, ecc.

  Il tutto supportato da diversi laboratori per l'attività di ricerca e sperimentazione dei vari ricercatori.

1.5 Obiettivo del lavoro

Il presente lavoro si inserisce quindi nel progetto CALZARE 4.0 come parte del lavoro condotto dall'Università Politecnica delle Marche.

In particolare, lo studio riguarda la parte di introduzione e implementazione del servizio di recupero della scarpa attraverso sostituzione della suola.

Il lavoro ha l'obiettivo di comprendere il task da svolgere da parte del macchinario, per poi proporre una possibile soluzione, progettando la cella robot che svolgerà

l'operazione e analizzando possibili aggiunte nella sensoristica del sistema per facilitare l'esecuzione della lavorazione.

Gli enti coinvolti nel progetto hanno già ipotizzato l'utilizzo di un cobot per eseguire l'operazione, integrandolo con un sistema di visione per assicurare precisione, includendo un algoritmo di intelligenza artificiale per permettere maggiore flessibilità; perciò, lo studio volgerà principalmente su questi aspetti, sebbene siano state comunque prese in considerazione eventuali alternative.

Capitolo 2

Identificazione delle cuciture

2.1 Obiettivo del sistema di visione

Il task da eseguire può essere riassunto in tal modo: la scarpa viene posizionata all'interno della cella robotizzata, nella quale il robot si avvicinerà e provvederà a rimuovere le cuciture, seguendone l'andamento lungo il bordo della suola.

L'operazione avviene su scarpe usate: ciò complica la determinazione della posizione delle cuciture dato che, a differenza di un modello nuovo, possono essere presenti deformazioni dovute all'uso della scarpa.

L'impossibilità di usare traiettorie predefinite per determinare il movimento del robot rende necessaria la presenza di un sistema di acquisizione che misuri la posizione dei vari punti di cucitura, in modo che si conoscano, per ogni scarpa che si può presentare, i punti su cui eseguire il lavoro di rimozione.

L'obiettivo è quindi ricercare un sistema di misura che fornisca la posizione spaziale dei vari punti di cucitura; dato che il sistema dovrà operare in corrispondenza del centro della cucitura, sarà necessario individuare solo la posizione di questo.

2.2 Algoritmi di matching per estrazione feature da immagine

Il primo studio per il riconoscimento delle cuciture è stato condotto considerando l'uso di algoritmi di pattern matching sull'immagine.

Le due principali tecniche di matching nelle immagini sono *pattern matching* e *geometric matching*:

- *Pattern matching*: basato su confronto tra template e l'immagine.

  È necessaria l'esistenza di una immagine template, ovvero un'immagine da usare come riferimento per eseguire l'algoritmo: il programma di matching sovrappone il template a una porzione dell'immagine, per poi confrontare i valori dei pixel corrispondenti.

- *Geometric matching*: basato sull'identificazione delle primitive nell'immagine, ovvero si cercano le strutture geometriche (linee, curve, ...) presenti nell'immagine.

> Le curve vengono ricavate attraverso l'applicazione, sull'intera immagine, di trasformazioni che identificano le equazioni costitutive dei luoghi geometrici: la più diffusa è la trasformata di Hough. [11]

È stato eseguito un primo test per entrambi i metodi, ma è stato scartato il geometric matching: esso risulta più sensibile al rumore e alla condizione dell'immagine, risultando meno flessibile.

L'immagine presa per eseguire il test non è stata scattata con l'hardware dedicato; tuttavia, i risultati migliori ottenuti fin da subito per il pattern matching hanno confermato la preferenza per questo ultimo metodo.

Inoltre, bisogna considerare che è richiesta l'identificazione dei singoli punti di cucitura, compito non ottenibile facilmente attraverso il matching geometrico.

Si riportano il codice di prova e i risultati ottenuti.

```
"""code that receives an image and gives the hough transform og the main
    edges of the image"""
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Loads a grayscale version of the image
image = cv2.imread(r"image_path", cv2.IMREAD_GRAYSCALE)

# Application of the Canny filter to get the edges
edges = cv2.Canny(image, 100, 250, apertureSize=3)
kernel = np.ones((9,9),np.uint8)
closing = cv2.morphologyEx(edges, cv2.MORPH_CLOSE, kernel)
plt.imshow(closing, cmap='gray')
plt.title('Detected Patterns')
plt.show()

# Apply Hough tranform
lines = cv2.HoughLines(edges, 1, np.pi / 180, 150)

# Return the image to RGB
image_color = cv2.cvtColor(image, cv2.COLOR_GRAY2BGR)

# Verification of the identification of lines
if lines is not None:
    for rho, theta in lines[:, 0]:
        # Find the external points on the lines
        a = np.cos(theta)
        b = np.sin(theta)
        x0 = a * rho
        y0 = b * rho
        x1 = int(x0 + 1000 * (-b))
        y1 = int(y0 + 1000 * (a))
```

```
        x2 = int(x0 - 4000 * (-b))
        y2 = int(y0 - 4000 * (a))
        if rho > 40 or rho < -40 :
            continue
        else:
        # Draw the line
            cv2.line(image_color, (x1, y1), (x2, y2), (0, 0, 255), 2)

# Show image
cv2.imshow('Punti allineati - Linee trovate', image_color)
plt.imshow(image_color, cmap='gray')
plt.title('Detected Patterns')
plt.show()
```

Figura 2.1: Immagine di partenza per l'identificazione delle primitive

Dalla Figura 2.3 si può notare immediatamente la difficoltà dell'algoritmo nel trovare la linea delle cuciture, concentrandosi piuttosto sul bordo della scarpa. Si procede quindi eseguendo un algoritmo di pattern matching sulle immagini delle cuciture.

2.2.1 Pattern matching

Il procedimento consiste nell'identificare il pattern da trovare, ricavarlo dall'immagine originale o da un'altra immagine in cui sia presente, per poi darlo in input come template di riferimento all'algoritmo. Successivamente si inserisce anche l'immagine da cui ricavare il risultato.

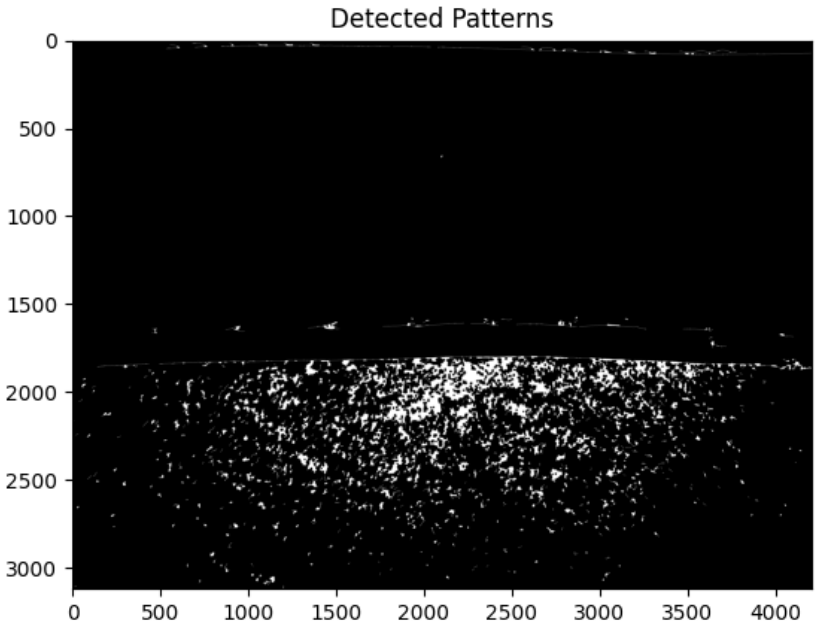

Figura 2.2: Rappresentazione di una binarizzazione per evidenziare dei punti su cui calcolare la trasformata

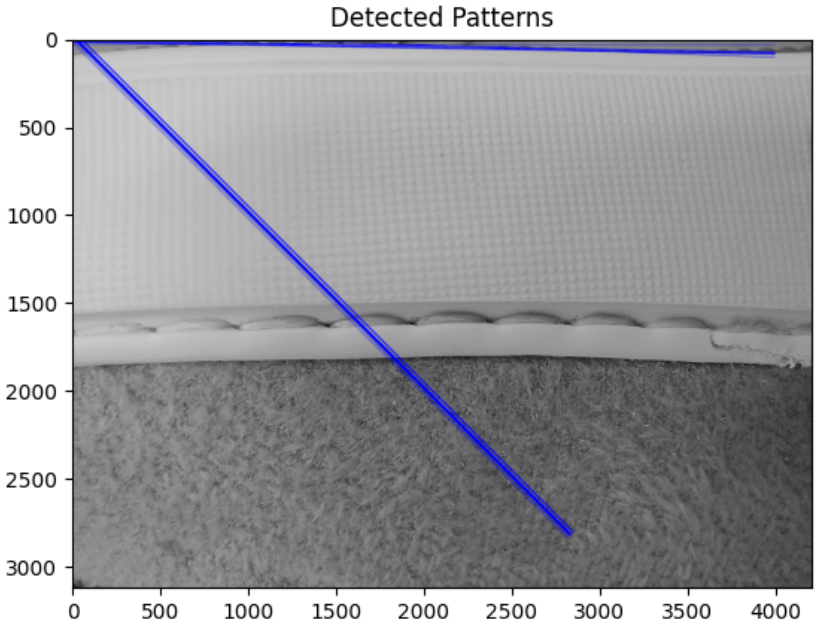

Figura 2.3: Sovrapposizione delle curve ottenute con Hough sull'immagine originale

L'algoritmo esegue il matching facendo scorrere il template lungo una direzione

rispetto l'immagine sotto analisi e confrontando i valori dei singoli pixel: si esegue poi una funzione di costo per dare uno SCORE, ovvero un livello di somiglianza, alla porzione di immagine confrontata con il template.

Se il valore di SCORE ottenuto è sufficientemente elevato, la porzione di immagine viene considerata contenente il pattern.

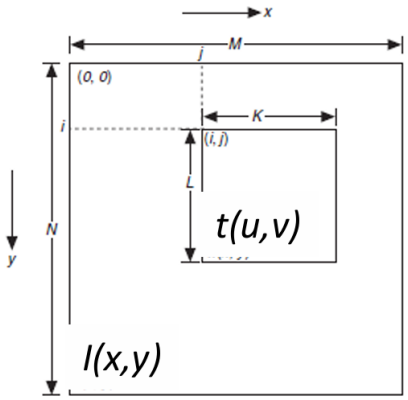

Figura 2.4: Rappresentazione schematica del funzionamento dell'algoritmo di pattern matching: T=I(i,j): porzione di immagine considerata; Template t=t(u,v) con u,v dimensioni del template in pixel

Il valore della somiglianza può essere calcolato con varie funzioni:

- SAD (sum of the absolute difference), dove si esegue una sommatoria delle differenze nei valori dei pixel dell'immagine e del template:

$$S_{\text{SAD}} = \frac{1}{n} \sum_{(u,v)\in T} |t(u,v) - I(u+i, v+j)| \tag{2.1}$$

- SSD (sum of squared value difference), in cui si valuta la differenza quadratica anziché la differenza semplice; questo metodo riduce la probabilità di falsi positivi nei risultati, dato che eventuali errori verranno elevati al quadrato:

$$S_{\text{SSD}} = \frac{1}{n} \sum_{(u,v)\in T} [t(u,v) - I(u+i, v+j)]^2 \tag{2.2}$$

- NCC (normalized cross correlation); in questo metodo si includono anche gli effetti di proprietà statistiche quali deviazioni standard e valore medio dei pixel:

$$S_{\text{NCC}} = \frac{1}{n} \sum_{(u,v)\in T} \left(\frac{t(u,v) - m_t}{\sqrt{\sigma_t^2}} \cdot \frac{I(u+i, v+j) - m_I(i,j)}{\sqrt{\sigma_I^2}} \right) \tag{2.3}$$

Dove m_t, m_I (i,j): valori medi di template e porzione di immagine; σ_t^2, σ_I^2: deviazioni standard;

In caso di corrispondenza perfetta tra template e immagine si ha SAD = SSD = 0, oppure NCC = 1, evento raramente ottenibile; perciò, si definisce una soglia al di sopra della quale si considera la corrispondenza. Al variare del valore di soglia, i risultati saranno più o meno attendibili. La soglia influisce anche sulla robustezza dell'algoritmo a variazioni di sfondo nell'immagine e sull'eliminazione del rumore.

2.2.2 Applicazione del pattern matching

Nel caso specifico del lavoro svolto, l'algoritmo è usato per identificare il profilo della cucitura e ottenere una curva di fitting che rappresenti il suo andamento.

L'idea consiste nell'usare la curva per controllare il movimento del robot e correggerne la traiettoria. In ingresso al codice sono necessari un template e un'immagine su cui eseguire il confronto.

Figura 2.5: Immagine di riferimento per il template matching

Come riferimento si prende una semplice fotografia delle cuciture di una scarpa Santoni (Figura 2.5), dalla quale è stato poi ritagliato un singolo punto di cucitura che farà da template.

Successivamente si ricava un template da usare nell'algoritmo ritagliandolo dall'immagine originale (Figura 2.6).

Si riporta di seguito il codice utilizzato, seguito da una sua descrizione:

```
"""Code for template recognition, clustering of the found templates, and
    extraction of curves from the obtained points.
A second-degree polynomial interpolation system was used
"""
```

Figura 2.6: Ritaglio dell'immagine utilizzato come template

```
import cv2
import numpy as np
from scipy.stats import mode
from scipy.interpolate import interp1d
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN


# Load the main image and the template (wire path image)
main_image = cv2.imread(r"image_path", cv2.IMREAD_GRAYSCALE)
template = cv2.imread(r"image_path", cv2.IMREAD_GRAYSCALE)

#load image and manage optioin for saving file
original =
    cv2.imread(r"C:\Users\feder\Desktop\tirocinio\immagini\im4.jpg",
    cv2.IMREAD_GRAYSCALE)
filename = 'test_im4_interpolation'
save_option = False
# Perform template matching
result = cv2.matchTemplate(main_image, template, cv2.TM_CCOEFF_NORMED)

# Set a threshold to detect multiple patterns
threshold = 0.55 # Adjust this threshold based on your use case

# Find locations where the matching result exceeds the threshold
locations = list(np.where(result >= threshold)) #location[0] is the y
    coordinate
# Get the dimensions of the template
h, w = template.shape
locations_c = locations.copy()
points = np.column_stack((locations[1],locations[0])) #reshaped for next
    operation
for i in range(len(points)):
    points[i][0] += w//2
    points[i][1] += h//2

# Scale the x-coordinate to de-emphasize horizontal distance
```

```
scaled_points = points.copy()
scaled_points[:,0] = points[:,0] * 0.01 # Scale down x-axis

# Apply DBSCAN [clustering]
dbscan = DBSCAN(eps=600, min_samples=2)
labels = dbscan.fit_predict(points)
clustered_points = mode(labels, keepdims = False)[0]
labeled_points = np.column_stack((points,labels)).tolist() #created 3d
    points with 3rd dimension = label
nested = [sublist for sublist in labeled_points if clustered_points in
    sublist]
plot_points = np.delete(np.asarray(nested),2,1) #removed label after
    nesting

#Point verification along x axis -- see clustering_problem image
medium_value = np.mean(points[:,0])
right = medium_value + w
left = medium_value - w
selected = []
for point in points:
    if point not in plot_points:
        if left < point[0] < right:
            selected.append(point)
    else:
        pass
try:
    plot_points = np.concatenate((plot_points,np.asarray(selected)))
except Exception as Ex:
    print('Raised exception at point verification step') #used when
        selected is empty

#repeating the clustering to select the single points for every sewing
    point
dbscan_restricted = DBSCAN(eps=100, min_samples=2)
labels_restricted = dbscan_restricted.fit_predict(plot_points)
clustered_restricted= mode(labels_restricted, keepdims = False)[0]
labeled_restricted =
    np.column_stack((plot_points,labels_restricted)).tolist() #3d points
#nested_restricted = [sublist for sublist in labeled_restricted if
    clustered_restricted in sublist]

set_labels = list(set(labels_restricted))

clusters = []
for label in set_labels:
    clusters.append([item for item in labeled_restricted if item[2] ==
        label])
```

```
    clusters[label] = np.delete(np.asarray(clusters[label]),2,1)
    #getting points on different clusters

#median point per cluster
median_points = []
for cluster in clusters:
    median_points.append(np.median(cluster,axis = 0))
#convertiti punti in array per il plot
npa = np.asarray(median_points)

#fitting spline ###################################
npa = np.sort(npa,axis = 0)
out_spline = interp1d(npa[:,0],npa[:,1], kind = 'quadratic')
x_d = np.linspace(npa[:,0].min(),npa[:,0].max(),100)
y_d = out_spline(x_d)

# Loop over the locations and draw rectangles around matched regions
for pt in zip(*locations_c[::-1]):
    top_left = pt
    bottom_right = (top_left[0] + w, top_left[1] + h)
    cv2.rectangle(main_image, top_left, bottom_right, 25, 2)

#fitting line (for comparison with the spline)
coefficients = np.polyfit(npa[:,0], npa[:,1], deg=1) #con deg = 2 create 2
    linee invece che 1 parabola
fit_line = np.polyval(coefficients, npa[:,0])

######################################## Display the result

fig = plt.figure()
fig.add_subplot(221)
plt.imshow(original,cmap='gray')
plt.title('Original')
plt.axis('off')

fig.add_subplot(222)
plt.imshow(main_image,cmap='gray')
plt.scatter(points[:, 0], points[:, 1], c=labels)
plt.title('Detected patterns')
plt.axis('off')

fig.add_subplot(223)
plt.imshow(original,cmap='gray')
plt.scatter(npa[:,0],npa[:,1] , c='black')
plt.title('Selected points for fit')
plt.axis('off')
```

```
fig.add_subplot(224)
plt.imshow(original,cmap='gray')
plt.plot(x_d, y_d, label='Spline Curve')
plt.title('Fitting line')
plt.axis('off')
if save_option:
    plt.savefig(filename + '.pdf')
else:
    pass
plt.show()

#plotting the points and the interpolation curves
plt.plot(x_d, y_d, label='Spline Curve')
plt.plot(npa[:,0], fit_line, color='red',linewidth = 2.0)
plt.scatter(npa[:,0],npa[:,1])
if save_option:
    plt.savefig(filename + ' graph.pdf')
else:
    pass
plt.show()
```

Il matching viene effettuato attraverso la funzione *matchTemplate*, disponibile nella libreria *Opencv*. [12]

La funzione ha come risultato una mappa dell'immagine dove sono rappresentati i vari risultati del confronto tra template e immagine di sfondo. Viene quindi definita una soglia, impostata a 0,55. La soglia è utilizzata poi per selezionare solo i punti dell'output che soddisfano valori superiori, in modo da ridurre i casi di falsi match.

I punti in output dal pattern matching coincidono con il pixel in alto a sinistra rispetto quella che è la forma del template: si esegue quindi una traslazione in modo da avere in ingresso all'algoritmo di clustering i punti centrali dei pattern trovati.

In seguito all'ottenimento dei punti di match si esegue poi un algoritmo di clustering. Dopo aver riordinato i punti in un vettore, adattandoli quindi per poter essere processati, si utilizza la funzione *dbscan*, appartenente alla libreria *Sklearn*. [13]

Questo specifico metodo è creato allo scopo di identificare punti nello spazio 2D appartenenti a cluster spaziali: i punti vengono distinti in base alla relativa distanza uno dall'altro.

Una ulteriore caratteristica è che non richiede in ingresso un numero di cluster predefinito da parte dell'utente, a differenza del più diffuso algoritmo di clustering della stessa libreria *kmeans*. Ciò lo rende più conveniente rispetto agli altri algoritmi che processano valori scalari o punti bidimensionali.

Per evitare errori dovuti a falsi positivi si esegue un ulteriore controllo sui punti trovati: ipotizzando che l'immagine mostri le cuciture disposte verticalmente, viene modificata la funzione di clustering riducendo l'impatto delle ascisse sull'algoritmo.

Le cuciture identificate non coincidono con tutte quelle presenti nell'immagine, evento in generale non voluto ma dipendente dal tipo di acquisizione eseguita, oltre che

dal template stesso, e in generale minimizzabile ma non eliminabile completamente.

La riduzione dell'effetto della coordinata x rischia tuttavia di includere falsi positivi nell'immagine. Per evitare possibili errori, si esegue una media delle ascisse dei punti, per poi eliminare quelli che non rientrano all'interno di un certo range, impostato pari al doppio della lunghezza stessa del template, ovvero si eliminano punti che sarebbero troppo lontani da quella che è la linea di cucitura.

Si continua poi con la divisione dei vari cluster trovati, in modo da identificare un punto medio per ogni cluster. Ciò viene eseguito ripetendo l'algoritmo di Clustering, in questo caso con valori più restrittivi sulle distanze necessarie per definire il cluster: in particolare, si scelgono soglie di valore inferiore alle dimensioni del template stesso.

Dato che le cuciture mediamente sono sempre a una distanza costante l'una dall'altra, la presenza di eventuali punti vicini tra loro di solo alcuni pixel vorrebbe dire che l'algoritmo ha ottenuto valori elevati nelle vicinanze del centro esatto: per ridurre il carico sulla memoria, e non tenere in considerazione più volte lo stesso punto, si calcola il punto medio delle misure ottenute, per poi non tenere più in conto gli output multipli.

I punti trovati in ogni cluster sono quindi appartenenti a template riferiti alla stessa cucitura: dopo una semplice riorganizzazione dei punti trovati, il risultato sarà quindi una lista contenente quelli appartenenti alla stessa cucitura.

Il risultato finale è quindi un punto posizionato mediamente al centro delle varie cuciture.

Il processo si conclude eseguendo un fitting, ovvero una interpolazione: i vari punti vengono presi come base per definire una linea o una curva interpolante.

Essendo essi distribuiti fisicamente su una superficie curva, si esegue una interpolazione usando un polinomio di secondo grado. L'algoritmo per l'interpolazione è disponibile attraverso la libreria *numpy* [14]:il comando usato è *interp1d*. I punti vengono infine mostrati su un grafico e sovrapposti all'immagine originale (Figura 2.7).

Per verificare l'efficacia del metodo, si modifica l'immagine di riferimento, mantenendo il template iniziale; il risultato è riportato in Figura 2.8. Per semplicità, la fotografia viene scattata in modo da avere condizioni simili alla precedente. Tale scelta è accettabile in quanto si ipotizza che le immagini catturate dalla telecamera, una volta montata sul robot, siano simili tra loro, in quanto prese in successione.

Un problema nato in seguito all'osservazione dei risultati è la tendenza della curva interpolante ad avere comportamenti non voluti (Figura 2.9): eseguendo una curva di terzo grado, è stata notata la tendenza a descrivere una traiettoria che ritorna su sé stessa, come visto nella schematizzazione in Figura 2.10. Comportamento meno evidente ma comunque accennato nel caso della curva quadratica. Tali andamenti rischiano di riflettersi negativamente sulla traiettoria che dovrà seguire il robot.

Il codice richiede comunque la definizione manuale di un valore di soglia per definire i punti da prendere in considerazione per il clustering.

Inoltre, il controllo è valido solo per pattern orientati verticalmente rispetto l'immagine: il metodo è espandibile, includendo anche la possibilità di cuciture

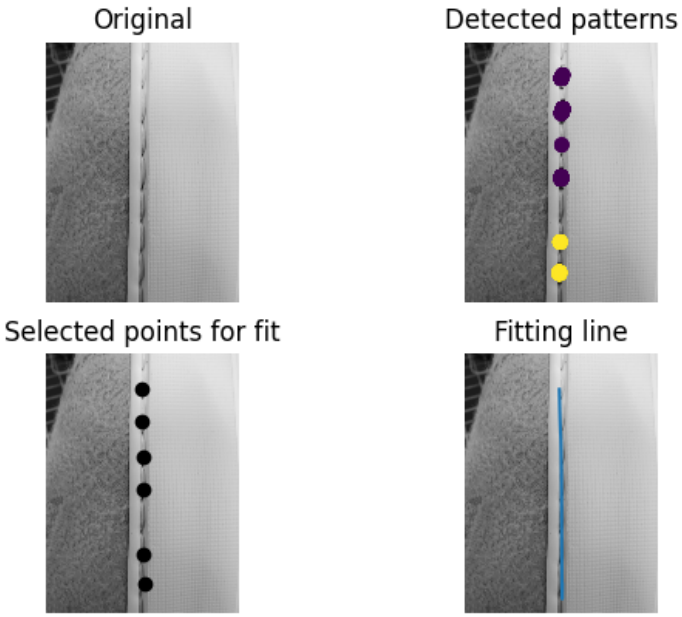

Figura 2.7: Risultati ottenuti dallo script per il pattern matching

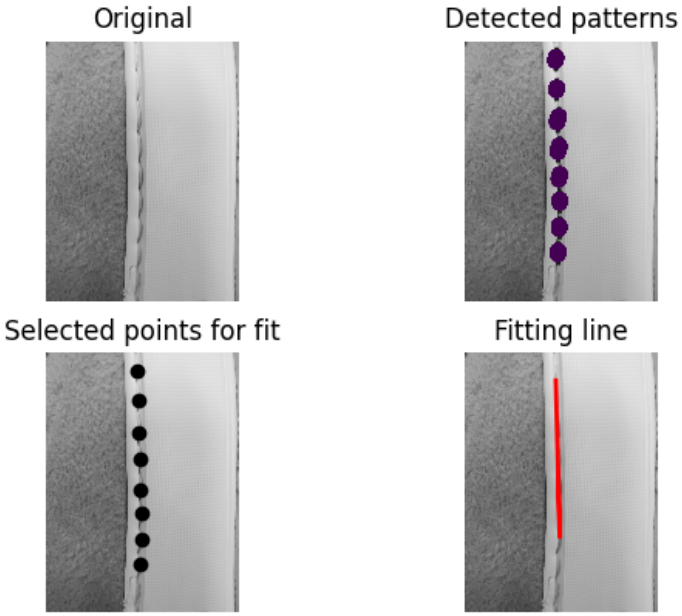

Figura 2.8: Risultati ottenuti per il pattern matching con diversa immagine di riferimento

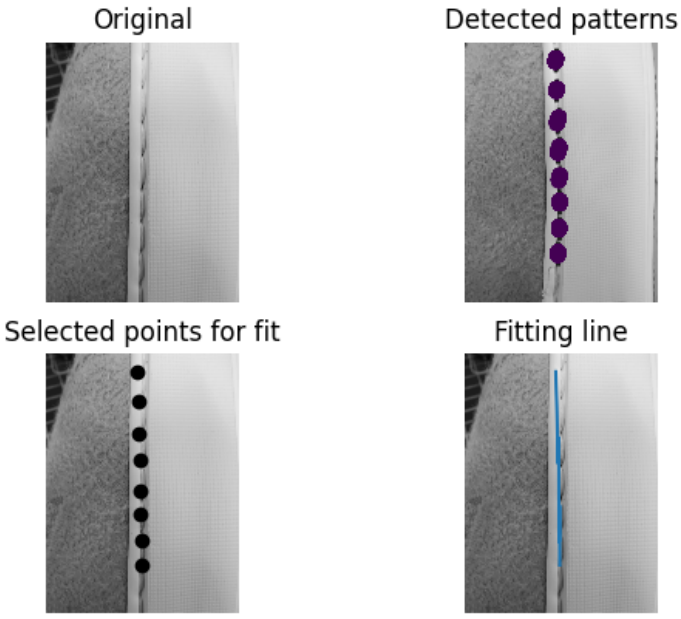

Figura 2.9: Risultati ottenuti per il pattern matching, caso con interpolazione di terzo grado

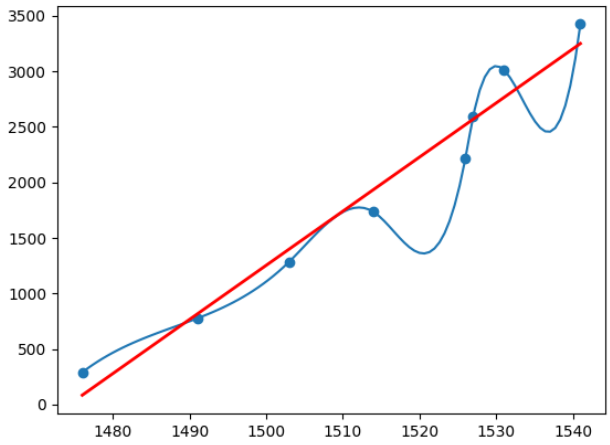

Figura 2.10: Rappresentazione grafica dell'andamento della curva di terzo grado rispetto una linea interpolante di primo grado

posizionate orizzontalmente, ma si procede ipotizzando che, una volta montata la telecamera sul terminale del robot, questa acquisisca immagini sempre allo stesso modo, e quindi con cuciture rivolte sempre nella stessa direzione. Nonostante ciò, resta comunque una riduzione della flessibilità del programma.

Infine, il programma ha riscontrato evidenti difficoltà a generalizzare, in quanto se si riesegue l'algoritmo cambiando immagine di riferimento, difficilmente si ottengono delle identificazioni adeguate delle cuciture.

Queste limitazioni, oltre al fatto che la curva interpolante può avere comportamenti instabili, rendono il metodo non sufficientemente flessibile e robusto a un'applicazione industriale. Ciò è provato dal test dell'algoritmo su diverse immagini, riscontrando risultati non soddisfacenti nel matching.

Come alternativa più adattabile si è optato per un algoritmo di intelligenza artificiale basato su rete neurale.

2.3 Intelligenza artificiale

Il processo di identificazione delle cuciture è fondamentale per garantire precisione da parte del robot durante la lavorazione: dato che le cuciture devono essere sfilate, è richiesto che il tool sia posizionato in corrispondenza del centro della cucitura.

Sono ammissibili piccole imprecisioni della posizione, ma sono comunque da limitare all'ordine del millimetro, e in generale si vogliono comunque minimizzare.

Dato che il modello di scarpa potrebbe variare in base al cliente che usufruisce del servizio, è necessario un software molto flessibile, in grado di adattarsi a diversi tipi di condizioni di lavoro. Si ricorda inoltre che il servizio è fornito per il recupero della scarpa; quindi, il robot andrebbe a operare su calzature usate, le cui superfici sono tipicamente deformate e consumate.

Questo conferma ulteriormente la necessità di un processo in grado di adattarsi a situazioni svariate. Per questo motivo è stato scelto un modello di intelligenza artificiale basato su deep learning e CNN: Ultralytics YOLO (You Only Look Once). [15]

Il modello è infatti stato creato per operare su immagini o video, per eseguire task come identificazione di oggetti o segmentazione dell'immagine.

L'implementazione dell'algoritmo è molto semplice ed eseguibile con poche righe di codice, grazie anche agli esempi forniti direttamente dagli autori [15].L'intero framework è inoltre open source, quindi gratuito e disponibile a eventuali modifiche.

Vengono pubblicate continuamente nuove versioni migliorate dei modelli YOLO: per l'esecuzione dell'addestramento nella presente applicazione è stato impiegato YOLOv9.

È possibile addestrare un modello YOLO in diversi modi: si può eseguire il training di un modello "da zero", ovvero non addestrato precedentemente; in alternativa, è possibile scegliere un modello già addestrato su dataset standard [16] ed eseguire un ulteriore training su dataset specifici per utilizzazioni particolari. Per l'applicazione in questione, si è scelto di utilizzare un modello da zero.

2.3.1 Preparazione dei dati

L'addestramento di un modello YOLO consiste nel fornire in input delle immagini in cui le feature sono già state identificate da persone o da altri modelli. In questo caso le soluzioni sono state trovate manualmente. Il successivo processo di training viene poi gestito automaticamente.

L'acquisizione, l'annotazione e l'organizzazione finale delle immagini sono i passaggi che costituiscono l'organizzazione dei dati di training, e sono gli elementi più importanti nel processo di addestramento, dato che un training basato su dati con errori, incompleti o imprecisi genera facilmente un modello con basse performance.

I dati da fornire sono costituiti da immagini annotate: acquisizioni in cui sono presenti gli elementi che il modello deve imparare a riconoscere, i quali sono poi identificati spazialmente attraverso un rettangolo che li racchiude, chiamato bounding box.

Il dataset è stato strutturato in due componenti principali:

- Set di immagini: un certo numero di immagini con relative bounding box, che verranno poi date in input al modello durante l'addestramento e confrontate con le predizioni effettuate;
- Set di etichette (labels): sono file di testo che contengono i dati sulle bounding box identificate; nel caso delle reti YOLO, le etichette hanno una struttura precisa: ogni immagine ha infatti un file di testo associato nel quale sono presenti tante righe quante sono le etichette; ogni riga ha una sequenza di numeri che indicano rispettivamente:
  - Tipo di classe;
  - Posizione in x del centro dell'etichetta;
  - Posizione in y del centro dell'etichetta;
  - Larghezza dell'etichetta;
  - Altezza dell'etichetta.

Le coordinate x, y e le dimensioni dell'etichetta sono comprese tra zero e uno, in quanto sono normalizzate rispetto alle dimensioni dell'immagine. [17]

class	x	y	w	h
0	0.170408	0.401220	0.128571	0.026829
0	0.343878	0.384146	0.202041	0.026829
0	0.550000	0.379268	0.206122	0.017073
0	0.696939	0.382927	0.251020	0.019512
0	0.910204	0.392683	0.146939	0.019512

Tabella 2.1: Tabella che rappresenta una possibile struttura di alcune etichette

Nella tabella 2.1 si riportano alcuni esempi di come potrebbe essere un file di etichette. Per completezza si riporta, in Figura 2.11, anche l'immagine a cui sono collegate.

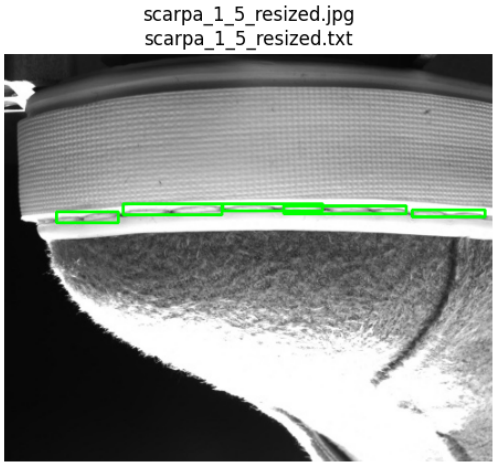

Figura 2.11: Immagine contenente le etichette riportate in tabella 2.1

È necessario, inoltre, che i file di testo delle etichette abbiano lo stesso nome dei file immagine, per permettere all'algoritmo di YOLO di riconoscere e associare i vari file.

2.3.2 Acquisizione delle immagini

È possibile trovare online dei dataset di immagini già annotate e pronte per essere date in input al modello da addestrare. La presente applicazione, tuttavia, è molto specifica e non sono stati trovati dataset pronti con delle immagini di scarpe in cui sono state annotate le cuciture sulla suola. Si procede quindi ad acquisire le immagini manualmente e poi annotarle.

Le immagini sono state acquisite cercando di imitare le condizioni finali in cui il modello dovrà essere applicato, ipotizzando una distanza tra oggetto e telecamera di circa 5-7 cm, eseguite con una camera industriale monocromatica. Per l'acquisizione è stata resa disponibile una camera basler da parte del laboratorio di misure ottiche dell'università. Sono state eseguite delle immagini su tre scarpe rese disponibili da parte dell'azienda, più una presente nel laboratorio. Le immagini sono state eseguite in corrispondenza della cucitura, a distanza simile a quella ipotizzata nell'applicazione finale, e ripetute per avere acquisizioni per tutto il perimetro della scarpa. (Figura 2.11)

Le scarpe Santoni erano state precedentemente usate per eseguire test di rimozione della cucitura; quindi, non tutte le immagini presentano cuciture accettabili per essere identificate. Le immagini senza cuciture verranno comunque usate per l'addestramento: queste sono associate a file di annotazione vuoti e servono per

facilitare il compito del modello nella distinzione tra oggetto da identificare e sfondo dell'immagine. Tali acquisizioni contengono infatti solo "sfondo". (Figura 2.14) In totale sono state acquisite 39 immagini su 4 scarpe diverse, delle quali non tutte contengono cuciture da identificare.

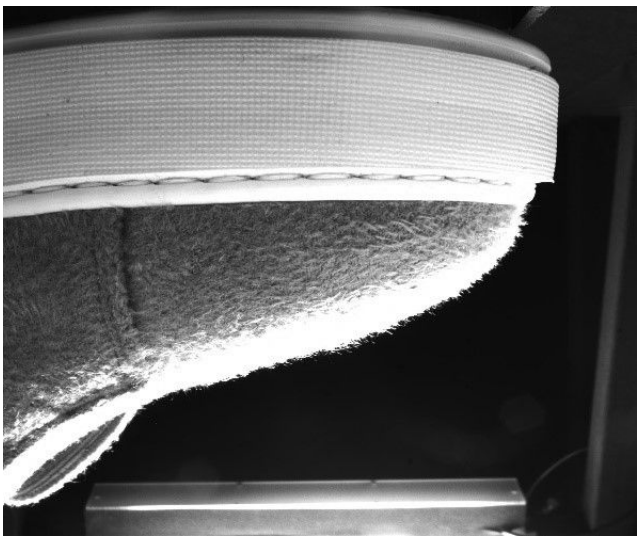

Figura 2.12: Esempio di immagine acquisita con camera industriale, scarpa Santoni

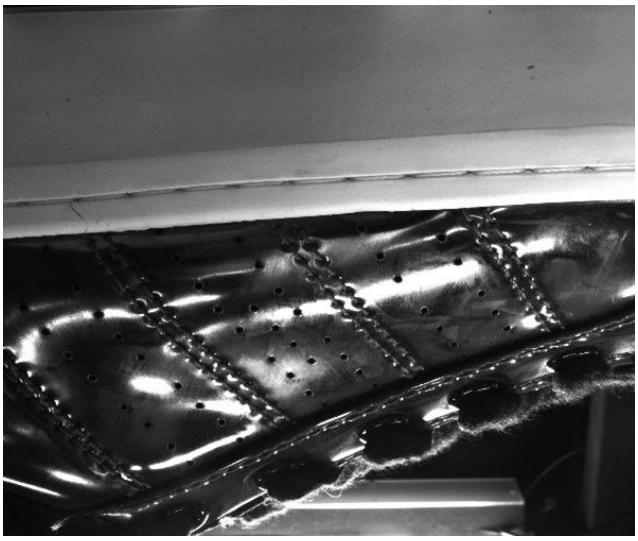

Figura 2.13: Esempio di immagine acquisita con camera industriale, scarpa alternativa

2.3.3 Data augmentation

Sebbene non esista un numero minimo di immagini da fornire all'addestramento, è preferibile avere la maggior quantità possibile di dati. Gli stessi modelli preaddestrati,

Figura 2.14: Esempio di immagine senza etichette

accennati precedentemente, sono preparati su dei dataset contenenti diverse migliaia di immagini.

Questo perché più immagini indicano più dati su cui il modello può essere addestrato, e quindi si avranno,in generale, risultati migliori. Esse, inoltre, devono essere variabili per quanto riguarda posizione delle feature e sfondo, in modo da rendere più flessibile il modello.

A causa del materiale di riferimento limitato, tuttavia, si è scelto di acquisire un numero inferiore di immagini, per poi eseguire una tecnica di data augmentation.

La data augmentation è un'operazione che si esegue sui dati di addestramento delle reti neurali per aumentare artificialmente la quantità di dati disponibili.

Nel caso di immagini, queste ultime vengono duplicate e trasformate attraverso varie operazioni, come la rotazione o l'ingrandimento, in modo da avere un numero maggiore di immagini disponibili.

Vi sono vari strumenti per effettuare la data augmentation, tra cui diverse librerie in python. In questo caso è stata scelta la libreria *Augmentor* [18], la quale accetta in ingresso semplicemente le cartelle di input e output dell'operazione, successivamente si descrivono le trasformazioni che si vogliono eseguire sulle singole immagini: dato che l'applicazione finale non prevede una forte variazione rotazionale degli oggetti delle immagini, si è scelto di limitare l'operazione di rotazione in favore di altre; in totale si hanno le seguenti trasformazioni date in input all'algoritmo:

- *Rotate*: effettuata rotazione casuale dell'immagine all'interno di un range definito;
- *Zoomrandom*: effettuato uno zoom casuale di una certa percentuale;
- *Flipleftright*: inserita la probabilità di poter "specchiare" l'immagine;

- *Shear*: effettuata la deformazione dell'immagine, attraverso allungamenti a destra e sinistra.

Il codice scritto per eseguire le trasformazioni è relativamente semplice, come riportato di seguito:

```
    import Augmentor

input_dir = r'image_path'
output_dir = r'image_path'

def augmentation(input_dir,output_dir,sample_num):
    p = Augmentor.Pipeline(source_directory=input_dir,
        output_directory=output_dir)

    #Transform
    p.rotate(probability=0.7, max_left_rotation=5, max_right_rotation=5)
    p.zoom_random(probability=0.5, percentage_area=0.8)
    p.flip_left_right(probability=0.5)
    p.shear(probability=0.5,max_shear_left=15,max_shear_right=15)

    # Data augmentation for all images
    p.sample(sample_num)

if __name__ == '__main__':
    augmentation(input_dir,output_dir)
```

A causa dell'automazione da parte dell'algoritmo nel rinominare le immagini aumentate, sono stati aggiunti dei semplici script per la rinomina automatica dell'immagine ottenuta e un suo ridimensionamento.

```
    import cv2
import matplotlib.pyplot as plt
import os

count = 3
in_dir = f'folder_path'
out_dir = f'folder_path'
def image_resizing(in_dir,out_dir,remove_original = True):
    x = 640
    y = 640
    i = 0
    for img in os.listdir(in_dir):
        path = f'{in_dir}\\{img}'
        image = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
    # change image dimensions
        resized_image = cv2.resize(image,(x,y),
            interpolation=cv2.INTER_AREA)
        name = f'resized_{i}.jpg'
```

```
        out_path = os.path.join(out_dir,name)
    # show new image
        cv2.imwrite(out_path,resized_image)
        if remove_original:
            os.remove(path)
        print(image.shape)
        print(resized_image.shape)
        i += 1

if __name__ == '__main__':
    image_resizing(in_dir,out_dir)
```

Sia nello script per il ridimensionamento che per la rinomina, il codice è disposto in modo da creare una funzione, così che possa essere utilizzabile nello script principale.

```
    def rename_image(in_dir,out_dir,ext:str,shoe_num):
    count = 0

    for filename in os.listdir(in_dir):
        new_name = f'scarpa_{shoe_num}_{count}_resized.{ext}'
        old = os.path.join(in_dir,filename)
        new = os.path.join(out_dir,new_name)
        os.rename(old,new)
        count += 1
```

L'operazione di ridimensionamento è stata effettuata poiché le dimensioni delle immagini influiscono sulla velocità di addestramento del modello: delle immagini di 5MP, come quelle ottenute dalla telecamera usata, richiederebbero svariate ore per essere processate. Perciò esse vengono anche ridimensionate. La dimensione finale è di 640x640, scelta perché i modelli in YOLO usati accettano di default immagini con queste dimensioni.

Il codice finale racchiude le funzioni per effettuare augmentation, ridimensionamento e rinomina automatici.

Le immagini originali sono suddivise in quattro cartelle in base alla scarpa alla quale sono riferite.

Dato che le immagini non contenenti cuciture sono presenti principalmente nelle foto delle scarpe 2 e 3, il processo di data augmentation è stato concentrato sulle scarpe 1 e 4, delle quali sono state riprodotte 100 immagini ciascuna. Per le restanti, sono state ricreate 50 immagini. In totale si hanno 300 immagini da fornire nel processo di addestramento.

2.3.4 Annotazione

Le immagini per l'addestramento devono essere quindi preparate eseguendo un'annotazione, ovvero ogni immagine va osservata e le feature in essa presenti vanno identificate: si deve quindi "disegnare" il rettangolo della bounding box per ogni feature.

Per questa operazione sono disponibili vari editor di immagini, i quali permettono di disegnare manualmente la bounding box per poi generare il file di testo contenente le etichette.

Per quanto riguarda il processo di annotazione delle immagini ottenute, è stato scelto l'uso di un programma apposito [19] che permette di usare i modelli SAM per *image segmentation* di Meta [20] per effettuare una annotazione "intelligente" delle immagini: lo script stampa a schermo l'immagine, sulla quale è possibile annotare rettangoli definendo i vertici; successivamente viene applicato il modello SAM (*segment anything*) di Meta per riconoscere gli oggetti all'interno dell'etichetta creata e aggiustarne le dimensioni in base al risultato ottenuto.

Tale processo permette di avere delle etichette più precise rispetto alla semplice annotazione manuale. Ciò viene eseguito per tutte le immagini. Per confrontare gli effetti delle scelte per l'annotazione, sono stati effettuati due processi di etichettatura: uno in cui le etichette includevano due punti di cucitura (Figura 2.15), un secondo con una etichetta per ogni singolo punto di cucitura (Figura 2.16).

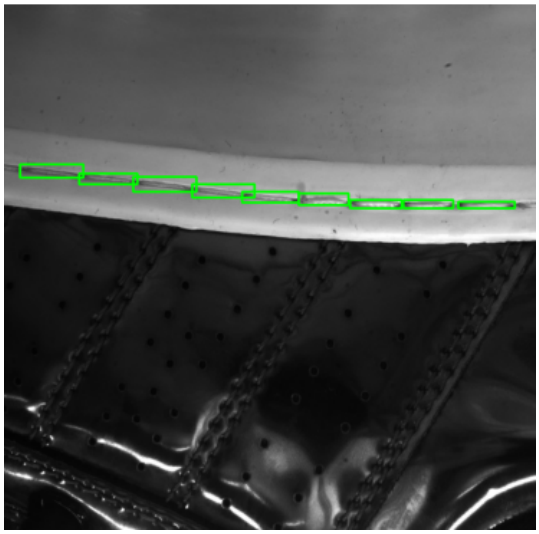

Figura 2.15: Immagine annotata selezionando una singola cucitura per ogni etichetta

Una volta terminato il processo di annotazione, si esegue la divisione delle immagini in immagini di training e di validazione. Il processo è automatizzato attraverso uno script apposito [19], producendo 276 immagini di training, lasciando le restanti per la validazione.

La disposizione delle cartelle è teoricamente ininfluente, tuttavia si preferisce solitamente definire una cartella data in cui sono presenti due sottocartelle: immagini e labels; in ognuna delle quali sono presenti a loro volta delle cartelle con i dati di training e validation.

L'input per il modello YOLO e il comando di training definisce anche un file di configurazione, in cui sono presenti il percorso principale della cartella data, i percorsi relativi delle immagini di training e validation, e una sezione in cui sono elencate le

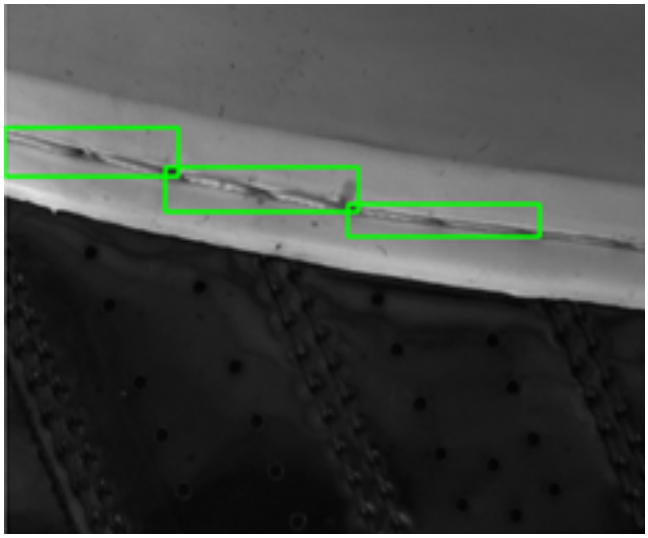

Figura 2.16: Immagine annotata selezionando due punti di cucitura per ogni etichetta

diverse classi. Dato che il modello finale dovrà identificare solo delle cuciture, viene definita solo la classe *"stitching"*, identificata alla posizione 0.

2.3.5 Organizzazione delle immagini

In seguito ad alcuni errori di identificazione, sono state rinominate le foto ottenute in modo da renderle sequenzialmente ordinate: è stato infatti notato che l'algoritmo tende a considerare anche la posizione dei vari file nella cartella, oltre che il loro nome; questo ha portato all'associazione di alcuni file di label con delle immagini non corrispondenti, risultando quindi in etichette posizionate nella zona sbagliata dell'immagine. Tale comportamento è evidente nell'esempio riportato in Figura 2.17.

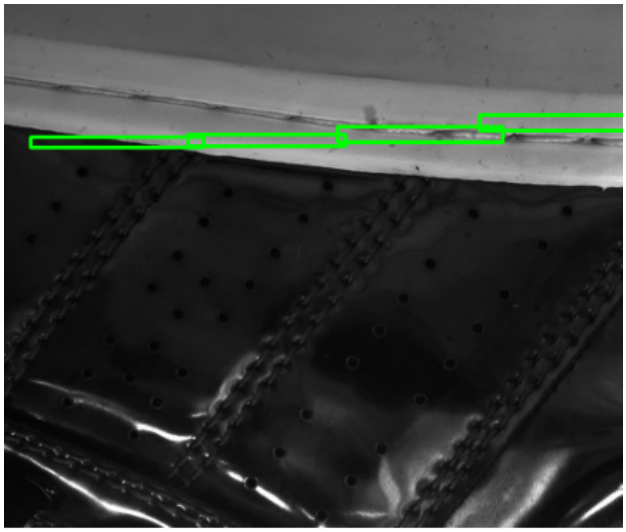

Figura 2.17: Immagine con errore di annotazione

2.3.6 Nota su hardware usato

Le capacità di calcolo richieste per poter eseguire SAM all'interno del codice per l'annotazione intelligente sono molto elevate: per questo motivo si è scelta l'applicazione di metodi per sfruttare le capacità di calcolo da parte della GPU.

Le unità grafiche sono infatti create per eseguire calcolo parallelo: avendo dei dati in ingresso, si richiede la loro elaborazione eseguendo operazioni relativamente semplici ma che devono essere ripetute un elevato numero di volte.

Ciò è particolarmente evidente nell'elaborazione di immagini e nella creazione di elementi grafici: il processo di elaborazione va eseguito per ogni pixel nell'immagine e l'aumento di pixel fa aumentare notevolmente la quantità di operazioni da svolgere. Per questo motivo l'elaborazione grafica nei computer viene affidata a schede dedicate: le unità di processamento grafico o GPU. Sebbene il loro uso comune si limiti alla rappresentazione grafica, è possibile sfruttare le loro capacità di calcolo parallelo per altre applicazioni usando appositi strumenti.

Per velocizzare l'operazione di annotazione con l'ausilio di SAM è stato scelto CUDA Toolkit, il quale permette di utilizzare l'architettura CUDA, sviluppata da NVIDIA [21] per l'esecuzione di programmi che richiedono calcolo da parte del processore grafico. La scheda grafica usata è una NVIDIA GeForce MX130.

2.3.7 Nota su Google Colab

Il processo di addestramento è molto lungo e richiede molte risorse hardware per poter essere svolto velocemente. Sebbene inizialmente siano state eseguite prove su hardware locale, i lunghi tempi impiegati hanno imposto la scelta di una elaborazione in cloud: a tale scopo è stato considerato Google Colab. [22]

Colab è uno strumento di Google che permette di usare da remoto dei server con elevate capacità di calcolo e processori appositamente creati per l'elaborazione di grandi quantità di dati e l'addestramento di algoritmi di intelligenza artificiale.

Il servizio prevede offerte gratuite con messa a disposizione, per un tempo limitato, di processori grafici o TPU [23], elementi hardware ottimizzati per lavorare con tensori, particolarmente indicati per l'addestramento di IA.

2.3.8 Training

Come detto in precedenza, YOLO è stato ottimizzato per poter eseguire il comando di addestramento usando poche righe di codice. Tuttavia, è necessario scaricare un modello [24] da cui far partire l'addestramento. Essendo "da zero", il modello scaricato contiene solo la struttura di base dell'algoritmo IA, e i pesi nelle funzioni di attivazione dei vari neuroni sono casuali.

L'addestramento consiste infatti nel far eseguire l'algoritmo sulle immagini fornite, il quale produrrà una predizione della posizione degli oggetti da identificare, dando in uscita delle etichette. Un successivo confronto con le etichette originali fornirà

indicazioni sull'errore commesso, la precisione ed altri parametri di valutazione poi organizzati nei risultati finali.

Il processo appena descritto viene eseguito non su singole immagini ma su batch di immagini: insieme di immagini accostate tra loro a formarne una unica, la quale viene poi fornita in ingresso al modello.

L'intero processo di predizione e verifica viene poi ripetuto per un certo numero di volte: ciò è descritto nel parametro delle epoche.

L'input per il modello YOLO e il comando di training richiede un file di configurazione, in cui sono presenti:

- il percorso principale della cartella data;
- i percorsi relativi delle immagini di addestramento e validazione;
- una sezione in cui sono elencate le diverse classi.

Per completezza si riporta un esempio di struttura di file di configurazione:

```
   path: 'path' # dataset root dir
train: images/train # train images (relative to 'path')
val: images/val # val images (relative to 'path')

# Classes
names:
  0: stitching # number of classes
```

Dato che il modello finale dovrà identificare solo delle cuciture, viene definita solo la classe "stitching", identificata come classe 0. Sono stati inseriti anche due ulteriori parametri per l'addestramento:

- single-cls: parametro di gestione dell'addestramento che riduce le operazioni da eseguire per l'identificazione delle classi, specificando che per questo training è necessario identificare solo una classe di oggetti;
- patience: parametro che definisce la "pazienza" dell'addestramento; a ogni ciclo di addestramento vengono misurati dei parametri di valutazione, aspettandosi che al ciclo successivo queste valutazioni siano migliorate.

  Se questo non avviene, si ipotizza che il modello sia pronto e l'addestramento viene interrotto prematuramente. In questo caso sono state impostate 25 epoche in cui si accetta un mancato miglioramento dei risultati.

Questi ultimi sono stati inseriti in seguito a una prima prova per verificare il funzionamento dell'addestramento. I parametri considerati vengono inseriti nel comando *train*, ovvero il comando che avvia e poi gestisce l'addestramento:

```
model = YOLO('yolov9c.yaml')
ROOT_DIR = 'dir_path'
```

```
results = model.train(data=os.path.join(ROOT_DIR,
    "stitching_config.yaml"),
                epochs=200,
                batch = 16,
                single_cls = True,
                patience = 25)
```

2.3.9 Risultati

L'addestramento produce in output dei file che contengono il modello da inserire poi nel processo di predizione; questo è costituito da quelli ricavati durante l'addestramento. Vengono forniti sia gli ultimi pesi ricavati che quelli considerati "migliori", ovvero quei pesi che durante l'addestramento hanno determinato minori errori e prestazioni maggiori.

Oltre questi risultati, vengono generati anche dei file, con relative rappresentazioni grafiche, contenenti l'andamento di diversi parametri, i quali permettono una valutazione del processo di training e del modello.

In particolare, tra i risultati sono presenti:

- Un file contenente tutti i parametri considerati per l'addestramento;
- Delle matrici di confusione;
- Un grafico della curva F1;
- Dei file contenenti informazioni sulle etichette dei dati di addestramento;
- Una rappresentazione della curva di Precisione;
- Un grafico della curva di Richiamo;
- Un grafico della curva Precisione-Richiamo;
- Una tabella, con associati relativi grafici, degli andamenti delle varie metriche: precisione, richiamo, funzione costo per classe, bounding box, precisione media al 50% della IOU e al 95% della IOU;
- Alcuni esempi dei batch utilizzati durante il training, oltre che un batch di validazione, in modo da vedere graficamente eventuali problemi con le etichette, le immagini o il modello.

Si riportano i risultati ottenuti dall'addestramento eseguito con immagini annotate sulle singole cuciture.

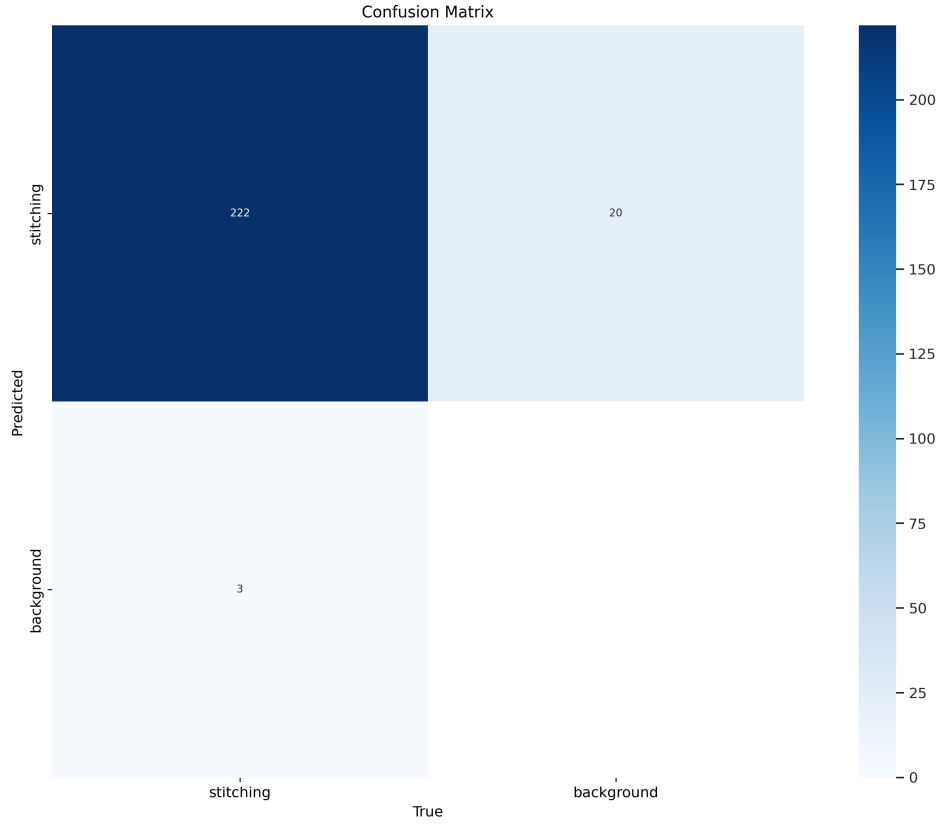

Figura 2.18: Matrice di confusione semplice

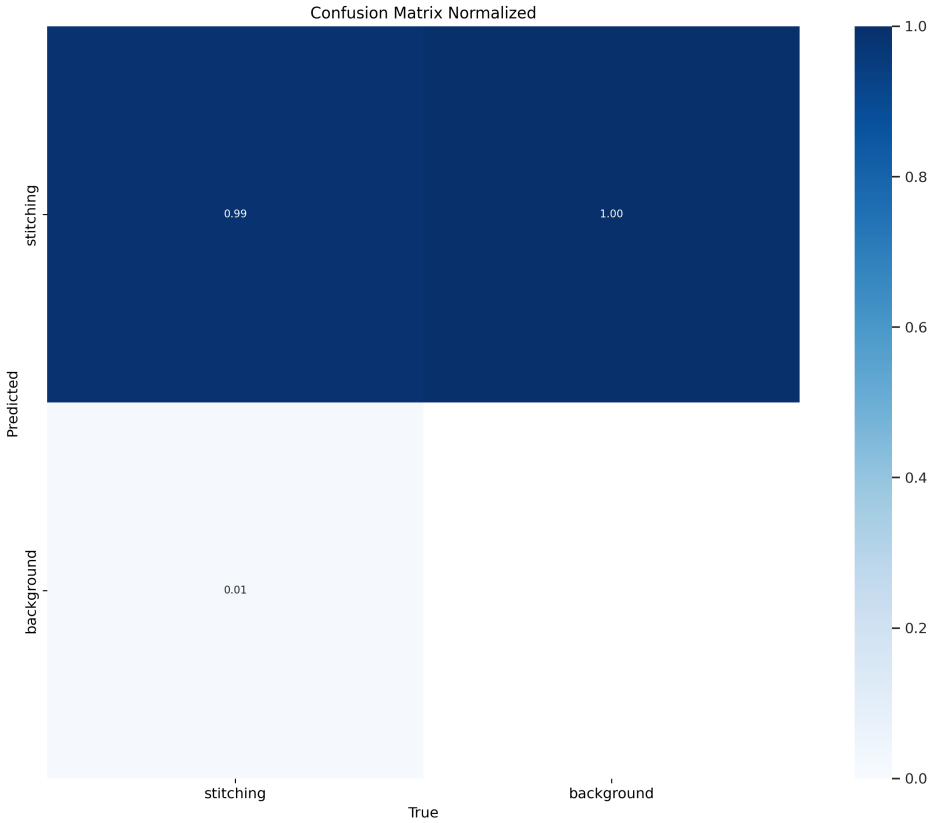

Figura 2.19: Matrice di confusione con valori normalizzati

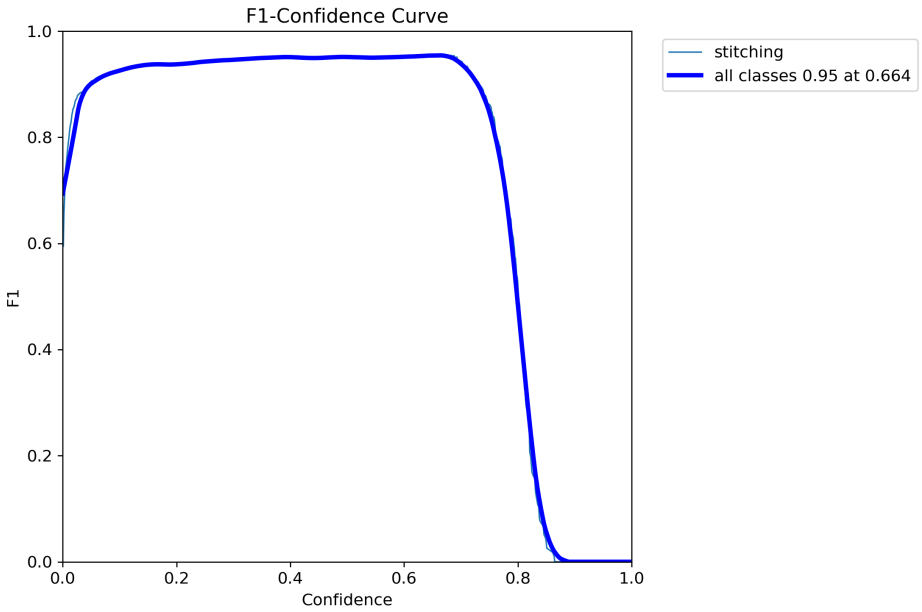

Figura 2.20: Rappresentazione grafica dell'andamento della funzione F1

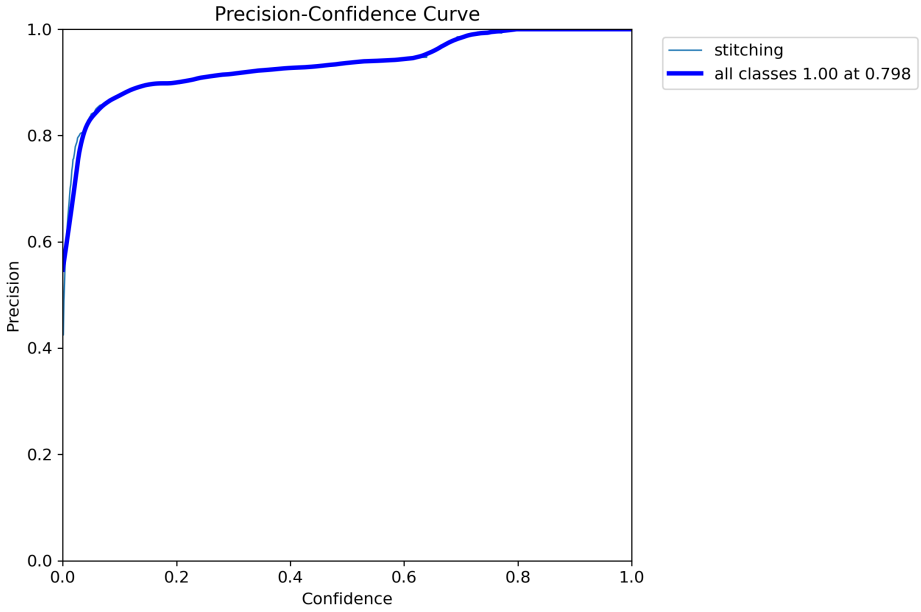

Figura 2.21: Rappresentazione grafica dell'andamento della funzione di precisione (*precision*)

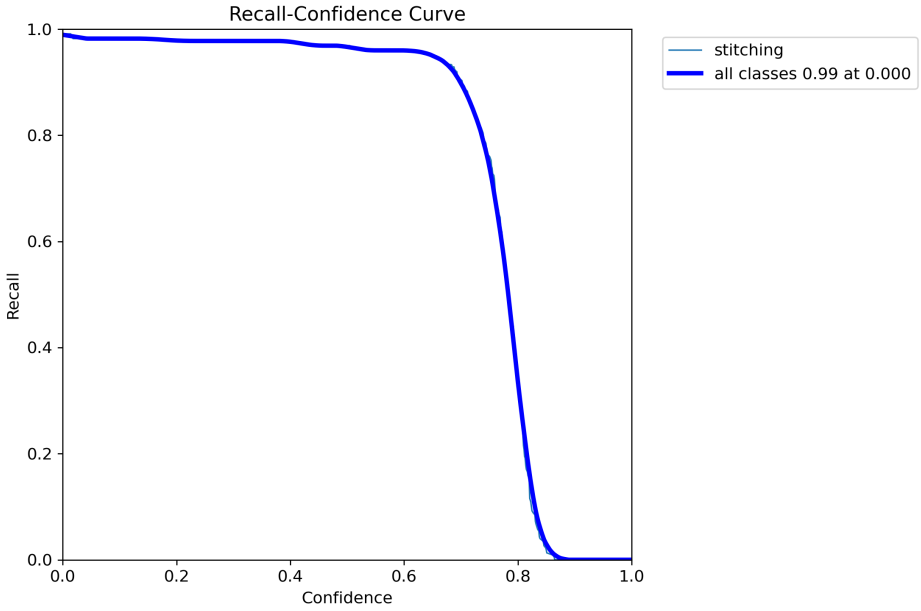

Figura 2.22: Rappresentazione grafica dell'andamento della funzione di richiamo (*recall*)

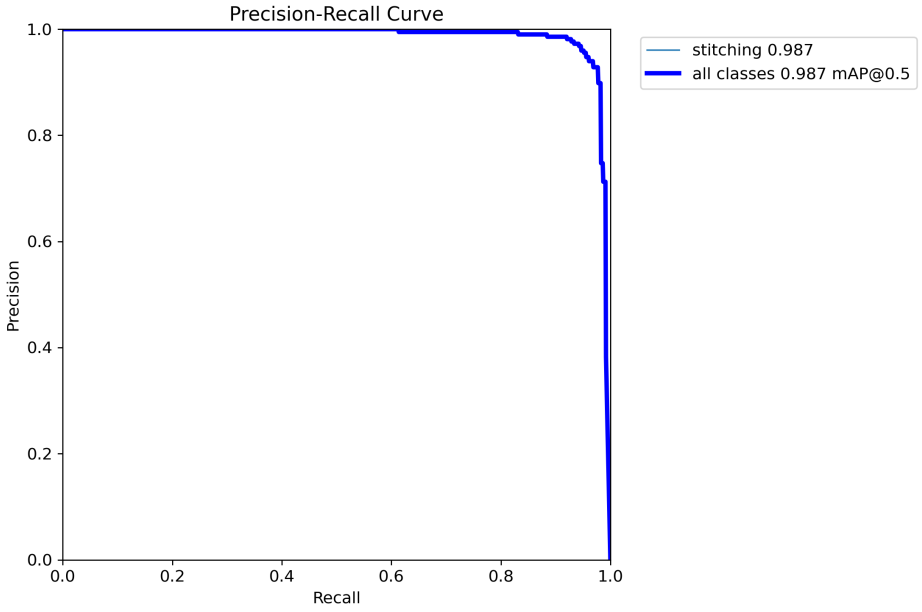

Figura 2.23: Rappresentazione grafica dell'andamento relativo tra precisione e richiamo

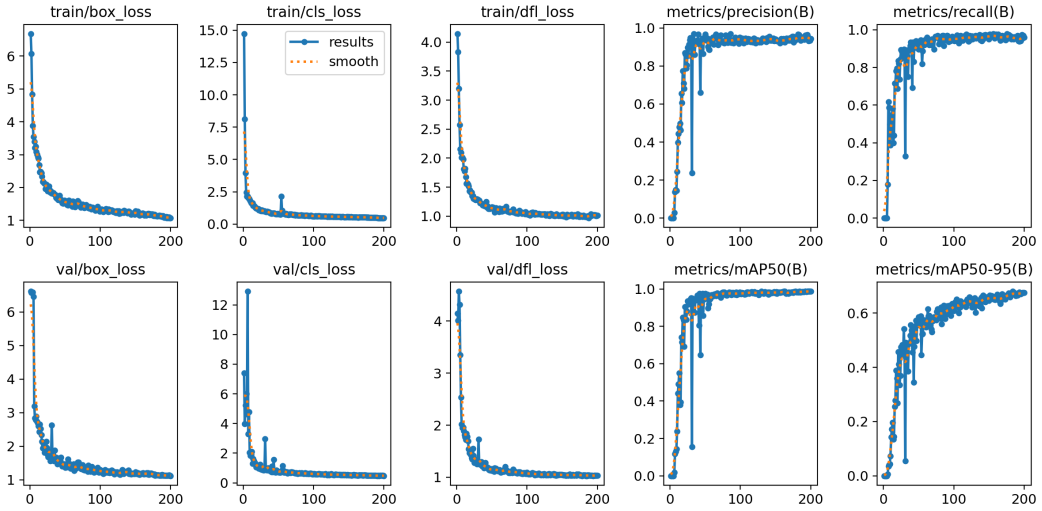

Figura 2.24: Grafici per l'andamento delle funzioni di costo, precisione, richiamo e la loro combinazione (mAP)

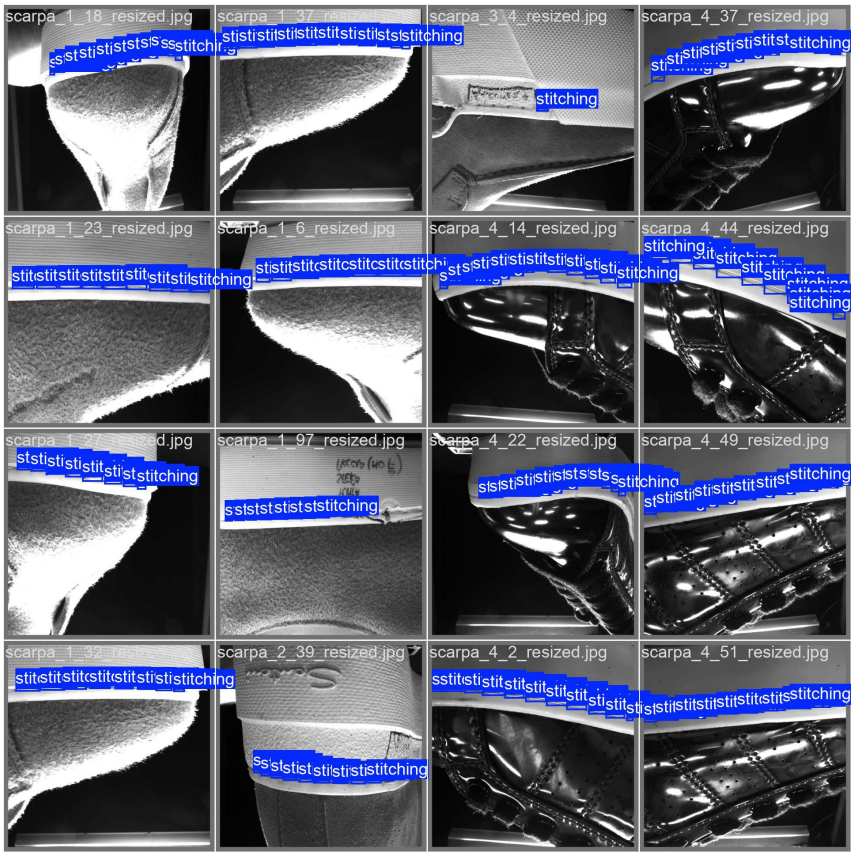

Figura 2.25: Esempio di uno dei batch di validazione usato nella rispettiva fase durante l'addestramento

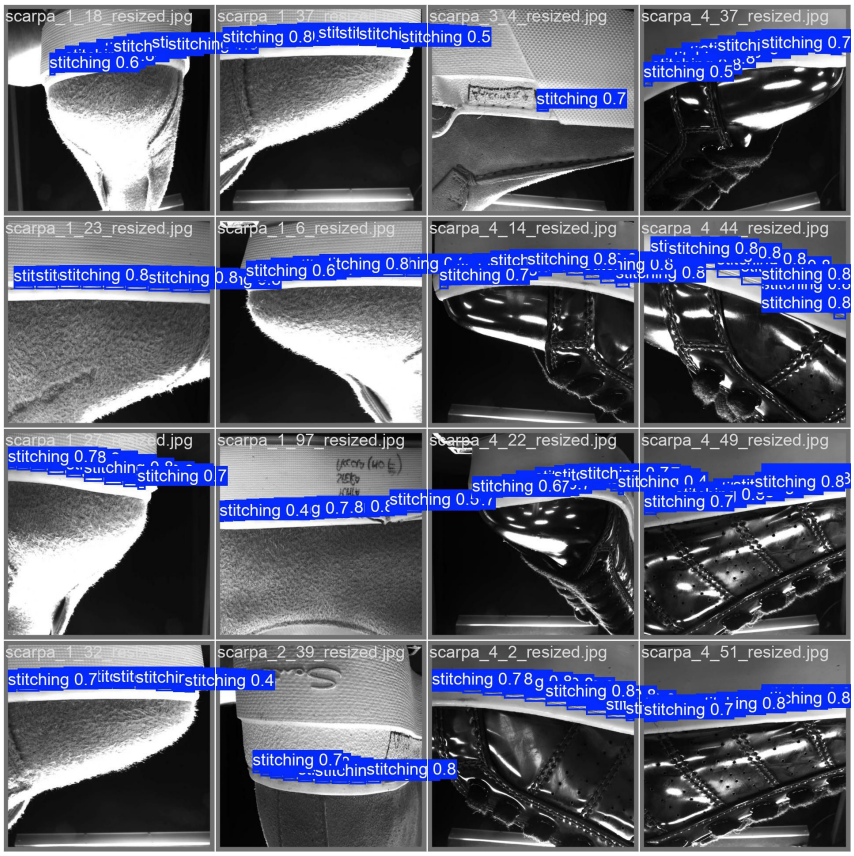

Figura 2.26: Risultati della previsione effettuata sul batch di immagini in Figura 2.25

2.3.10 Studio risultati forniti dopo l'addestramento di un modello YOLO-metriche di valutazione

I risultati forniti dall'algoritmo di addestramento sono basati su determinate valutazioni metriche.

Esistono molti tipi di queste metriche; i modelli YOLO sono verificati, in particolare, sulla base di tre metriche principali:

- I.O.U. (*intersection over union*);
- Precisione (*Precision*)
- Richiamo (*Recall*)

Dalle quali si possono successivamente ricavare ulteriori parametri di valutazione, quali:

- F1 *score*;
- *Average precision*;
- *Mean average precision*;
- *Confidence*

La maggior parte delle metriche sono basate sui risultati ottenuti; in particolare si considerano quattro tipi di risultati:

- Veri positivi (VP), oggetti presenti nell'immagine che sono stati correttamente identificati dal modello;
- Veri negativi (VN), oggetti non presenti nell'immagine che non sono stati identificati dal modello e sono stati ignorati;
- Falsi positivi (FP), occorrenze non presenti nell'immagine che sono state identificate erroneamente dal modello;
- Falsi negativi (FN), oggetti presenti nell'immagine che il modello non è riuscito a identificare.

Questi rappresentano i confronti tra i dati forniti in ingresso al processo di addestramento e i risultati forniti dal modello in un certo momento del training. La loro definizione è stata declinata nel processo di identificazione di oggetti nell'immagine, ma in generale sono usati in tutti i processi di addestramento e validazione di modelli IA.

In generale, le metriche di valutazione permettono di ricavare prestazioni del modello ottenuto, eventuali errori durante l'addestramento o nei dati in ingresso:

- IOU: parametro che indica quanto il modello riesce a localizzare le features;

- Precisione: usata per valutare se il modello restituisce troppe identificazioni false;

- Richiamo: necessario per valutare se il modello è in grado di identificare tutte le istanze presenti in un'immagine:

- F1: utile per valutare il bilanciamento tra precisione e richiamo;

- Funzioni di costo: utile per diagnosticare problemi di *overfitting* e *underfitting*. In breve, l'overfitting è definibile come addestramento eccessivo sui dati di training, in quanto il modello fornisce risultati eccellenti se applicato sugli stessi dati di addestramento, ma non è in grado di effettuare predizioni su dati nuovi. Il problema di underfitting, invece, accade quando il modello non riesce a restituire risultati sufficienti neanche nei dati di training. [25]

Sulla base dei risultati, oltre che alle informazioni date dalle bounding box fornite in addestramento e quelle predette dal modello, vengono definite le varie metriche di valutazione [26]:

- Intersection over union (IOU): è un metodo per valutare se la bounding box predetta è simile alla bounding box effettiva. In caso di sovrapposizione esatta, il valore di IOU è uguale a 1. Il valore della IOU è ottenuto dai valori dei punti in alto a sinistra e in basso a destra delle bounding box vera e predetta, ovvero i lati dei rettangoli. Per l'intersezione, si effettua la ricerca del massimo tra i punti in alto a sinistra e del minimo in basso a destra, per poi definire le aree dei due rettangoli. L'unione si ottiene a partire dalla somma delle aree meno la loro intersezione. L'equazione finale può essere rappresentata come rapporto tra intersezione e unione.

$$\text{IOU} = \frac{\text{intersezione}}{\text{unione}} \tag{2.4}$$

  In Figura 2.27 è mostrata anche una sua rappresentazione grafica.

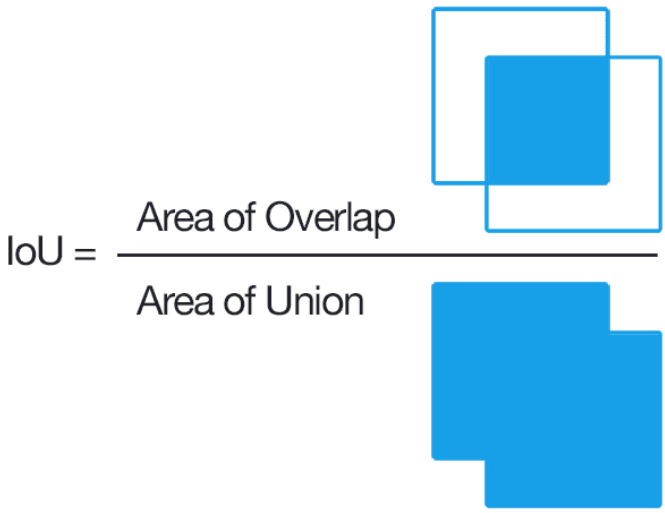

Figura 2.27: Rappresentazione grafica dell'equazione per ricavare la IOU

- Precisione: livello di precisione nell'identificare elementi corretti, ovvero percentuale di rilevamenti corretti rispetto ai totali effettuati dal modello. Ovviamente si vuole che questa valutazione aumenti nel corso dell'addestramento.

$$\text{Precision} = \frac{\text{veri negativi classificati erroneamente}}{\text{veri negativi totali}} = \frac{FP}{FP + VN} \tag{2.5}$$

  Il grafico ottenuto per il test condotto è riportato in Figura 2.21

- Richiamo: percentuale di rilevamenti predetti rispetto al numero di rilevamenti reali. Questo indica che un modello con richiamo maggiore ha più possibilità di riconoscere degli oggetti rispetto allo sfondo dell'immagine. Anche in questo caso, si desidera che la valutazione aumenti nel corso del training.

$$\text{Recall} = \frac{\text{veri positivi classificati correttamente}}{\text{veri positivi totali}} = \frac{VP}{VP + FN} \tag{2.6}$$

  Il grafico ottenuto per il test condotto è riportato in Figura 2.22

- AP (*average precision*) calcola area della curva precisione-richiamo, sintetizza gli andamenti dei due parametri in un unico valore. Rappresenta in una unica metrica di valutazione sia l'andamento della precisione che del richiamo. Una AP crescente indica che sia precisione che richiamo stanno aumentando all'avanzare delle epoche.

$$\sum_{k=0}^{n-1} [(\text{ecalls}(k) - \text{recalls}(k+1)) \cdot \text{precision}(k)] \tag{2.7}$$

- mAP (*mean average precision*): calcola la AP per classi con oggetti multipli e ne ricava una media. Questa metrica è particolarmente utile quando sono

presenti più classi, ma in questa applicazione non verrà considerata, in quanto presente una sola classe di oggetti (Figura 2.24).

- F1: descritta come media armonica di precisione e richiamo, viene definita per un certo valore di confidenza, ovvero una stima del modello sulla bontà della predizione fatta su una certa immagine. Nel caso dei modelli YOLO, si usa la probabilità che un oggetto identificato appartenga a una classe piuttosto che a un'altra. Per questa metrica si vuole che i valori ottenuti restino alti lungo tutto il range di confidenza, tra 0 e 1. Nel caso del test condotto, l'andamento è riportato in Figura 2.20.

  Ciò indicherebbe che la media di precisione e richiamo resti elevata sia quando il modello non è sicuro che la predizione sia giusta, sia quando ha una forte confidenza. In caso non si riesca a mantenere tale andamento, si cerca comunque di tenerlo più alto nella zona con alta confidenza, dato che poi questi valori sono quelli che vengono valutati principalmente.

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.8}$$

- Confidenza (*Confidence*): non è una effettiva metrica di valutazione, ma rappresenta la sicurezza del modello nel considerare una previsione come esatta, ovvero la probabilità che il modello abbia identificato correttamente una feature.

$$\text{Confidence} = P(\text{Object}) \times \text{IoU}(b, b_{\text{gt}}) \tag{2.9}$$

- *Loss Function*: indica il risultato dell'esecuzione di una funzione di costo durante l'apprendimento del modello.

  In ogni ciclo di addestramento, il modello esegue la convoluzione e usa i pesi della rete neurale per ottenere i valori di confidenza e la posizione delle bounding box. Questi valori sono inseriti nella funzione di costo per valutare l'errore commesso dal modello. Un apposito algoritmo, detto di retro-propagazione, userà il gradiente dell'errore ottenuto nel corso del training per modificare i pesi della rete, per poi cominciare un nuovo ciclo. YOLO fornisce nei risultati gli andamenti di tre funzioni di costo:

  - *Box loss*: funzione che confronta la bounding box predetta dal modello con quella effettiva nell'immagine, attraverso la funzione seguente:

$$\text{LCIoU} = 1 - \text{IoU} \tag{2.10}$$

  - *Cls loss* (*classification*): funzione di costo associata alla classificazione, ovvero la capacità del modello di collegare una feature identificata nella

giusta classe di appartenenza. La funzione usata è relativa alla cross-entropia binaria, dato che in questo caso si opera solo su una singola classe:

$$L_{\text{cls}} = -\frac{1}{n}\sum [y_i \log(p_i) + (1 - y_i)\log(1 - p_i)] \tag{2.11}$$

Dove y_i è l'etichetta reale, mentre p_i è la probabilità della predizione.

– *Dfl* (*distribution focal loss*): funzione di costo per migliorare l'accuratezza nell'identificazione; si concentra su "esempi complessi da riconoscere", ovvero modifica i pesi nel modello in base ai risultati ottenuti nel riconoscimento delle classi: riduce pesi in base alle classi che sono facili da riconoscere e li aumenta per le classi più difficili. La funzione in sé è simile alla classification loss, ma poi segue un diverso algoritmo per aggiustare i parametri del modello.

Oltre alla rappresentazione grafica delle metriche descritte, viene fornita anche una curva dell'andamento della precisione rispetto al richiamo. Anche in questo caso, valori elevati indicano prestazioni migliori del modello.

Vengono riportate inoltre delle matrici di confusione: strutture contenenti tutti i risultati dell'addestramento, intesi come veri positivi o negativi e falsi positivi o negativi. La sua struttura è rappresentata in tabella 2.2.

La matrice organizza i risultati in base alla classe di oggetti considerata, elencando le etichette effettive lungo le righe e quelle predette lungo le colonne: in questo modo, i valori lungo la diagonale della matrice indicano i risultati positivi dell'identificazione, poiché l'etichetta prevista coincide con quella reale, mentre al di fuori della diagonale si trovano i vari errori commessi dal modello, in cui sono rappresentati eventuali confusioni tra classi.

Va inoltre considerato che tra le classi valutate è presente lo sfondo delle immagini, dato che il modello potrebbe anche prevedere la presenza di una feature che in realtà non esiste o, viceversa, non identificare affatto la presenza di un oggetto che in realtà è presente.

Dato che il modello è addestrato per riconoscere una sola classe, la matrice ha dimensioni 2x2, ovvero la classe cucitura (*stitching*) e lo sfondo. Viene anche fornita una versione della stessa matrice ma con valori normalizzati tra 0 e 1. (Figura 2.19)

	positivo (reale)	negativo (reale)
positivo (predetto)	TP	FP
negativo (predetto)	FN	TN

Tabella 2.2: Tabella di rappresentazione della struttura di una matrice di confusione, caso con due classi di oggetti

Infine, tra i risultati vengono forniti anche dei grafici che rappresentano comportamenti e proprietà delle etichette utilizzate nel processo di training. Anche in questo caso i valori sono normalizzati per semplicità.

Il primo insieme di schemi rappresenta le caratteristiche principali delle etichette durante il processo:

- Il primo grafico rappresenta un istogramma del numero di istanze per ogni classe. Naturalmente in questo caso la rappresentazione collassa: l'istogramma è quindi una singola colonna che include tutte le etichette, dato che è definita solo una classe;
- Il secondo grafico riporta le varie dimensioni delle etichette fornite in ingresso. Non dà informazioni sui risultati ottenuti, ma può essere utile per valutare eventuali imprecisioni o andamenti anomali nei dati in ingresso;
- Il terzo grafico è un istogramma bidimensionale dove si riscontrano gli andamenti delle labels, in particolare delle posizioni x e y;
- Il quarto grafico è simile al precedente, riportando in questo caso i valori di altezza e larghezza delle etichette.

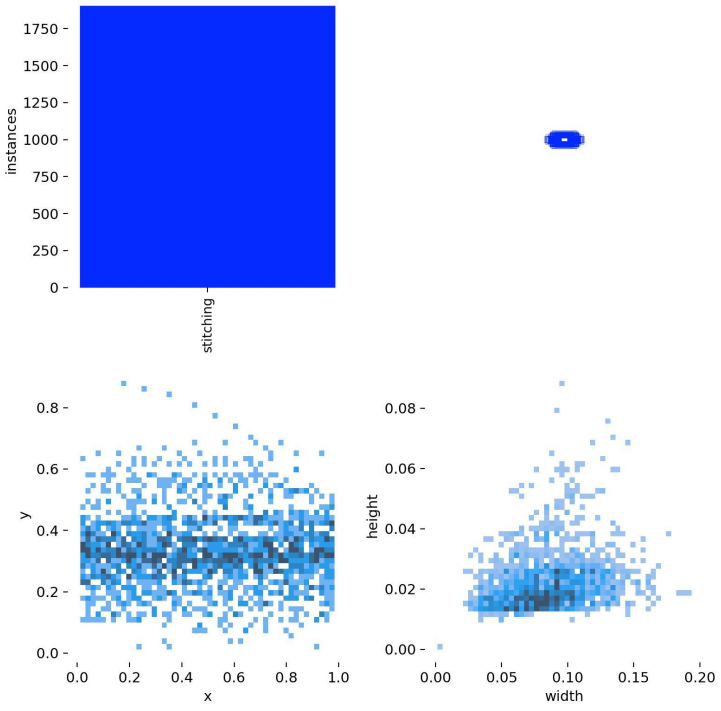

Figura 2.28: Grafici con comportamenti e caratteristiche delle labels utilizzate

Il secondo insieme riporta dei correlogrammi delle etichette, ossia degli istogrammi che riportano i diversi comportamenti di tutti i valori delle labels in confronto tra loro. Si ricorda che ogni etichetta è rappresentata con i valori di posizione (x,y) e di dimensioni (w,h). Anche in questo caso, i vari istogrammi danno informazioni sui dati in ingresso al modello, per verificare che siano corretti per quanto riguarda le etichette.

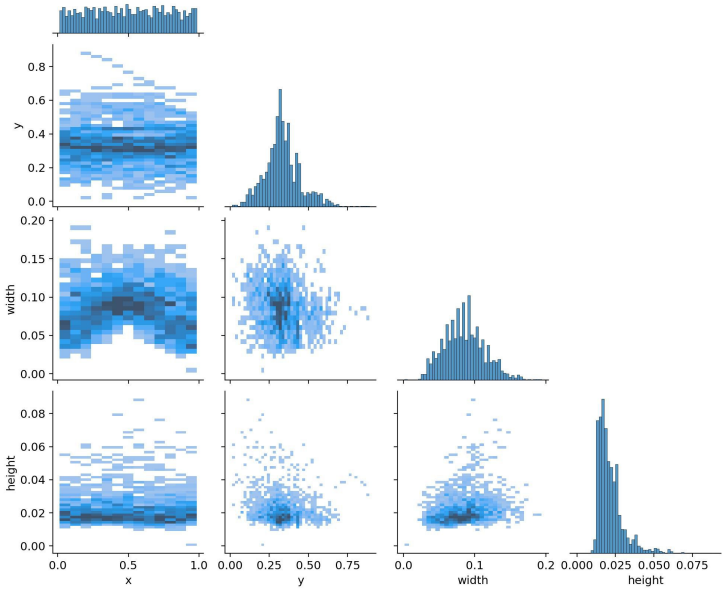

Figura 2.29: Grafici con istogrammi di correlazione delle labels utilizzate

2.3.11 Studio degli iperparametri per l'ottimizzazione

Nell'ambito dell'addestramento di una rete neurale, gli iperparametri sono dei settaggi di alto livello per l'algoritmo di IA. Essi sono impostati prima del processo di training e non cambiano durante il processo.

Esistono molti iperparametri definibili per un algoritmo, tuttavia è stata portata più attenzione ai parametri di default utilizzati nei modelli YOLO [27]:

- epoche: inteso come numero di cicli per cui ripetere il processo di addestramento. Troppe epoche portano a overfitting, poche epoche non permettono un training adeguato. Il numero di epoche è regolato in base alla patience: se per un certo numero di epoche i valori delle metriche di validazione restano invariati, il training si arresta prima del dovuto (*early stopping*);

- *Learning rate*: determina lo step per cui i parametri vengono aggiornati, ovvero quanto velocemente i pesi del modello vengono riscritti durante il training. Il learning rate influenza la velocità della convergenza della rete a una soluzione e quanto riesce a generalizzare. Con learning rate più basso la rete migliora la convergenza e la generalizzazione. Learning rate alto permette di raggiungere soluzioni più velocemente ma rischia di rendere la rete troppo instabile e non portare a convergenza. Il learning rate di partenza è definito *lr0*, e si riduce durante l'addestramento con andamento tipicamente lineare fino a una sua frazione determinata da *lrf*;

- momento: aiuta l'ottimizzatore a muoversi lungo l'andamento della funzione di costo, per evitare che un minimo locale della funzione venga considerato il minimo assoluto. Permette anche di limitare il rumore nelle stime della rete, migliorando la generalizzazione;

- *weight decay*: tecnica di regolarizzazione usata per prevenire overfitting, aggiungendo un termine di penalità alla funzione di costo. Esso consiste in un parametro modificabile moltiplicato per la somma dei quadrati dei pesi; quindi, il termine è proporzionale al quadrato dei pesi. Esso viene aggiunto alla funzione di costo per ridurre la probabilità che i pesi assegnati diventino troppo elevati, creando overfitting. Il parametro di weight decay controlla il termine per determinare quanto i pesi si riducono;

- *class probability weight*, *object probability weight*: funzioni di costo basate sulla cross-entropy loss, usata per distinguere tra oggetto e non-oggetto. La perdita è calcolata in base alla probabilità di ogni pixel di appartenere o meno a un oggetto rispetto alle etichette 'vere'. I valori tipici sono tra 1 e 10 ma possono essere anche al di fuori, in base al problema. Nelle recenti versioni di YOLO sono rappresentate dagli iperparametri box e cls; in cui il primo favorisce la determinazione degli oggetti, il secondo l'accuratezza delle bounding box, ovvero quanto è probabile che l'oggetto interno al rettangolo sia appartenente a una certa classe;

- *warmup epochs*: epoche in cui il learning rate viene aumentato gradualmente fino a lr0, per avere una stabilizzazione iniziale del processo di training;

- *warmup momentum*: momento iniziale che viene modificato fino al valore base, similmente alle warmup epochs;

- IOU *threshold* (*IOU-t* oppure IOU): valore minimo per cui si accetta la IOU, valore che indica quanto la bounding box predetta deve essere sovrapposta a quella effettiva per essere considerata valida. Valori elevati di *IOU-t* portano a meno rilevazioni, ma più accurate, mentre si ha l'effetto opposto riducendo il parametro;

- *anchor t*: threshold per le ancore (bounding box predefinite di diverse dimensioni e proporzioni); durante il training il modello riconosce le dimensioni tipiche degli oggetti per migliorare la loro corrispondenza nell'immagine, la threshold evita che queste modifiche siano troppo elevate, portando a non riconoscere oggetti o a riconoscerne troppi. Il parametro controlla quanto queste ancore possono essere modificate solitamente per quanto riguarda le loro dimensioni (es. se *anchor* $t = 2$, l'ancora modificata può essere al massimo di dimensioni doppie rispetto l'originale).

La variazione dei parametri può essere impostata specificando il parametro nel comando di addestramento, seguito dal valore voluto.

Dopo diverse prove, sono stati impostati gli iperparametri elencati in 2.3.

iperparametro	valore assegnato
epochs	200
IOU	0.7
lr0	0.01
lrf	0.01
momentum	0.937
warmup epochs	3
warmup momentum	0.8
box	7.5
cls	0.5

Tabella 2.3: Tabella con i valori impostati per gli iperparametri più comuni

Si riportano anche i risultati ottenuti considerando un labelling con inclusi due punti di cucitura. In questo caso è possibile notare immediatamente le prestazioni inferiori: maggior numero di falsi negativi nella matrice di confusione(Figura 2.30), valori inferiori nelle varie metriche(Figura 2.31,Figura 2.32,Figura 2.33), e andamenti più irregolari durante le varie epoche (Figura 2.34).

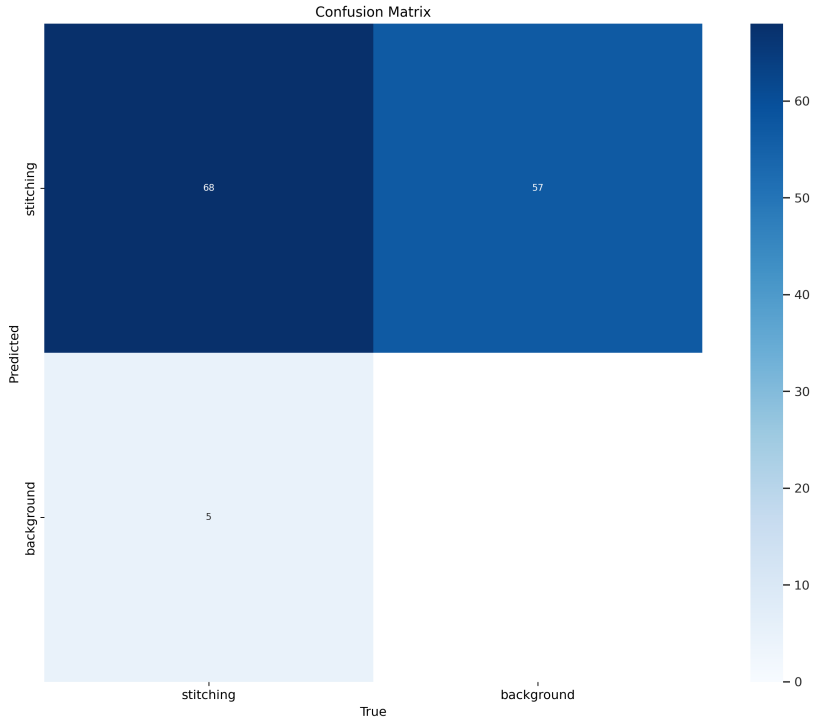

Figura 2.30: Matrice di confusione semplice, caso con etichette contenenti una doppia cucitura

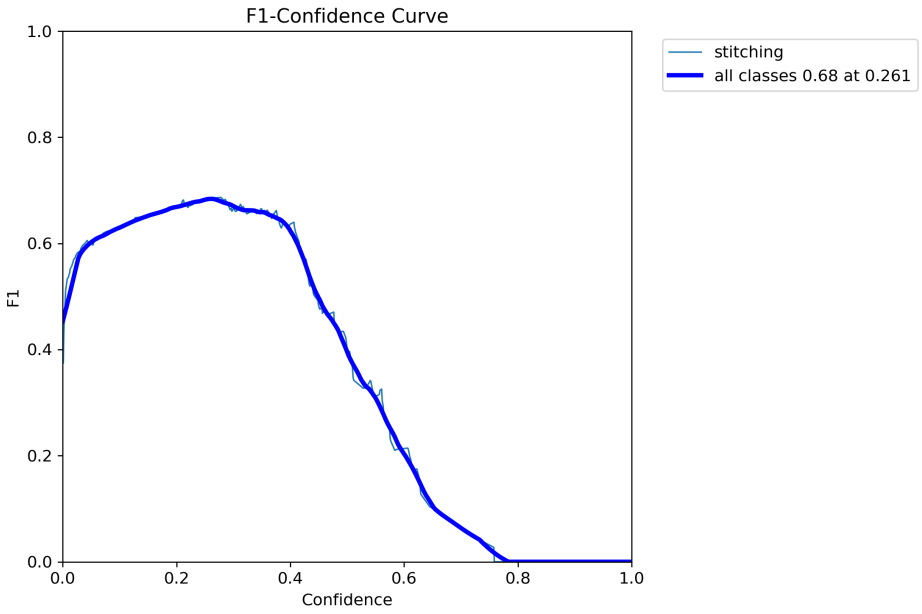

Figura 2.31: Rappresentazione grafica dell'andamento della funzione F1, caso con etichette che includono due cuciture

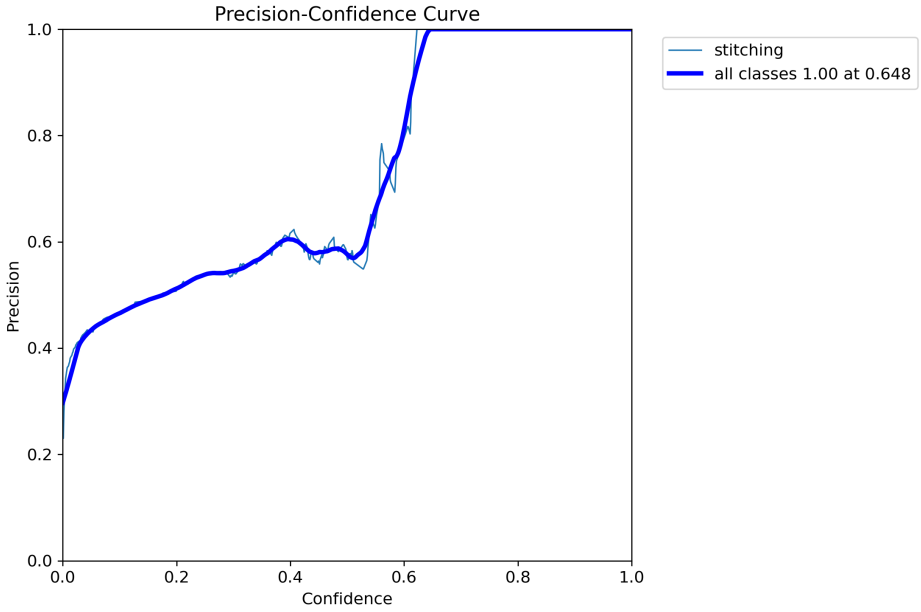

Figura 2.32: Rappresentazione grafica dell'andamento della funzione di precisione (*precision*), caso con etichette che includono due cuciture

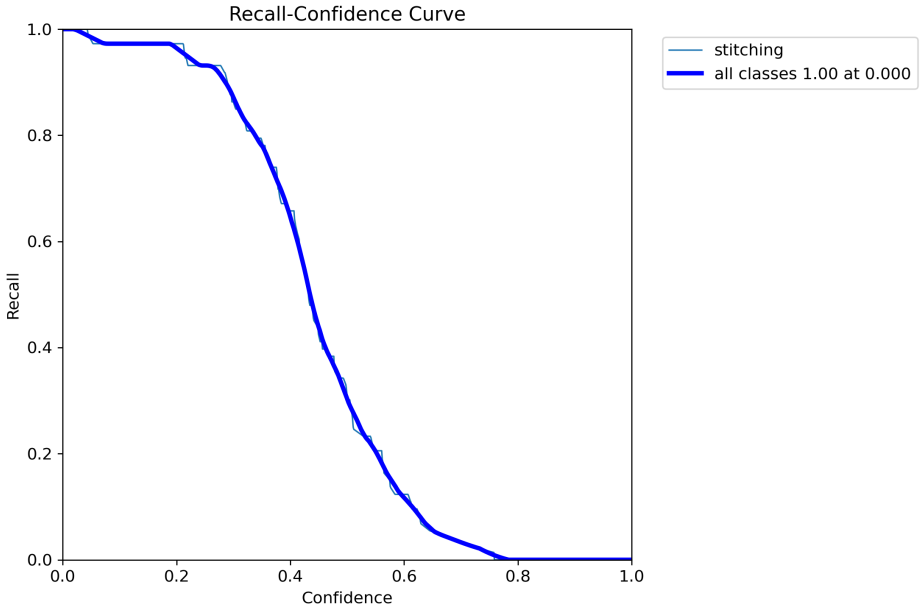

Figura 2.33: Rappresentazione grafica dell'andamento della funzione di richiamo (*recall*), caso con etichette che includono due cuciture

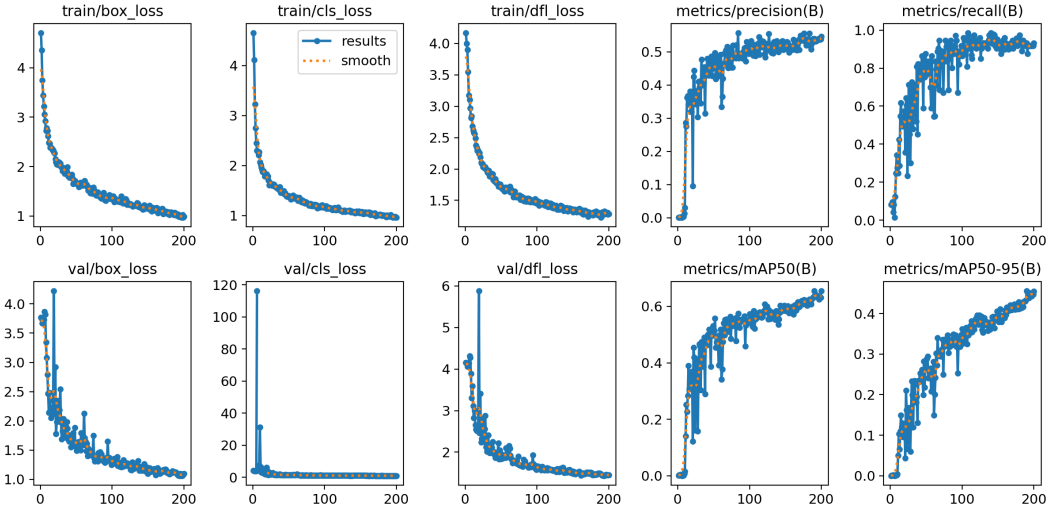

Figura 2.34: Grafici per l'andamento delle funzioni di costo, precisione, richiamo e la loro combinazione (mAP), caso con etichette che includono due cuciture

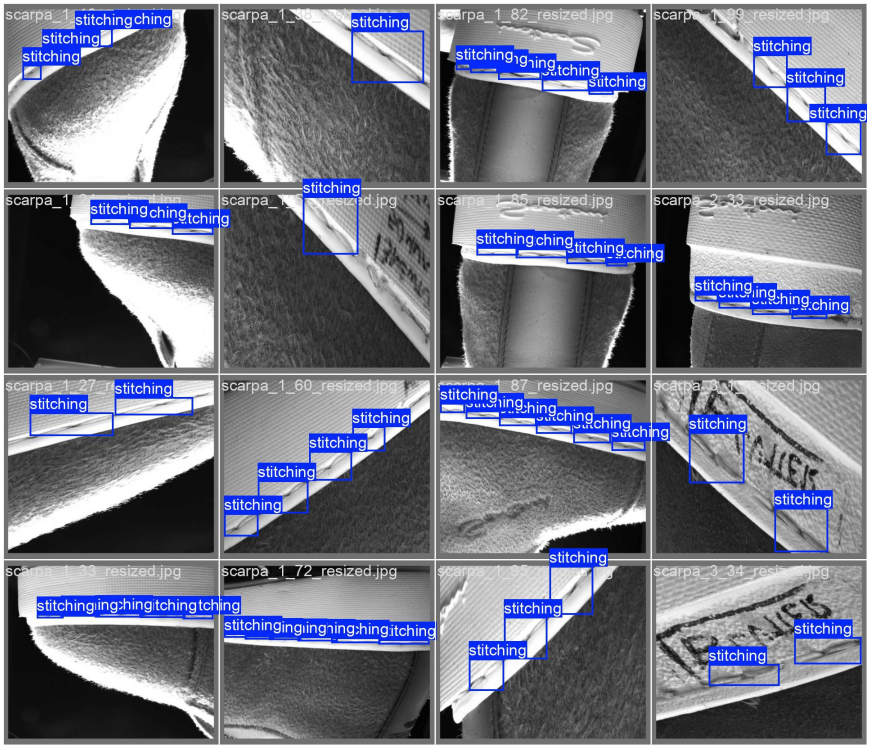

Figura 2.35: Esempio di uno dei batch di validazione usato nella rispettiva fase durante l'addestramento

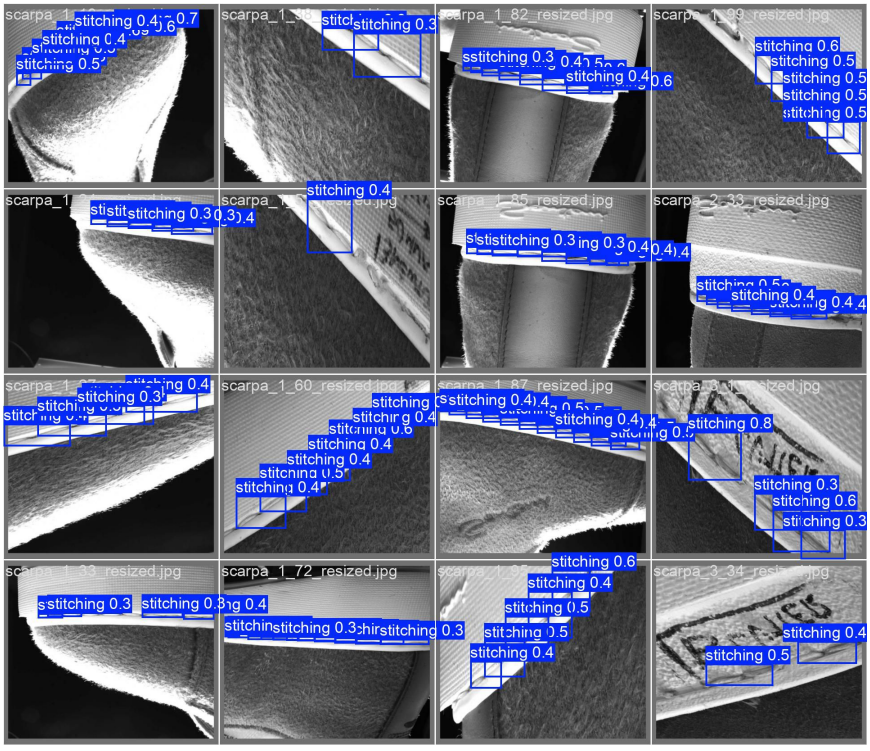

Figura 2.36: Risultati della previsione effettuata sul batch di immagini in Figura 2.36

Come visibile in Figura 2.35 e Figura 2.36 il modello identifica meno cuciture rispetto al caso con etichette su singola cucitura. Inoltre, i livelli di confidenza sono mediamente inferiori, con rari casi che raggiungono il 70% (valore 0.7 in Figura 2.36).

2.4 Studio e scelta dei componenti

Per l'ottenimento dei punti di cucitura sono stati presi in considerazione due metodi di acquisizione:

- Uso di un sistema composto da una telecamera e un sensore di distanza laser;
- Uso di un profilometro.

In entrambi i casi, si considera una disposizione del sistema di misura adiacente al tool per la rimozione delle cuciture, in modo da avere informazioni sui punti successivi sui quali il tool andrà a operare, oltre che per poter effettuare misurazioni spostando gli strumenti di acquisizione tramite il robot.

Il sistema robot ha un limite di peso trasportabile di circa 10 kg: dato che il sistema deve già tenere conto della massa del tool in sé, si cerca di limitare il più possibile il peso del sistema di misura.

Considerazioni simili possono essere fatte per l'ingombro spaziale: un assieme composto da un tool e un sistema di acquisizione di dimensioni notevoli limiterebbe la libertà di movimento del robot, oltre a incrementare il momento di inerzia, e quindi il carico percepito sul terminale.

La superficie su cui si trovano i punti di cucitura ha andamento in generale complesso nello spazio, oltre a essere irregolare, dato che si sta trattando una scarpa usata. Per questo è richiesta una precisa conoscenza della posizione nello spazio dei punti: quindi delle coordinate x, y, z.

Il sistema di acquisizione dovrà dunque essere in grado di ottenere tutte e tre le coordinate.

2.4.1 Profilometro

Si prende in considerazione l'uso di un profilometro laser: questo sensore emette un fascio bidimensionale (linea laser) il quale poi acquisisce la posizione (in x e z) dei punti sulla linea, ovvero la distanza dei vari punti tra loro e la loro distanza dalla sorgente laser (Figura 2.37). La posizione in y può essere ottenuta in modo indiretto facendo traslare il profilometro lungo una direzione, a posizione e velocità note.

Solitamente si usa un sistema con guide lineari azionate da motori elettrici, con incluso un sistema di misura della posizione, tipicamente un encoder. È possibile anche usare profilometri che includono nell'hardware interno un sistema di traslazione del fascio laser, non rendendo quindi necessario l'uso di sistemi di spostamento e di misura della posizione.

Per questa applicazione, si ipotizza che il profilometro sia montato sul terminale del robot, e che quindi sia in grado di muoversi e di conoscere la sua posizione spaziale.

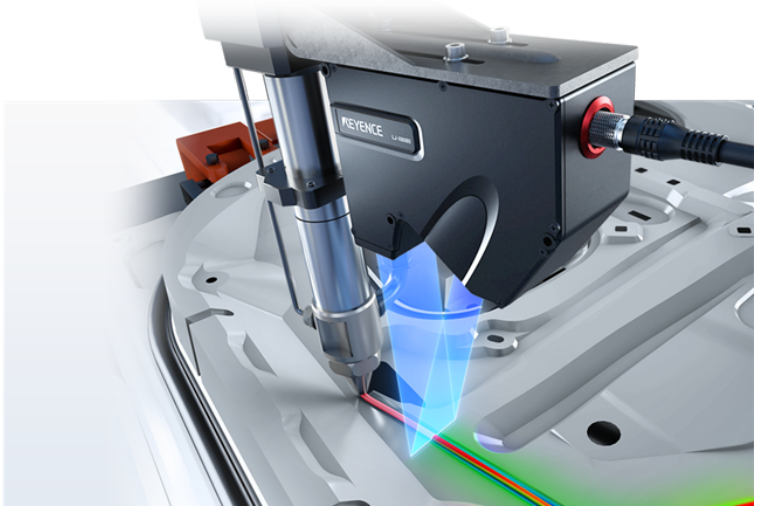

Figura 2.37: Esempio di applicazione industriale con uso di un profilometro

2.4.2 Test con profilometro

Per verificare la fattibilità dell'acquisizione e valutare i risultati ottenibili è stato eseguito un test di misura impiegando un profilometro disponibile all'interno dell'università.

Il dispositivo è composto da un profilometro montato su una guida mobile con sensori di posizione per conoscere la distanza percorsa durante la misurazione. Il test consiste in una semplice acquisizione della superficie della scarpa. Si riportano i risultati ottenuti in Figura 2.38:

L'acquisizione in sé restituisce una nuvola di punti: la memorizzazione dei dati avviene attraverso una matrice, dove righe e colonne indicano lunghezza e altezza dei punti nello spazio. Il valore riportato in ogni elemento della matrice rappresenta la profondità misurata in quel punto, per un totale di tre dimensioni spaziali. In seguito, i dati puntuali ottenuti sono stati rielaborati per poter rappresentare la nuvola di punti sottoforma di superficie (Figura 2.39).

Nel corso dello studio è stato effettuato un incontro con l'azienda keyence. [28]

Keyence è specializzata nella produzione e vendita di sensori di vario tipo, tra cui sensori di distanza laser e profilometri. Durante l'incontro sono state eseguite alcune prove dimostrative del funzionamento di un modello di profilometro con incorporato un sistema di movimento della lama laser.

Il sistema osservato è stato inoltre collegato ad uno di controllo proprietario, con apposito hardware per la gestione dei dati di misura e un software dedicato per la visione e l'editing dei risultati ottenuti.

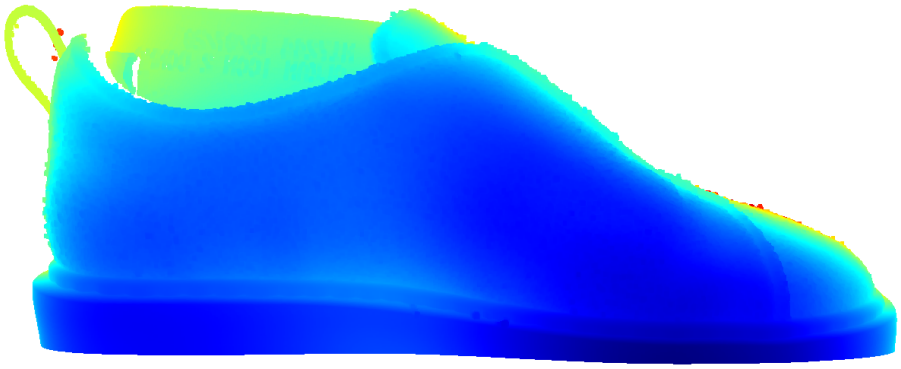

Figura 2.38: Immagine della nuvola di punti

Figura 2.39: Immagine della superficie ottenuta dopo aver elaborato la nuvola di punti

Si riporta di seguito(Figura 2.40) una schermata dell'interfaccia del software e dei risultati ottenuti dalla misura effettuata su una cucitura, seguita da un esempio di come il software può ricavare il punto medio in base alla posizione dei punti di cucitura, grazie alla differenza di profondità percepita.(Figura 2.41)

A seguito del test effettuato e dell'incontro con l'azienda produttrice, sono stati tratti i seguenti risultati: l'utilizzo di un profilometro per ricavare la posizione delle cuciture è teoricamente fattibile; tuttavia, l'ingombro dell'apparecchio e il suo peso rendono complessa la sua installazione sul terminale del robot.

Il profilometro proposto, nonostante sia in grado di misurare autonomamente su un'intera area, non è in grado di acquisire l'intera scarpa, né un solo lato.

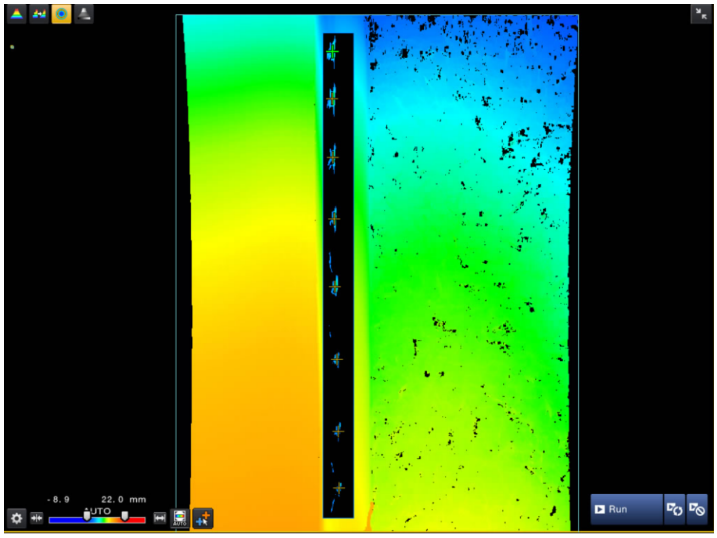

Figura 2.40: Acquisizione eseguita dal profilometro ad area

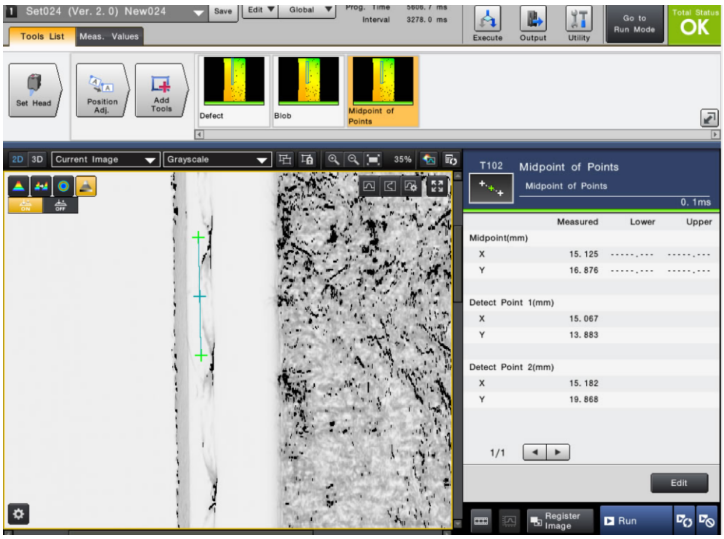

Figura 2.41: Esempio di elaborazione per ricavare punto medio della cucitura

Per ottenere un'area maggiore si deve allontanare il sensore dall'oggetto, ma ciò è fattibile solo fino a un certo limite, inoltre l'accuratezza cala all'aumentare della distanza. Nel caso dell'operazione da svolgere, inoltre, si prevede di mantenere il terminale del robot vicino alla superficie della scarpa: questo impedisce di acquisire aree di grandi dimensioni.

Va inoltre considerato il fattore di costo dell'hardware: un tale sensore risulta mediamente più costoso di eventuali alternative, tra cui l'opzione di una telecamera associata a sensore laser.

2.4.3 Camera con sensore laser

Il sistema consiste in una telecamera che acquisisce immagini nelle vicinanze dei punti di cucitura che il robot dovrà raggiungere, per poi processarle e definire la posizione 2D dei vari punti presenti nell'immagine.

La sola telecamera non può conoscere la distanza delle cuciture da se stessa; quindi, è necessario integrare un ulteriore sistema di misura per conoscere quella che sarebbe la terza coordinata della posizione dei punti: come apparecchio di acquisizione è stato ipotizzato un distanziometro a puntatore laser.

Per quanto riguarda la telecamera, la scelta di un modello specifico è da fare considerando alcuni vincoli dovuti alle condizioni operative:

- La camera verrà posta in vicinanza del tool che lavora la cucitura: la distanza tra camera e oggetto da osservare è quindi molto ridotta. Viene ipotizzata una distanza di lavoro di circa 5 cm;
- La camera è installata a fianco del tool di lavorazione, sulla flangia del robot: è quindi preferibile una camera che non sia troppo voluminosa o pesante, dato che entrambi questi fattori influiscono negativamente sullo spazio disponibile per il movimento del robot e per il carico trasportabile;
- Naturalmente va anche preso in considerazione il costo dell'hardware: in caso di prestazioni e caratteristiche simili, si deciderà per l'opzione più conveniente.

Sulla base di questi vincoli, è stato preso in considerazione il seguente modello, con la relativa ottica:

- Teledyne dals vision camera genie nanoG3-GM11M2020 monochrome 2048x153653fpsC-mount GV (Figura 2.42);
- TUSS Mega-pixel Fixed Focal Lens 2/3 50mm F2.8.

L'ottica associata è scelta in modo da avere una distanza ottimale di lavoro di 50 mm, in coerenza con la distanza ipotizzata per le acquisizioni di circa 5-7 cm. Dalla Figura 2.42 si possono anche intuire le dimensioni molto ridotte del sensore:29x44x21 mm.

L'apparecchiatura appena elencata include solo gli elementi fondamentali del sistema, senza quindi considerare accessori quali cavi di connessione e interfacce software; tuttavia, anche considerando il loro prezzo, il sistema completo resta comunque più conveniente rispetto l'uso di un profilometro.

Essendo disponibili molte varianti, è stata scelta una telecamera di dimensioni contenute, in modo da limitare ingombro e peso che poi vanno a scaricarsi sul robot. Questo è un ulteriore vantaggio rispetto il profilometro, il quale ha peso e dimensioni molto maggiori.

Figura 2.42: Camera scelta per l'applicazione

Capitolo 3

Studio e test del processo robot

3.1 Obiettivo del processo

Il processo sfrutta lo studio eseguito in precedenza sull'identificazione delle cuciture per guidare il movimento del robot: esso si avvicinerà alla scarpa da lavorare, per poi posizionarsi all'incirca nella zona della cucitura.

Il sistema di visione con il modello IA incluso acquisirà un'immagine delle vicinanze del robot per poi guidarlo nella posizione esatta della cucitura, su cui verrà fatta l'operazione di scucitura dall'apposito tool.

Il processo prosegue con il robot che si muove lungo la linea di cuciture, continuamente monitorato e corretto dal sistema di visione. Si procede quindi con lo studio del movimento del robot ed eventuali controlli ulteriori per la traiettoria da seguire.

3.2 Tool per il test

Il tool per il test, inizialmente, doveva non scucire i vari punti ma tagliare, o comunque distruggere, l'intero filo di cucitura. Questa opzione, seppur molto semplice, permetterebbe una lavorazione molto veloce della scarpa.

Tuttavia, eseguire una lavorazione che vada a rompere il filo, lascerebbe detriti nella parte interna della tomaia, rendendo poi necessario un ulteriore intervento, in questo caso da parte di operatori, per rimuovere i frammenti rimasti.

Questo problema ha portato a una modifica del tipo di task da eseguire: dalla semplice rottura del filo, si provvederà invece a sfilarlo dalla scarpa. Rimuovendo il filo senza rotture non resterebbero sfridi all'interno della scarpa. Il nuovo obiettivo richiede però un attrezzo appositamente creato per scucire, perciò è necessario crearlo da zero.

La sua progettazione è affidata all'azienda robot System Automation (RSA), la quale si occuperà del suo studio, in collaborazione con l'Università Politecnica delle Marche, oltre che alla sua costruzione. Al momento dell'esecuzione dello studio sulla cella robotica il tool non aveva ancora iniziato la sua progettazione; quindi è stato necessario utilizzare un sistema sostitutivo che poi andrà rimpiazzato in quello finale.

È stato ipotizzato che il sistema collegato al robot abbia dimensioni comprese tra 10 e 20 cm; per questo motivo è stata creata una punta mobile: il sistema ha una

semplice interfaccia di collegamento con il terminale del robot, che poi si estrude per una certa distanza in direzione perpendicolare alla normale della flangia, di una quantità variabile, grazie alla presenza di una vite collegata alla punta effettiva.

Il modello CAD include, per semplicità, un cilindro che rappresenta la vite nel caso in cui si considera la sua estensione massima, visibile in Figura 3.1.

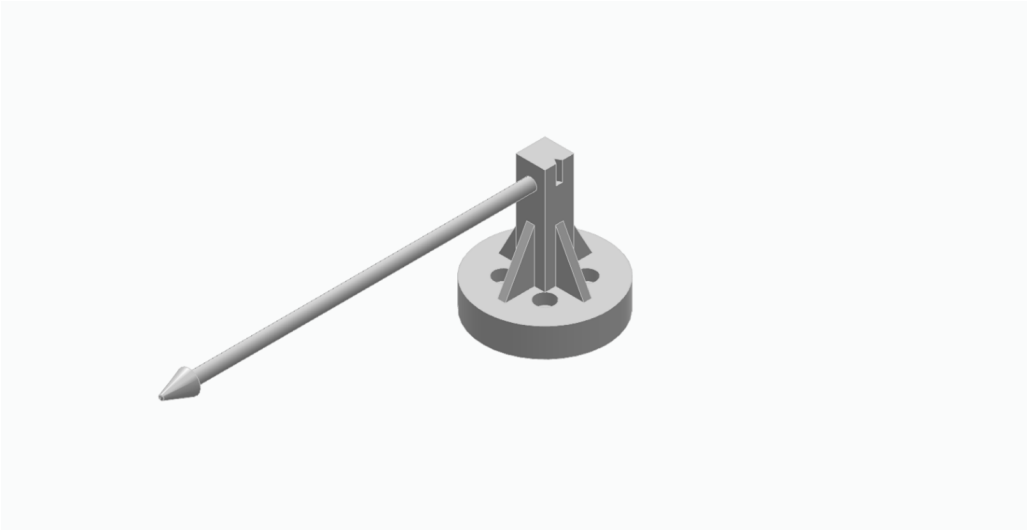

Figura 3.1: Modello CAD del tool sostitutivo utilizzato nelle prove

La punta è modellata in software CAD [29] e stampata in 3D. Aver definito in modo esatto sistemi di riferimento e dimensioni del tool permetterà di semplificare i passi successivi della progettazione del software robot.

Il collegamento alla flangia avviene attraverso quattro viti M5, disposte in modo perpendicolare all'asse z del sistema di riferimento del terminale. Il tool stampato e installato sulla flangia è mostrato in Figura 3.2.

Esso è predefinito sul robot, ed è disposto in modo che l'asse z sia uscente dalla flangia, mentre l'asse x sia parallelo alla sua superficie, disposto verso destra se si osserva la flangia, con asse y disposto, di conseguenza, secondo la regola della mano destra. Per semplicità, la base del tool è disposta in modo che la punta sia perpendicolare alla flangia del robot.

Nei test svolti, la punta si limita a toccare la scarpa nella zona adiacente la cucitura. Ottenere una posizione precisa non è necessario, in quanto le coordinate esatte dei singoli punti verranno forniti dal sistema di visione.

3.3 Programmazione del robot

Lo studio e i relativi test per la cella robotizzata sono stati eseguiti su un robot interno al dipartimento universitario. Nel particolare, si tratta di un Fanuc crx 10-iA/L. [30]

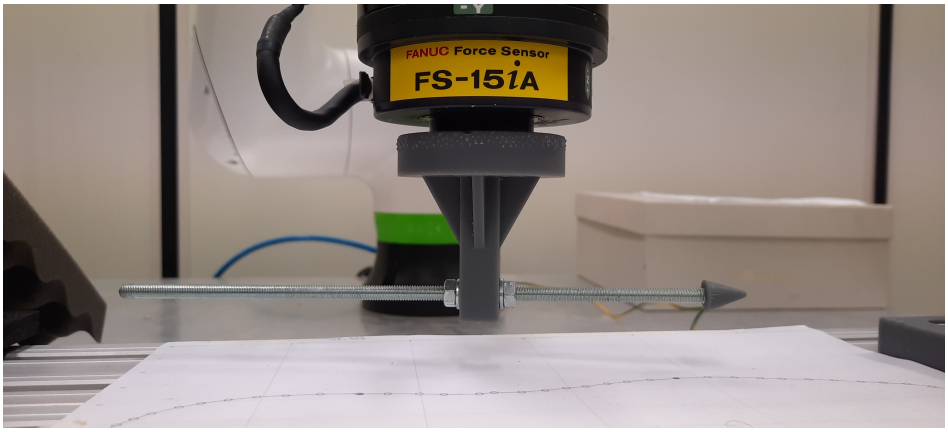

Figura 3.2: Controparte reale del tool utilizzato per i test di traiettoria e controllo di forza

Fanuc è una delle principali aziende produttrici di robot sul mercato, e i modelli della serie crx sono di tipo collaborativo. Essi, infatti, si distinguono per l'assenza di giunture in cui possono restare incastrate dita o braccia degli operatori, oltre che per una struttura in plastica caratterizzata da geometrie curvilinee, finalizzate a ridurre i danni dovuti agli impatti.

Il modello utilizzato ha un peso nominale di soli 39 kg e una portata di carico di 10 kg.

Per quanto riguarda la cinematica, il crx-10-iA/L è parte della categoria dei robot articolati; quindi, composto da una serie di bracci posti in serie tra loro: al termine di un braccio ha sede il motore che muove il braccio successivo.

In totale il robot ha sei gradi di libertà, con un'area raggiungibile semisferica, se non si considerano eventuali tool che possono aumentarla o diminuirla in base ai casi.

La flangia del terminale ha un'interfaccia di connessione per tool di vario tipo, con quattro viti per il fissaggio meccanico e una porta di connessione per collegamento dei sensori o di tool elettrici. Sul terminale, oltre al tool per le lavorazioni, sono installabili anche diversi sensori, più comunemente sensori di forza o di visione.

Oltre al robot in sé, è presente un controllore R-30iB Mini Plus, ovvero un computer adibito al controllo del robot, al suo azionamento e alla gestione dei dati in ingresso e in uscita, oltre che al collegamento con l'interfaccia utente: in questo caso attraverso un Tablet Teach Pendant (Tablet T.P.), ovvero un tablet appositamente installato per essere connesso al sistema robot.

Il controllore possiede diversi terminali per input e output, così da poter ottenere all'esterno informazioni da parte di sensori montati sul robot, oppure ricevere comandi per il movimento o l'attivazione del tool: questi saranno utilizzati in futuro per collegare il sistema di visione al robot, in modo da correggere la traiettoria lungo la linea di cucitura, oltre ad attivare o disattivare il dispositivo per la rimozione del filo.

La programmazione del robot può essere eseguita direttamente dal T.P., attraverso un'apposita interfaccia di programmazione a blocchi, ottimizzata per facilitare e

velocizzare la generazione di programmi per eseguire i task.

Per quanto semplice e intuitiva, la programmazione diretta attraverso T.P. non risulta adeguata all'operazione da svolgere. Sono infatti disponibili dei comandi per eseguire movimenti lineari, circolari, o dei singoli giunti, ma non di curve complesse, se non attraverso combinazioni dei comandi precedenti. Essendo la curva da seguire particolarmente articolata, il processo di creazione del movimento da eseguire richiederebbe molto tempo per essere programmato, oltre a non assicurare un buon livello di inseguimento della superficie.

Per questo motivo si sceglie di generare il programma di movimento del robot attraverso il software ROBOGUIDE. [31]

3.4 ROBOGUIDE

ROBOGUIDE è un software di proprietà di Fanuc, pensato per progettare e simulare celle di lavoro in cui sono presenti sistemi robotici Fanuc: il suo utilizzo permette di ridurre drasticamente il tempo di creazione di nuovi programmi da implementare sulle celle.

Il software permette di simulare non solo il robot in studio, ma l'intera cella di lavoro, includendo eventuali ostacoli, sensori, parti da spostare o su cui eseguire operazioni. L'ambiente di simulazione dispone di un ampio catalogo di modelli 3D di robot, sensori, tool, ecc. Viene inclusa anche la possibilità di importare modelli CAD esterni per poter eseguire simulazioni direttamente con oggetti che poi saranno coinvolti nell'applicazione reale.

La simulazione di una cella parte dalla definizione di un robot, sul quale è impostato il sistema di riferimento di base di tutto il programma. Tra i modelli disponibili nel catalogo si sceglie lo stesso cobot su cui si eseguiranno i test reali, quindi il crx-10iA/L. I passi successivi consistono nell'inserimento degli elementi fondamentali che costituiranno il sistema finale:

- Si ipotizza un piano di appoggio sul quale sarà posizionata la scarpa, per cui si inserisce nella simulazione una fixture: queste sono considerabili come dei componenti simili a parti di lavorazione, la cui caratteristica però è di rimanere fisse nello spazio;

- Il robot può eseguire lavorazioni su elementi denominati parti, per cui ne è stata creata una apposita sulla quale poter svolgere la scucitura. Per semplicità si importa una semplice riproduzione della suola;

- Il tool da collegare al terminale del robot è specifico per l'operazione di rimozione scuciture e, come detto in precedenza, nei test si utilizza un elemento di sostituzione che si limita a toccare la cucitura. Per il suo inserimento nella simulazione si ricorre alla capacità di importazione di file CAD esterni in ROBOGUIDE;

- Il robot utilizzato dispone di un apposito sensore di forza: il Force Sensor FS-15iA. Esso verrà usato per effettuare controllo di forza, e nella simulazione va aggiunto un suo modello in quanto modifica le dimensioni del terminale, allungando l'interfaccia della flangia di collegamento.

- ROBOGUIDE include comandi semplificati per la generazione di traiettorie; in particolare, si può definire una curva in corrispondenza della parte da lavorare, generabile a partire da spigoli o linee di riferimento oppure eseguibile manualmente, mediante la quale si può creare la traiettoria che il robot deve seguire.

3.5 Preparazione della cella

La versione reale della cella consiste in un semplice sistema di appoggio per la scarpa, posizionato entro i limiti di lavoro del robot.

In generale si vuole assicurare corrispondenza tra posizione reale della scarpa e posizione nell'ambiente simulato, in quanto il robot eseguirà una traiettoria riferendosi alla parte simulata, ma operando su quella reale.

La scarpa nella sua interezza è in realtà di poco interesse; al contrario, si vuole sapere principalmente la posizione della sola linea di cucitura. Si ipotizza che essa coincida con il profilo esterno della scarpa; quindi, per conoscere la curva è sufficiente estrarre una linea dal bordo della scarpa.

Per semplicità, si utilizza come profilo di partenza il bordo visibile osservando la parte inferiore della suola, riportato in Figura 3.3. Per l'estrazione si utilizza uno scanner, in modo da assicurare un'acquisizione esatta. La foto scannerizzata potrà poi essere elaborata per estrarre il bordo esterno. (Figura 3.4)

L'estrazione del bordo non genera una curva ma una serie di punti di passaggio (75 in totale), dai quali viene ottenuta per interpolazione. I punti vengono importati in un programma CAD (Solid Edge) per ottenere la linea di passaggio, e da essa estrudere un modello semplificato della suola. Il solido ottenuto verrà poi importato in ROBOGUIDE come parte su cui eseguire la lavorazione. (Figura 3.5)

Nell'applicazione reale si ipotizza che la scarpa resti ferma durante la lavorazione. Per questo motivo vengono estratti dei negativi della suola, utilizzandoli poi per stampare in 3D degli afferraggi che mantengano la scarpa nella posizione desiderata. Questi vengono inoltre fissati a terra, in modo da assicurare che la posizione relativa tra robot e scarpa resti sempre la stessa, anche cambiando scarpa. (Figura 3.6)

Dopo aver inserito gli elementi nella simulazione, si procede con la definizione dei sistemi di riferimento. Essi sono gli elementi con cui ROBOGUIDE identifica i vari oggetti, in quanto a ognuno associa un proprio sistema di coordinate, per poi definire posizione spaziale, orientamento ed eventuali spostamenti rispetto al riferimento solidale.

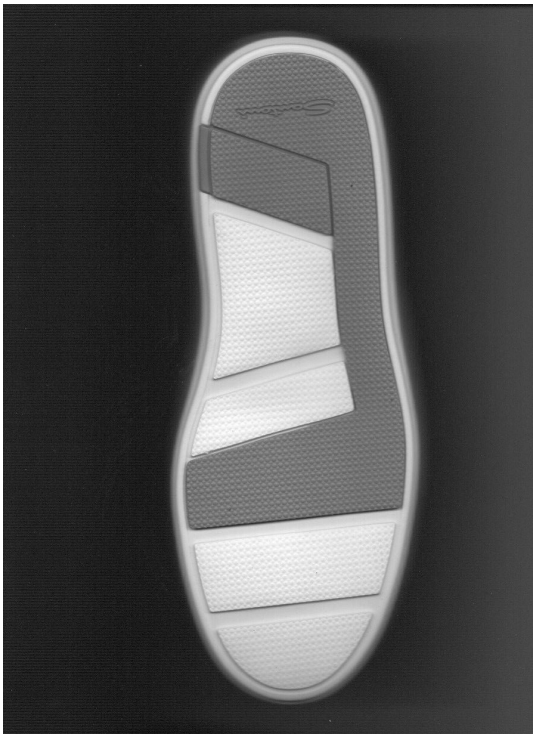

Figura 3.3: Acquisizione della suola da parte dello scanner

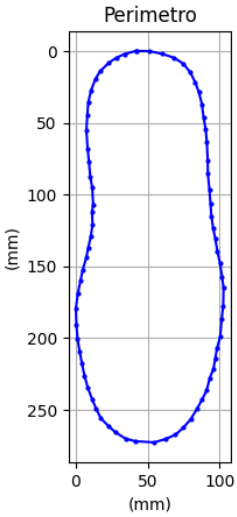

Figura 3.4: Grafico della disposizione dei punti ricavati dall'immagine scannerizzata

L'origine del sistema di riferimento del robot è in corrispondenza del primo giunto, misurata a circa 245 mm di altezza da terra, con asse z in direzione verticale e asse x disposto in modo da apparire entrante nelle porte di collegamento del robot. Il sistema di riferimento è già definito in ROBOGUIDE, quindi non è necessario aggiungerlo.

Di particolare importanza è il sistema di riferimento del tool o *tool center point*

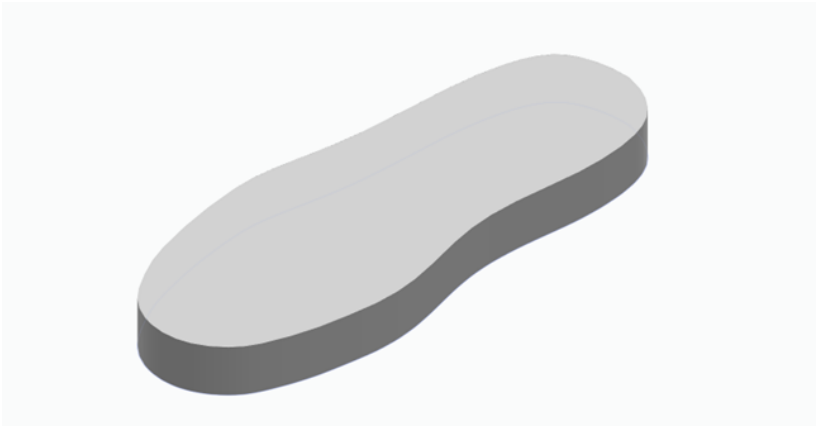

Figura 3.5: Modello 3D ottenuto dai punti estratti in Figura 3.4

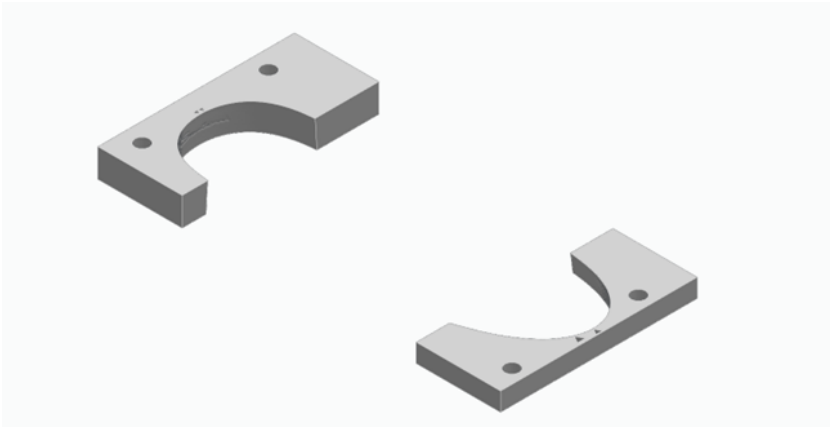

Figura 3.6: Sistema di fissaggio della scarpa

(T.C.P), dato che la sua origine deve coincidere con il punto che entra in contatto con la scarpa: deve quindi essere posizionato in corrispondenza della punta del tool creato. Per la scelta delle orientazioni degli assi si imposta una disposizione tale per cui l'asse z del T.C.P sia uscente dalla punta del tool, con asse y tangente alla direzione in cui deve spostarsi il robot e asse x scelto di conseguenza.

La scarpa reale non ha, ovviamente, un suo sistema di riferimento, né possiede punti notevoli adatti a essere presi come tali. Perciò si sceglie un'alternativa alla semplice impostazione di un punto casuale: l'estrazione del profilo.

Il modello CAD della suola condivide l'origine del sistema di riferimento con quello definito per i punti dello scanner: la suola scannerizzata viene trattata come immagine, quindi si applica lo standard, comune per tutte le immagini digitali, per cui l'origine del sistema di riferimento dell'immagine si trova in alto a sinistra, con asse x rivolto verso destra e asse y verso il basso. La terna di riferimento data dallo scanner viene considerata quindi come il sistema solidale alla scarpa.

ROBOGUIDE definisce tutti i sistemi di riferimento rispetto l'origine del robot; tuttavia, il sistema solidale alla scarpa non coincide con qualche punto noto su di

essa, per cui è difficile definire la sua posizione relativa nello spazio di simulazione. In particolare, anche inserendo il modello virtuale nel programma, si dovrebbe trovare una posizione per cui la versione virtuale coincida con quella reale.

Per garantire coerenza tra reale e virtuale e conoscere la posizione della scarpa rispetto il robot, si usa uno script Matlab [32] creato appositamente a tale scopo. Il codice esegue l'inversa di una funzione di rototraslazione.

In cinematica, per passare da un sistema di riferimento a un altro, si utilizza una matrice di trasformazione che esegue le rotazioni (definite con la matrice mostrata in (3.3)) e le traslazioni necessarie per ridefinire le coordinate di un punto.

$$p^A = {}^A_B T p^B \tag{3.1}$$

$${}^A_B T = \begin{bmatrix} {}^A_B R & O^A_B \\ 0 \quad 0 & 0 \quad 1 \end{bmatrix} \tag{3.2}$$

$${}^A_B R = \begin{bmatrix} \cos\vartheta & -\sin\vartheta & 0 \\ \sin\vartheta & \cos\vartheta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

La trasformazione completa è definita in (3.1). A e B sono due sistemi di riferimento, p^A, p^B sono punti espressi rispettivamente con coordinate rispetto ad A e a B, O^A_B è la posizione dell'origine di B rispetto ad A, ${}^A_B R$ è la matrice di rotazione che porta gli assi del sistema di riferimento B a coincidere con gli assi di A, eseguendo una rotazione di ϑ.

Declinando l'equazione (3.1) nel caso in studio, si considerano i punti rispetto al sistema di riferimento del robot come il risultato di una rototraslazione dei punti definiti rispetto al riferimento dello scanner: una volta ricavati i punti sul robot, è possibile definire l'origine dei punti rispetto allo scanner.

Questo perché la matrice di trasformazione contiene i valori di rotazione e di traslazione tra i due sistemi di riferimento; dunque la conoscenza della matrice permette di conoscere anche la posizione relativa tra robot e scarpa.

Da notare che la matrice di rotazione ${}^A_B R$ è definita considerando una rotazione intorno l'asse z. Quindi, sebbene la curva delle cuciture in generale abbia un andamento variabile in tutte le direzioni, si ipotizza che sia contenuta in un piano parallelo all'asse z, per semplificare il calcolo.

L'operazione richiede sia i punti visti dal robot che quelli visti dallo scanner:

- Per i punti ottenuti dallo scanner, sono già disponibili quelli estratti per definire il profilo (Figura 3.4);

- Per i punti rispetto al robot, si usa una misura ricavata dalla scarpa: fissando la scarpa sugli afferraggi, se ne ricavano sei punti, andando a toccare il bordo della suola con la punta del tool. La misura è possibile in quanto il robot può

fornire la posizione in tempo reale del T.C.P. I punti misurati sono riportati nella tabella 3.1.

x	y	z
731.395	-141.609	-149
696.492	-74.315	-149
686.893	36.238	-149
727.982	128.744	-149
788.906	44.763	-149
779.844	-77.809	-149

Tabella 3.1: Tabella con i valori delle coordinate dei sei punti misurati tramite posizione del TCP del robot

Inserendo i punti misurati e i punti dati dallo scanner nello script matlab, viene eseguita la trasformazione. Essendo i punti sul robot misurati, quindi contenenti un certo errore di misura, la matrice non sarà esatta ma solo approssimata. Per ridurre l'errore, si esegue la ricerca di un minimo: a partire da una matrice iniziale, ipotizzando dei parametri di primo tentativo nulli, si esegue più volte l'estrazione della matrice di trasformazione e la si applica a sei dei settantacinque punti definiti rispetto lo scanner.

Si calcola poi la distanza tra i punti trasformati e i punti misurati rispetto al robot. Se la misura fosse esatta, dovrebbero esistere sei punti che sono sovrapposti a quelli trasformati a partire dal sistema dello scanner, quindi aventi distanza nulla. Essendoci sempre un errore di misura, ci si limita a prendere i punti che minimizzano queste distanze. Il risultato di questa sovrapposizione è mostrato in Figura 3.7.

L'operazione viene ripetuta modificando i parametri ottenuti per la matrice di trasformazione fino a ottenere nuovamente un valore minimo dell'errore, definito in *fminsearch* nel codice matlab.

```
clc
close all
clear
global punti_s P_r
punti_s=importdata("perimeter_points.xlsx");

p1 = [707.730 170.129 -115]';
p2 = [671.978 204.892 -115]';
p3 = [661.978 346.481 -115]';
p4 = [705.393 439.219 -115]';
p5 = [760.077 341.608 -115]';
p6 = [749.018 233.129 -115]';
P_r = [p1 p2 p3 p4 p5 p6; ones(1,6)];
punti_s=[punti_s';ones(1,size(punti_s,1))];
scatter(P_r(1,:),P_r(2,:))
```

```
guess_par=[0 0 0 0]; % presi valori iniziali ipotizzati per la posizione
    del sist. rif.scanner
e=err(guess_par);
options =
    optimset('MaxIter',10^8,'MaxFunEvals',10^8,'TolFun',10^-10,'TolX',10^-10);
par=fminsearch(@err,guess_par,options)

q=par(1);
Os=par(2:4);

Rz=[cos(q) -sin(q) 0;
    sin(q) cos(q) 0;
    0          0  1];
T=[Rz Os'; 0 0 0 1];
punti_r=T*punti_s;

figure (1)
plot(punti_r(1,:),punti_r(2,:));
hold on
plot(P_r(1,:),P_r(2,:),'o')
axis equal

function e=err(guess_par)
global punti_s P_r
q=guess_par(1);
Os=guess_par(2:4);

Rz=[cos(q) -sin(q) 0;
    sin(q) cos(q)  0;
    0          0   1];
T=[  Rz Os';
    0 0 0 1];
punti_r=T*punti_s;
for i=1:size(P_r,2)
    for j=1:size(punti_r,2)
        %calcolo distanza tra punti misurati e punti rototraslati secondo
        % la matrice trovata
    distanze(i,j)=norm(P_r(:,i)-punti_r(:,j));
    end
end
e=norm(min(distanze,[],2));
end
```

La tabella 3.2 mostra la struttura della matrice di rototraslazione ottenuta a seguito della minimizzazione degli errori nei parametri. I valori riportati nell'ultima colonna equivalgono alla posizione del centro del sistema di riferimento dello scanner rispetto

al riferimento del robot: sono dunque i risultati da portare in ROBOGUIDE.

0.9998	0.0193	0	589.1420
-0.0193	0.9998	0	163.1001
686.893	36.238	1	-115
0	0	0	1

Tabella 3.2: Tabella con i valori della matrice di rototraslazione ottenuta dal codice Matlab

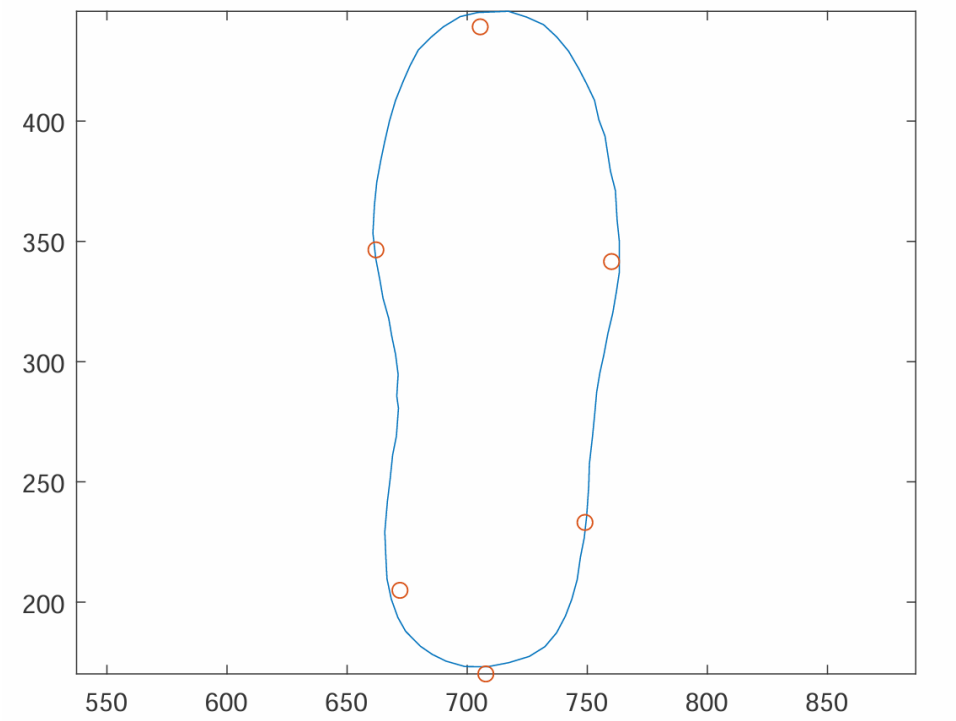

Figura 3.7: Risultato fornito dal codice Matlab: mostrati i sei punti misurati con l'insieme dei punti dello scanner dopo la trasformazione

Una volta trovato il minimo, si inseriscono i dati trovati in ROBOGUIDE per definire la posizione del modello. Grazie al processo di misura dei punti reali, il sistema corrisponde sia nel caso reale che in quello virtuale, a meno di un errore che però è stato minimizzato, e quindi trascurabile.

Si hanno pertanto tutti gli elementi definiti e posizionati come nel caso reale, per cui si procede con la definizione della traiettoria che il robot dovrà eseguire.

3.6 Controllo di posizione

Il comando di generazione della traiettoria è *cad to path*: semplicemente selezionando la curva generata in precedenza, esso produce una possibile traiettoria eseguibile

dal robot. ROBOGUIDE permette anche di visualizzare il comportamento del tool lungo la linea selezionata, in modo da correggere eventuali errori o modificare il punto di contatto o la direzione normale al movimento.

Ipotizzando che il tool esegua l'operazione di scucitura sempre restando verticale rispetto ad essa, si imposta una traiettoria che mantenga il terminale del robot perpendicolare alla superficie della scarpa.

Essendo previsto un controllo in forza, si definisce anche un offset del punto di contatto di 2 mm.

Tra le opzioni disponibili vi è anche l'aggiunta di un semplice movimento di avvicinamento e allontanamento dai punti di inizio e fine lavorazione; questo viene aggiunto in modo che il robot si avvicini con il tool perpendicolare alla superficie da lavorare a partire da una distanza di 10 cm, per poi allontanarsi, una volta completata la lavorazione, della stessa distanza.

Una volta completata la definizione della traiettoria, ROBOGUIDE genera il file di programmazione da eseguire sul robot.

3.7 Controllo di forza

Il processo di scucitura richiede che il robot vada in contatto con la superficie laterale della scarpa, per poi eseguire l'operazione. Per avere maggiore controllo sulla lavorazione, si utilizza un controllo di forza.

In generale il controllo di forza è utilizzabile nei robot collaborativi, in quanto è richiesto conoscere la forza applicata dal robot per evitare che questa sia eccessiva e quindi cadere in situazioni di pericolo per gli operatori. Nell'applicazione considerata, si utilizza una cella di carico installata sul terminale del robot: il sensore di forza FS-15iA (visibile sul robot in Figura 3.2).

Il sensore può misurare forze e momenti applicati sul terminale del robot, per poi inviare i dati ottenuti al controllore del sistema per eseguire il controllo effettivo.

A livello di programmazione, ROBOGUIDE non permette una gestione semplificata del controllo di forza, per cui è necessario aggiungere i comandi direttamente tramite T.P. e impostarli manualmente.

Per assicurare la corretta esecuzione del controllo di forza, è stato eseguito un tuning del sensore: il robot calibra il sensore portandosi al punto di applicazione della forza, per poi eseguire autonomamente un avvicinamento fino al contatto della superficie e imprimendo una forza data.

L'operazione viene ripetuta più volte in modo da ottenere una calibrazione del sensore di forza e modificare i parametri del controllore.

Considerando che si vuole controllare la forza continuamente lungo tutta la traiettoria, si utilizza il comando di *force contouring*. Esso è suddiviso in due sottocomandi che vengono posti all'inizio e alla fine del percorso di cui si vuole controllare la forza: *force contouring start* e *force contouring end*.

Il comando di partenza richiede il punto iniziale per far partire il controllo, il sensore da usare, la forza applicata e in quale direzione applicarla. Si imposta quindi una forza agente lungo l'asse z del T.C.P, eseguendo varie prove con valori di forza tra 1 N e 5 N. La velocità di avvicinamento è anch'essa definibile nel comando di contouring, includendo anche di quanto il tool può "affondare" nella suola.

Il programma generato da ROBOGUIDE può essere semplicemente copiato nei registri del controllore robot, in quanto è possibile estrarre il file con il codice direttamente utilizzabile nel crx.

I vari test sono stati eseguiti in quanto ci si aspetta che, a partire dall'offset di 2 mm definito nel controllo di posizione, il robot si muova verso la superficie fino a toccarla con una forza coincidente con quella definita nel comando di contouring.

È stato notato nei vari test che lungo la curva della suola, in particolare in corrispondenza della punta, il robot tende a staccarsi da essa, in quanto la curvatura stretta provoca una riduzione della forza applicabile; se la velocità della lavorazione è troppo alta, il robot non riesce a esercitare la forza lungo i tratti con curvatura più elevata; quindi, tende ad allontanarsi dalla superficie in lavorazione.

Questo problema ha portato a limitare la velocità di avanzamento del robot, in modo che possa restare sempre in contatto con la scarpa e continuare il controllo di forza. Successive prove hanno portato a una determinazione del tempo medio impiegato per compiere una rivoluzione completa intorno la suola.

Capitolo 4

Risultati ottenuti

4.1 Cella Robot

Grazie ai controlli di posizione e di forza, il tool riesce a seguire adeguatamente la superficie laterale della suola.

Sebbene i test siano stati condotti sulla stessa scarpa, il movimento è generalizzabile a qualsiasi scarpa dello stesso numero, o in generale a calzature con forma e dimensioni simili.

Il programma generato da ROBOGUIDE ha semplificato il processo di creazione della traiettoria grazie all'uso del modello CAD della suola, ottenuto tramite scansione. La traiettoria ottenuta è visibile nella schermata in Figura 4.1.

Eseguendo più test sullo stesso programma e nelle stesse condizioni, sono state valutate delle variazioni nei parametri del controllo di forza per ricercare la velocità massima per cui il movimento resta accettabile e il tool non si allontana dalla superficie: procedendo per tentativi, sono stati ottenuti tempi di percorrenza del perimetro della scarpa inferiori al minuto: impostando una velocità media e cronometrando il tempo impiegato per raggiungere il punto di avvicinamento e allontanamento, disposti a 10 cm dalla superficie, sono stati contati 55 secondi circa.

L'uso dello script matlab è necessario solo nel momento in cui si definisce la posizione della scarpa; una volta fissata, i sistemi di riferimento non cambiano relativamente tra loro. Il codice resta comunque utilizzabile in caso fosse necessario modificare il posizionamento della scarpa, visibile nel caso reale in Figura 4.2.

Ulteriori prove sono state condotte modificando le dimensioni del tool: grazie alla barra filettata, il tool è regolabile nella sua lunghezza rispetto al centro della flangia. A causa della modifica delle dimensioni del tool è necessario ricreare il programma di inseguimento della traiettoria. Tuttavia, grazie a ROBOGUIDE, bisogna semplicemente traslare il T.C.P del tool simulato al valore desiderato e rigenerare il codice eseguibile.

Il processo di modifica richiede solo qualche minuto, e una volta importato il programma nel robot è facilmente eseguibile. Ciò faciliterà le fasi successive di progettazione, in quanto il tool definitivo per la lavorazione potrà poi essere reso virtuale e utilizzato per generare il programma finale.

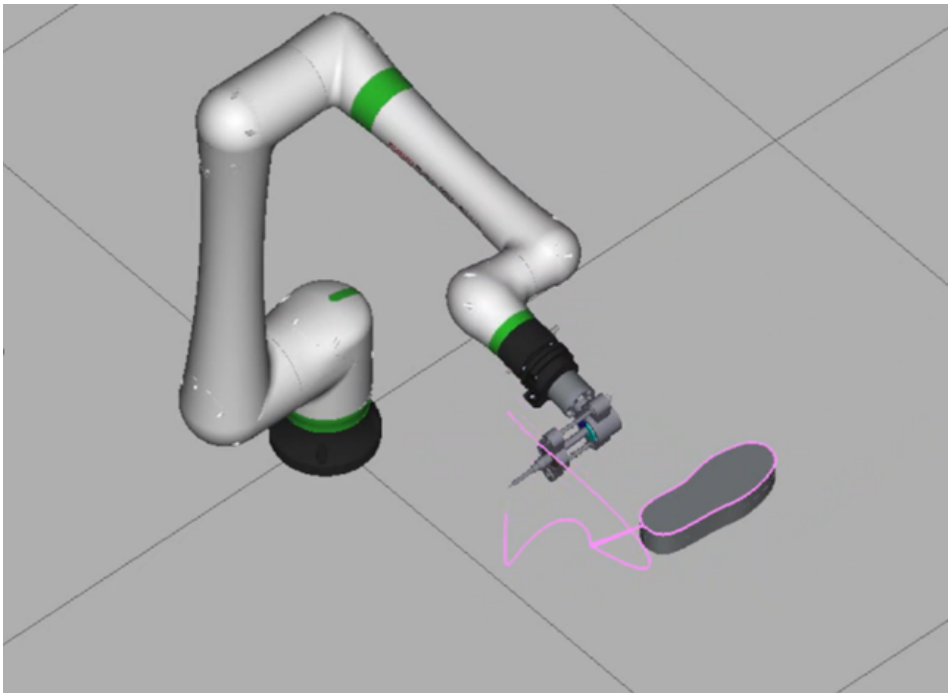

Figura 4.1: Rappresentazione virtuale ottenuta in ROBOGUIDE della cella di lavoro

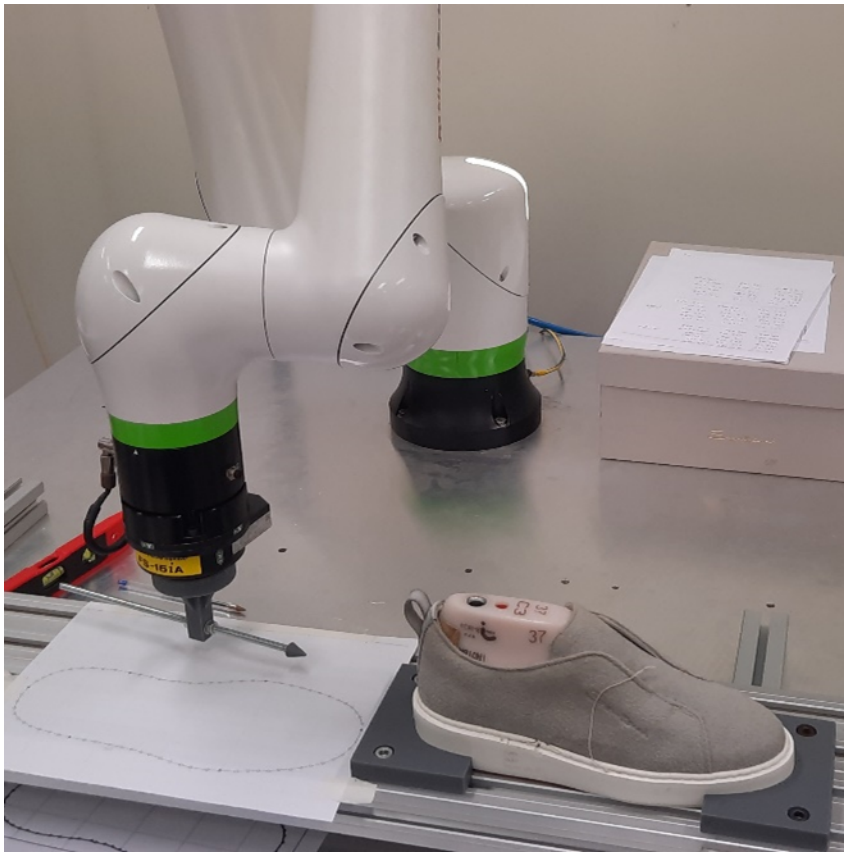

Figura 4.2: Controparte reale della cella di lavoro

4.2 Sistema di visione

Per quanto riguarda il sistema di visione con algoritmo di intelligenza artificiale, è stato eseguito un ultimo test per verificare se il modello fosse utilizzabile anche a partire da file video, in modo da osservare il suo comportamento in caso sia necessario implementarlo con un sistema di visione che implichi la registrazione di video piuttosto che acquisizione di immagini. L'esecuzione è stata definita su una semplice ripresa da vicino delle cuciture della scarpa. Non essendo disponibile l'hardware finale, è stata considerata una semplice ripresa della durata di 15 secondi. Per quanto riguarda l'esecuzione, è stato recuperato e adattato un approccio di riconoscimento oggetti disponibile online [33] sebbene sia stato preso come esempio di applicazione e quindi non sia ottimizzato per un uso industriale.

Il codice si limita, dopo una prima rielaborazione del file video, a prendere ogni frame e applicarvi il modello addestrato. I risultati forniti, ovvero le predizioni con le varie bounding box, sono poi mostrate a schermo. Dato che esse devono definire il profilo esterno delle cuciture, viene stampato anche un punto, dopo aver ricavato le posizioni intermedie dei rettangoli. Questi coincidono in media con i centri delle cuciture e sono i punti sui quali verrà applicato il tool per eseguire la scucitura del filo. Si riporta in Figura 4.3 un frame della ripresa.

L'elaborazione dei dati di ogni singolo frame, il successivo calcolo della posizione centrale e la stampa a schermo dei risultati ottenuti rallentano notevolmente la velocità di riproduzione del video.

Tuttavia, è da considerare che il codice non è ottimizzato per un uso industriale, e la ripresa eseguita e l'hardware utilizzato non sono quelli finali su cui verrà elaborato il programma. Nonostante ciò, il risultato è comunque considerabile di qualità sufficiente.

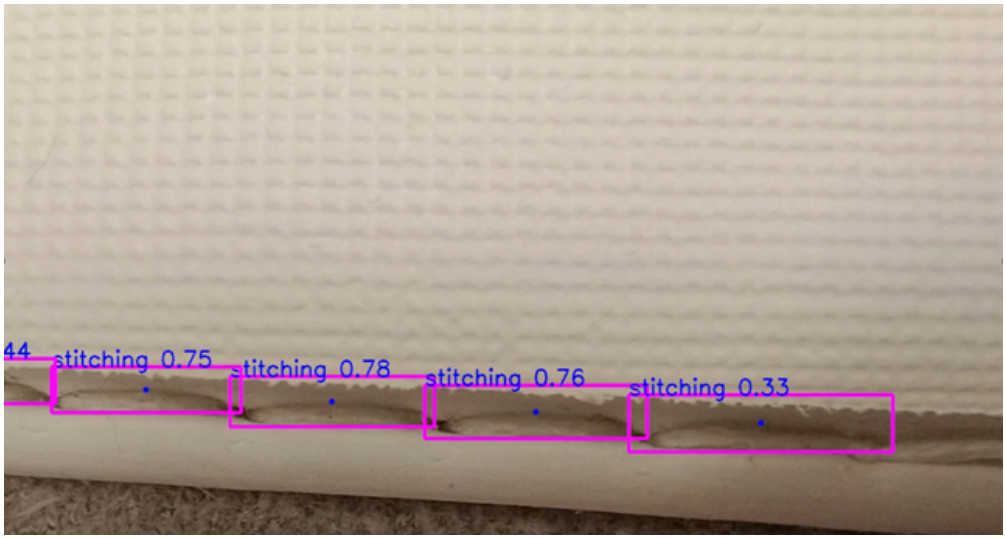

Figura 4.3: Frame ricavato dalla ripresa video

Capitolo 5

Conclusioni

Si ricordano le richieste del progetto: studio della fattibilità di una cella di lavoro robotizzata per la scucitura delle scarpe nell'ambito del riciclo della suola e del riuso della tomaia, definendo una possibile proposta di struttura e funzionamento di base della cella. Lo studio svolto rientra nel progetto di innovazione e ammodernamento del sistema produttivo di calzature CALZARE 4.0, portato avanti da Santoni S.p.A.

Il processo di scucitura verrà svolto da un robot collaborativo, grazie anche a un tool appositamente progettato per la rimozione di cuciture senza rotture del filo. Il tool, collegato al terminale del robot, dovrà avvicinarsi a una scarpa che viene posta nella cella, per poi scorrere lungo il perimetro della scarpa rimuovendo la cucitura.

Al termine dell'operazione, si avrà in uscita la scarpa senza il filo per connettere suola e tomaia, senza ulteriori scarti di lavorazione. In particolare, non resteranno frammenti di filo nella parte interna della scarpa, grazie al fatto che la lavorazione non prevede rotture ma solo sfilamenti.

Come cobot per lo studio è stato preso il Fanuc crx-10iA/L, con inclusa una cella di carico per poter eseguire il controllo di forza sulla scarpa.

Dai test condotti, il cobot risulta in grado di seguire la traiettoria descritta dalla linea di cucitura, eseguendo allo stesso tempo un controllo della forza esercitata sulla superficie della scarpa. Grazie al programma di simulazione ROBOGUIDE, la generazione dell'eseguibile per la definizione del processo di movimento e controllo è risultata semplice e veloce, oltre che rapidamente modificabile in base a eventuali cambiamenti nei parametri del processo.

A seguito dei test eseguiti, sono stati raggiunti tempi di esecuzione inferiori al minuto, in coerenza con l'ambiente di lavoro industriale. I risultati sull'andamento della traiettoria generata sono visibili osservando la curva seguita dalla punta del tool in Figura 4.1.

Per quanto riguarda l'estrazione delle singole cuciture, è stato addestrato un modello di rete neurale per riconoscere punti di cucitura a partire da acquisizioni visive (risultati visibili in Figura 4.3).

Lo studio sulla sensoristica utilizzabile ha tenuto in considerazione due possibili sistemi di acquisizione: un profilometro o una telecamera in aggiunta a un sensore di distanza. Sebbene il profilometro non abbia bisogno di ulteriori sensori per ricavare

le informazioni spaziali ricercate, il suo ingombro notevole e il suo costo più elevato hanno favorito la scelta della telecamera, più compatta e conveniente.

Tale modello di riconoscimento potrà essere implementato nel sistema finale, per migliorare la precisione del robot nella fase di avvicinamento alla cucitura.

Per quanto riguarda sviluppi futuri dello studio condotto:

- Il modello è stato addestrato su un numero di dati di training limitati, conseguendo tuttavia risultati accettabili, grazie anche all'uso della data augmentation. Un maggiore quantitativo di dati migliorerebbe senza dubbio i risultati ottenuti.

- Sebbene il programma generato sia flessibile, è necessario eseguire test considerando il tool definitivo che verrà montato sul robot per poter valutare quali saranno le performance finali dell'intero sistema, oltre che l'utilizzo di software e hardware industriali per assicurare le stesse condizioni di quelle effettive.

- Finora è stato studiato il processo di lavorazione in sé, senza tenere conto dei vari sistemi accessori e considerazioni aggiuntive per l'effettiva applicazione industriale della cella, quali dei sistemi di riferimento definiti tra robot e scarpa, sistemi ausiliari di afferraggio, implementazione in tempo reale dell'algoritmo di IA.

- Il codice per l'esecuzione dell'algoritmo su riprese video deve essere ottimizzato se lo si vuole includere nel futuro sistema di visione industriale.

- In assenza di collegamento diretto tra robot e sistema di visione, è necessaria la progettazione e scrittura di un programma che utilizzi i dati ricevuti dai risultati del modello e li converta in azioni di correzione della traiettoria del robot, in modo da assicurare precisione elevata durante l'operazione.

In conclusione, è stato ottenuto un esempio di funzionamento di base per una cella robot per l'applicazione desiderata, e si è visto come potrebbe essere strutturato un sistema robotizzato per la scucitura delle scarpe, facilmente utilizzabile come punto di partenza per la progettazione avanzata del sistema industriale (rappresentati in Figura 4.2 per il caso reale e Figura 4.1 per la versione virtuale).

Le velocità di esecuzione e l'utilizzo di robot collaborativi ridurranno il lavoro manuale richiesto agli operatori, assicurando elevate velocità di esecuzione. La reale fattibilità del sistema apre la possibilità di introdurre sistemi robotici nella filiera di produzione calzaturiera, supportando e agevolando la lavorazione artigianale.

La sua implementazione garantisce una gestione semplice del sistema automatizzato e, all'interno del processo di recupero della scarpa, favorisce l'adesione ai paradigmi di sostenibilità sostenuti da CALZARE 4.0.

Bibliografia

[1] M.Forlini. Integrazione di un robot mobile con un manipolatore collaborativo. Master's thesis, Università Politecnica delle marche, 2021.

[2] G. Mazzuto. Smart factories. Università Politecnica delle marche, 2024. appunti del corso di Smart Factories.

[3] Innovation Post. Industria 5.0: cos'è la quinta rivoluzione industriale, vantaggi e rischi e quale ruolo ha l'europa. [online] available: https://www.innovationpost.it/tecnologie/industrial-it/industria-5-0-cose-e-come-cambia-modo-di-fare-impresa/#:~:text=L'Industria%205.0%20cos%C3%AC%20come,impatto%20ambientale%20degli%20processi%20produttivi. Accessed: 2025.

[4] International Organization for Standardization, Geneva, Switzerland. *ISO 8373:2021 Robotics — Vocabulary*, 2021. Available at https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-3:v1:en.

[5] G. Palmieri. Università Politecnica delle marche, 2024. appunti del corso di Robotica Industriale.

[6] Treccani. Intelligenza artificiale. [online] available: https://www.treccani.it/enciclopedia/intelligenza-artificiale_(Enciclopedia-della-Scienza-e-della-Tecnica)/, 2008. Accessed: 2025.

[7] Ultralytics. Apprendimento automatico. [online] available: https://www.ultralytics.com/it/glossary/machine-learning-ml. Accessed: 2025.

[8] Mathworks. rete neurale. [online] available: https://it.mathworks.com/discovery/neural-network.html. Accessed: 2025.

[9] Mathworks. rete neurale convoluzionale. [online] available: https://it.mathworks.com/discovery/convolutional-neural-network.html. Accessed: 2025.

[10] Michele Germani et al. Matteo Forlini, Marianna Ciccarelli. Advancing human–robot collaboration in handcrafted manufacturing: cobot-assisted polishing design boosted by virtual reality and human-in-the-loop. *Int J Adv Manuf Technol*, 2024.

[11] P. Castellini. Algoritmi di matching. Università Politecnica delle marche, 2024. appunti del corso di Sistemi di Misura e Visione.

[12] Open CV Team. [online] available: https://opencv.org/. Accessed: 2025.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[14] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[15] Glenn Jocher, Jing Qiu, and Ayush Chaurasia. Ultralytics YOLO, January 2023.

[16] CoCo128 Dataset. [online] available: `https://www.kaggle.com/datasets/ultralytics/coco128`. Accessed: 2025.

[17] Ultralytics. Object detection datasets overview. [online] available: `https://docs.ultralytics.com/datasets/detect/#ultralytics-yolo-format`. Accessed: 2025.

[18] mdbloice. Augmentor. [online] available: `https://github.com/mdbloice/Augmentor`. Accessed: 2025.

[19] G.Zoppi. Recognition and anticipation of human actions in a human-robot collaborative assembly scenario. Master's thesis, Università Politecnica delle marche, 2024.

[20] Meta. Segment anything. [online] available: https://segment-anything.com/. Accessed: 2025.

[21] Nvidia. Cuda toolkit. [online] available: https://developer.nvidia.com/cuda-toolkit. Accessed: 2025.

[22] Google. Colaboratory. [online] available: https://colab.research.google.com/. Accessed: 2025.

[23] Google. Gpu architecture. [online] available: `https://colab.research.google.com/github/d2l-ai/d2l-tvm-colab/blob/master/chapter_gpu_schedules/arch.ipynb`. Accessed: 2025.

[24] Ultralytics. Yolov9. [online] available: https://docs.ultralytics.com/models/yolov9/#performance-on-ms-coco-dataset. Accessed: 2025.

[25] Ultralytics. Ultralytics glossary. [online] available: https://www.ultralytics.com/it/glossary. Accessed: 2025.

[26] Google. accuracy-precision-recall. [online] available: https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall?hl=it. Accessed: 2025.

[27] Ultralytics. what are hyperparameters. [online] available: https://docs.ultralytics.com/guides/hyperparameter-tuning/#what-are-hyperparameters. Accessed: 2025.

[28] Keyence italia s.p.a. Homepage. [online] available: https://www.keyence.it/. Accessed: 2025.

[29] Siemens. Solid edge. [online] available: https://solidedge.siemens.com/it/. Accessed: 2025.

[30] Fanuc. Home. [online] available: https://www.fanuc.eu/it/it. Accessed: 2025.

[31] Fanuc. Roboguide. [online] available: https://www.fanuc.eu/it/it/robot/accessori/simulation-software-roboguide. Accessed: 2025.

[32] Mathworks. Matlab. [online] available: https://it.mathworks.com/products/matlab.html. Accessed: 2025.

[33] Dipankar Medhi. Real-time object detection with yolo and webcam: Enhancing your computer vision skills. [online] available: https://dipankarmedh1.medium.com/\protect\penalty\z@real-time-object-detection-with-yolo-and-webcam-\protect\penalty\z@enhancing-your-computer-vision-skills-861b97c78993. Accessed: 2025.