UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea magistrale in *Ingegneria Informatica e dell'Automazione*

Progettazione e sviluppo di un sistema di visione per il riconoscimento di incisioni su pneumatici attraverso tecniche di analisi di immagini.

Design and development of a vision system for the recognition of tire engravings through image analysis techniques.

Relatore:
Prof. Adriano Mancini
Correlatore:
Dott. Marco Brutti

Laureando:
Francesco Zerbino Di Bernardo

Anno Accademico 2023/2024

Prefazione

Oggigiorno la tendenza industriale è volta a dare sempre maggiore attenzione all'ottimizzazione del processo produttivo e al migliorare la qualità dei propri prodotti per garantire standard di qualità elevati.

Diventa quindi di fondamentale importanza la fase di ispezione, in tempo reale, delle superfici con pattern e texture complesse, attraverso l'utilizzo di tecniche di intelligenza artificiale applicate alla visione. Quindi, assume un ruolo centrale, la possibilità di disporre di un sistema di visione intelligente perfettamente integrato alla propria linea produttiva, per il monitoraggio di aspetti e caratteristiche chiave del prodotto.

Nella produzione di pneumatici stradali vi è la problematica di riuscire a identificare anomalie come incisioni o decifrare sigle e codici dalla spalla laterale. Sulla spalla dello pneumatico possiamo trovare informazioni chiavi, come dimensioni o codice identificato, ed è una priorità riuscire a recuperarle facilmente e correttamente. Detto ciò, diventa fondamentale per il produttore determinare pneumatici anomali cosi che possa intervenire immediatamente.

L'obiettivo della tesi è di progettare e sviluppare una metodologia che riesca ad individuare il Department of Transportation (DOT), un codice seriale identificativo e quindi indispensabile per ogni pneumatico, attraverso l'utilizzo di un sistema di visione e tecniche di analisi di immagini.

Verrà utilizzata una scansione ad alta risoluzione della spalla laterale dello pneumatico, per individuare IL DOT attraverso una precisa analisi ed elaborazione dell'immagine e l'utilizzo di modelli intelligenti per l'optical character recognition (OCR).

L'individuazione del DOT è da considerare come un primo step per le future analisi. Ogni pneumatico avrà sulla propria spalla un singolo DOT, e proprio questa caratteristica verrà sfruttata per dar vita ad un sistema di coordinate che consideri il DOT come il punto di origine per ogni pneumatico.

Indice

Capitolo 1

Introduzione

Il lavoro di tesi si concentra nel investigare per individuare e decodificare scritte sulla spalla laterale di pneumatici, per riuscire ad estrapolare caratteristiche uniche dello stesso, nel nostro caso ci soffermeremo sul codice di 3 caratteri "DOT".

Il DOT copre un ruolo centrale in questo studio. Il DOT è un codice presente una sola volta su entrambi i lati di ogni pneumatico, quindi è perfetto come punto di partenza per riuscire a costruire un sistema di coordinate, con la quale poter mappare la totalità delle scritte presenti sulla spalla. Quindi, l'idea alla base è riuscire ad individuare un punto sempre presente sullo pneumatico, per impostarlo come riferimento centrale e successivamente riuscire a mappare le restanti sigle avendo un approccio generale e applicabile a qualsiasi pneumatico. Da ricerche approfondite, si è scoperto che ogni pneumatico di moderna produzione deve aver presente il codice DOT, ovvero un insieme ordinato di stringhe alfanumeriche che identificano caratteristiche chiavi dello pneumatico, su entrambi i lati.

La tesi, nello specifico, si concentra sulla progettazione e sviluppo di un sistema di visione e analisi di immagini, per poi verificarne l'applicabilità e l'accuratezza di un sistema di visione e analisi di immagini. Il sistema utilizzerà un profilometro laser ad alta risoluzione pensato per scansionare le superfici laterali dello pneumatico, e vari script python per la successiva analisi ed elaborazione delle acquisizioni.

Le varie analisi ed elaborazioni sulla scansione dello pneumatico, hanno l'obbiettivo di migliorare alcuni aspetti dell'immagine (es. la nitidezza dei caratteri, rimuovere rumore) per avere una immagine binaria che possa essere data in input al modello OCR.

Il progetto di tesi unisce tecnologie di visione industriale e analisi di immagini con una parte di OCR (Optical Character Recognition). Le prime consentono l'ispezione, la misurazione di prodotti o l'estrazione di particolari dettagli e caratteristiche fondamentali su oggetti attraverso camere. La parte di analisi di immagini vuole estrarre una gamma di caratteristiche identificative da un'immagine digitale. In fine l'OCR si occupa di rilevare e decifrare caratteri presenti in immagini restituendo il testo corrispondente.

Il presente lavoro di tesi verrà cosi strutturato: nella prima parte saranno introdotti i principi teorici riguardanti le tecnologie utilizzate, concentrandoci su aspetti generali che aiutino ad

introdurre i passaggi successivi e a giustificare le scelte fatte in fase di sviluppo e test. Nella seconda parte, verranno presentate nel dettaglio come le specifiche tecnologie studiate sono state impiegate all'interno del progetto. In fine saranno riportati alcuni risultati ottenuti durante i nostri test.

Nello specifico:

Sistema di visione industriale , verrà mostrato in cosa consiste un sistema di visione, come lavora e quali sono i suoi punti di forza, considerando la sua applicazione in un impianto industriale. In fine viene presentata come può essere integrato con tecniche di intelligenza artificiale.

Analisi immagini , è riportata una panoramica di cosa di intende con analisi di immagini, partendo dal concetto di immagine digitale e come questa può essere analizzata, passando dalle fasi di acquisizione, elaborazione e studio dei risultati.

OCR , viene presentato l'OCR come software intelligente nel comprendere porzioni di testo in immagini digitali ed inoltre, sono presentate due tecnologie OCR come Tesseract e EasyOCR.

1.1 Sistemi di visione

I sistemi di visione sono formati da uno o più dispositivi, che si occupano dell'acquisizione di dati (immagini) e da un dispositivo in grado di ricevere e analizzare questi dati, in genere un PC, mediante algoritmi opportunamente studiati.

Il cuore del sistema di visione artificiale è dunque costituito da una o più telecamere e da sorgenti di luce (illuminatori) opportunamente scelte. A cui si aggiungono componenti meccaniche e un sistema intelligente per il processing delle immagini, rappresentato dagli algoritmi di visione, indispensabile per poter comunicare con altri sistemi ma anche con gli operatori.

Il sistema ottico forma un'immagine sul sensore della telecamera che produce un segnale elettrico in uscita. Questo segnale verrà digitalizzato e memorizzato da una apposita scheda di acquisizione dati chiamata "frame grabber". L'immagine catturata, e resa in questo modo "comprensibile" da un calcolatore, deve contenere le informazioni richieste per assolvere alla funzione a cui è destinato il sistema di visione. Questa verrà quindi elaborata con un apposito software, che comprende particolari algoritmi di calcolo e di analisi, per filtrare il rumore di sottofondo, correggere gli effetti degli errori di distorsione o eventuali riflessi, in modo che sia possibile individuare le caratteristiche dell'immagine ed amplificarne alcuni aspetti, come ad esempio i bordi o il contorno, allo scopo di eseguire controlli, verifiche e misurazioni per i quali il sistema è stato concepito.

Sulla base dei risultati dell'elaborazione, il sistema prenderà decisioni in merito alla destinazione dell'oggetto scansionato, ad esempio smistarlo fra i "buoni" o scartarlo, e fornirà le informazioni opportune al resto del sistema produttivo per il controllo delle retroazioni.

Possiamo schematizzare i componenti costitutivi di un sistema di visione, come segue (18):

1. **Componenti meccanici** Componenti meccanici per il supporto e la movimentazione. Consistono spesso, in sistemi di movimentazione automatica per il posizionamento degli oggetti da ispezionare di fronte a camere e illuminatori in modo appropriato, cosi da evidenziare il più possibile le caratteristiche dell'oggetto.

2. **Illuminazione e ottiche** Progettazione e messa a punto della illuminazione e scelta delle ottiche. Tiene conto delle esigenze e delle problematiche del caso, implica che ci sia una approfondita analisi dei requisiti e solo a seguire l'effettiva scelta e messa a punto di illuminatori e ottiche.

3. **Scelta delle camere da impiegare** Come nel caso di illuminazione e ottiche, anche la scelta delle camere è fortemente influenzata dai requisiti da rispettare.

4. **Analisi ed elaborazione delle immagini acquisite** Consiste nella parte intelligente del sistema, attraverso tecniche di computer vision riesce ad estrarre caratteristiche utili.

5. **Interpretazione dei risultati ottenuti** In conclusione, vengono interpretate le caratteristiche ottenute.

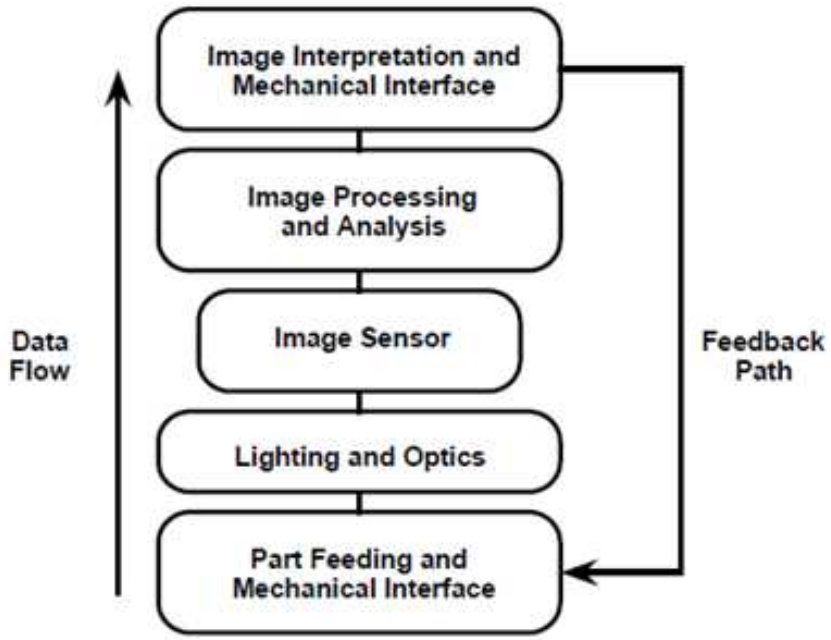


Figura 1.1: Sintesi workflow di un sistema di visione (18)

Dunque, sistemi del genere vengono progettati e realizzati con l'obiettivo di compiere analisi su immagini acquisite digitalmente e garantiscono importanti vantaggi, quali:

Costanza, affidabilità, oggettività dei controlli Consentono di eseguire un controllo su tutti i pezzi e mantengono nel tempo la costanza dei criteri di valutazione senza alcuna variazione delle prestazioni anche al variare, entro un certo range, delle condizioni operative, come ad esempio l'illuminazione. Garantiscono inoltre una omogeneità nel giudizio. Tutti questi aspetti sarebbero irraggiungibili da operatori umani.

Operabilità in ambienti ostili Operano in condizioni ambientali limite come ambienti molto rumorosi, esposti ad agenti chimici, temperature molto elevate o molto fredde, aree nel raggio di azione di macchine o sistemi di movimentazione, spazi ristretti.

Elevate velocità di controllo Sono in grado di svolgere operazioni di verifica in frazioni di secondo anche su oggetti in movimento molto veloce. La velocità di controllo genera due importanti opportunità fra loro correlate:

- Possibilità di ripensare il sistema di controllo qualità, passando da verifiche a campione a controlli totali (100% dei prodotti) con tutti i vantaggi che ne conseguono.
- Possibilità di ridisegnare i processi e le linee di produzione, introducendo sistemi di movimentazione automatici ed evitando problematiche intermedie introdotte precedentemente dai controlli qualità e dal trasferimento dei pezzi da una fase all'altra.

Controllo su oggetti di piccole dimensioni o non accessibili per altri strumenti Consentono di analizzare particolari non visibili o difficilmente identificabili dall'uomo grazie a ottiche e software specifici. Inoltre, consentono il controllo su parti non accessibili agli strumenti di misura a contatto.

Elevata precisione del controllo Consentono, anche in presenza di oggetti caratterizzati da tolleranze molto ristrette, di raggiungere una precisione ed un'accuratezza del controllo di gran lunga superiore a quella di molti strumenti di misura a contatto.

Generazione di dati sul processo Oltre ad assolvere ai compiti di controllo, sono in grado di generare e memorizzare dati sul processo in tempo reale, evidenziando scostamenti dai parametri ottimali. Questo consente di individuare eventuali segnali o trend di peggioramento del processo, fornendo così i dati necessari per intraprendere azioni correttive in modo reattivo e dinamico.

1.2 Progettazione di un sistema di visione industriale

La progettazione di sistemi di visione industriale richiede la conoscenza di una varia gamma di discipline. Queste includono progettazione di hardware e software, applicazione di ottica, illuminazione e la progettazione di sistemi meccanici.

La progettazione e successivo deploy di un sistema di visione artificiale non deve ostacolare il funzionamento complessivo della linea di produzione. L'introduzione di un sistema di visione artificiale in un processo di produzione, senza considerare appieno tutte le implicazioni, si tradurrà in false aspettative sulle capacità del sistema. Prima della progettazione e del successivo deploy di un sistema di visione, potrebbe essere utile rispondere ad alcune domande:

- L'inclusione del sistema di visione artificiale come influenzerà la velocità di produzione?
- Il processo di produzione dovrà essere modificato per accogliere l'introduzione del sistema di visione?

- La linea di produzione dovrà essere dotata di un sistema di visione automatizzato oppure l'integratore di visione avrà il controllo totale sull'ambiente di ispezione?
- Il sistema di visione richiederà modifiche personalizzate all'impianto, al processo e/o all'ambiente?
- Man mano che le esigenze di produzione cambiano, è possibile riconfigurare facilmente il sistema di visione?

La loro progettazione risulta essere estremamente personalizzabile, in quanto la scelta dei dispositivi è strettamente legata alle esigenze di produzione e ai requisiti di analisi. Quindi sono molteplici le scelte legate ai dispositivi per acquisizione ed illuminazione. In base al tipo di analisi l'acquisizione può infatti essere svolta con molteplici strumenti.

Il sistema di visione può essere realizzato con architetture hardware e software anche molto diverse fra loro in funzione delle singole esigenze applicative, della necessità di disporre di potenze di calcolo più o meno grandi e di altre caratteristiche richieste al sistema.

Per quanto riguarda la telecamera, questo è il dispositivo optoelettronico che comprende ottica, sensore ed elettronica ed ha il compito di tradurre l'immagine dell'oggetto in un segnale video con certe caratteristiche.

Le varie tipologie di telecamera disponibili oggi sul mercato si distinguono per:

- Forma del sensore: lineare o matriciale.
- Risoluzione
- Sensibilità del sensore: colore, pancromatico, vicino infrarosso, ultravioletto, termico...
- Tipo di segnale in uscita: analogico o digitale
- Frequenza di acquisizione

Il numero e la dimensione fisica dei pixel determina la risoluzione effettiva della telecamera, cioè la dimensione del più piccolo particolare distinguibile. Per avere un'elevata risoluzione sono necessari dunque molti pixel, ma ridurre la dimensione dei pixel ne riduce anche la capacità di raccogliere la luce e quindi la sensibilità. Risoluzione e sensibilità sono caratteristiche correlate ed in opposizione fra loro, il compromesso migliore dipenderà dalle condizioni applicative. Le telecamere dispongono di una serie di regolazioni per ottenere immagini adatte allo scopo che ci si prefigge anche in caso di variazioni delle condizioni operative.

Le regolazioni più comuni sono il tempo di esposizione, la regolazione della messa a fuoco, e il controllo automatico del guadagno che permette di ottenere immagini con una distribuzione ottimale dei livelli di luminosità. Un altro importante controllo è il segnale di sincronismo (`trigger`), che viene utilizzato per comandare l'acquisizione dell'immagine in un istante preciso, utile nel caso di ripresa di un oggetto in movimento.

L'ottica del sistema invece, ha la funzione di focalizzare l'immagine dell'oggetto da esaminare sul sensore della telecamera. Le caratteristiche della lente influiscono profondamente sull'immagine che verrà acquisita dalla telecamera. Alcune caratteristiche fondamentali delle lenti sono:

- Lunghezza focale, da cui dipendono parametri quali le dimensioni dell'immagine e la distanza tra telecamera e oggetto da esaminare.
- Rapporto focale o apertura, dalle cui dimensioni dipendono la luminosità e la risoluzione dell'immagine ottenuta
- Profondità di campo

Una componente essenziale e spesso critica del sistema di visione è l'illuminazione, infatti questa determina il modo con cui l'immagine verrà acquisita e successivamente elaborata. Se l'illuminazione non è adeguata al tipo di analisi che si intende fare, le successive operazioni di acquisizione ed elaborazione sono destinate a fallire.

Le principali tecniche di illuminazione sono:

- Illuminazione direzionale.
- Illuminazione diffusa.
- Illuminazione coassiale o omnidirezionale.
- Illuminazione diascopica.
- Illuminazione strutturata.

Un altro aspetto fondamentale nella progettazione di un sistema di visione riguarda la sua integrazione con il processo produttivo. Tra gli aspetti che devono essere progettati ad hoc per ogni sistema i principali sono: il controllo delle fasi di movimentazione automatica per il posizionamento delle parti da esaminare nel campo visivo delle telecamere, la sincronizzazione temporale tra il processo ed il sistema di visione, i comandi alle fasi successive del processo sulla base dei controlli effettuati (ad esempio al sistema di smistamento per l'accettazione o lo scarto posto a valle del sistema di visione), la gestione degli allarmi e delle correzioni da effettuare nel caso in cui il sistema di monitoraggio rilevi uno scostamento del processo dai parametri normali di funzionamento e la trasmissione dei dati relativi al controllo o l'elaborazione e la trasmissione periodica dei dati statistici.

Un sistema di visione artificiale è dunque un mezzo potente che sta diventando sempre più essenziale all'interno delle aziende. Garantisce precisione, affidabilità e stabilità; grazie alle elevate velocità raggiungibili automatizzando i processi, la produttività aumenta. La produzione inoltre è maggiormente controllata: l'analisi eseguita è oggettiva e interamente tracciata e documentata. La tipologia di dati che si possono raccogliere è molto ampia e garantisce una visione ancora più ampia sul processo, e non più limitata al singolo prodotto.

Queste caratteristiche rendono i sistemi di controllo visivo perfettamente compatibili con i requisiti dell'Industria 4.0: non sono semplici selettori di buono/scarto ma sono sistemi in grado di comunicare e interagire con altri dispositivi, oltre a essere una fonte preziosissima di dati, da cui trarre informazioni essenziali al fine di individuare le parti critiche di un processo produttivo. Consentono infine di intercettare eventuali derive prima che possano portare alla produzione di pezzi difettosi o al danneggiamento di macchinari.

1.3 Immagine

Un'immagine è una metodica di rappresentazione secondo coordinate spaziali indipendenti di un oggetto o di una scena. Contiene informazioni descrittive riferite all'oggetto e alla scena che rappresenta; pertanto, l'immagine è una distribuzione (che può essere bi o tri-dimensionale) di un'entità fisica.

Le immagini sono generate dalla combinazione di una sorgente di energia (elettromagnetica o ultrasuoni) e dalla riflessione dell'energia emessa dalla sorgente da parte di oggetti in una scena.

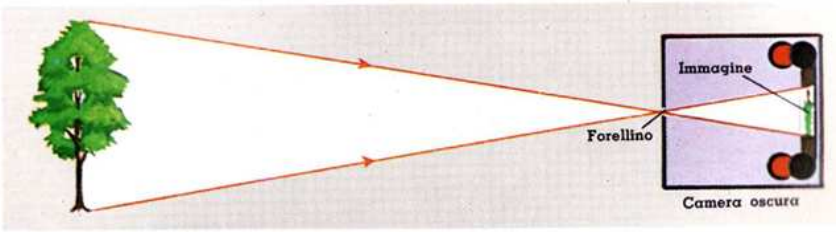


Figura 1.2: Principio per l'acquisizione di immagini (27)

Digitalizzazione

Possiamo distinguere immagini come Analogiche e Digitali; le prime sono definite da valori continui, ad esempio ottenute tramite macchina fotografica. Le seconde sono costituite da valori discreti.

Il passaggio da un'immagine Analogica ad una Digitale è chiamata Digitalizzazione. Il processo è una conversione che applicato alla misurazione di un fenomeno naturale o fisico ne determina il passaggio dal campo dei valori continui a quello dei valori discreti, quindi dalla discretizzazione dei valori continui di una foto analogica riusciamo ad ottenere un'immagine digitale.

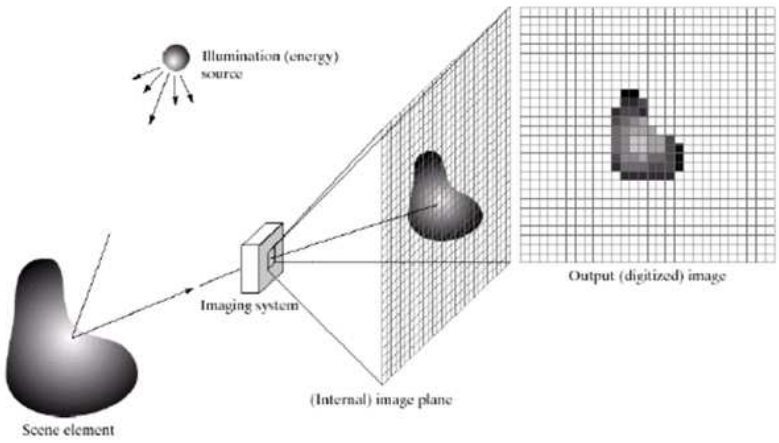


Figura 1.3: Principio di digitalizzazione immagini (27)

Dalla descrizione della digitalizzazione, possiamo introdurre la definizione puntuale di immagine digitale; un'immagine digitale è un insieme ordinato di numeri interi ognuno dei quali rappresenta

l'intensità luminosa media di un'areola corrispondente nell'immagine sorgente, detta pixel (PICture Element).

Il pixel rappresenta il più piccolo elemento dell'immagine, è caratterizzato da una propria posizione all'interno dell'immagine (orizzontale e verticale), e da un valore di intensità colore, variabile in funzione del sistema di rappresentazione adottato (scala di grigi, RGB o HSV ecc) (17).

Profondità di colore

Con risoluzione indichiamo la capacità di un sistema di digitalizzazione di eseguire misurazioni più o meno fini, nel nostro caso parleremo di profondità del colore.

La profondità di colore per un'immagine, rappresenta la minima differenza di luminosità rilevabile dal sensore ottico ovvero, trasposto in bit, è la quantità minima di informazione che serve a discernere tra diverse possibili situazioni, rappresenta il numero di colori utilizzati nella rappresentazione dell'immagine stessa.

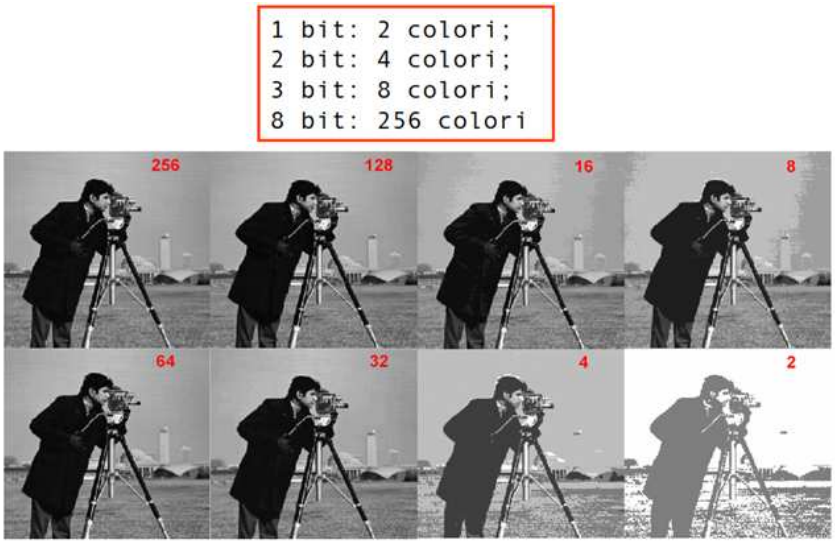


Figura 1.4: Esempio di come cambia una stessa immagine al variare della profondità di colore (27)

La risoluzione geometrica è legata all'ampiezza delle areole, minori sono i passi di campionamento (durante la discretizzazione) dX e dY, maggiore è la risoluzione geometrica del dispositivo. Essa viene misurata in punti per pollice o DPI (dots per inch) o numero di pixel orizzontali per numero pixel verticali.

Le immagini digitali possono essere memorizzate attraverso diverse tipologie di file che sfruttano differenti algoritmi di compressione. I formati più comuni sono quelli non compressi, con compressione lossless e con compressione lossy.

Caratteristiche formati non compressi (.raw o .bmp):

1. hanno richieste di elaborazione minima, non essendo necessari algoritmi di compressione (in fase di scrittura) e decompressione (in fase di lettura).

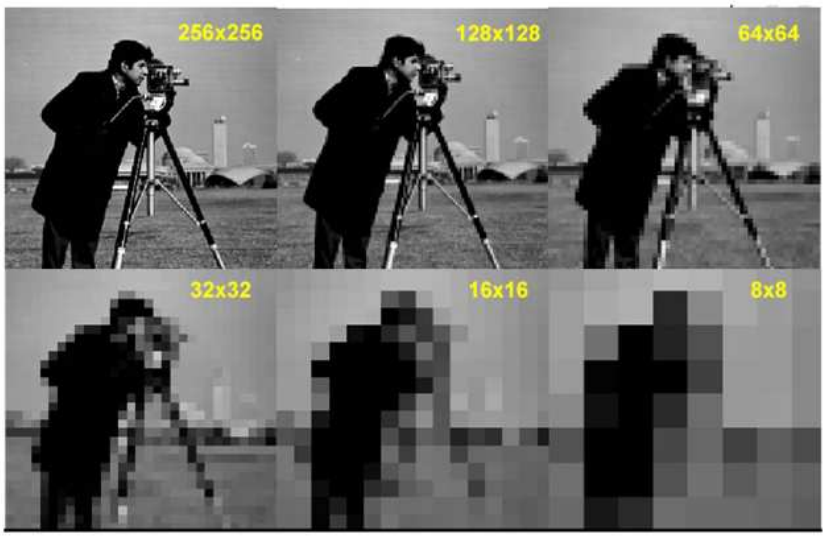


Figura 1.5: Esempio del variare della risoluzione geometrica (27)

2. Risultano particolarmente voluminosi

Caratteristiche formati compressi lossless (.png o .tiff):

1. Le immagini salvate con un algoritmo di compressione dati lossless occupano meno spazio nei dispositivi di memorizzazione.
2. Mantenendo inalterata tutta l'informazione originale.

Caratteristiche formati compressi lossy (.jpeg):

- Anche qui le immagini occupano meno spazio nei dispositivi di memorizzazione.
- La compressione introduce una perdita di informazione.
- Sono sconsigliate per l'utilizzo con software di fotoritocco, successive modifiche porterebbero ad un progressivo degrado dell'immagine.
- Sono particolarmente indicate per la trasmissione di immagini, data la ridotta dimensione del file.

1.4 Analisi di immagini

Analisi di immagini è un termine generale con il quale si indicano tutte quelle procedure o tecniche che vengono usate per interpretare e parametrizzare informazioni da immagini o gruppi di immagini. La scelta delle tecniche di analisi dipende fortemente dalla natura delle osservazioni e dal tipo di domande a cui si vuole dare una risposta.

Una prima analisi è la visualizzazione. È un'analisi importante per immagini semplici, con rappresentazioni correttamente orientate, colorate o in scala di grigi. Invece per immagini

complicate non basta la sola visualizzazione ma bisogna usare algoritmi appositamente studiati per estrarre caratteristiche significative.

1. Acquisizione
2. Elaborazione o pre-processing
3. Analisi e misuraione dell'immagine
4. Interpretazione risultati

1.4.1 Acquisizione

L'acquisizione di un'immagine è un processo attraverso il quale si ottiene una rappresentazione digitale di una scena, Questa rappresentazione è nota come immagine digitale e suoi elementi sono chiamati pixel. E' necessario un dispositivo elettronico fotosensibile capace di catturare la scena inquadrata, i sensori più diffusi sono il CCD (dispositivo ad accoppiamento di carica) o il CMOS (Complementary Metal-Oxide-Semiconductor). Il principio di funzionamento resta semplice: la lunghezza d'onda della luce viene catturata dai sensori e in base alla quantità di luce incidente riesce ad acquisire una carica maggiore o minore.

Le tecnologie sopra menzionate, CCD e CMOS, conferiscono caratteristiche uniche, che definiscono il tipo di dati che una telecamera può fornire con un certo grado di robustezza. Esistono differenze fondamentali nei tipi di prestazioni offerte dai diversi sensori. Negli ultimi anni, la tecnologia CMOS ha sovraperformato i CCD nella maggior parte delle applicazioni di visione.

Quando si seleziona un sensore (per una fotocamera), la tecnologia CCD introduce meno rumore e produce immagini di qualità superiore, soprattutto nelle scene con cattiva illuminazione ed ha una profondità di colore superiore. D'altro canto, i sensori CMOS sono più veloce nell'elaborazione delle immagini, inoltre a causa dell'architettura hardware necessitano di meno energia elettrica per funzionare, consentono l'elaborazione di una regione di interesse sul dispositivo e sono più economici dei CCD.

1.4.2 Elaborazione

L'elaborazione di un'immagine è un aspetto importante dell'analisi di immagini. Lo scopo della elaborazione delle immagini è migliorare il contrasto ed eliminare il rumore per evidenziare gli oggetti di interesse. Questo processo può essere estremamente utile per migliorare la qualità dell'estrazione delle caratteristiche e l'analisi dell'immagine a valle.

La elaborazione può includere operazioni semplici come il ritaglio dell'immagine, miglioramento del contrasto o altro, in modo significativo operazioni più complesse come la riscalatura. La qualità e il tipo di immagine sono fondamentale per selezionare un tipo di procedura di elaborazione efficiente. Tuttavia, la elaborazione è un passaggio fondamentale che può migliorare l'analisi dell'immagine e talvolta renderla possibile.

Operazione di modifica o editing

Sono operazioni molto comuni in ambito grafico, rivolte principalmente a modificare la visualizzazione dell'immagine nel suo complesso. Le più comuni sono la selezioni di regioni d'interesse, la riscalatura delle immagini e la loro rotazione.

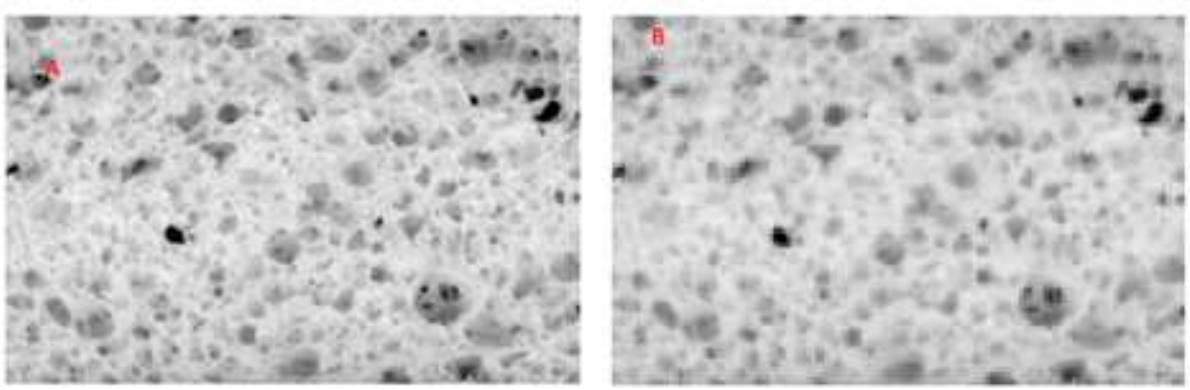


Figura 1.6: Differenza di scalatura tra le due immagini. Riscalatura da 150 dpi in A alla B con 60 dpi (27)

Operazioni di elaborazione a punto singolo

In questo caso i nuovi valori dei parametri cromatici (livello di grigio, RGB, HSV, ecc.) dei singoli pixel non dipendono dalla posizione del pixel stesso, ma solo dai valori di partenza dei parametri cromatici. Esempi comuni di operazioni di questo tipo sono le tecniche che prevedono una modifica dell'istogramma delle immagini in toni di grigio:

- **Aumento del contrasto** (contrast enanchement)
  La funzione di trasformazione è molto semplice: introduce una corrispondenza lineare tra i valori di luminosità dell'immagine prima e dopo la trasformazione.

  In figura 1.7 si può notare l'effetto dell'aumento del contrasto. Nel riquadro vi è graficato l'effetto della trasformazione dei valori di intensità: in ascissa i valori originali e in ordinata i nuovi valori.

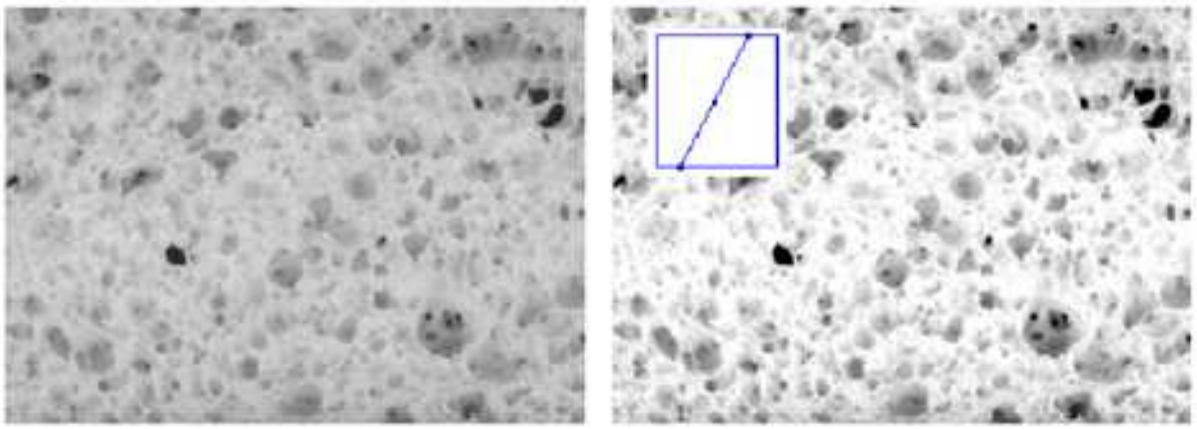


Figura 1.7: Esempio di incremento del contrasto (27)

- **Equalizzazione**
  Ridistribuisce i valori d'intensità lungo la loro scala, in base alla loro frequenza. Criteri per la ridistibuzione, a seconda della regione del range d'intensità che si vuole mettere maggiormente in evidenza: best fit, lineare, a campana, logaritmico, esponenziale

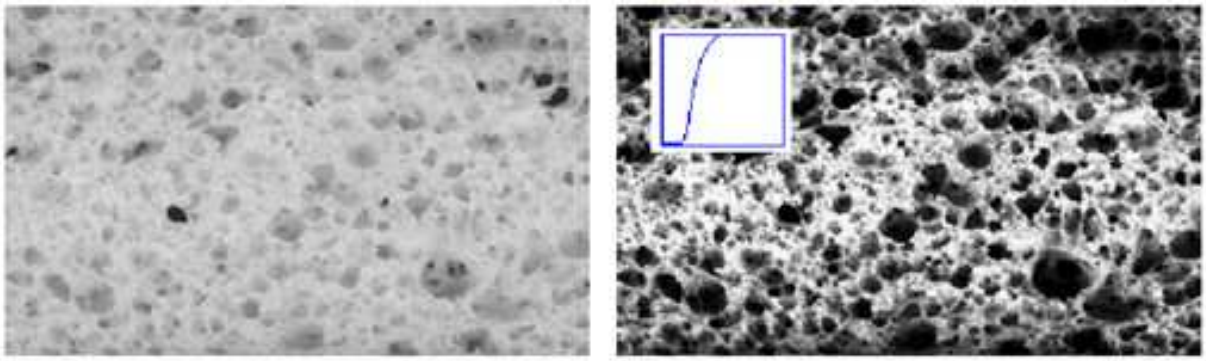


Figura 1.8: esempio di equalizzazione con best fit (27)

- **Binarizzazione**
  Tecnica per realizzare la segmentazione di un'immagine, cioè la suddivisione dell'immagine in sfondo e gli oggetti d'interesse. Ai valori d'intensità al di sotto di una certa soglia (threshold) viene attribuito il valore 0 (nero), ai restanti il valore 255 (bianco).

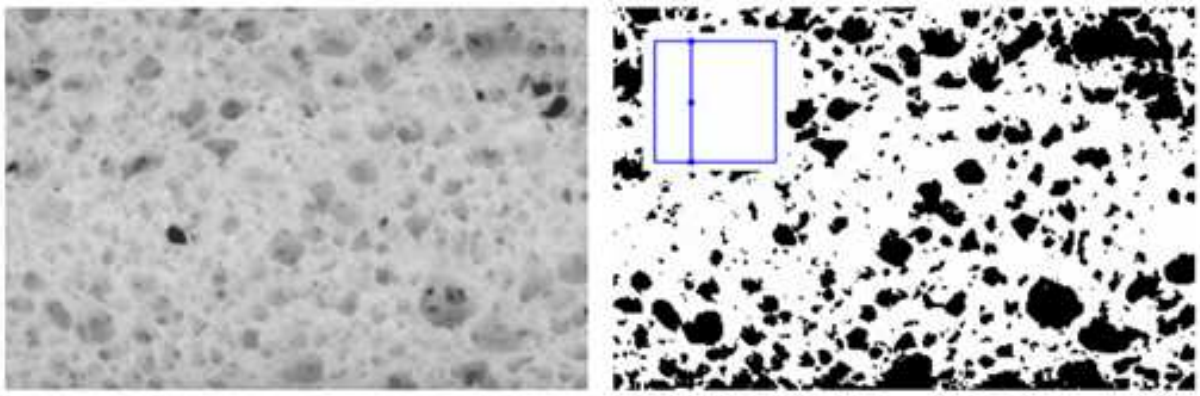


Figura 1.9: Esempio di binarizzazione con soglia a 75 (27)

Operazioni di trasformazioni spaziali

In questo caso, la trasformazione dei pixel avviene manipolando direttamente i pixel stessi, mediante l'uso di maschere o filtri spaziali.

La maschera è una matrice di pixel nell'intorno del pixel considerato, i valori d'intensità dei pixel della maschera sono utilizzati per la determinazione della trasformazione del pixel.

Tra i filtri possiamo distinguerne lineari e non lineari; In filtri lineari il valore d'intensità di un pixel è la media ponderata dell'intensità dei pixel compresi nella maschera, dove i coefficienti per il calcolo della media sono caratteristici di ogni filtro. I filtri non lineari considerano sempre una matrice di pixel per operare la trasformazione, ma non utilizzano coefficienti. Ma

metodi statistici o formule matematiche che ricavano un nuovo valore per l'intensità del pixel considerato, a partire dall'insieme dei valori della matrice.

Filtri lineari:

- **filtro passa-basso** (lowpass) o di sfocatura (blurring): riduce il rumore presente nelle immagini, ma tende a peggiorare la definizione dei dettagli, in particolare dei bordi.
- **filtro passa-alto** (highpass): evidenza le differenze d'intensità tra i pixel, in particolare i contorni degli oggetti, riducendo gli altri dettagli a sfondo
- **filtro "sharpen":** accentua i contorni, ma aumenta il rumore nell'immagine
- **filtri di contorno orizzontale e verticale** (horizontal and vertical edge): permettono l'individuazione di bordi con un orientamento particolare.

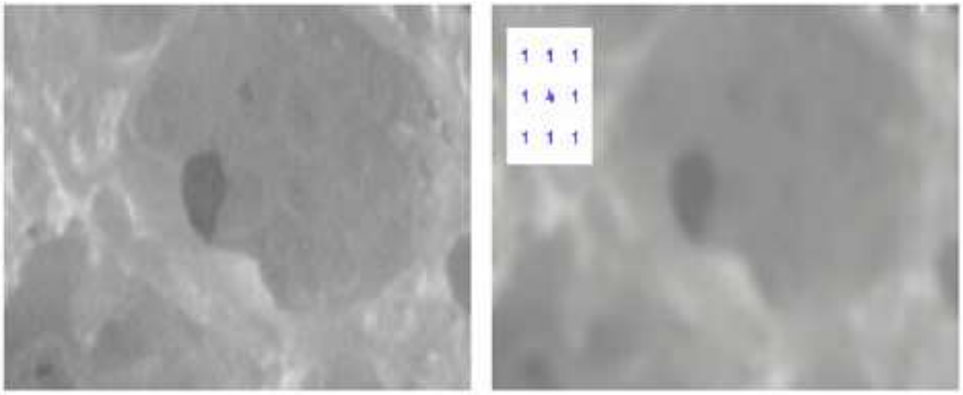


Figura 1.10: esmpio effetto sfocatura (27)

In figura 1.10 un esempio di filtro di sfocatura. A sinistra: immagine di una sezione di pane acquisita mediante microscopio ottico e videocamera analogica; A destra: applicazione di un filtro di sfuocatura 3x3 ripetuta 5 volte, nel riquadro la matrice utilizzata.

Filtri non lineari.

- **filtro mediano**: sostituisce il valore del pixel considerato con la mediana dei valori della matrice, riduce il rumore nelle immagini, ma preserva maggiormente i dettagli e i contorni del filtro di sfocatura.
- **filtri di dilatazione ed erosione**: il valore del pixel viene sostituito rispettivamente con il valore massimo e minimo della matrice, il primo espande gli oggetti presenti nell'immagine, il secondo li riduce.

In figura 1.11 un esempio di filtro mediano. A sinistra: immagine di una sezione di pane acquisita mediante microscopio ottico e videocamera analogica. A destra: applicazione di un filtro mediano 3x3 ripetuta 5 volte, si più osservare come rispetto all'esempio precedente si abbia un'analoga riduzione del rumore, ma una maggiore preservazione dei dettagli dei contorni.

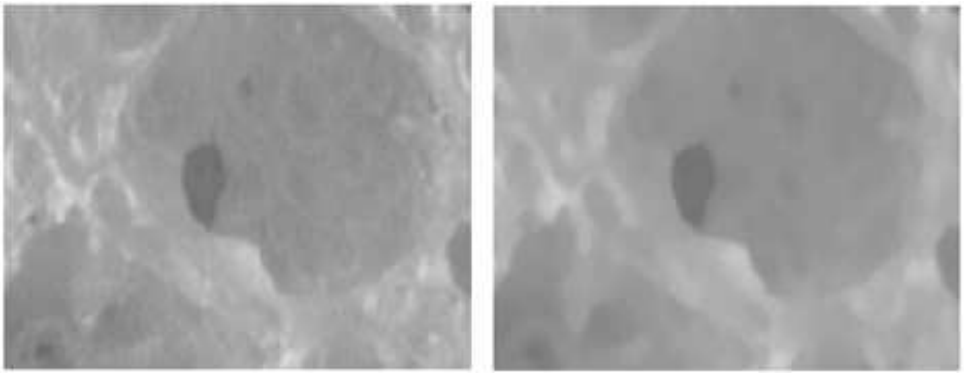


Figura 1.11: Esempio effetto di filtro mediano (27)

Operazioni aritmetiche e logiche

Queste operazioni possono essere effettuate o tra due immagini o tra un'immagine e un valore numerico costante, utilizzando operatori come: addizione, sottrazione, moltiplicazione, divisione, logaritmo, media, massimo, minimo; o operatori logici (booleani) quali AND, OR, NOT, XOR. Spesso usate per la rimozione di informazione relative allo sfondo in sequenze di immagini in movimento, la mascheratura di porzioni d'immagine, l'ottenimento della media di più immagini. In figura 1.12 un esempio di operazione di sottrazione. Sinistra: superficie di piadina acquisita

Figura 1.12: Esempio di applicazione di operazioni aritmetiche e logiche (27)

mediante uno scanner piano; Centro: Maschera di individuazione delle aree imbrunite; Destra: Risultato: Sinistra - Centro, tutti i punti per i quali il risultato dell'operazione è negativo sono stati posti uguali a zero (sfondo nero).

Filtri per trasformazioni morfologiche

Appartengono a questa categoria quei filtri per i quali il risultato della trasformazione dipende dalle proprietà di forma degli oggetti presenti nell'immagine.

Erosione Consiste nel rimuove i pixel del contorno degli oggetti, tende a isolare gli oggetti congiunti da protuberanze sottili e a rimuovere i piccoli oggetti isolati.

Dilatazione Consiste nel aggiunge nuovi pixel al contorno degli oggetti, tende ad unire oggetti vicini e a rimuovere i buchi presenti negli oggetti.

Apertura Viene effettuata mediante una o più erosioni seguite da una o più dilatazioni, rende il contorno degli oggetti più regolare, isola gli oggetti, rimuove i piccoli oggetti e i buchi.

Chiusura E' l'inverso dell'apertura, ha caratteristiche analoghe all'apertura, ma tende a separare gli oggetti.

Skeletonizzazione Rimuove ripetutamente i pixel del contorno fino a ridurre l'oggetto a uno "scheletro" largo un pixel.

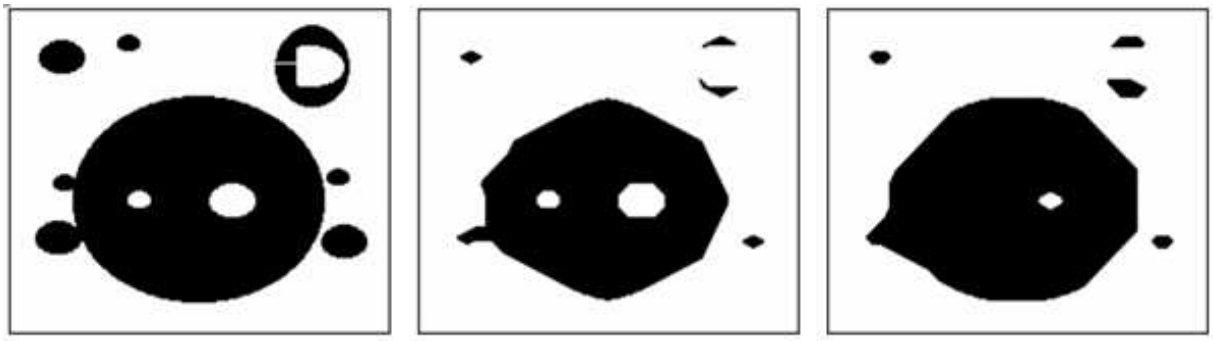


Figura 1.13: Esempio di applicazione di trasformazioni morfologiche (27)

In figrua 1.13 sinistra immagine di Input; Centro: erosione dell'immagine di input; Destra: apertura sull'immagine di input In figura 1.14 sinistra: Dilatazione dell'immagine di input

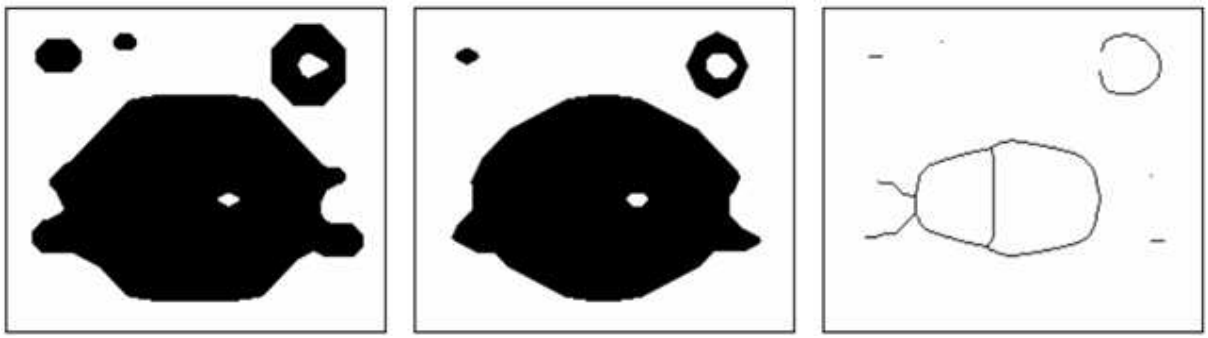


Figura 1.14: Esempio di applicazione di trasformazioni morfologiche (27)

precedente; Centro: Chiusura dell'immagine di input precedente; Destra: Skeletonozzazione dell'immagine di input precedente.

Operazioni di segmentazione

Tale operazione permette di isolare gli oggetti da analizzare. Esistono due approcci fondamentali al problema della segmentazione: quello per discontinuità e quello per similarità.

- L'approccio per discontinuità cerca di individuare punti isolati, linee, bordi e quindi di ricostruire i contorni degli oggetti da isolare.

- L'approccio per similarità viene applicato nel caso della segmentazione per imposizione di una soglia d'intensità (thresholding) e dei metodi basati sulla crescita, sulla separazione (scissione) e sulla fusione (merging) di regioni.

A seguire un esempio di segmentazione.

Figura 1.15: Pagnotta di pane in cottura (27)

In figura 1.15 è riportata una pagnotta di pane durante la sua cottura, invece in 1.16 è presente l'istogramma realtivo alla distribuzione dei livelli di intensità dopo aver convertito l'immagine 1.15 in scala di grigi. Volendo applicare una segmentazione facciamo riferimeto a figura 1.17,

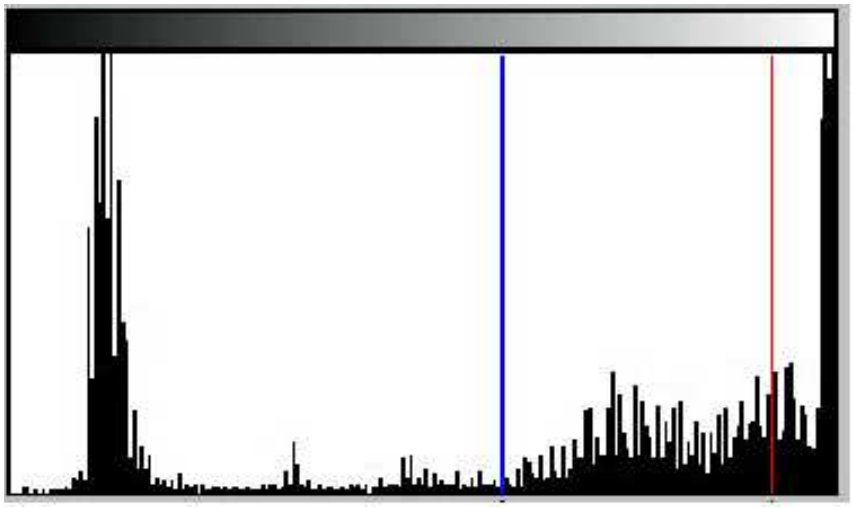


Figura 1.16: Istogramma d'intensità di figura 1.15 (27)

dove i contorni delle segmentazioni sono ottenuti applicando una soglia d'intensità di 155 (blu) e 235 (rosso), le stesse riportate nell'istogramma: la prima permette l'individuazione del bordo superiore, ma non di quello inferiore; la seconda mostra come non sia possibile individuare il bordo inferiore mediante l'applicazione di una soglia d'intensità.

Un ulteriore segmentazione è presente in figura 1.18, ottenuta mediante una procedura mista che prevede l'uso di filtri di individuazione dei bordi e della tecnica di crescita delle regioni, si può notare la correttezza della segmentazione.

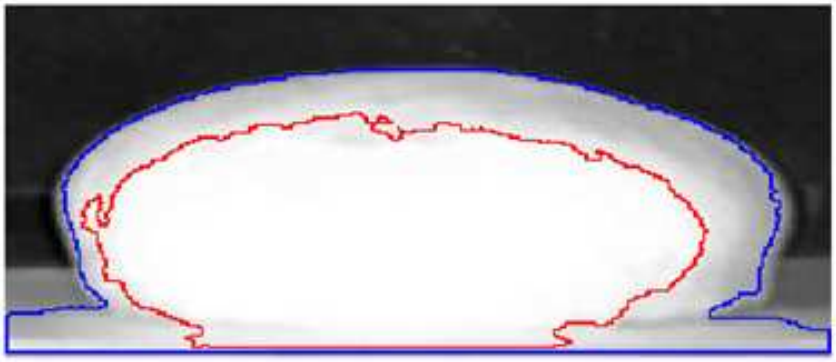


Figura 1.17: Esempio di segmentazione su doppia soglia (27)

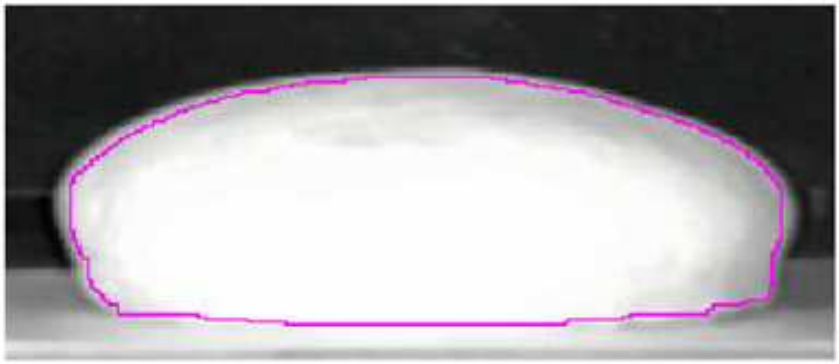


Figura 1.18: Esempio di segmentazione singola (27)

1.4.3 Analisi o misurazione dell'immagine

La misura dell'immagine si esegue mediante opportuni algoritmi di calcolo matriciale (direttamente implementati nel software dedicato o previa esportazione in debito formato) che caratterizzano gli elementi isolati (perimetro, area, shape factor, dimensione minima e massima, etc.) e, successivamente, si possono ottenere informazioni statistiche sulla loro distribuzione.

Misure relative a proprietà specifiche sono i parametri dimensionali di un oggetto (area, lunghezza, larghezza, perimetro, diametro equivalente) o di forma (shape factor, aspect ratio, dimensione frattale, numero di buchi, indici di convessità e solidità), o ancora di chroma o intensità (valori R,G,B e loro trasformazione nello spazio H,S,V o gray scale).

Misure globali dell'immagine sono la determinazione di frazioni di superficie, la numerazione degli oggetti, la determinazione di lunghezza e curvatura delle linee e di vari tipi di parametri di gradiente o di orientamento.

1.4.4 Interpretazione risultati

Una volta conclusa la fase di elaborazione, ovvero dopo avere eseguire ripetute misurazioni, si passa alla loro valutazione statistica e l'idonea rappresentazione dei risultati, ovvero la fase di Interpretazione. La capacità di gestire e saper interpretare al meglio le caratteristiche estratte, comporta ottenere risultati utili e conformi al problema che si sta affrontando. Inoltre, la scelta delle caratteristiche che verranno estratte dall'immagine sarà fortemente influenzata dal tipo di

analisi che si sta trattando.

1.5 OCR optical character recognition

OCR è l'acronimo per Optical Character Recognition; in generale si considera come un software che consente automaticamente di riconoscere caratteri o scritte da immagini, traducendole in testo con codifica ASCII o UNICODE.

L'OCR è in grado di riconoscere sia il testo scritto a mano che stampato, ma le prestazioni dipendono direttamente dalla qualità dei documenti in ingresso.

L'OCR è progettato per elaborare immagini che sono costituite quasi interamente da testo, con pochissimo rumore, partendo da immagini che si possono ottenere da camere o scansioni di documenti.

Quindi è possibile convertire diversi tipi di documenti, come documenti cartacei digitalizzati, file PDF o immagini catturate da una fotocamera digitale in dati testuali modificabili e ricercabili. Spesso, le immagini di input contengono difetti come la distorsione ai bordi e luce attenuata, che rendono difficile per la maggior parte delle applicazioni OCR riconoscere correttamente il testo.

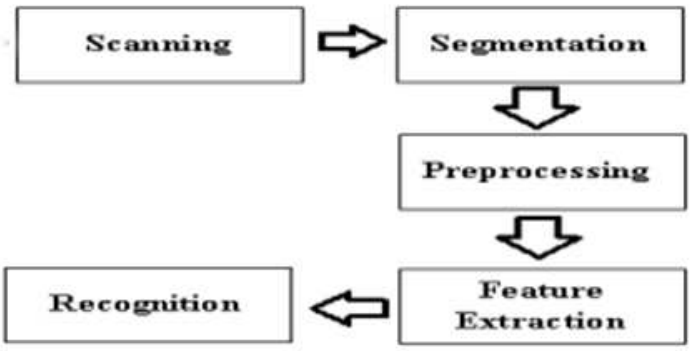


Figura 1.19: Sintesi della struttura di un OCR (22)

Per eseguire il riconoscimento dei caratteri, l'applicazione deve passare attraverso cinque task (22):

Scansione o acquisizione immagine Si può dire che la scansione consiste nel trasformare il documento originale in un'immagine in scala di grigi. Inoltre, si può valutare l'utilizzo di una fotocamera per'acquisizione di un'immagine contenente testo da codificare.

E' necessario un'immagine in scala di grigio, per facilitare l'operazione di thresholding. Ovvero, un processo con la quale è facile ottenere un'immagine binaria. Per l'OCR è utile avere dei caratteri neri su sfondo bianco.

Segmentazione La segmentazione è il processo di individuazione delle regioni di interesse rispetto al resto della'immagine. Nel caso di OCR si tratta di estrapolare zone contenti testo, isolandole dal resto della'immagine.

Il problema principale della segmentazione è la confusione che può crearsi nel individuare parole nel caso di caratteri uniti e non divisi. Di solito, le spaccature e le giunzioni nei caratteri sono conseguenza di una scansione e acquisizione non corretta.

Preprocessing La presenza di rumore e disturbi nella'immagine potrebbe compromettere il corretto riconoscimento dei caratteri. Quindi, il preprocessing consiste in operazioni di elaborazione dell'immagine per gestire disturbi e rumori cosi da migliorare l'accuratezza del'OCR.

Estrazioni features Consiste nel'estrapolare caratteristiche importanti dalle scritte. Ogni scritta individuata è caratterizzata da specifici attributi o caratteristiche, quindi vengono estratte per facilitare la rappresentazione.

Recognition o classificazione Riesce nel riconoscimento dei caratteri e parole. Quindi, provvede alla conversione concreta di scritte presenti in immagini ai flussi di testo corrispondente.

1.5.1 Tesseract

Tesseract è un OCR engine open source. È stato inizialmente sviluppato tra il 1984 e il 1994 presso HP. Nel 1995 è stato inviato a UNLV per il test annuale sulla precisione dell'OCR dopo il progetto congiunto tra gli HP Labs di Bristol e la divisione scanner di HP in Colorado. Infine, nel 2005, è stato rilasciato come open soources da HP.

Per lo studio del funzionamento di Tesseract, facciamo riferimento alla documentazione ufficiale rilasciata da Ray Smith (26).

Tesseract presuppone che il suo input sia un'immagine binaria (nero e bianco) con regioni di testo ben definite. Tesseract è in grado di gestire sia il tradizionale testo nero su bianco che situazione inversa, ovvero bianco su nero.

L'elaborazione segue una tradizionale pipeline di un OCR. Il primo passo è un'analisi delle componenti connesse in cui vengono memorizzati i contorni delle componenti. Attraverso lo studio dei contorni è semplice individuare il testo bianco su sfondo nero con la stessa facilità del testo nero su bianco.

Tesseract è stato probabilmente il primo motore OCR in grado di gestire così banalmente il testo bianco su nero.

L'elaborazione inizia con l'analisi dei contorni, creando una gerarchia tra essi individuando elementi caratteristici chiamati blob. I blob non sono altro che "macchie" identificate dai contorni estratti dagli elementi testuali all'interno dell'immagine.

Successivamente i blob saranno organizzati in linee di testo e poi analizzate considerando lo spazio fisso e le proporzioni presenti tra caratteri o parole. Le linee di testo vengono suddivise in parole in maniere differente a seconda del tipo di spaziatura tra i caratteri. Il testo verrà tagliato in celle contenti singoli caratteri.

Il riconoscimento procede quindi come un processo in due passaggi. Nel primo passaggio si tenta di riconoscere ciascuna parola. Ogni parola riconosciuta in maniera soddisfacente viene passata a un classificatore adattivo come dati di addestramento. Il classificatore adattivo ha quindi la possibilità di riconoscere in modo più accurato il restante testo nella immagine.

Poiché il classificatore adattivo potrebbe aver imparato qualcosa di nuovo troppo tardi per fornire un contributo migliore nella parte iniziale della immagine, viene eseguito un secondo passaggio, in cui le parole che non sono state riconosciute in precedenza verranno analizzate dal classificatore aggiornato.

Ricerca linee di testo

L'algoritmo di ricerca delle linee di testo è progettato in modo che testo inclinato possa essere riconosciuto senza dover essere corretto, evitando così la perdita di qualità nell'immagine.

Le parti fondamentali del processo sono il filtraggio dei blobs e la costruzione della linea stessa. Supponendo che l'analisi della immagine abbia già fornito aree di testo, si passa al filtraggio della dimensione del testo. Verrà eliminato un percentile sulle altezze dei caratteri cosi da rimuovere lettere maiuscole e caratteri molto sviluppati in altezza, cosi che l'altezza mediana si avvicina alla dimensione reale del testo. Quindi è possibile filtrare i blobs più piccoli di una frazione dell'altezza mediana, cosi da eliminare eventuale rumore presente.

L'ordinamento e l'elaborazione dei blob in base alla loro altezza rende possibile assegnargli una riga di testo univoca, monitorando anche la pendenza, con una notevole riduzione del rischio di non riuscire ad individuare testo inclinato.

Una volta trovate le righe di testo, viene trovata una linea di base utilizzando una spline quadratica. Le linee di base vengono adattate suddividendo i blob in gruppi con un andamento ragionevolmente continuo rispetto alla riga di testo originale.

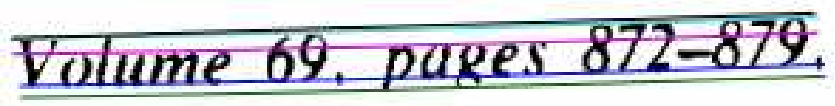


Figura 1.20: Esempio di linee di testo, in blue linea base e in rosa linea mediana (26)

Tesseract una volta individuata la baseline di una riga, passa a determinare e ad analizzare il passo fisso, cosi da dividere le varie parole. Una parola a passo fisso ha tutte le sue lettere ad una distanza fissa tra loro.

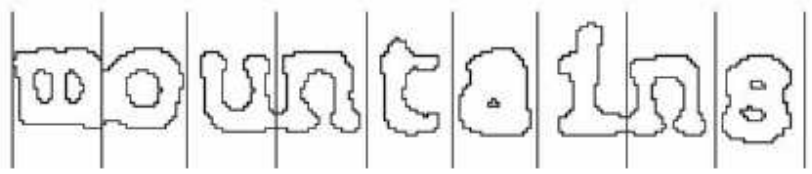


Figura 1.21: Esempio di parola con spaziatura fissa (26)

Il discorso cambia in parole con spaziature non costante, dove il passo non può essere usato come discriminante. Nella figura 1.22 la parola "financial" non ha una spaziatura costante tra i caratteri. Per risolvere questo problema Tesseract si concentra sulla regione che va dalla baseline e le linee di testo della mediana delle altezze dei caratteri, successivamente sfoca il tutto così da individuare facilmente i blob che si sfiorano e trovare le singole parole.

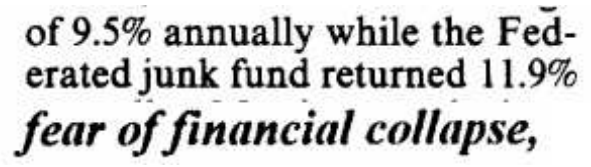


Figura 1.22: Esempio frase con spaziatura non fissa (26)

Segmentazione dei caratteri

Una delle parti più importanti in qualsiasi motore di OCR consiste nella segmentazione della parola in carrettieri, ovvero individuare i carrettieri presenti all'interno della parola. L'analisi passa attraverso lo studio del punto di taglio dei contorni delle singole lettere. I punti possono essere angoli concavi o convessi tra segmenti che costituiscono i poligoni del contorno.

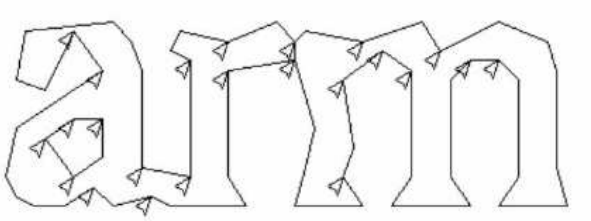


Figura 1.23: Esempio punti di taglio su caratteri (26)

Estrazione features

L'idea intrapresa per l'estrazione di features in parte deriva dal uso di segmenti appartenenti alla approssimazione poligonale attorno al carattere, comunque questo approccio avrebbe un grande problema nel gestire caratteri spezzati. Allora, la soluzione riguarda l'usare caratteristiche per la fase di inferenza (le caratteristiche del carattere incognito da usare per interpretare la classi di appartenenza) diverse da quelle per la fase di addestramento.

In fase di addestramento vengo usate features che sono segmenti estratti direttamente dal poligono del carattere, invece in classificazione le features estratte dal carattere ignoto verranno confrontate con features estratte da un prototipo normalizzato partendo dai poligoni originali.

Possiamo veder un prototipo come un "modello di prova" associato ad un carattere specifico (ad esempio A avrà il suo prototipo, cosi come B) e le features estratte dalla lettera incognita verranno confrontate con questi prototipi.

Dalla figura 1.24 si ha che in (a) abbiamo 'h' originale sul quale è stato addestrato il modello ocr e in (b) abbiamo la 'h' da riconoscere. Ad usare features puntali e identiche sia in train e inferenza,

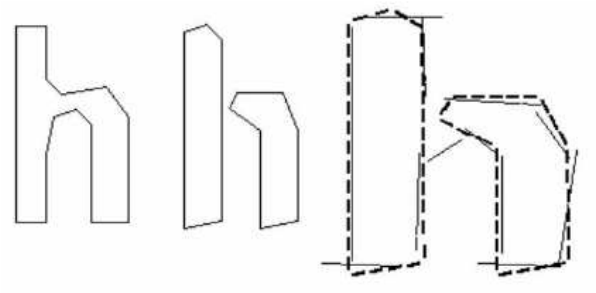


Figura 1.24: in ordine: 'h' originale (a), 'h' rotta (b), 'h' features (c) (26)

in questo caso non sarebbe possibile confrontare le caratteristiche e quindi ci troveremmo in una situazione in cui le lettere spezzate sarebbe fraintese dal modello.

Sempre dalla 1.24 le linee tratteggiate vengono estratte dal carattere non incognita e i segmenti continui sono presi dal prototipo di riferimento.

Visto che Tesseract riesce a gestire caratteri spezzati, il dataset di addestramento ha molti meno campioni di caratteri rispetto ad altri classificatori ocr.

Classificatore adattivo

Poiché un classificatore statico deve essere in grado di generalizzare a qualsiasi tipo di carattere, la sua capacità di discriminare tra caratteri diversi è nettamente minore rispetto a quella di un classificatore adattivo, dove ha maggiore sensibile, addestratosi dall'output del classificatore statico.

Un classificatore adattivo avrà una maggiore discriminazione nel decifrare testi con lo scorrere della immagine analizzata, quindi l'accuratezza del modello migliorerà a fine immagine ed ecco perché sarà necessaria una seconda analisi OCR con il classificatore aggiornato.

1.5.2 EasyOCR

EasyOCR è un'opzione affidabile per gli sviluppatori Python grazie alla sua versatilità nella gestione dei caratteri tipografici e dei layout di testo, nonché alla sua attenzione alla precisione e alla velocità. EasyOCR semplifica il processo di estrazione del testo dalle foto da utilizzare in vari progetti Python, tra cui software desktop, applicazioni online e altri.

È possibile automatizzare facilmente le operazioni relative al testo, migliorare l'estrazione dei dati dai documenti scansionati e sfruttare le capacità di riconoscimento del testo nei progetti di analisi delle immagini incorporando EasyOCR nei programmi Python. È uno strumento utile per le applicazioni di computer vision e un modo semplice per utilizzare l'OCR nei vostri progetti Python.

Per lo studio si EasyOCR abbiamo fatto riferimento alla documentazione ufficiale rilasciata da Jaided Ai(3).

EasyOCR supporta attualmente più di 80 lingue e altre sono in fase di sviluppo.

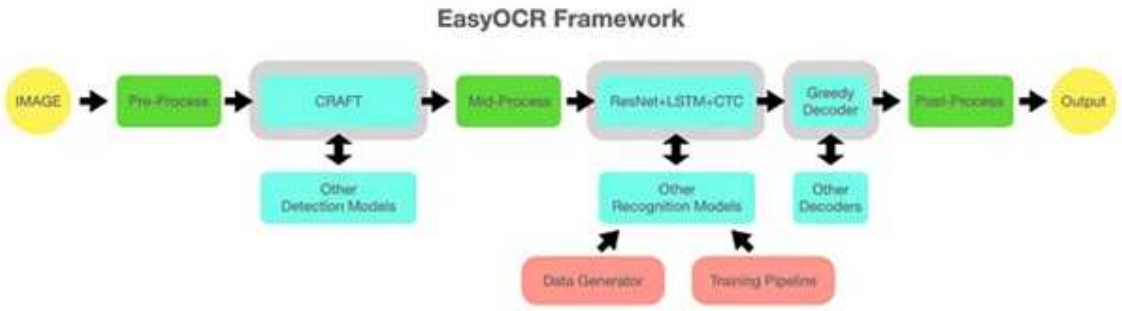


Figura 1.25: Framework del OCR EasyOCR (1)

Architettura OCR

I tre componenti principali di EasyOCR sono l'estrazione delle features, l'etichettatura delle sequenze e la decodifica. Per estrarre caratteristiche utili dall'immagine di ingresso, vengono utilizzati modelli di deep learning come ResNet e VGG.

Queste caratteristiche sono essenziali per il riconoscimento del testo nelle immagini. L'etichettatura della sequenza, la fase successiva, utilizza reti LSTM (Long Short-Term Memory) per interpretare il contesto delle caratteristiche estratte. Il riconoscimento e la strutturazione del testo sono compiti cruciali per le reti LSTM.

Infine, la parte di decodifica interpreta e trascrive le sequenze etichettate nel testo effettivamente riconosciuto, utilizzando l'algoritmo Connectionist Temporal Classification (CTC).

Questi tre elementi funzionano come un'unità per consentire a EasyOCR di estrarre in modo affidabile ed efficace il testo dalle immagini. La pipeline di addestramento si basa sul framework deep-text-recognition-benchmark, che migliora il riconoscimento del testo nelle immagini e offre una solida base per l'esecuzione del OCR.

Estrazione delle features (Resnet e VGG)

Il primo passo per usare modelli di riconoscimento è l'estrazione delle features, cosi per creare una serie di caratteristiche che possano essere utilizzate in fase di analisi.

In EasyOCR vengono utilizzate VGG e Resnet.

Resnet, nota anche come Residual Networks, è un tipo di rete neurale convoluzionale (CNN) che bypassa alcuni livelli utilizzando scorciatoie o connessioni. In questo modo, la rete può essere più profonda e ancora addestrabile, risolvendo i problemi sul gradiente. L'apprendimento basato su rappresentazioni dei residui del segnale piuttosto che della rappresentazione diretta del segnale è il principio fondamentale dell'architettura di Resnet. Questo riduce la complessità del modello e facilita il processo di apprendimento da parte della rete.

L'altra rete CNN è chiamata Visual Geometry Group, o VGG. La sua architettura omogenea e la sua semplicità sono ben note. La sua architettura consiste in filtri di convoluzione molto piccoli

(3x3) impilati sempre più in profondità l'uno sull'altro. Questo semplifica la comprensione e la modifica della rete, riducendo il numero di calcoli e parametri.

Etichettatura (LSTM)

L'etichettatura avviene dopo l'estrazione delle caratteristiche. A questo scopo si utilizzano le reti LSTM (Long Short-Term Memory).

Le reti neurali ricorrenti (RNN) di tipo long-sequence learning and memory, o LSTM, sono in grado di modellare le dipendenze temporali di più fasi temporali. Il design distinto delle LSTM, a differenza delle RNN convenzionali, aiuta a prevenire i problemi sul gradiente. Questi elementi conferiscono a LSTM una grande efficacia nei compiti di etichettatura.

Decodifica (CTC)

La decodifica è l'ultima fase del modello di riconoscimento e a tal fine si utilizza la Connectionist Temporal Classification (CTC).

Un tipo di funzione di Loss chiamata CTC che viene applicata su serie temporali in cui il tempo è incerto. Si applica in situazioni in cui non si è certi dell'allineamento tra le etichette e i dati di input, come accade spesso nel riconoscimento del parlato e della scrittura. Per fornire una predizione di lunghezza variabile, CTC aggiunge un'etichetta vuota alle etichette esistenti. È quindi perfetto per i compiti di decodifica OCR, perché calcola la perdita sommando tutti i possibili allineamenti delle sequenze di input e di output.

Una versione modificata dell'architettura del deep-text-recognition-benchmark serve come pipeline di addestramento per l'esecuzione del riconoscimento di EasyOCR. Grazie a questa struttura, i modelli di riconoscimento del testo possono essere addestrati su una varietà di set di dati, rendendo EasyOCR incredibilmente versatile ed efficace.

1.6 Pneumatici e struttura DOT

Nella produzione degli pneumatici una fase critica è la procedura di lettering. Durante questa fase di ispezione si controlla ogni volta che tutte le scritte ed i simboli impressi sul pneumatico siano corretti e ben posizionati. Per quelle normative internazionali che richiedono la presenza di simboli e scritte definite sulla superficie dello pneumatico e per la qualità percepita dal cliente del pneumatico stesso, il controllo delle scritte è una procedura cruciale nel processo di produzione del pneumatico.

Tenendo conto della scheda tecnica degli pneumatici in prova, costituita da un elenco di tutte le scritte e simboli che devono essere impressi sullo pneumatico, bisogna verifica se tutti gli elementi presenti nell'elenco sono correttamente impressi e posizionati. Ulteriori controlli possono essere effettuati, come l'analisi del deterioramento dello stampo, dove l'altezza del timbro dei caratteri

sullo pneumatico in prova viene confrontata con quella prevista. È possibile impostare una tolleranza limite per attivare un avviso quando lo stampo si sporca, si deteriora o si danneggia.

1.6.1 Sezione pneumatico

Nello specifico, per i vari test sono stati usati pneumatici stradali. Siamo consapevoli che le considerazioni fatte su pneumatici di queste dimensioni cadono nel momento che si andranno ad usare gomme di diversa grandezza. Alcuni aspetti da tenere in considerazione potrebbero essere: l'aumento in altezza della spalla da scansionare, l'incremento della dimensione della scritta DOT e il suo posizionamento all'interno della spalla laterale. Di conseguenza anche l'intero sistema di acquisizione e movimentazione dovrà essere aggiornato e reso congruo ai nuovi pneumatici. Dopo aver fatto questa breve premessa, passiamo alla descrizione della sezione di uno pneumatico.

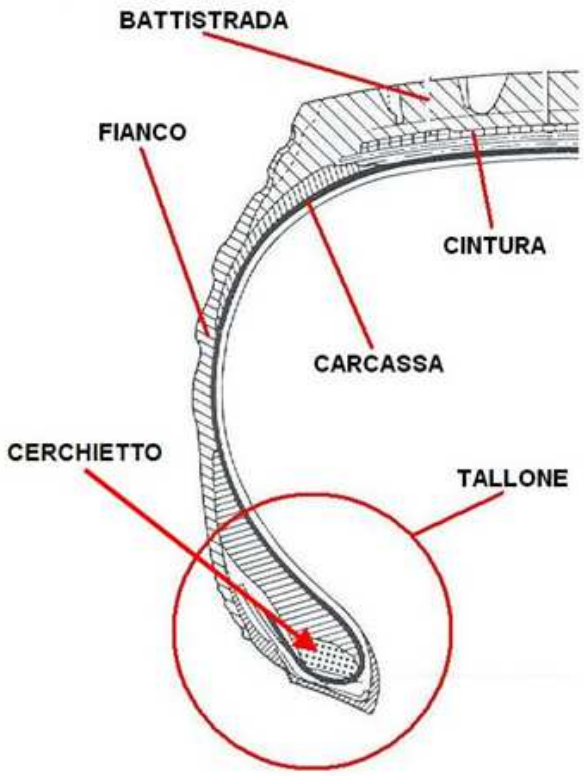


Figura 1.26: Sezione pneumatico

Come mostrato in foto 1.26 lo pneumatico è diviso in 6 porzioni (dall'alto verso il basso):

- **Battistrada** è lo strato di gomma dello pneumatico che poggia direttamente sul asfalto durante la guida. È resistente all'abrasione ed è formato da una serie di incavi, scanalature e buchi. La conformazione del battistrada ha lo scopo di incanalare l'acqua presente sulla strada e rigettarla, evitando così che le gomme scivolino facendo perdere il controllo del veicolo. Infatti, viaggiare con un battistrada usurato aumenta il rischio di acquaplaning.
- **La cintura** è una struttura a nido d'ape che è realizzata con fili di ferro. Il rinforzo fornisce resistenza al usura, al carico e alle vibrazioni.

- **La carcassa** realizzata in materiale resistente come tessuto, nylon o poliestere, è una parte cruciale dello pneumatico perché fornisce la base di resistenza e di supporto per gli altri strati. Ha la funzione di sorreggere il peso del auto e di assorbire gli urti;
- **Il fianco o spalla** costituisce la parte più esterna della gomma. E' a compreso tra lo spigolo del battistrada e l'inizio del tallone. Ha il compito di garantire la stabilità in curva e mantenere la traiettoria. È costituito da uno strato di mescola che deve proteggere lo pneumatico dagli urti laterali.

  Il fianco è la porzione che ci interessa, qui sono incise varie caratteristiche dello pneumatico, come il DOT o la tripla di parametri dimensionali (es 205/50/R17).
- **Il tallone** è uno strato di tela gommata che unisce copertura e cerchio, e conferisce alla struttura una maggiore robustezza. Evita il deterioramento dello pneumatico (che sfrega contro il cerchio) e garantisce la tenuta dell'aria in pressione.
- **Il cerchietto** è un anello metallico composto da fili d'acciaio al quale è ancorato ermeticamente lo pneumatico.

Lo pneumatico è suddiviso in porzione ben distinte, dove ognuna ha il proprio ruolo e caratteristiche specifiche. Nel nostro caso ci concentreremo sulla sola parte laterale, ovvero fianco,che è compresa tra la porzione inferiore del battistrada e parte esterna superiore del tallone. Per semplicità nell'intero documento indicheremo l'interezza della porzione in esame come Fianco o Spalla.

Ora verranno descritte sia il seriale DOT che il seriale dimensionale, sono i due codici con maggiore informazione sugli pneumatici. Sarà descritta la struttura e cosa riporta ogni singolo sotto codice.

1.6.2 DOT

Il codice DOT certifica la conformità del produttore dello pneumatico agli standard di sicurezza del National Highway Traffic Safety Administration (NHTSA) del Dipartimento dei Trasporti degli Stati Uniti (DOT). Gli pneumatici hanno il numero di serie DOT situato sulla parte bassa del fianco, in una posizione facilmente raggiungibile.

Il codice "DOT" è seguito da otto o tredici lettere e/o numeri che identificano il luogo di produzione dello pneumatico, le dimensioni dello pneumatico e il codice del produttore, oltre alla settimana e all'anno di produzione. Quindi riuscendo ad analizzare il seriale DOT è possibile estrapolare informazioni essenziali sullo pneumatico stesso. Resta una fonte informativa importante in caso di incidente, per riuscire a diagnosticare correttamente le cause del guasto.

Facendo riferimento alla figura 1.27 dettagliamo gli elementi costitutivi del DOT.

- **DOT** Il numero di identificazione di ogni pneumatico inizia con le lettere DOT (Department of Transportation). Si tratta di un indicatore che verifica che i pneumatici siano conformi alle linee guida del Dipartimento dei Trasporti Americano. Serve anche come punto di partenza chiaro e universale.

Figura 1.27: Esempio DOT (16)

- **Tire Plant Code** [XX] Il gruppo successivo di numeri contiene due caratteri che indicano il codice dello stabilimento di produzione dello pneumatico. Ogni stabilimento di produzione di pneumatici ha un codice unico di due caratteri, spesso composto da un numero e una lettera. In caso di ritiro del pneumatico, di problemi di sicurezza o di altri problemi legati al pneumatico, è possibile risalire allo stabilimento che ha prodotto il pneumatico in questione.
- **Tire Size Code** [YY] Rappresentano il codice della misura dello pneumatico. Il codice della misura degli pneumatici è un po' meno semplice di altre indicazioni. Il DOT lascia che i produttori determinino il proprio codice per le dimensioni degli pneumatici. Questa agevolazione nella produzione di nuovi pneumatici da parte dei produttori, obbliga all'NHTSA [1] di aggiornare l'elenco dei codici. Sfortunatamente, questo rende difficile tradurre le dimensioni dello pneumatico sul numero DOT del pneumatico. Fortunatamente, esiste un modo più semplice per conoscere le dimensioni del pneumatico che verrà introdotto a seguire.
- **Tire Manufacturer Characteristics** [ZXY] Si tratta essenzialmente di uno spazio in cui i produttori possono indicare la distinzione dello pneumatico o altre caratteristiche specifiche del marchio. Il Registro Federale degli Stati Uniti afferma: "Il terzo raggruppamento può essere utilizzato a discrezione del produttore per fornire qualsiasi altra caratteristica significativa dello pneumatico. I produttori presentano l'elenco dei codici alla NHTSA.
- **Tire Age** [4519] Le ultime quattro cifre del numero DOT indicano l'età dello pneumatico. I primi due numeri di questo raggruppamento indicano la settimana dell'anno di produzione dello pneumatico. Le ultime due cifre indicano l'anno di produzione dello pneumatico.

1.6.3 Parametri dimensionali

Le dimensioni degli pneumatici sono in realtà una combinazione di due elementi: la larghezza dello pneumatico e il diametro della ruota a cui si adatta. La larghezza è misurata in

[1] National Highway Traffic Safety Administration

millimetri, il diametro della ruota si misura in pollici. Le dimensioni degli pneumatici sono così importanti perché influiscono su diversi aspetti, tra cui l'aderenza del pneumatico, la resistenza al rotolamento e l'efficienza del carburante dell'auto.

Prendendo in esempio il seriale della foto 1.28 possiamo dettagliare i suoi componenti.

Figura 1.28: Parametri dimensionali (14)

- **Sezione** [195]. Rappresenta la sezione di larghezza dello pneumatico che entra a contatto con la strada.
- **Rapporto d'aspetto** [55]. indica in percentuale del rapporto esistente tra l'altezza del fianco e la sua larghezza.
- **Diametro interno** [R 16]. È il diametro interno dello pneumatico, dove si inserisce il cerchio e la R sta ad indicare "radial"
- **Indice di carico** [87]. Il numero di indice di carico è un'informazione essenziale necessaria quando si acquistano pneumatici per auto. Il numero di indice di carico viene utilizzato per determinare il peso massimo che uno pneumatico può sostenere. Questa informazione si trova di solito sul fianco dello pneumatico ed è importante assicurarsi che il numero di indice di carico corrisponda ai requisiti del veicolo.
- **Indice di velocità** [V]. Rappresenta la classificazione dei pneumatici in base alla velocità. La classificazione della velocità di un pneumatico indica la velocità massima a cui il pneumatico può essere utilizzato in sicurezza. La classificazione si basa su test condotti dal produttore in condizioni controllate. La classificazione della velocità è solitamente espressa con una lettera: "A" è la più bassa e "Y" la più alta.

Capitolo 2

Materiali e metodologie utilizzate

Per le acquisizioni degli pneumatici si è scelto di usare un profilometro a lama laser. Il motivo di questa scelta è legata alla movimentazione e alla conformità della gomma stessa. Avendo un oggetto circolare, c'è bisogno di una metodologia per ottenere delle immagini con scritte linearizzata della spalla, in più, si voleva sfruttare il fatto che le scritte laterali sono in rilievo; quindi, queste due caratteristiche hanno fortemente motivato la scelta e il successivo utilizzo di un profilometro.

Facendo un confronto con una camera lineare; non avremmo potuto sfruttare le informazioni delle altezza delle sporgenze e avremmo aggiunto una complessità in più per ottenere delle scritte dritte..

La profilometria Laser, una tecnica basata sulla triangolazione tra una lama laser e una telecamera a scansione d'area, consente al sistema di avere una ricostruzione tridimensionale della superficie del pneumatico con la massima risoluzione raggiungibile considerando le specifiche della strumentazione e del tempo ciclo desiderato.

Una volta ottenuti i dati raccolti dal profilometro, sarà necessario utilizzarli per renderizzare un'immagine della spalla dello pneumatico. Il rendering è stato sviluppato interamente in python. La scelta è stata forzata, tenendo conto delle molteplici librerie esistenti in fatto di manipolazione d'immagini e dalla facilità di debug per un linguaggio scriptato.

Solo dopo avere ottenuto l'immagine desiderata, si è passati alla identificazione del DOT tramite un modello OCR. Nello specifico, è stato utilizzato sempre Python, cosi da facilitare l'interfacciamento con lo script di rendering con il modelli OCR. I due modelli OCR:

- **Tesseract** Nello specifico Pytesseract, una libreria python che permette di interfacciarsi facilmente con Tesseract (13).
- **EasyOCR** Anche essa una libreria free che mette a disposizione modelli preaddestrati per l'OCR (4).

2.1 Profilometro ed encoder

Il sistema di visione impiegato è costituito da un profilometro laser pilotato da un encoder incrementale, cosi da riuscire ad ottenere misurazioni 3D sulla spalla laterale del pneumatico con risoluzione desiderata.

A seguire dettaglieremo i principi di funzionamento di un profilometro ed encoder.

2.1.1 Profilometro

Tecnologie innovative come la misurazione senza contatto, sono sempre più utilizzate per il controllo di qualità e manutenzione nell'industria 4.0. Spesso, i metodi di misurazione con contatto non sono applicabili a causa della natura fragile dell'oggetto da esaminare, per la sensibilità della superficie al contatto meccanico, per la presenza di temperature elevate, per le dimensioni ridotte dell'oggetto o per il limite temporale di misurazione.

Con il progresso tecnologico in campo optoelettronico si hanno a disposizione misuratori laser, che oggigiorno consentono la misurazione automatizzata, rapida ad elevata precisione di superfici. Con la capacità di visualizzarne una rappresentazione in formato 3D.

Il principio di triangolazione di un profilometro laser si basa sulla misura dell'intensità della porzione di laser rimbalzato dalla superficie dell'oggetto che si vuole scansionare.

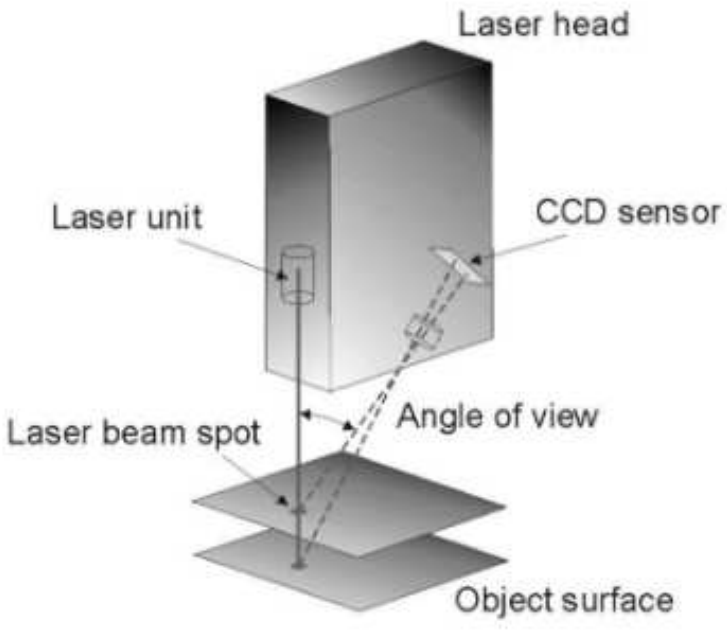


Figura 2.1: Principio di funzionamento profilometro laser (19)

Si ha una unità laser che proietta un fascio di luce, la quantità di luce riflessa dall'oggetto viene catturata dal sensore Charge-Coupled Device (CCD) che riesce a catturare una mappa di pixel considerando l'intensità della luce del laser. Il raggio catturato dal CCD risente della triangolazione che si viene a creare tra la sorgente del raggio luminoso, la superficie e la posizione del sensore CCD (19).

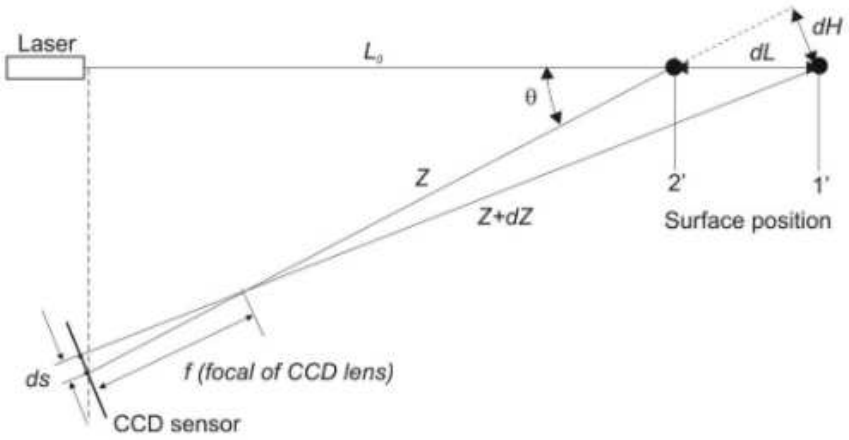


Figura 2.2: Principio di triangolazione per profilometro laser (19)

La sensibilità della misurazione è espressa in funzione dell'angolo di triangolazione θ, della distanza di base Z e della distanza focale F 2.1.

$$\frac{\partial s}{\partial Z} = \frac{f}{Z} \sin\theta \tag{2.1}$$

Solitamente i sensori laser compatti si caratterizzano per una risoluzione che si aggira nell'ordine di micrometri. Ciò non implica che per distanze più brevi, i sensori di triangolazione laser sono in grado di fornire una risoluzione inferiore al micron.

La triangolazione laser è un metodo basato su punti. Per la visualizzazione 3D della superficie ispezionata, i valori acquisiti vengono utilizzati per il successivo processo di scansione punto per punto. Nel processo di scansione, dai dati registrati viene creata una matrice bidimensionale come presentazione digitale della superficie scansionata [1].

Questo metodo di misurazione deve fare i conti con alcune limitazioni che influiscono sul rilevamento quindi sui risultati della misurazione. Esistono limitazioni causate dalla natura fisica della triangolazione, dal materiale dell'oggetto che influisce sulla capacità riflessiva della superficie, dalla complessità della texture, dalle ombre create per occlusione, dai disturbi sul raggio laser da parte di sporgenze, dalla diffusione e assorbimento della luce.

Per migliorare la visualizzazione delle immagini è necessaria l'elaborazione dei dati con l'utilizzo di funzioni di filtraggio.

I profilometri laser sono sensori di spostamento che raccolgono dati sulle altezze dei singoli punti per ogni linea laser (o profilo) affinché si possa ricostruire la superficie scansionata. Questo consente di eseguire misurazioni 2D/3D basate su differenze di altezza, sulla distanza tra ogni punto e tra ogni riga e sul angolo di incidenza utilizzando sempre un unico sensore. Oltre ai dati di altezza, i profilometri raccolgono anche dati sull'intensità del segnale luminosa riflesso cosi per fornire una soluzione stabile e affidabile per misurazioni e ispezioni.

[1] Con acquisizione facciamo riferimento al operazione di catturare i valori in altezza dei singoli punti. Con scansione indichiamo l'operazione con la quale, partendo dai punti acquisiti, si riesce ad ottenere una rappresentazione della superficie (modello 3D o semplice immagine).

Alcune considerazioni verranno fatte su risoluzione, field of view e situazioni di occlusione. Verranno presentate brevemente le problematiche che introdurre e come queste potrebbero essere gestite.

Risoluzione in X e Y Come abbiamo visto precedentemente, il profilometro riesce a misurare la superficie catturando punti consecutivi per un singolo profilo, per poi affiancare i profili ottenuti cosi da ricostruire una matrice di punti, questa matrice sarà la scansione dell'oggetto in esame. Considerando la matrice W dove ogni riga è associata ad un profilo, possiamo introdurre due parametri fisici legati alla distanza tra due punti in uno stesso profilo e la distanza tra un profilo e il successivo; nel primo caso parleremo di Risoluzione in X e nel secondo caso di Risoluzione in Y.

Per quanto riguarda la Risoluzione in X, essa è legata ai limiti fisici dello strumento. Nel dattaglio è vincolata dalla triangolazione con la quale il produttore ha costruito lo strumento. Non potendo essere modificata, nel caso si necessiti di una risoluzione differente, bisognerà considerare l'utilizzo di un nuovo profilometro.

La Risoluzione in Y, diversamente dalla precedente, può essere settata come si desidera. E' possibile intervenire sulla risoluzione in Y andando a gestire, in maniera opportuna, la frequenza di campionamento del profilometro.

Field of View (FOV) o campo visivo di un possibile dispositivo ottico. Si riferisce alla massima area e l'angolo massimo che il dispositivo riesce a coprire. Quindi, maggiore è il FOV e maggiore è mondo osservabile. Anche la FOV è vincolata dalla costruzione del profilometro, ogni strumento avrà la sua FOV in base alla triangolazione associata dal produttore.

Di conseguenza possiamo affermare che la FOV del profilomtero è fissa e non modificabile.

Occlusione Considerando una scena tridimensionale, nell'immagine bidimensionale che si otterrebbe, gli oggetti in primo piano copriranno (occludono) gli oggetti dietro di loro o porzioni di scena, non riuscendo a catturare l'interezza della scena perdendo molte informazioni. Un esempio di occlusione, in figura 2.3, si possono distinguere 2 situazioni.

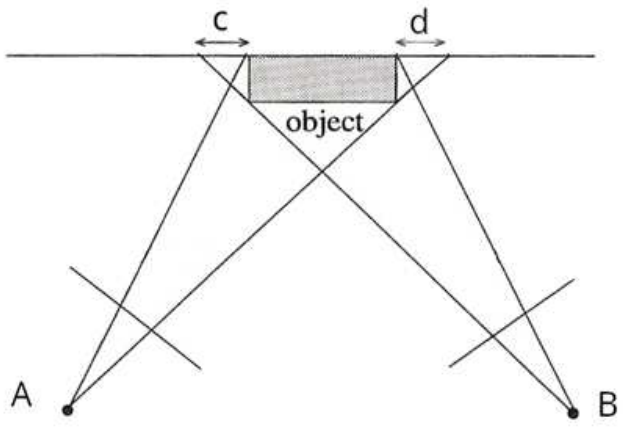


Figura 2.3: Esempio occlusione

Nel primo caso, avendo una camera nel punto A, non riusciremo a catturare oggetti o

porzioni di scena nella zona indicata con la lattera “d”. Situazione identica con la camera posizionata in B, che perderà tutta la porzione di scena in “c”.

Considerando la struttura meccanica con cui è costruito un profilometro, si verificheranno situazioni di occlusioni in presenza di sporgenze o incavature. L’occlusione diventa un problema importante se non prontamente gestito, il tutto potrebbe essere parzialmente arginato intervenendo sul corretto posizionamento dello strumento sull’oggetto da scansionare.

2.1.2 Encoder

L’encoder è un trasduttore di spostamento e di velocità che trasforma un movimento meccanico angolare o lineare in una serie di impulsi elettrici digitali. Questi impulsi elettrici saranno utilizzati per controllare la frequenza di campionamento di un qualsiasi dispositivo ad esso collegato.

L’encoder è composto da:

- **Interfaccia meccanica** la quale consiste nell’insieme di componenti che consentono l’accoppiamento dell’encoder alla macchina o dispositivo di applicazione.

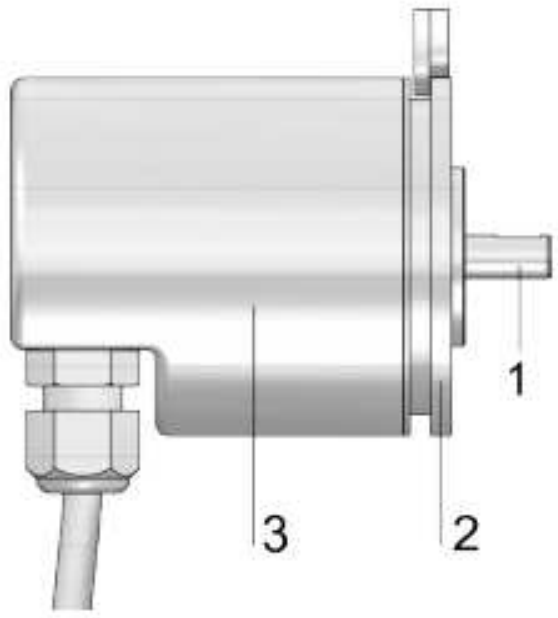


Figura 2.4: Interfaccia meccanica di un encoder

1. Indica l’asse collegato all’albero della macchina rotante;
2. Flangia che collega e adatta l’encoder al supporto d’installazione;
3. Involucro che contiene dischi e componenti elettroniche.

- **Disco** (o riga ottica, o attuatore con magnete). Il disco dell’encoder determina il codice di trasmissione degli impulsi ed è formato da un supporto (il materiale può essere di tipo

plastico, vetro o metallico) sul quale vengono impresse o ricavate strisce (o tacche) chiare (trasparenti), alternate a strisce scure (opache), inoltre è possibile distinguere encoder incrementali o assoluti.

- **Interfaccia elettronica**. È l'insieme di componenti di input ed output che consentono sia l'alimentazione dell'encoder che la trasmissione dei segnali elettrici.

Encoder incrementale

Gli Encoder incrementali sono specifici encoder, costituiti da segnali incrementali la cui variazione rispetto ad una posizione assunta come riferimento consente di rilevare rotazioni, velocità e accelerazioni basandosi sul conteggio degli impulsi inviati dal encoder alla macchina da pilotare.

Negli encoder incrementali possiamo distinguere due segnali A e B, che emettono impulsi quando il dispositivo viene spostato. Insieme, i segnali di output A e B indicano sia il verificarsi del movimento che la direzione. In più è presente un segnale di uscita aggiuntivo, tipicamente indicato con Z, che indica che l'encoder si trova in una particolare posizione di riferimento.

I segnali A e B vengono rappresentati con delle corone (dischi precedentemente introdotti) aventi tacche trasparenti alternate ad altre opache, il numero di queste tacche indica il numero di impulsi e determina la risoluzione del dispositivo. A differenza di un encoder assoluto, uno incrementale non indica la posizione assoluta, ma riporta solo i cambiamenti di posizione e la direzione del movimento.

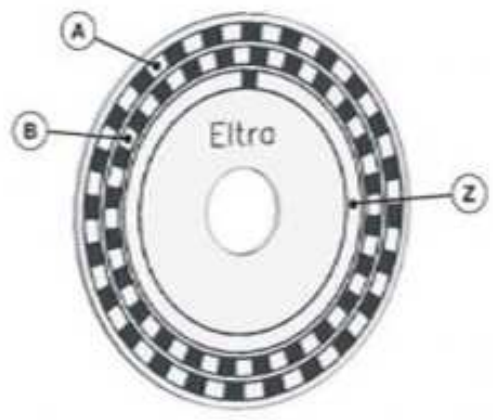


Figura 2.5: Esempio segnali A, B e Z su un encoder incrementale (2)

Come già discusso, un encoder incrementale lavora su due segnali A e B sfalsati di 90° e questo introduce 4 stati logici possibili in uscita al encoder. La risoluzione del encoder sta ad indicare il numero di volte che il segnale passa da un livello logico alto ad uno basso, per ogni giro del albero.

Nella rotazione del asse dell'encoder osservando il segnali A e B, passare successivamente da situazioni di stato logico 1 (alto) a 0 (basso) è possibile determinare movimento, velocità di rotazione e la direzione di rotazione dell'encoder.

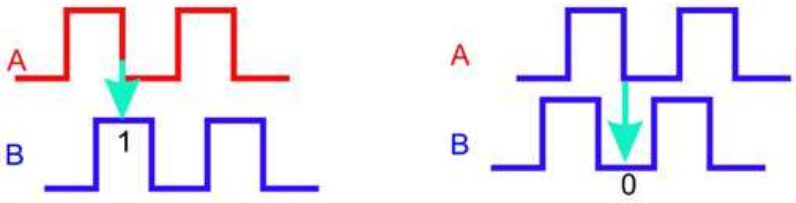


Figura 2.6: Esempio segnale a doppia onda quadra (15)

Per conoscere la direzione di rotazione, in senso orario o antiorario, si confrontano le due fasi A e B:

Rotazione in senso orario (incrementa): in questo ci troveremo con un fronte di discesa della fase A (quando il segnale logico passa da 1 a 0) in corrispondenza di 1 logico sul segnale della fase B.

Rotazione in senso antiorario (decrementa): in questo caso ci troveremo con un fronte di discesa della fase A in corrispondenza di uno 0 logico sul segnale della fase B. fase B.

Per conoscere la velocità di rotazione basterà contare quanti passi sono stati generati nell'unità di tempo.

2.2 Sistema di movimentazione e acquisizione pneumatico

In questo paragrafo verrà introdotto e descritto il sistema utilizzato per la movimentazione e acquisizione degli pneumatici.

Nello specifico, il sistema consiste in un supporto orizzontale per la rotazione degli pneumatici e un sistema di visione composto da un profilometro laser pilotato da un encoder incrementale. Lo pneumatico sarà adagiato su un pianale e messo in rotazione da un motore elettrico trifase, cosicché ruotando possa essere correttamente scansionato. La scelta di posizionare e far ruotare gli pneumatici su un piano orizzontale, è dovuta dopo aver notato che in questa maniera si riuscivano a contenere le oscillazioni dovute dalla rotazione.

Va fatta una nota sulla distribuzione della massa di un pneumatico. La forma di ogni pneumatico ricorda quella di una "ciambella", dove l'intera gomma è presente ai bordi lasciando la parte centrale vuota (là dove sarà alloggiata la ruota del mezzo), quindi in rotazione ci troveremo a dover gestire una situazione molto delicata. Con una struttura ti questo tipo, anche una piccola interferenza può dar vita ad enormi oscillazioni.

La strumentazione di visione è posizionata in maniere da riuscire a catturare l'intera altezza della spalla con la risoluzione desiderata. La distanza del profilometro dalla gomma è stata settata tenendo conto la FOV che forniva il produttore per il proprio profilometro. La frequenza di campionamento del profilometro è gestita attraverso un encoder, a sua volta collegato all'asse di rotazione del motore elettrico. Questo fa sì che il profilometro riesca ad acquisire scansioni perfettamente campionate.

Dall'immagine 2.7 si riesce facilmente ad individuare i vari elementi. Seguendo le etichette marcate di rosso, partendo dall'alto troviamo:

- **P** Profilometro laser Mech-Mind lnx 8300.
- **Pn** Uno degli pneumatici usati per i test.
- **S** I 4 supporti per ancorare lo pneumatico al pianale orizzontatale.
- **M** Motore trifase per la rotazione dello pneumatico.
- **E** Encoder Kubler 5825.

Figura 2.7: Foto del sistema di acquisizione e movimentazione

Mech Mind lnx 8300

Il profilometro usato è il modello Lnx 8300, prodotto e commercializzato dalla Mech Mind. Il modello Lnx 8300 si adatta perfettamente alle esigenze del nostro caso, visto che fornisce una semplice interfaccia per un encoder incrementale, ed è stato progettato con una risoluzione in X e una FOV che permettono di digitalizzare perfettamente i nostri pneumatici.

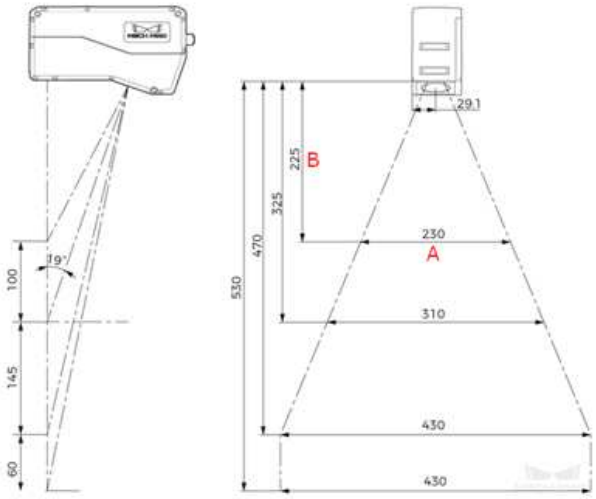
Il Mech Mind Lnx 8300 è dotato di risoluzione 4K+ e di alte velocità di scansione (fino a 15 kHz) per fornire dati 3D altamente accurati di piccole sporgenze e difetti a una velocità più elevata. Questo sensore è stato pensato per applicazioni di misura e ispezioni in industria elettronica e automobilistica.

Caratteristiche chiave:

- **Risoluzione 4K** Fornisce 4.096 punti dati per singolo profilo, riuscendo ad eseguire ispezioni 3D ad alta risoluzione
- **Ultra-High Scan Rates** Può raggiungere frequenze di scansione fino a 3,3 kHz. Dispone di ottiche avanzate e algoritmi avanzati per fornire dati 3D altamente accurati a una alta velocità di scansione.
- **Precisione micrometrica** La risoluzione X fino a 9 μm e una ripetibilità Z fino a 0,2 μm consentendo l'ispezione e la misurazione di piccolissime caratteristiche e difetti.
- **Single-Shot HDR** La funzione consente di scansionare superfici scure (bassa riflettività) e riflettenti (alta riflettività) in un'unica acquisizione.
- **SDK e API** Fornisce interfacce estensibile in vari linguaggi per l'acquisizione di immagini, tra cui C ++/C / API di Python. Inoltre è conforme agli standard GenICam3.0 e GigE Vision.
- **Possibili scenari di applicazione** Ispezioni e misurazioni di dettagli molto fini nell'ordine di μm.

Alla sinistra della figura 2.8 è riportata la FOV del lnx 8300.
https://docs.mech-mind.net/en/eye-3d-profiler/latest/hardware/manual-lnx-8000.html[2]

[2] *verificato il 5/7/2024*



Model			LNX-8300
Data points per profile			4096
Scan rate			3.3–15 kHz
Reference distance (RD)			325 mm
Measurement range	Z-axis		305 mm
	X-axis	Near	230 mm
		RD	310 mm
		Far	430 mm
X-axis resolution			105 µm
Z-axis repeatability			2 µm
Z-axis linearity			± 0.02% of F.S.
Weight			About 1.2 kg
Dimensions			About 195 × 61 × 109 mm

Figura 2.8: Caratteristiche costruttive del Mech Mind lnx 8300 (6)

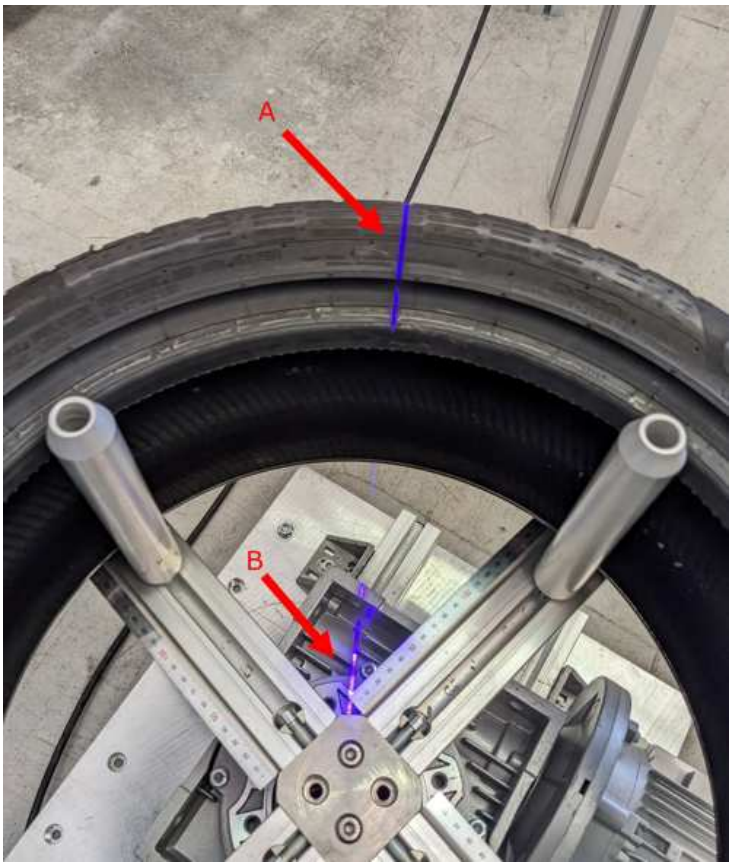


Figura 2.9: Lama laser profilometro

In figura 2.9 si può notare come la lama laser del profilometro incida la superficie laterale dello pneumatico. In particolare riusciamo ad individuare in A la porzione di lama acquisire lo pneumatico e in B la porzione di lama che fuori esce, andando ad incidere sul sistema di movimentazione. Eravamo consapevoli del verificarsi di una situazione del genere, dato che

ci trovavamo a dover scansionare una porzione di gomma molto stretta rispetto alla massima larghezza fornita dalla FOV . Questa situazione ci costretto ad acquisire dati anche in zone al di fuori dello pneumatico, appesantendo leggermente l'acquisizione. Inoltre non era possibile ridurre la ROI lungo il profilo di acquisizione (evitando di acquisire l'intero profilo ma solo la porzione in corrispondenza dello pneumatico) cosi da diminuire i dati inoltrati dal profilometro. Quindi eravamo vincolati a gestire profili dove una sola piccola porzione di punti faceva riferimento allo pneumatico.

Di conseguenza molta dell'informazione che il profilometro acquisiva è informazione non necessaria ,sarà eliminata soltanto dopo aver completato l'acquisizione, nella successiva fase di processing.

Encoder Kubler 5825 10000 impulsi

L'encoder incrementale usato è il 5825 di kubler con 10000 impulsi per giro. Questo encoder si prestava perfettamente al nostro studio, per le sue dimensioni contenute, le sue proprietà tecniche e il numero di impulsi così elevati *riferimenti:* (5).

Un encoder con 10000 impulsi ci ha permesso di gestire al meglio le acquisizioni, riuscendo a campionare le scansioni con risoluzione e le proporzioni desiderate.

2.3 Python

Per l'intera sessione di analisi e processing di immagini è stato usato Python. Nello specifico si è utilizzato Anaconda, per gestire un ambiente di sviluppo personalizzato con python 3.10 e per poter utilizzare il gestire dei pacchetti Conda presente di default in Anaconda.

Introduciamo le principali librerie usate e come queste possono essere installate.

2.3.1 OpenCV

OpenCV, abbreviazione di Open Source Computer Vision Library, è un'enorme libreria open source per la visione artificiale, l'apprendimento automatico e l'elaborazione delle immagini. Originariamente sviluppato da Intel, è ora gestito da una comunità di sviluppatori sotto la OpenCV Foundation. Supporta un'ampia varietà di linguaggi di programmazione come Python, C++, Java, ecc. Può elaborare immagini e video per identificare oggetti, volti o persino la scrittura di un essere umano. Quando è integrato con varie librerie, come Numpy, libreria altamente ottimizzata per lavorare su array (8).

La libreria permette una semplice gestione di immagini trattandole come "matrici di pixel", alle quali è possibile accedervi in maniera molto semplice e rapida.Diamo una sguardo a quella che si può definire l'Image Processing, cioè l'Elaborazione delle Immagini attraverso la libreria in questione.

Consideriamo un'immagine da voler analizzare, mettiamo per caso che essa presenta del rumore e come è possibile immaginare il rumore rendere inefficiente qualsiasi operazione di analisi. OpenCV mette a disposizione tecniche di elaborazione immagini. Arrivando a migliorare questa immagine in modo tale da facilitare tutte le successive analisi ed elaborazione delle immagini, fino ad eliminare completamente il rumore e ottenere un'immagine pulita.

Per l'installazione della libreria si possono seguire due strade; usare il gestore del pacchetto integrato in Python e quindi digitare da terminale:

```
pip install opencv-python
```

Oppure, se si sta utilizzando un ambiente Anaconda si può usare il suo gestore di pacchetti Conda:

```
conda install conda-forge::opencv
```

2.3.2 Scikit-image

Scikit-image (noto anche come skimage) è una delle librerie open source di elaborazione delle immagini per Python. Fornisce un potente toolbox di algoritmi e funzioni per varie attività di elaborazione delle immagini e visione artificiale. Ed è costruito sulla base di librerie scientifiche popolari come NumPy e SciPy.ndimage (10).

Le immagini in scikit-image sono rappresentate da matrici di pixel NumPy. Pertanto, molte operazioni comuni possono essere eseguite utilizzando i metodi NumPy standard per manipolare gli array. Inoltre, fornisce una vasta raccolta di algoritmi di elaborazione delle immagini come filtraggio, segmentazione, estrazione di caratteristiche, morfologia e altro.

Consente facilmente di interfacciarsi con strumenti scientifici di Python: come scikit-learn e Scipy. Ciò consente agli utenti di combinare l'elaborazione delle immagini con altre attività di elaborazione scientifica, come l'analisi dei dati, l'apprendimento automatico e la visualizzazione.

Scikit-image fornisce un'ampia gamma di strumenti e algoritmi per attività di elaborazione delle immagini. Include filtri immagine completi, operazioni morfologiche, trasformazioni di immagini, estrazione di caratteristiche e altro ancora. Questi strumenti consentono all'utente di eseguire operazioni complesse di elaborazione delle immagini con facilità e flessibilità.

Per configurare l'ambiente per scikit-image, è possibile utilizzare il gestore di pacchetti PIP o conda per installare scikit-image e le sue dipendenze.

Per installare scikit-image utilizzando pip, esegui semplicemente il comando seguente nel prompt dei comandi:

```
pip install scikit-image
```

Se invece si sta utilizzando un ambiente Anaconda, è possibile usare il suo gestore di pacchetti Conda:

```
conda install conda-forge::scikit-image
```

2.3.3 Scikit-learn

Scikit-learn è una libreria che fornisce una gamma di algoritmi di apprendimento supervisionati e non supervisionati utilizzando python. La libreria è basata su SciPy (Scientific Python) che deve essere installata prima di poter usare scikit-learn (12).

Estensioni o moduli per SciPy sono convenzionalmente denominati SciKits . In quanto tale, il modulo fornisce algoritmi di apprendimento ed è denominato scikit-learn. La libreria fornisce una robustezza e una facilità d'uso che permette di essere impiegata anche in sistemi di produzione.

La libreria è focalizzata sulla modellazione di dati. Non si concentra sul caricamento, la manipolazione e il riepilogo dei dati. Per queste funzionalità, si può fare riferimento a NumPy e Pandas.

Alcuni modelli popolari forniti da scikit-learn includono:

- **Clustering** Per raggruppare dati senza etichetta come KMeans
- **Cross Validation** Per stimare le performance di modelli supervisionati su dati non ancora classificati.
- **Set di dati** Per testare set di dati o per generare set di dati con proprietà specifiche.
- **Riduzione dimensionale** Per ridurre il numero di attributi nei dati per il riepilogo, la visualizzazione e per l'utilizzo di tecniche come PCA.
- **Ensemble learning** Per migliorare le performance di classificazione, combinando le previsioni di più modelli supervisionati.
- **Estrazione delle features** Per definire gli attributi chiavi da un insieme di dati, immagini o serie temporali.
- **Selezione delle featrures** Per identificare gli attributi significativi con la quale creare modelli supervisionati.
- **Regolazione dei parametri** Per ottenere il massimo dai modelli supervisionati.
- **Apprendimento multidimensionale** Per riassumere e rappresentare dati multidimensionali complessi.
- **Modelli supervisionati** Una vasta gamma di modelli intelligenti: reti neurali, support vector machine (SVM) e alberi decisionali ecc.

Per installare Scikit-learn basta digitare da terminale

```
pip install scikit-learn
```

O se si desidera usare conda:

```
conda install conda-forge::scikit-learn
```

Per l'analisi e interpretazione del testo sono stati svolti vari test con Pytesseract (Tesseract) e EasyOCR. I test ci hanno aiutato a valutare quale metodologia poteva aiutarci ad ottenere risultati più accurati. Introduciamo una breve descrizione su come è possibile settare l'ambiente di sviluppo per Tesseract (Pytesseract) e EasyOCR.

2.3.4 Pytesseract

Pytesseract è un wrapper per l'engine OCR Tesseract. E' utile perché facilita l'utilizzo di Tesseract in uno script Python, e supporta la maggior parte dei formati immagini, tra cui jpeg, png, gif, bmp, tiff. Inoltre è possibile lavorare direttamente con il testo riconosciuto, cosa non possibile con Tesseract che scrive su file il testo decifrato. (9)

Installazione

Pytesseract richiede Python 3 o superiore ed è necessaria una libreria tra Python Imaging Library (PIL o Pillow) oppure OpenCV. Prima di installare Pytesseract, bisgna essere sicuri di avere installato Google Tesseract OCR sul proprio computer, disponibile per Windows,Linux e Mac OS. Per verificare la presenza di Tesseract sul pc basterà richiamare il comando:

```
tesseract
```

Se così non fosse, ad esempio perché Tesseract non è nel **PATH**, bisognerà modificare le variabili:

```
tesseract_cmd
pytesseract.pytesseract.tesseract_cmd
```

in Windows. In Debian/Ubuntu si può usare il pacchetto tesseract-ocr.deb. Per gli utenti di Mac OS, installare il pacchetto homebrew tesseract.

Installazione Pytesseract via pip.

```
pip install pytesseract
```

Installazione Pytesseract partendo dai sorgenti.

```
pip install -U git+https://github.com/madmaze/pytesseract.git
```

attraverso Conda:

```
conda install conda-forge::pytesseract
```

2.3.5 EasyOCR

Come già descritto in precedenza, EasyOCR è una libreria Python per l'estrazione di testo immagini immagini. Si tratta di un OCR generale che può leggere sia il testo da scena naturali che il testo da un documento. (4)

Installazione EasyOCR attraverso pip:

```
pip install easyocr
```

Utilizzare i sorgenti:

```
pip install git+git://github.com/jaidedai/easyocr.git
```

Attraverso Conda:

```
conda install conda-forge::easyocr
```

Capitolo 3

Sviluppo progetto

L'intero codice è consultabile al seguente repository GitHub: https://github.com/Francesco7760/ocr_tires

Il progetto sviluppato consiste nel individuare e decodificare la sigla DOT (compresa dei soli 3 caratteri D,O,T) presente su di ogni pneumatico, partendo da una scansione ad alta risoluzione della sua spalla laterale. Come già spiegato in precedenza, si è usato un profilometro per la scansione, ottenendo così una matrice di punti contenente i valori delle altezze acquisite.

Dai valori raccolti sarà facile ottenere una depth map, ovvero un'immagine dove diverse altezze (o profondità) sono legate a diverse intensità di colore, dalla quale sarà possibile estrarre un'immagine binaria. Questo perché si desidera mettere in risalto i caratteri delle scritte rispetto lo sfondo, in particolare si vogliono scritte nere su sfondo bianco [1].

Per il progetto sono stati usati due laptop; il primo con a bordo Windows 10 collegato al sistema di acquisizione, quindi usato per interfacciarsi con profilometro tramite Gigabit Ethernet. Invece, il secondo con una distro Linux, nello specifico Arch Linux con una versione del kernel 6.6; con cui è stata sviluppata tutta la parte di elaborazione e analisi.

Data la necessità di disporre di ottime acquisizioni per far si che le successive analisi potessero dare risultati accurati, il lavoro di tesi è stato diviso in due macro parti; la prima ha riguardato il testare più configurazione per il sistema di visione, passando per il posizionamento corretto del profilometro al settaggio dei suoi parametri. Invece, nella seconda parte ci si è concentrati nella analisi e nel elaborazione delle scansioni con l'obiettivo di migliorare l'accuratezza dei modelli OCR.

Come è facile comprendere, le due parti sono fortemente connesse, non solo da un ordine logico (prima acquisizioni e solo a seguire elaborazione immagine), ma anche dalla possibilità di ottenere ottimi risultatati. Partire da una acquisizione corretta influirà positivamente sui risultati finali, e quindi faciliterà di molto tutta la parte di analisi e OCR.

[1] La scelta di lavorare su immagini binarie, è stata presa dopo aver notato che i modelli OCR utilizzati, avevano risultati miglior se in input avevano immagini cosi fatte.

Figura 3.1: Depth Map

Figura 3.2: Depth Map elaborata dopo Baseline correction

Figura 3.3: Immagine binaria ottenuta dalla Depth Map 3.2

Da 3.1 si può notare come, al variare delle altezza acquisite dal profilometro varia l'intensità colore. L'immagine è ottenuta normalizzando le altezze sul range $[0 - 255]$. *Grigi più chiari indicano altezze maggiori, si notano caratteri o la parte superiore dell'immagine. Invece grigi più scuri indicano altezze minori, si possono notare la parte centrale e inferiore dell'immagine. Da 3.2 notiamo che scompare la variazione sulla intensità colore. Questo perché, l'immagine è ottenuta come risultato di una* ***baseline correction*** *3.4.2 In 3.3 abbiamo un'immagine binaria che sarà data in input ai modelli OCR. Notiamo come i caratteri sono in netto contrasto rispetto lo sfondo.*

Detto ciò, nel dettagliare lo sviluppo del progetto seguiremo l'ordine cronologico degli eventi; in breve vengono elencati i task principali.

- **Acquisizione** Siamo partiti con il posizionamento dello pneumatico e del profilometro, centrati e fissati sul supporto meccanico. A seguire è stato utilizzato un software di acquisizione, settato con i parametri corretti, cosi da ottenere le acquisizioni con le giuste risoluzione e proporzioni. Le scansioni ottenute sono salvate sotto forma di file testuale o binario (.csv o .bin) e la loro struttura rappresenta una matrice rettangolare di punti.

- **Rendering immagine** Procede con il rendering della matrice di punti acquisita per ottenere la depth map corrispondente. Quindi, ci sarà una operazione di *Baseline Correction* per mettere in risalto i picchi presenti sui profili acquisiti sapendo che il profilo ha una andamento concavo e non lineare. Una volta linearizzati i singoli profili sarà possibile ottenere una depth map corretta e tramite un semplice threshold adattivo estrarre un'immagine binaria.

  Detto ciò, il cuore del rendering passa dalla ricerca della baseline migliore (precesso di baseline correction), dalla quale sarà possibile estrarre informazioni sui picchi locali per ogni profilo. I picchi stessi ci dicono qualcosa sulla presenza di caratteri in rilievo rispetto lo sfondo e una volta estratti sarà facile estrapolare un'immagine binaria.

- **OCR** Avendo un'immagine con caratteri in netto contrasto con lo sfondo (immagine binaria) sarà facile individuare il DOT attraverso Tesseract e EasyOCR. L'immagine binaria sarà analizzata dai modelli OCR attraverso un sistema di Tiling o sfruttando l'algoritmo di edge deteciotn Canny[2].

[2]Famoso algoritmo per l'individuazione di bordi di oggetti.

3.1 Acquisizioni

Come già accennato in precedenza il fulcro per aver risultati ottimi parte da delle ottime acquisizioni. Con *ottime acquisizione* ci riferiamo ad acquisizioni che rispecchino le proporzioni reali della superficie, che riesca a scansionare l'intero oggetto e soprattutto che non introduca rumore (o quanto meno, rumore che possa essere facilmente gestito) il tutto con alla massima risoluzione consentita.

Possiamo dire che ci sono 3 passi chiavi per avere una scansione ottima:

- **Posizionare correttamente il profilometro** Il profilometro deve trovarsi ad un altezza, dallo pneumatico, tale che riesca ad acquisire la spalla per la sua interezza. Inoltre la lama laser deve incidere sulla superficie in maniera ortogonale per evitare occlusioni e distorsioni indesiderate. Cosi da riuscire ad avere un scansione risoluta e senza deformazioni.

  Per determinare l'altezza ottimale si è tenuta in considerazione la Field Of View 2.8) dello strumento.

- **Settare correttamente i parametri** Per aver una acquisizione proporzionata e luminosa bisogna lavorare sulla risoluzione in X, Y e sul tempo di esposizione. Mec Mind fornisce una documentazione dettagliata su come poter settare al meglio questi parametri.

- **Numero massimo di profili da catturare** In fine bisogna decidere il numero massimo di profili da catturare per riuscire ad acquisire l'intero pneumatico. Questo parametro è stato arrotondato per eccesso cosi da avere a disposizione un leggero margine di sicurezza.

Configurazione profilometro

Prima di intervenire sui parametri da configurare sarà necessario posizionare correttamente lo strumento.

Figura 3.4: Allineamento corretto

Figura 3.5: Allineamento scorretto in senso antiorario

Figura 3.6: Allineamento scorretto in senso orario

Allineamento lama laser

Esempio del effetto di uno scorretto allineamento della lama laser (segmento blu). In 3.4 è riportata un allineamento corretto, come vediamo dalla scritta EXAMPLE, si riesce ad ottenere una scansione non distorta e corretta. Nei casi 3.5 e 3.6 ci troviamo in due situazione in cui la scansione ottenuta è fortemente distorta. Colpa di un allineamento sbagliato.

Il profilometro è posizionato in modo da catturare l'intera spalla, considerando l'altezza della spalla di circa 150 mm, molto inferiore dispetto range minimo in X della lama laser pari a 230 mm (etichetta A in 2.8). Questo non crea nessun tipo problema o vincolo in quanto con 230 mm si riusciva a catturare una zona molto più ampia della superficie interessata. Quindi, abbiamo posizionato il profilometro a circa **250 mm** dalla gomma. In verità ci saremmo potuti avvicinare ancora, ma abbiamo voluto mantenerci ad una distanza tale che ci avrebbe concesso una un minimo margine di sicurezza.

Un ulteriore aspetto da tenere in considerazione è il periodo di rotazione dello pneumatico. Il periodo di rotazione è vincolato alla velocità con il quale il profilometro riesce a catturare correttamente ogni singolo profilo. Se il pneumatico ruota lentamente il profilometro riesce ad acquisire correttamente, ma di contro bisognerà aspettare un lasso di tempo maggiore per ottenere una scansione completa. Se invece ci troviamo nel caso opposto, con un pneumatico che ruota troppo velocemente, il profilometro non riuscirà a stargli dietro e rischieremo di perdere dati durante la scansione. Perciò va trovato un compromesso sul tempo di rotazione, per far si che si riesca ad acquisire correttamente senza impiegarci eccessivo lasso di tempo.

Abbiamo settato la velocità dello pneumatico in modo da aver circa 26 secondi per una rotazione completa. Come spiegato prima, ci siamo mantenuti al di sopra del limite per non rischiare di perdere profili durante l'acquisizione.

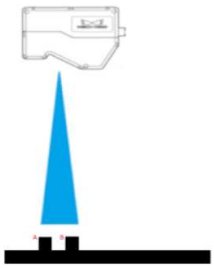


Figura 3.7: Profilometro perfettamente perpendicolare alla superficie

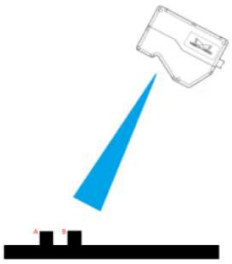


Figura 3.8: Profilometro non perpendicolare, inclinato verso destra

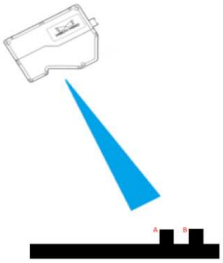


Figura 3.9: Profilometro non perpendicolare, inclinato verso sinistra

Figura 3.10: Inclinazione profilometro

Esempio sulla corretta inclinazione del profilometro. Nel caso 3.7 si riescono ad acquisire le sporgenze A e B nella maniera corretta. Nel caso 3.8 ci troviamo con lama inclinata verso destra e non si riesce ad acquisire correttamente A e B, abbiamo una forte occlusione da parte di B. Situazione simile in 3.9 dove c'è un occlusione invertita rispetto al caso precedente.

Un altro aspetto importante è l'allineamento della lama laser rispetto alla spalla dello pneumatico. La lama deve essere allineata in maniera corretta per evitare distorsioni e perpendicolare alla superficie per evitare occlusioni indesiderate. Per una migliore spiegazione, far riferimento alle figure 3.4 e 3.10.

Va fatta una premessa, ovvero il posizionamento dello pneumatico può risentire di alcune piccole incertezze, dovute alla natura meccanica del sistema di ancoraggio (nel fissare le varie viti ci siamo resi conto di non riuscire a mantenere la configurazione prestabilita). Nello specifico, per quanto concerne aver la lama laser ortogonale alla superficie da scansionare, ci siamo accorti che è pressoché impossibile ottenere un allineamento perfetto e che il tutto introduce rumore indesiderato che sarà prontamente gestito in fase di elaborazione.

Passiamo al settaggio dei parametri del profilometro. Nel settare i parametri abbiamo seguito la documentazione fornita da Mech Mind. In primis, si è fissato il tempo di esposizione, cercando di avere un profilo abbastanza luminoso e senza introdurre un eccessivo ritardo nell'acquisizione. A seguire si è impostata la risoluzione in Y ed infine si è scelto il numero di profili da catturare.

Il **tempo di esposizione** interviene sulla porzione di luce che il sensore fotosensibile riesce a catturare, quindi con un tempo di esposizione maggiore si avrà un profilo più luminoso ma anche un tempo maggiorato per l'acquisizione del profilo stesso. Quindi la scelta corretta è un compromesso tra brillantezza e velocità massima di acquisizione. Il tempo di esposizione influirà pesantemente sulla la qualità della scansione finale.

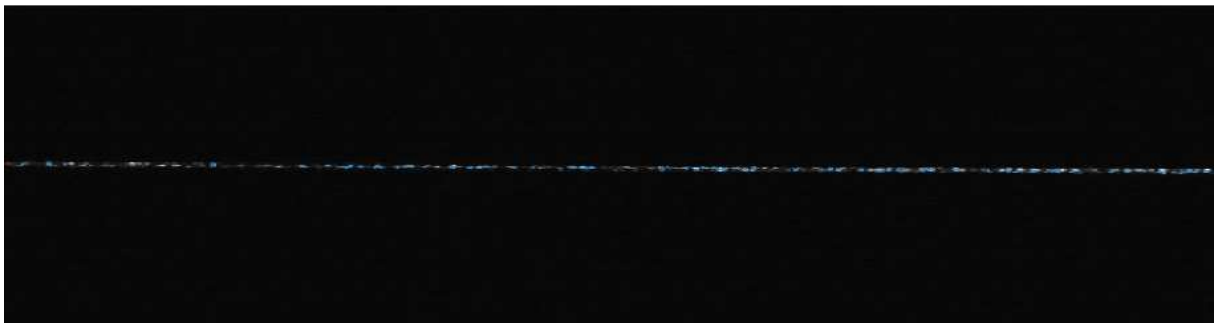


Figura 3.11: Profilo tempo esposizione 40 μs

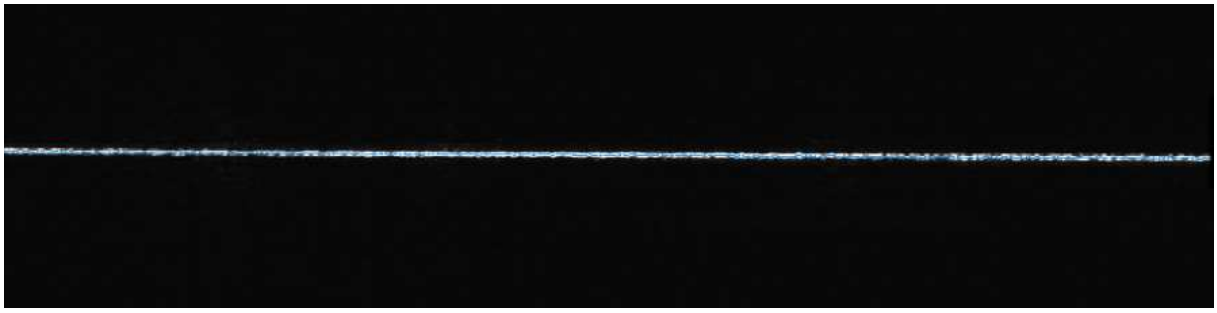


Figura 3.12: Profilo tempo esposizione 200 μs

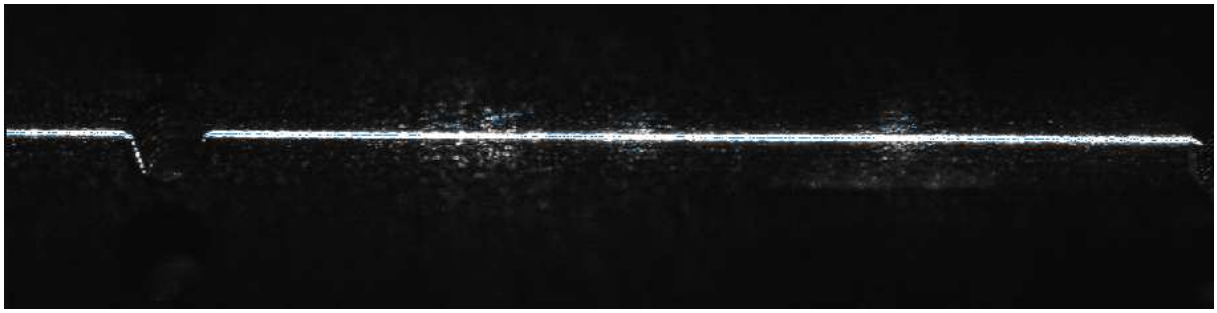


Figura 3.13: Profilo tempo esposizione 1500 μs

Figura 3.14: Tempo di esposizione (6)

Esempio di uno stesso profilo campione acquisito con tempi di esposizione diversi. 3.11 con un tempo di esposizione di 40μs, 3.12 con un tempo di esposizione di 200μs, 3.13 con un tempo di esposizione di 1500μs. Si può notare l'aumento della brillantezza con l'incrementare del tempo di esposizione.

In conclusione si è scelto un **tempo di esposizione** di **1000 μs**.

Proseguendo, ci siamo concentrati sul configurare la **risoluzione in Y**.

La risoluzione in Y è definita come la distanza in μm tra un profilo e il successivo, invece la risoluzione in X è la distanza in μm tra due punti consecutivi lungo un profilo. Nel personalizzare l'acquisizione si può intervenire solo sulla risoluzione in Y, al contrario quella in X è vincolata dalla costruzione meccanica del profilometro. Quindi se otteniamo un'immagine troppo schiacciata, vuol dire che abbiamo una distanza lungo l'asse Y maggiore della distanza effettiva, quindi la risoluzione andrà diminuita. Al contrario se ci troviamo con un'immagine dilatata, vuol dire che abbiamo una distanza tra due profili lungo l'asse Y minore della distanza reale, perciò andrà aumentata 3.16.

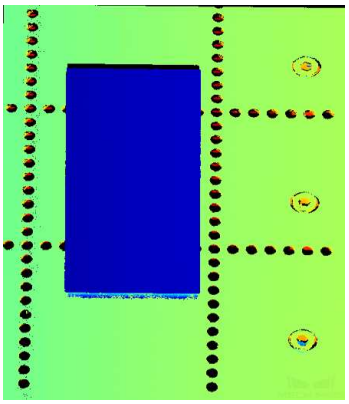


Figura 3.15: Scansione perfettamente proporzionate

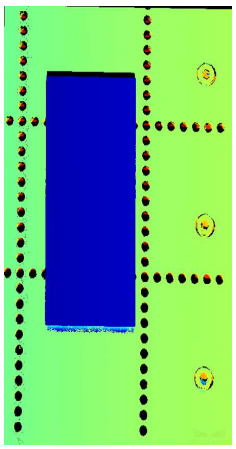


Figura 3.16: Scansione non proporzionata

Risoluzione in Y

Esempio sul corretto uso della Risoluzione in Y. Possiamo notare in 3.15 un'immagine con Risoluzione in Y e X uguali. Situazione diversa in 3.16 dove abbiamo la stessa immagine ma con una risoluzione in Y 3 volte quella in X, il risultato è un'immagine schiacciata lungo l'asse Y.

Nel nostro caso, vogliamo mantenere le proporzioni reali setteremo la risoluzioni in Y e X il più possibile uguali (non sarà possibile avere le risoluzioni perfettamente identiche, dato che stiamo ragionando con operazioni di campionamento e discretizzazione).

Per quanto riguarda la risoluzione, Mech Mind fornisce una formula 3.1,con la quale è possibile determinare la risoluzione desiderata in funzione di due parametri chiavi: **Trigger Interval** e **Trigger Signal Mode**.

Il calcolo è stato cosi svolto: volendo mantenere le proporzioni reali, si è impostata la risoluzione in Y uguale a quella in X, quindi pari a 105 μm (impostata dal produttore). Di conseguenza dettagliamo in conti svolti:

$$Resolution(\mu m) = EncoderResolution(\mu m) \times TriggerInterval \div TriggerSignalCountingMode \times 4 \tag{3.1}$$

- **Resolution Y** Volendo una risoluzione lungo l'asse Y pari a 105 μm e avendo una risoluzione dell'encoder fissa (fornita dall'encoder stesso), i due parametri su cui è possibile intervenire sono Trigger Interval e Trigger Signal Counting Mode.
- **Trigger Interval (TI)** È il parametro usato per settare il numero di trigger da considerare sul segnale proveniente dal encoder per acquisire un profilo.
- **Trigger Signal Counting Mode (TSCM)** Rappresenta il numero di trigger presenti in un periodo del segnale proveniente dal encoder. In ogni periodo possiamo avere 4 trigger.

  Dalla foto 3.17 è chiaro come i trigger vengono considerati all'interno di un periodo del segnale, Mech Mind permette di settare TSCM con 3 valori:

  - **1x** conta un solo trigger, nella foto è in corrispondenza alla 1° linea tratteggiata
  - **2x** conta 2 trigger, nella foto corrispondono alle 1° e 3° linee tratteggiate

– **4x** conta 4 trigger, nella foto corrispondono alle 1°,2°,3° e 4° linee tratteggiate

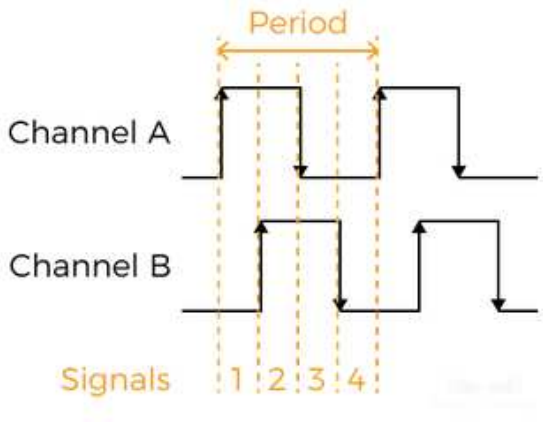


Figura 3.17: Trigger signal (6)

Trigger Interval e Trigger Counting Mode determinano la risoluzione dell'asse Y, influenzando così la precisione e la proporzionalità della scansione dell'oggetto.

- **Encoder Resolution** Misura la distanza sullo pneumatico che il profilometro copre nel intervallo di tempo tra una quadratura dell'encoder e la successiva. Per chiarire cosa si indica con intervallo tra due quadrature facciamo riferimento alla figura 3.17. L'intervallo tra due quadrature rappresenta l'arco temporale tra due linee tratteggiate.

  Quindi, per ottenere la risoluzione del encoder:

$$Encoder\_Resolution(\mu m) = Circonferenza\_DOT \div Impulsi\_di\_rotazione\_pneumatico \times 4 \tag{3.2}$$

  – La **circonferenza** utilizzata è la circonferenza della fascia laterale dello pneumatico su cui giace la scritta DOT.
  – Gli **impulsi di rotazione dello pneumatico**, indicano il numero di impulsi inviati dal encoder durante l'intero periodo di rotazione dello pneumatico. Vanno fatte alcune considerazioni: l'encoder è collegato sul asse di rotazione del motore trifase. Per far ruotare lo pneumatico viene utilizzata una trasmissione con rapporto 1/100 ovvero, saranno necessari 100 giri dell'asse di rotazione del motore per far compiere un giro completo alla gomma. Detto ciò, gli impulsi di rotazione sullo pneumatico sono pari al rapporto di trasmissione e impulsi del encoder.

Di conseguenza possiamo dire l'**Encoder Resolution** è pari a **1,57** μm.

Per determinare il numero massimo di profili da considerare, bisogna ragionare sulla circonferenza della fascia laterale su cui giace la scritta DOT da scansionare e la risoluzione in Y, una volta conosciute le due incognite sarà facile determinare il numero di profili necessari facendo una semplice divisione.

$$NumeroProfili = CirconferenzaDOT \div RisoluzioneY \tag{3.3}$$

Dalla figura 3.18 in rosso la circonferenza massima dello pneumatico, in verde la circonferenza della superficie da scansionare e in A è sintetizzato il concetto di risoluzione in Y (tra le linee

tratteggiate). Perciò la circonferenza utilizzata non è la circonferenza massima dello pneumatico perché ci interessa concentrarci sulla fascia dove è presente il DOT, di conseguenza il diametro usato è di 500 mm.

$$CirconferenzaDOT = \pi \times 500mm = 1570,79mm \tag{3.4}$$

Figura 3.18: Circonferenza DOT in verde e circonferenza massima in rosso. In A, corrispondete della risoluzione in Y. (20)

Configurazione finale

Ricapitolando, si è scelto di configurare il profilometro con **Trigger Interval** pari a **1x** e **Trigger Signal Counting Mode** a **67**, per quanto riguarda il numero di profili per ottenere una scansione completa si parla 14960 unità, ma si è deciso di optare per **16000 profili** per essere abbastanza sicuri di scansionare l'intera ruota anche in caso malfunzionamenti.

Per consultare direttamente le formule impiegate, visitare il seguente link: https://docs.mech-mind.net/en/eye-3d-profiler/latest/appendix/y-axis-resolution.html [3]

[3] verificato il 5/7/2024

3.2 Software acquisizioni

Il produttore Mech Mind, fornisce insieme alla documentazione tecnica, anche delle API con annesse references. Inoltre sono rilasciati alcuni esempi di codice, con i quali si può immediatamente passare all'acquisizione. Infatti per velocizzare il tutto, ci siamo affidati al loro codice per mettere su un software capace di acquisire scansioni con la nostra configurazione.

Brevemente: il software è stato scritto partendo da un loro esempi in C++ con API 2.3.1, dove sono state riportate alcune modifiche che riguardavano i nostri setting.

Mech Mind mette a disposizione un repository GitHub dove è possibile consultare vari esempi https://github.com/MechMindRobotics/mecheye_cpp_samples/tree/master/profiler[4]

Al momento della scrittura di questo documento, nel repository è possibile trovare 5 samples distinti uno per ogni necessità. Passeremo velocemente ad una descrizione.

- **TriggerWithSoftwareAndFixedRate.cpp** Attiva l'acquisizione ad una velocità fissa, quindi recupera e salva i dati acquisiti.
- **TriggerWithExternalDeviceAndFixedRate.cpp** Innesca l'acquisizione con l'input di un dispositivo esterno, scansiona ad una velocità fissa, quindi recuperare e salvare i dati acquisiti.
- **TriggerWithSoftwareAndEncoder.cpp** Innesca l'acquisizione con l'input via software, scansiona con segnale proveniente dal encoder, quindi recupera e salva i dati acquisiti.
- **TriggerWithExternalDeviceAndEncoder.cpp** Attiva l'acquisizione con l'input di un dispositivo esterno, scansiona con segnale proveniente dal encoder, quindi recupera e salva i dati acquisiti.
- **TriggerMultipleProfilersSimultaneously.cpp** Gestisce più profilometri laser per acquisire in modo asincrono e recupera i dati acquisiti.

Nello specifico abbiamo usato il **TriggerWithSoftwareAndEncoder.cpp** dato che abbiamo a disposizione un encoder e possiamo pilotare i trigger via software. Quindi abbiamo apportato le seguenti modifiche (portiamo all'attenzione solo le linee di codice modificate o aggiunte):

- Tempo esposizione

```
currentUserSet.setIntValue(
    mmind::eye::brightness_settings::ExposureTime::name, 1000);
```

- Trigger signal counting mode

```
currentUserSet.setEnumValue(
    mmind::eye::trigger_settings::EncoderTriggerSignalCountingMode::
        name, static_cast<int>(
            mmind::eye::trigger_settings::
                EncoderTriggerSignalCountingMode::Value::Multiple_1));
```

[4] verificato il 5/7/2024

- Trigger interval

```
currentUser->Set.setIntValue(
    mmind::eye::trigger_settings::EncoderTriggerInterval::name, 67)
```

- Numero profili da catturare

```
currentUser->Set.setIntValue(
    mmind::eye::scan_settings::ScanLineCount::name, 16000);
```

Si è scelto di salvare le acquisizioni in due formati binario e testale (.csv). In binario si ha il vantaggio di avere un file più compatto e veloce per accedere in lettura e scrittura, di contro non è prontamente facile da aprire con un semplice editor di testo. Invece resta molto facile consultare un .csv con un editor di testo.

In entrambi i casi i file sono cosi strutturati; in testa è presente un array contenete le dimensioni della matrice acquisita, numero massimo profili (16000) e numero di punti per profilo (4096). Il file continua con un vettore contente i valore acquisiti dal profilometro, ogni riga presenta un punto acquisito. Si è deciso di salvare le dimensioni della matrice, per rendere più facile in un secondo momento il reshape della stessa.

In questa tesi sono stati considerati entrambi i file (.bin e .csv), la differenza di formato sarà gestita dallo script apposito che si occuperà di leggere entrambi i file e caricare la matrice corrispondente in memoria. Nonostante ciò, siamo consapevoli dei grandi vantaggi che porta l'utilizzo di file binari, perciò nel repository 3 è presente la possibilità di leggere sia acquisizioni in binario che in csv. Inoltre nella cartella tools sono presenti scripts per la conversione di file binario in csv e viceversa, dando cosi la possibilità a chi vuole, di testare il processo di analisi con il formato dati preferito.

Le acquisizione hanno una struttura a matrice dove ogni riga è un profilo e perciò ci ritroveremo a gestire delle matrice "verticali", ciò implica che in fase si rendering saranno ruotate per avere delle immagini che rispecchino l'andamento orizzontale del fianco dei pneumatici.

Figura 3.19: Acquisizione cosi come viene restituita dal profilometro

Figura 3.20: Acquisizione ruotata, per facilitarne l'elaborazione

3.3 Struttura repository

La prima parte del progetto ha riguardato la parte di acquisizione, ora continuiamo con la seconda.

La seconda parte del progetto riguarda l'analisi delle scansioni acquisite e di come si è riuscito ad estrarre la scritta DOT. Ora, verrà presentata la struttura del software scritto.

Il repository è cosi strutturato:

- **./binary_dir** Directory contenente il file in formato binario delle scansioni, saranno utili

nel momento si voglia velocizzare la lettura e scrittura dei file stessi.

- **./csv_dir** Directory contenente il file testuali in csv delle scansioni effettuate, i csv sono mantenuti saranno comunque usati, nonostante la loro dimensione maggiore.
- **./image_grey_dir** Directory che conterrà le immagini in scala di grigi (depth map) ottenute come risultato della elaborazione delle acquisizioni.
- **./thresh_dir** Directory usata per salvare le immagini ottenute tramite threshold adattivo per mettere in risalto le scritte rispetto allo sfondo (immagini binarie).
- **./tools** Vi sono presenti 2 script utili per trasformare un file binario in csv e viceversa.
- **main.py** E' lo script principale, il cuore del progetto all'interno della quale sono presenti le principali elaborazioni.
- **csv_to_image.py** Contiene tutte le funzioni con la quale è possibile estrarre un'immagine da una singola acquisizioni.
- **detect_DOT.py** Contiene funzioni necessarie per OCR, come il disegnare il testo decifrato e la propria boundingbox
- **morphological_transformations.py** Contiene le principali trasformazioni morfologiche: Opening, Closing, Dilation, Erode e Skeleton
- **config.ini** File contenente le principali configurazioni che l'utente può facilmente settare a piacimento.
- **requirement.txt** Contiene tutte le librerie e dipendenze da dover installare per replicare un ambiente di sviluppo funzionante.

La parte di scripting può essere divisa in due sezioni; la prima riguarda la gestione dei dati acquisiti dal profilometro per ottenere un'immagine pulita, questa verrà indicata come "*rendering immagine binaria*". La seconda parte si concentra nell'estrapolare la posizione del DOT al interno del immagine appena processata e verrà chiamata "individuazione e decodifica del DOT".

Nel "**rendering immagine binaria**" ci concentriamo nel determinare un'immagine favorevole ai modelli OCR. Con immagine favorevole si intende un'immagine dove sia netta la distinzione tra caratteri e sfondo (background), avendo scritte nero su bianco senza rumore o disturbi che possano trarre in inganno il modello .

Consapevoli del campionamento e delle discretizzazioni effettuati dal profilometro, si sa che durante l'acquisizione ci saranno zone non perfettamente catturate con punti mancanti quindi senza informazione utile. Inoltre, avremo del rumore dovuto alla forma della sezione dello pneumatico e alla natura meccanica del sistema di movimentazione. Sapendo che molti di questi errori non possono essere evitati (vedi discretizzazione profilometro e perdita di punto in acquisizione) si è cercato di compensare il più possibile in post-processing sia per i dati acquisiti sia per l'immagine renderizzata.

Invece in "**individuazione e decodifica del DOT**", facciamo riferimento a due sistemi di supporto all'OCR. Nello specifico abbiamo implementato un sistema basato su tiles, ovvero una

finestra che scorrerà lungo l'intera immagine fino ad individuare il DOT. Qui l'OCR effettuerà una analisi sulle porzioni di immagini ritagliate dalla tile. Il secondo sistema di supporto lavora su blobs, ovvero macchie di color nero corrispondenti alle scritte da decifrare. Questi blobs saranno ottenuti con l'applicazione di trasformazioni morfologiche. In fine l'OCR effettuerà l'analisi sui soli blobs che rispondono a specifiche caratteristiche geometriche.

3.4 Rendering immagine binaria

Discuteremo il modo con la quale siamo riusciti ad ottenere un'immagine binaria partendo da una matrici di valori. L'obbiettivo è di ottenere un'immagine binaria con scritte nere su bianco.

Date queste premesse, vanno discusse due problematiche: La presenza di porzioni di gomma non acquisite, perciò si avranno scansioni con informazioni mancanti e la forma concava dei profili acquisiti. E l'andamento concavo dei profili acquisiti. Quindi non si potranno usare le altezze cosi come sono state acquisite, ma andranno prima normalizzate rispetto una baseline.

La mancanza di dati è una situazione spiacevole dovuta dalla difficoltà del profilometro nel acquisire porzioni di gomma che sono limitrofe a sporgenze o concavità. Questo comportamento è dovuta all'occlusione di porzioni di gomma che svettano al di spora delle porzioni vicine coprendole dalla lama laser. Il problema riguarda maggiormente i singoli caratteri delle scritte, perché essendo in rilievo coprono ciò che li circonda, di conseguenze ci troviamo con scansioni aventi scritte con punti mancati in piccole parti del loro contorno. La situazione verrà gestita imputando i dati mancati.

La forma non lineare della sezione dello pneumatico comporta aver profilo con un andamento concavo, cosi che le altezze misurate non possano essere prese come dislivelli assoluti sulla superficie.

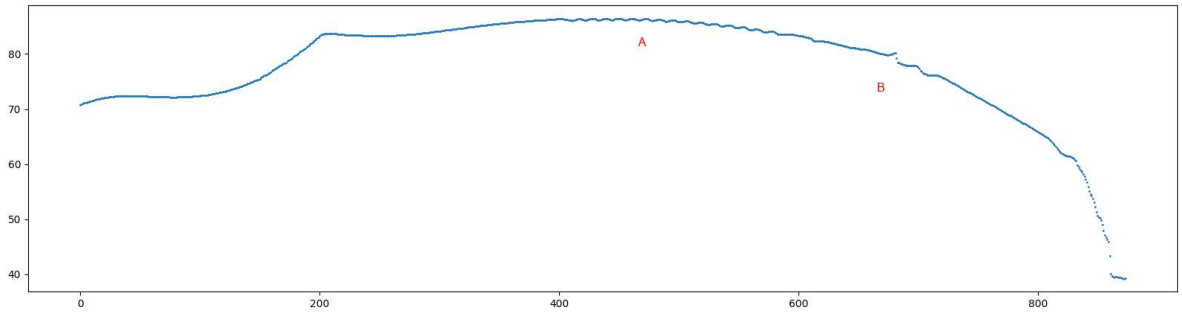


Figura 3.21: Esempio di profilo concavo

Esempio di un profilo acquisito, in ascisse le coordinate dei punti acquisiti (ci sono lo 1000 punti, perché abbiamo tagliato il profilo nei soli punti della spalla) e in ordinate i valori di altezza. Concentrandoci sui punti A e B, si notano picchi importanti e quindi informazioni da dover estrarre. Di contro non possiamo lavorare con i valori di altezza cosi come vengono catturati dal profilometro. L'andamento concavo del profilo proibisce l'utilizzare i valori di altezza nudi e crudi, c'è bisogna di un operazione di linearizzazione del profilo, una baseline correction 3.4.2

Il profilo ha un andamento concavo ciò comporta che non si può lavorare sul profilo nudo e crudo cosi come è stato acquisito, sarà necessario prima *linearizzarlo*. Con *linearizzato* ci

riferiamo ad una trasformazione con la quale vogliamo mantenere i picchi presenti su un profilo, ma adagiandoli virtualmente su un profilo che segua un andamento lineare.

Di conseguenza, il secondo passo nel processing consiste nel riuscire a linearizzare i profili per mettere in risalto le scritte ovviando alle difficoltà introdotte dalla concavità. Per fare ciò ci siamo affidati a tecniche di Baseline correction, con la quale individuare una baseline per poi linearizzare il profilo in esame.

3.4.1 Imputazione

L'imputazione dei dati mancanti è un'operazione delicata e importante al tempo stesso, nel nostro caso dovremmo ricostruire parte di dati che sono il cuore dello studio. Una cattiva imputazione potrebbe influire negativamente sulla lettere vicine e di conseguenza influire sul risultato del OCR.

Detto ciò, abbiamo deciso di fare affidamento alla libreria Scikit-learn che fornisce metodi di imputazione pronti al uso. In particolare si è deciso di provare due strade per imputare i valori mancanti; con il valore mediano o utilizzando K-Nearest Neighbors.

Prima va fatta una premessa importante, visto il nostro interessa nel concentrarci sul solo DOT e sapendo che si trova in una fascia bene precisa della spalla, prima di passare all'imputazione dei dati abbiamo effettuato un cropping della matrice.

Ci siamo concentrati semplicemente nella fascia dove siamo sicuri sia presente il DOT, ciò comporta ridurre la dimensione della matrice con cui lavorare e in parte ridurre anche il numero dei dati mancati. Nello specifico la scelta dei valori limiti dove tagliare la matrice possono essere configurati in *config.ini* 3.3 sui parametri *left_limit* e *right_limit* (considerando che stiamo ancora lavorando con una matrice "verticale"). 3.2)

Per quanto riguarda l'imputazione con valore mediano o KNN vanno fatte alcune considerazioni. I due approcci differiscono sulla metodologia impiegata:

- **Imputazione con valore mediano** 3.23 Risulta essere la pratica più semplice e veloce tra le due, ma anche quella che ritorna il risulto meno preciso, introducendo un alto livello di approssimazione. Considerando una scansione costituita da più profili acquisiti uniti tra loro, i punti mancanti vengono sostituiti banalmente dal valore medio calcolato lungo il profilo. Come detto in precedenza, molti valori mancanti sono vicino a scritte o increspature, quindi l'utilizzo del valo mediano va ad annullare l'influenza che potrebbe avere la presenza picchi sui punti da imputare, inoltre pone allo stesso livello ogni punto imputato. A favore si ha una imputazione molto veloce. (11)
- **Imputazione K-Nearest Neighbors** 3.24 Fornisce l'imputazione dei valori mancanti usando l'approccio k-Nearest Neighbors. Utilizza la distanza che i valori mancanti hanno rispetto ai loro vicini più vicini; ogni punto vicino viene pesato in base alla sua distanza, quindi un valore in prossimità del punto mancante avrà un peso maggiore rispetto a valori lontani. A fronte dei problemi presenti nel caso precedente, qui riusciamo ad imputare tenendo conto dei valori dei punti vicini, riuscendo così a tenere in considerazione la

presenza di picchi e ad ottenere una ricostruzione più accurata rispetto al caso con valore mediano. (11)

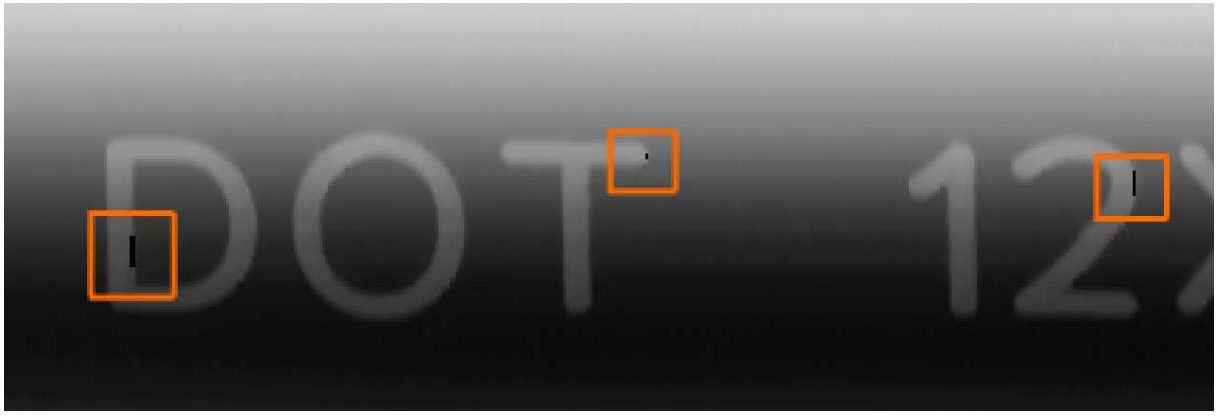


Figura 3.22: Scritta DOT senza imputazione

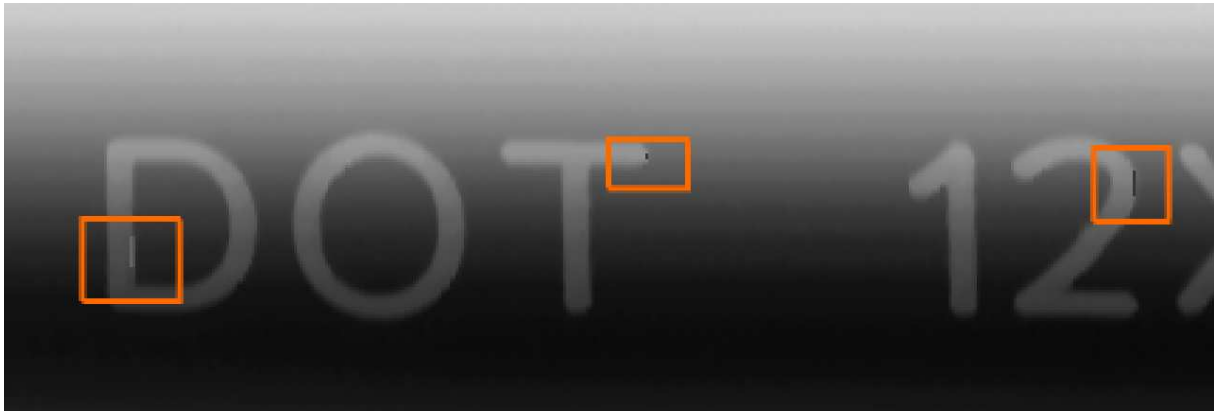


Figura 3.23: Scritta DOT con imputazione tramite valore medio

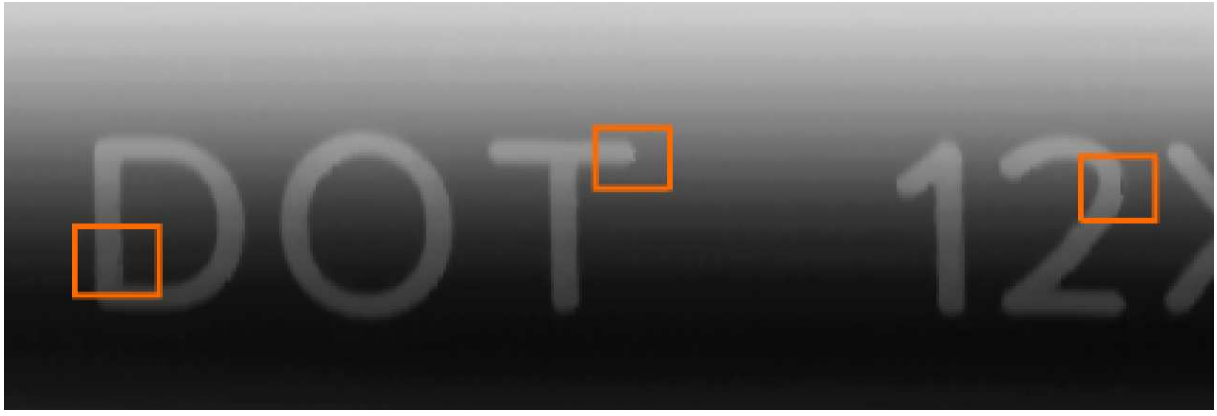


Figura 3.24: Scritta DOT con imputazione KNN

Dalla figura 3.22 sono marcate 3 situazioni in cui è evidente la mancanza di dati, partendo da sinistra verso destra: abbiamo la parte basse della D, alcuni punti in alto sulla destra della T e sulla destra del 2. Nella figura 3.23 i dati mancati sono stati sostituiti con il valore mediano della riga corrispondente, possiamo notare come la situazione sia migliorata ma rimane ancora molto grossolana e poco precisa. Nella figura 3.24 i dati mancanti sono stati imputati usando il KNN e possiamo notare un notevole miglioramento. In particolare sulla D e T, dove le porzioni mancanti sembrano essere state integrate perfettamente con i pixel circostanti. Quindi possiamo dire che tra le due metodologie, l'imputazione tramite KNN sia quella da preferire.

3.4.2 Baseline correction

Con Baseline correction ci riferiamo ad una serie di tecniche di pre-elaborazione che ci consentano di estrarre caratteristiche di interesse da una baseline di background. La dove le caratteristiche sono necessarie per le analisi e il background può essere inteso come rumore che si vuole rimuovere. Quindi lo scopo è rimuovere questo rumore di fondo per isolare e mettere in risalto le informazioni necessarie. (23)

Nel nostro caso, vogliamo lavorare sul singolo profilo per estrarre i picchi di altezze rispetto il fondo (con fondo ci riferiamo alla parte delle gomme senza informazioni utili), per ottenere un profilo linearizzato dove sia facile individuare le altezze delle scritte.

Abbiamo studiato e testato 2 tecniche per la baseline correction. A seguire verranno introdotte e dettagliate le singole tecniche, riportando i vantaggi, le difficoltà e la giustificazione che ci ha portato a consigliare l'utilizzo di uno o l'altro approccio.

Le tecniche di baseline correction studiate sono basate su:

- **Asymmetric Least Squares Smoothing ALSS**
- **Analisi wavelet**

Asymmetric Least Squares Smoothing

L'aymmetric least squares smoothing è una metodologia interessante per la stima di baseline, che basa il suo funzionamento sul Whittaker smoother. ALLS restituisce velocemente una stima di una linea di base anche per segnali di grandi dimensione, inoltre è di facile configurazione intervenendo direttamente sulla flessibilità della beseline.

L'idea è di ritrovare una funziona regolare che segua facilmente la basaline del segnale. Per fare ciò, si inizia da una funzione che si adatta al intero segnale, quindi calcolare le deviazioni, positive e negative, tra il segnale e la funzione adattata e usare queste differenze per continuare ad aggiornare la baseline.

Asymmetric perché il peso delle deviazioni è asimmetrico. Le deviazioni positive hanno un peso minore rispetto quelle negative, nello specifico si penalizzano di molto le deviazioni positive rispetto a quelle negative. In questo modo l'adattamento trascurerà i picchi e si adatterà molto meglio ai punti di base, che sono molto meno penalizzati.

Sia Y un segnale, con lunghezza d'onda M, supponiamo che sia stato campionato ad intervalli uguali. Sia Z un'altra serie, che dovrebbe avere le seguenti proprietà: avere un andamento dolce senza brusche impennate, ma anche essere fedele a Y.

Questi due obiettivi possono essere combinati minimizzando la funzione dei minimi quadrati:

$$S = \sum_i (y_i - z_i)^2 + \Delta \sum_i (\Delta^2 z_i)^2 \tag{3.5}$$

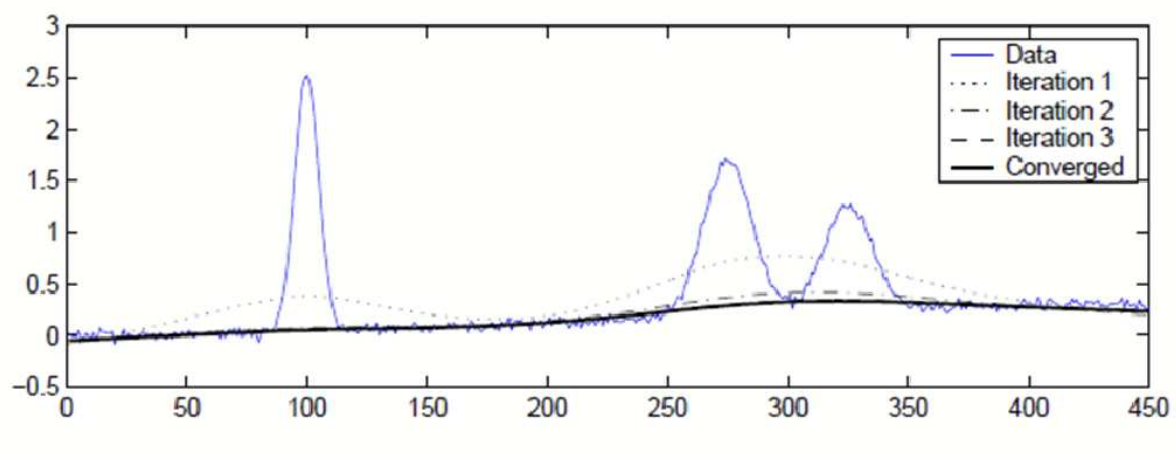


Figura 3.25: ALSS baseline correction (25)

dove

$$\Delta^2 z_i = (z_i - z_{i-1}) - (z_{i-1} - z_{i-2}) = z_i - 2z_{i-1} + z_{i-2} \tag{3.6}$$

Considerando S, il suo primo termine misura l'andamento di Z rispetto Y, invece il secondo termine può essere visto come la penalità su Z per il comportamento non regolare e l'equilibrio tra i due termini è gestito dal Δ.

Dalla formula 3.5 possiamo estrarre una sua generalizzazione con l'aggiunte di pesi attraverso un vettore di termini Wi, ottenendo cosi:

$$S = \sum_i w_i (y_i - z_i)^2 + \lambda \sum_i (\Delta^2 z_i)^2 \tag{3.7}$$

Per trovare i pesi basta risolver il seguente sistema di equazioni:

$$(W + \lambda D'D)z = Wy \tag{3.8}$$

dove W = diag(w) e D è una matrice ottenuta da

$$Dz = \Delta^2 z \tag{3.9}$$

Generalmente in 3.8 è riportato un sistema con M equazioni, risulta un sistema sparso, avente una matrice con le sole diagonale principale e le due sub-diagonali (sotto e sopra la principale) diverse da zero.

Dalla figura 3.25 vediamo come lavora ALSS su un segnale con 3 picchi (Data colore blue). Dalla tabella in alto a destra possiamo notare come variano gli smooting al aumentare il numero di volte di Iteration (Iteration sono intesi come iterazioni con cui ha lavorato lo smoother, per ogni iterazione i pesi dello smoother vengono aggiornati). Con Iteration = 1, si ottiene uno smoothing "leggero" che viene ancora fortemente influenzato dai 3 picchi ottenendo così una baseline non perfetta. Una baseline del genere non può essere considerata una buona baseline, perché il segnale corretto, ottenuto dalla differenza tra baseline e segnale originale, presenta

valori negativi e una buona baseline dovrebbe darci segnali con solo deviazioni positive. (25)

Quindi dalla formula 3.7 i residui Yi – Zi non giocano a nostro favore, visto che cosi si da la stessa importanza a deviazioni positive e negative. Risultati migliori si ottengono se riusciamo a differenziare l'importanza tra le due, dando maggiore importanza alle deviazioni negative, intervenendo direttamente sui pesi Wi.

Per la scelta dei pesi da usare, introduciamo un parametro P tale che Wi = P se Yi > Zi e Wi = 1-P altrimenti, l'obiettivo è trovare una soluzione di Z in funzione del parametro P che confermi i pesi usati in 3.8. Con i pesi 3.8 otteniamo una nuova stima di Z. Ripetendo i passaggi più volte, dove per ogni passaggio si aggiornano i pesi 3.8, fintando che i pesi non cambiano più. Si può affermare che con l'aumentare delle iterazioni la curva di smooting converge alla baseline desiderata. Bastano dalle 5 alle 10 iterazioni per ottenere già ottimi risultati.

Analisi Wavelet

Una wavelet è una forma d'onda di durata limitata, infatti presenta qualche oscillazione concentrata nel tempo, con valore medio nullo ed energia diversa da zero. Inoltre, a differenza della sinusoide, che è la funzione principale della trasformata di Fourier, le wavelet tendono ad avere un andamento irregolare e asimmetrico. Nella trasformata wavelet (WT), il segnale viene

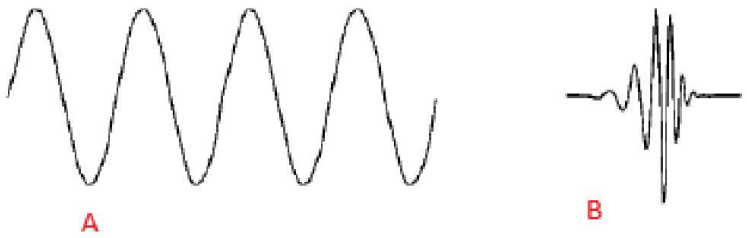


Figura 3.26: Confronto seno (A) e wavelet (B)

decomposto utilizzando una famiglia di funzioni, le quali, singolarmente, sono la versione scalata e traslata della funzione wavelet originale, chiamata wavelet madre. Il fattore di scala permette di dilatare o contrarre la wavelet: a fattori bassi di scala (che significa wavelet contratta), la WT possiede una buona risoluzione nel tempo, a scapito di quella in frequenza; vale il viceversa, a fattori di scala elevati (che significa wavelet dilatata) la WT presenta buona risoluzione in frequenza e peggiore risoluzione nel tempo. Esiste, quindi, una relazione frequenza-scala, per cui la WT permette di avere una buona risoluzione in frequenza alle basse frequenze, ma, contemporaneamente, una buona localizzazione temporale alle alte frequenze.

Quindi, è possibile effettuare una decomposizione del segnale multi-risoluzione, ovvero si decompone iterativamente il segnale a risoluzioni diverse, ciascuna corrispondente ad un differente valore di scala, per tutta la sua lunghezza.

Quindi verrà usata la decomposizione multi-risoluzione sul profilo, per annullare le compunti di ordine più basso cosi da mettere in risalto i valori di picchi. In verità qui non cerchiamo

una baseline per poi sottrarla al segnale originale cosi da ottenere un profilo corretto. In questo caso interveniamo annullando una singola componente del profilo originale per ottenere immediatamente il profilo corretto (24).

Per spiegare meglio il processo di decomposizione multi-risoluzione, introduciamo alcuni concetti chiavi.

Continuous wavelet transform

Come nella trasformata di Fourier, la Continuous Wavelet Transform (CWT) utilizza prodotti interni per valutare la somiglianza fra il segnale oggetto di studio e una funzione di analisi. In particolare, nella Short-time Fourier transform (STFT), la funzione di analisi è rappresentata da un'esponenziale complesso finestrato, mentre nella CWT si tratta di una wavelet.

Nello specifico la CWT confronta il segnale con versioni traslate e scalate di una wavelet, definita madre:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}}\psi(\frac{t-b}{a}) \tag{3.10}$$

dove a e b, rispettivamente, sono i parametri di scala e traslazione, con $a, b \in R, a > 0$ e $\psi(t)$è la wavelet "madre". Ogni versione scalata della wavelet madre deve trasportare la stessa quantità di energia, per cui si normalizza tramite $\frac{1}{\sqrt{a}}$.

Confrontando il segnale con le wavelet a differenti scale e posizioni, si ottiene una funzione a due variabili. Inoltre, le informazioni fornite dalla CWT a scale e localizzazioni temporali ravvicinate sono altamente correlate, per cui si avrà una rappresentazione ridondante del segnale, richiedendo un importante onere di calcolo.

In particolare, la CWT di un segnale $s(t)$ è così definita:

$$C_{a,b,\psi(t)} = \int_{-\infty}^{+\infty} s(t)\frac{1}{\sqrt{a}}\psi * (\frac{t-b}{a})dt \tag{3.11}$$

dove * denota il complesso coniugato. Dunque, per ogni valore di a e b, quanto più il segnale e la wavelet sono simili, quanto più il coefficiente restituito dalla CWT avrà un valore elevato. Da notare come, non solo la scelta dei parametri a e b influenzano i coefficienti della CWT, ma anche la scelta della wavelet.

Occorre fare una precisazione sulla relazione frequenza-scala. Infatti, quanto più il fattore di scala è elevato, ossia la wavelet risulta più dilatata, tanto più è larga la porzione di segnale a cui è confrontata, per cui più grossolane saranno le caratteristiche del segnale misurate dai coefficienti wavelet.

In sintesi, vale questa generale relazione fra scala e frequenza: a $\propto$ 1/f. Per cui:

- Per un fattore di scala basso ($a < 1$) avremo una wavelet compressa, i dettagli cambiano rapidamente e si avranno alte frequenze.

- Per un fattore di scala alto ($a > 1$) avremo una wavelet dilatata, i dettagli cambiano lentamente e si avranno basse frequenze.

Questa relazione può essere, quindi, sfruttata per massimizzare la quantità e la qualità delle informazioni su un segnale, utilizzando wavelet molto compresse da confrontare con porzioni di segnale alle frequenze più alte, e, via via, wavelet sempre più dilatate da confrontare con porzioni di segnale con frequenze sempre più basse.

Discrete wavelet transform

Nel paragrafo precedente si è messo in luce come la CWT abbia come risultato una forte ridondanza nei coefficienti, problema che può essere superato tramite l'utilizzo della Discrete Wavelet Transform (DWT). Essa prevede la discretizzazione dei parametri di scala e traslazione e viene modificata, di conseguenza, la rappresentazione della wavelet:

$$\psi_{j,k}(t) = \frac{1}{\sqrt{a_0^j}}\psi(\frac{t - kb_0a_0}{a_0^j}) \tag{3.12}$$

dove $j, k \in Z, a_0 > 1$ è un fissato step di dilatazione e b_0 è un fattore di traslazione che dipende da a_0. Di solito si fissa $a_0 = 2$ e $b_0 = 1$, in modo da avere un campionamento diadico sia nelle frequenze che nel tempo:

$$\psi_{j,k}(t) = 2^{-j/2}\psi(2^{-j}t - k) \tag{3.13}$$

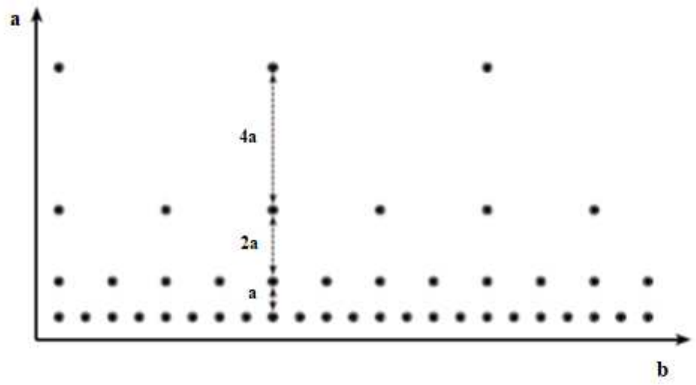


Figura 3.27: Localizzazione delle wavelet discrete nello spazio tempo-scala

La DWT può essere scritta come:

$$d_{j,k} = \int_{-\infty}^{+\infty} s(t)2^{-j/2}\psi * (2^{-j}t - k)dt \tag{3.14}$$

dove $d_{j,k}$ sono noti come i coefficienti o dettagli wavelet alla scala j e alla posizione j Dai dettagli $d_{j,k}$ è possibile ricavare il segnale originale $s(t)$ se è soddisfatta una condizione necessaria e

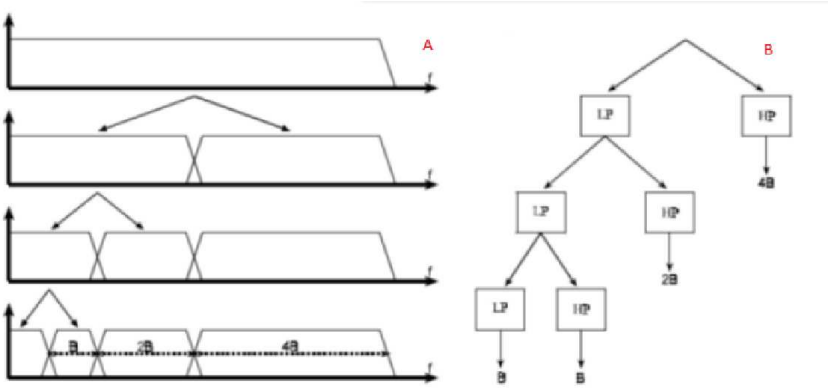


Figura 3.28: Decomposizione multirisoluzione
A Schema che mostra la separazione dello spettro del segnale; B Albero di decomposizione wavelet

sufficiente, ossia che l'energia dei coefficienti wavelet deve trovarsi fra due limiti positivi. Se i due limiti sono uguali, allora le wavelet discrete si comportano come una base ortonormale. Questo, inoltre, comporta che venga rimossa totalmente la ridondanza dalla WT.

Dunque, il segnale originale può essere ricostruito tramite tale formula:

$$s(t) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} d_{j,k} 2^{-j/2} \psi(2^{-j}t - k) \tag{3.15}$$

Decomposizione multirisoluzione

Nel analisi wavelet si parla spesso di approssimazioni e dettagli. I primi sono le componenti a bassa frequenza del segnale, mentre i secondi sono le componenti a frequenze elevate. Per dividere queste due componenti si usano due filtri: un passa-alto e un passa-basso. Se si filtra una sola volta il segnale d'interesse, si ottengono due segnali differenti: uno estratto dal filtraggio passa-alto, che rappresenta i dettagli (cioè le variazioni rapide); il secondo ottenuto dal filtro passa-basso, che rappresenta le approssimazioni (cioè l'andamento più grossolano). Tuttavia, il segnale risultato del filtraggio passa-basso, potrebbe contenere ancora dei dettagli, quindi si può procedere ad un ulteriore fase di filtraggio, ottenendo ulteriori due segnali. Questo processo può essere iterato finché non si è soddisfatti della suddivisione delle bande.

In realtà, nel caso in cui si stiano analizzando dei segnali digitali, oltre al filtraggio, bisogna attuare un'ulteriore operazione. Infatti, in questo caso, se si effettuasse soltanto il filtraggio, come risultato si avrebbero il doppio dei campioni rispetto al segnale originale. Questo perché ogni segnale risultato dal rispettivo filtraggio ha lo stesso numero di campioni del segnale originale. Per ovviare a questo problema bisogna effettuare un'operazione di downsampling, che permette di mantenere solo un campione ogni due, pur mantenendo il contenuto informativo invariato.

Infine, è interessante notare come si può ricostruire il segnale originale a partire dai dettagli e

dalle approssimazioni:

$$s(t) = \sum_{k=-\infty}^{k=\infty} a_{H,k} 2^{-H/2} \phi(2^{-H}t - k) + \sum_{j=-\infty}^{H} \sum_{k=-\infty}^{\infty} d_{j,k} 2^{-j/2} \psi(2^{-j}t - k) = A_H(t) + \sum_{j=-\infty}^{H} D_j(t) \tag{3.16}$$

dove H è il livello di decomposizione, $\psi(t)$ è la wavelet madre, $a_{H,k}$ sono i coefficienti delle approssimazioni al livello H, $d_{j,k}$ sono i dettagli wavelet e $\phi(t)$ è chiamata funzione di scala (scaling function) ed è legata alla wavelet madre. Anche questa funzione, come la wavelet madre, può essere scalata e traslata a formare una famiglia di funzioni di scala, nel dominio discreto:

$$\phi_{j,k}(t) = 2^{-j/2} \phi(2^{-j}t - k) \tag{3.17}$$

Per spiegare il ruolo di tale funzione bisogna fare una breve osservazione: per una wavelet, al crescere del parametro a, che aumenta di volta in volta di un fattore due nel dominio del tempo, in accordo con il campionamento diadico, si ha che la sua larghezza di banda, invece, si dimezza.

Questo significa che, teoricamente, ci vorranno un numero infinito di wavelet, per coprire lo spettro del segnale fino a zero, perché, al aumentare del fattore di scala, si coprirà solo la metà dello spettro rimanente. La scaling function, dunque, ha il ruolo di coprire l'intera porzione di spettro lasciata libera dalla wavelet e viene definita a partire da quest'ultima. Quindi, è giusto precisare che i filtri passa-alto sono associati alle funzioni wavelet, mentre i filtri passa-basso sono associati alle funzioni di scala.

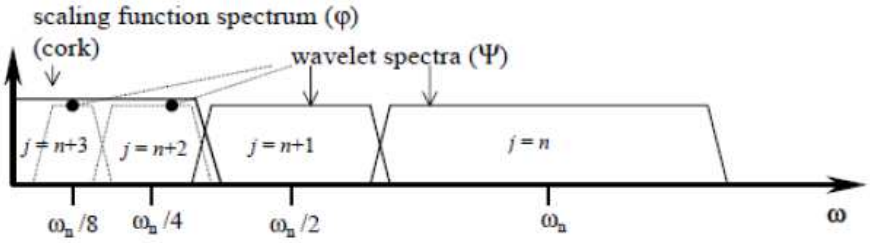


Figura 3.29: Come un'unica funzione di ridimensionamento può sostituire infinite wavelet

Infine, tornando alla formula 3.16, si nota come il segnale possa essere scomposto in un'approssimazione $A_h(t)$ al livello H (ultimo livello di decomposizione), che rappresenta l'andamento grossolano del segnale (contenente le frequenze più basse non contenute nei dettagli), e una successione di dettagli. La funzione di dettaglio $D_j(t)$ rappresenta l'errore residuo quando si passa dall'approssimazione a più alta risoluzione (o più bassa scala $j-1$) a quella a più bassa risoluzione (o più alta scala j).

Wavelet Packet Decomposition

La Wavelet Packet Decomposition (WPD) permette una più complessa e raffinata analisi, poiché, a differenza della DWT, anche i dettagli possono essere decomposti in due parti, usando lo stesso approccio che si usa nella DWT per decomporre le approssimazioni ad ogni livello (figura **??**). Ad un determinato livello H, quindi, la WPD produce 2^H insiemi di coefficienti, al contrario

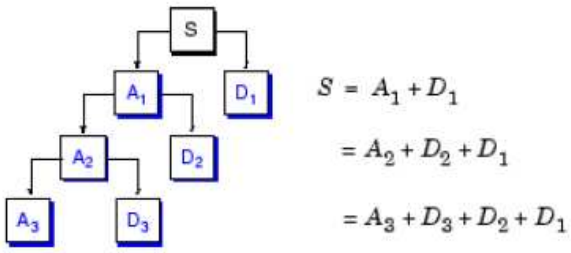


Figura 3.30: Esempio schematico di ricostruzione del segnale originale S a partire dalle approssimazioni e dai dettagli

della DWT che, al livello di decomposizione H, considera H+1 insieme di coefficienti (quelli dei dettagli da D_1 a D_H più l'approssimazione A_H).

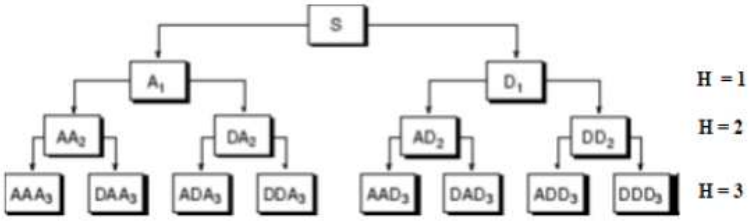


Figura 3.31: Albero di decomposizione della wavelet packet fino al livello 3

Le funzioni che sono usate nella WPD, sono forme d'onda del tipo:

$$W_{j,n,k}(x) = 2^{-j/2} W_n(2^{-j}x - k) \tag{3.18}$$

dove j è il parametro di localizzazione temporale, k è il parametro di scala e $n \in N$ viene definito un parametro di frequenza o di oscillazione. Nel dettaglio, $W_n(x)$ sono funzioni che derivano dalla applicazione ricorsiva di queste formule:

$$W_{2n}(x) = \sqrt{2} \sum_{k=0}^{2N-1} h(k) W_n(2x - k) \tag{3.19}$$

$$W_{2n+1}(x) = \sqrt{2} \sum_{k=0}^{2N-1} g(k) W_n(2x - k) \tag{3.20}$$

dove 2N è supposta essere la lunghezza dei filtri $h(k)$ e $g(k)$, rispettivamente un filtro passa-basso e uno passa-alto. In particolare, $W_0(x)$ è la scaling function e $W_1(x)$ è la wavelet madre.

Per cui, ad un fissato j e k la $W_{j,n,k}$ analizza le fluttuazioni del segnale nel intorno della posizione $2^j k$, per tutti i valori ammissibili di n. Osservando la figura 3.32, si può osservare come n = 0,1,...,7, che rappresenta l'ordine delle W_n, non corrisponda esattamente al numero delle oscillazioni, infatti:

Ordine n 0 1 2 3 4 5 6 7

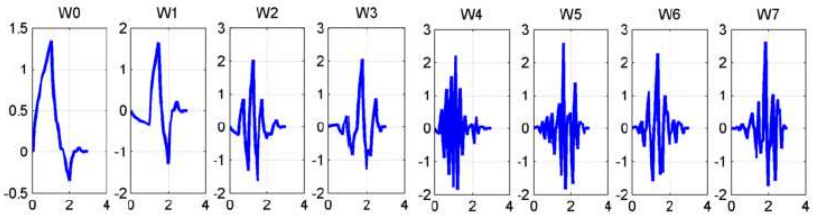


Figura 3.32: Esempio di funzioni $W_n(x)$, derivate da wavelet di Daubechies di ordine 2

Numero di attraversamenti dello zero di W_n 2 3 5 4 9 8 6 7

Conviene, quindi, definire l'ordine di frequenza (frequency order) r(n), in modo tale che la frequenza aumenti monotonamente con l'ordine n:

Ordine n 0 1 2 3 4 5 6 7

Frequency order r(n) 0 1 3 2 6 7 5 4

Si può osservare dalla Figura 3.32 che la $W_{r(n)}(x)$, infatti, oscilla approssimativamente circa n volte. Per analizzare il segnale è, dunque, meglio tracciare i coefficienti del pacchetto wavelet seguendo $r(n)$, piuttosto che n.

La totalità delle funzioni $W_{j,k} = W_{j,n,k}(x)$ rappresenta il pacchetto wavelet in corrispondenza dei parametri j e n. Per valori di j e n interi e positivi, il risultato della WPD è organizzato ad albero, dove j definisce la profondità e n la posizione a quel livello di profondità (figura 3.33). Per ogni valore di j, i, i valori possibili dei parametri n sono $0, 1...2^j - 1$.

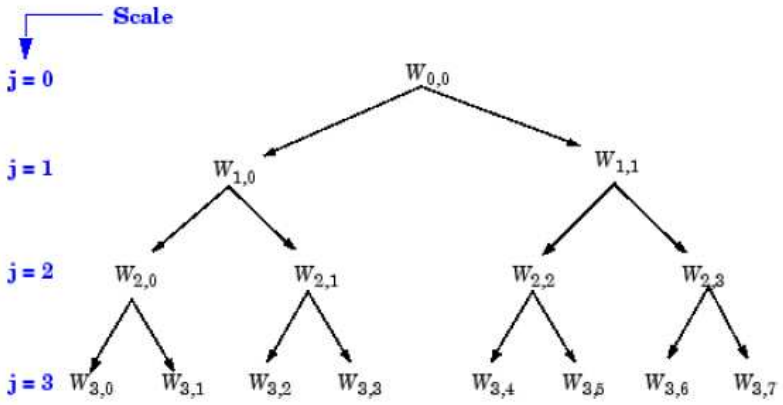


Figura 3.33: Pacchetti wavelet organizzati ad albero, nel particolare in figura si ha una decomposizione su tre livelli

Dunque, un segnale può essere decomposto in un numero di sotto strutture molto elevato, che dipende anche dalla profondità dell'albero. Dal momento che la gestione di tutti questi pacchetti può risultare onerosa, è interessante avere un metodo che permette di trovare una decomposizione ottimale rispetto ad un particolare criterio, calcolabile velocemente da un calcolatore. Solitamente il criterio più diffuso su cui basare la decomposizione ottimale è l'entropia, la quale è una misura di regolarità e di ordine. Infine, i coefficienti derivano, quindi,

dal prodotto interno fra il segnale da analizzare x(t) e ciascuna delle funzioni della WPD, ovvero $W_{j,n,k}(x)$ di cui appena discusso:

$$C_{j,n,k} = \int_{-\infty}^{+\infty} X(t) W_{j,n,k}(t) dt \tag{3.21}$$

Confronto baseline correction

Verranno fatte alcune analisi sui risultati ottenute dalle due tecniche di baseline correction. Per questo paragone, abbiamo calcolato la baseline partendo da uno stesso profilo originale, cercandone uno che possa avere caratteristiche importanti, ovvero picchi molto accentuati e un andamento non facile da approssimare. In ogni figura sono riportati il profilo originale (colore blue), la baseline (linea tratteggiata) e il profilo corretto, ottenuto come differenza tra profilo di partenza e baseline (curva arancio).

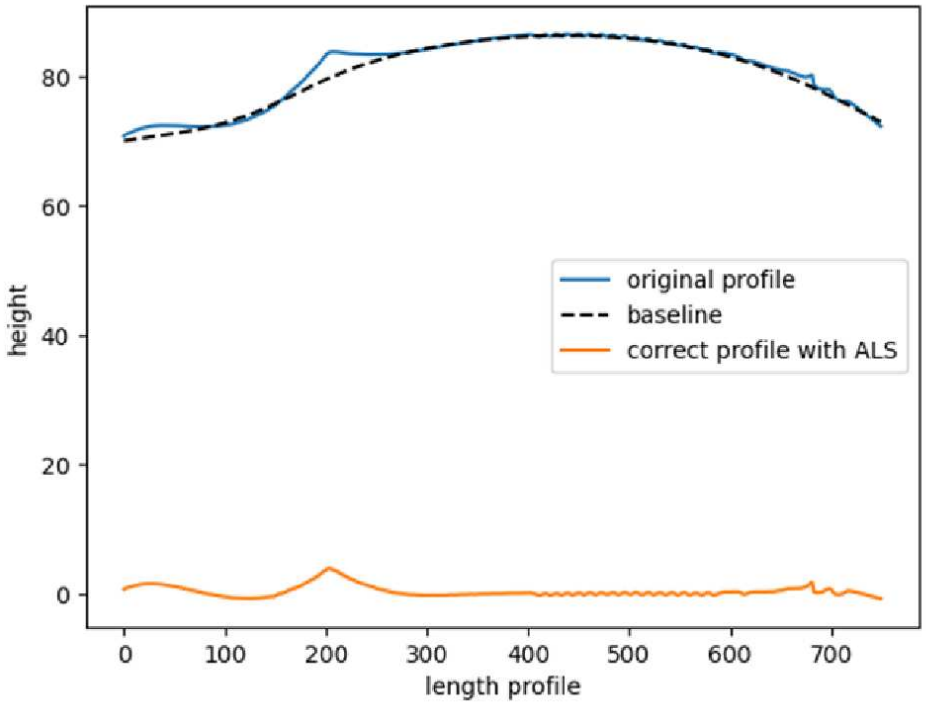


Figura 3.34: Baseline correction basato su ALSS

Dalla figura 3.34 possiamo notare come la baseline ottenuta tramite ALSS (curva tratteggiata) approssimi l'andamento del profilo principale senza seguire i picchi. Intorno a *lenght profile*=150 possiamo individuare alcuni punti del profilo corretto negativi, ma è facile comprendere come in questo caso l'errore introdotto è minimo e quindi accettabile.

In figura 3.35 vediamo come la baseline basata su Wavelet asseconda maggiormente i picchi del profilo originale, inoltre abbiamo un numero maggiore di deviazioni negative rispetto al caso precedente. Possiamo affermare che questa tecnica introduce un profilo corretto di qualità peggiore rispetto alla tecnica ALSS.

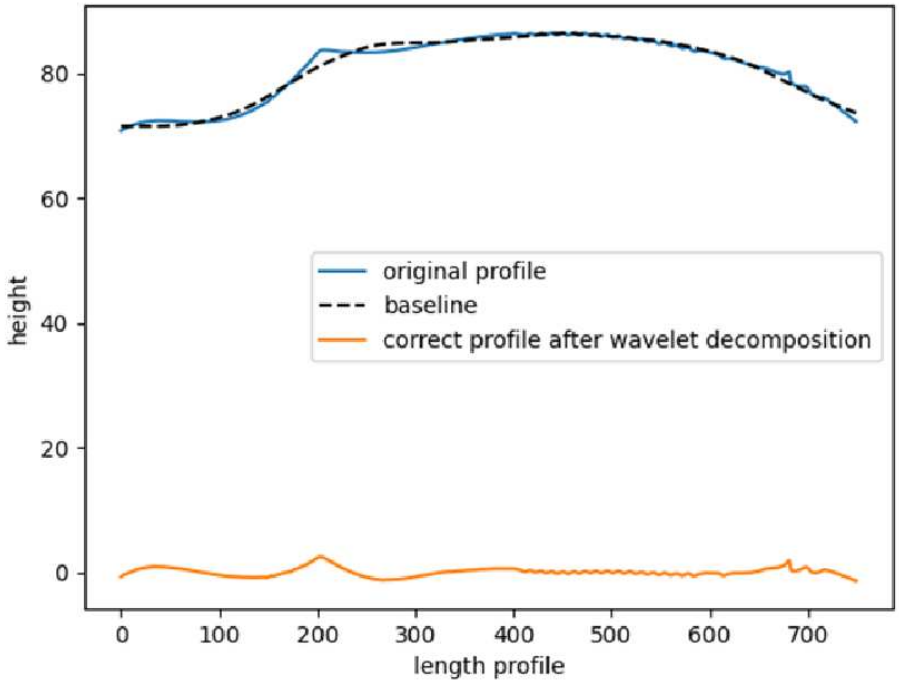


Figura 3.35: Baseline correction basato wavelet

Si può notare che la baseline correction con decomposizione su wavelet restituisce una profilo corretto senza passare per l'estrazione della baseline, come spiegato in 3.4.2, ma in questo paragrafo viene forzato il calcolo della baseline per poter riportare un confronto tra le due metodologie di baseline correction.

In conclusione è consigliato usare ALS come tecnica per baseline correction e dal *config.ini* 3.3 è possibile selezionare entrambe le tecniche, intervenendo nella sezione [general] e modificando la voce *baseline_engine*.

Figura 3.36: Rendering immagine con baseline correction basate su ALSS

Figura 3.37: Rendering immagine con baseline correction basata du decomposizione wavelet

Nelle figure 3.36 e 3.37 possono essere notate alcune differenze.

- **Brillantezza** La seconda immagine risulta più chiara della seconda. Possiamo dire che la seconda immagine è la più brillante tra le due.
- **Omogeneità** Nella prima foto notiamo una regolarità maggiore. Nella seconda sono presenti fasce orizzontali di intensità diverse e contrastanti. Precisamente; una fascia scura sul DOT e alternarsi di fasce chiare e scure sopra le scritte HRS Winter. Questa variazione di grigio caratterizza un andamento ondulatorio della superficie. Questo è un comportamento che vogliamo non sia presente nelle nostre immagini.
- **Nitidezza DOT** Varia anche la nitidezza del DOT. Nella prima immagine abbiamo un DOT nitido rispetto lo sfondo con la fascia bianca sottostante che non disturba. Situazione diversa per la seconda immagine. Qui abbiamo un DOT meno nitido e con una fascia bianca molto presente.

Date le differenze riscontrate e considerando matrici di dimensioni originali (16000x4096), si può affermare che in questo caso per il nostro progetto la tecnica di baseline correction consigliata è quella basata su l'ALSS.

3.4.3 Immagine in scala di grigi e immagine binaria

Nei paragrafi precedenti sono state presentate immagini in scala di grigi, ad esempio 3.36 3.37 ed immagini binarie, ad esempio 3.40 3.41. Ora andremo a descrivere il processo di rendering, partendo dalla matrice di punti per arrivare ad un'immagine in scala di grige e binaria.

Va ricordato che ci siamo concentrati in una sotto-porzione di matrice, ovvero nella porzione associata alla fascia laterale che contiene perfettamente le scritte. Quindi prima di passare al rendering, è stata effettuata una operazione di cropping[5], possiamo fare questo perché ci interessa la sola presenza della sigla DOT, quindi è inutile continuare a ragionare sull'intera scansione.

Questa scelta ha portato alcuni vantaggi.Uno riguarda la velocità nel tempo di elaborazione, ora ci troveremo a lavorare con un minor numero di punti. Concentrandoci su una porzione cosi fine, ci rendiamo subito conto che le differenze tra le due tecniche di baseline correction vengo parzialmente annullate[6].

Partendo da una matrice di 16000 righe 4096 colonne[7], abbiamo effettuato un cropping tra le colonne numero 1660 e 1770, riducendo il numero di colonne a sole 110. Dopo il ridimensionamento ci troveremo a dover gestire un numero minore di punti.

Con la nuova matrice passiamo alla sua linerizzazione per poi normalizzarla sul range $[0-255]$[8])

Partendo dalle tecniche di baseline correction già introdotte, cerchiamo di ottenere un profilo di baseline da usare per linerizzare ogni singolo profilo. Quindi verrà calcolata una sola baseline, ottenuta dal profilo 0 della matrice, e a sua volta sottratta ad ogni altro profilo linearizzandolo [9].

Infine la matrice aggiornata verrà normalizzata considerando i valori di minimo e di massimo.

$$normalizate(i) = \frac{i - MIN}{MAX - MIN} \times 255 \tag{3.22}$$

Dove i è l'i-esimo punto da normalizzare, MIN e MAX i valori di minimo e massimo nel intera matrice di punti. Il 255 è presente per ottenere un scala di grigi proporzionata.

[5] Operazione di ritaglio. Eseguendo un cropping del genere, ci troveremo a ragionare su profili più piccoli.

[6] Questo perché ci troviamo a linearizzare una porzione di profilo che ha una andamento concavo meno marcato. Quindi l'operazione di linearizzazione risulta nettamente facilitata

[7] Ricordiamo che il profilometro effettua acquisizioni "verticali".

[8] Volendo una scala di grigi, abbiamo mantenuto un solo canale colore.

[9] Ricordiamo che nei nostri test abbiamo utilizzato la ALSS. Nel caso di decomposizione tramite Wavelet non servirà effettuare differenze tra baseline e profilo originale

Figura 3.38: Immagine in scala di grigio con ALSS dopo cropping

Figura 3.39: Immagine in scala di grigio con ALSS dopo cropping

Riportiamo l'esempio di due immagini ottenute tramite baseline correction ALSS 3.38 e Wavelet 3.39 partendo dalla stessa acquisizione. Le due immagini risultato simili, è difficile trovare delle differenze. La mancanza di notevoli differenze è dovuto al fatto che ora ci troviamo a linearizzare profili più piccoli e con una concavità meno accentuata rispetto agli originale. Quindi il peso dell'elaborazione dovuta al tipo di baseline correction viene in parte annullato. Possiamo affermare che le differenze su queste due immagine sono minori rispetto al caso 3.4.2.

Dopo le considerazioni fatte, possiamo dire che per questo progetto lavorando con matrici ritagliate attorno alla scritta DOT, non notiamo sostanziali differenze tra le due tecniche di Baseline correction.

Per quanto riguarda le immagini binarie, ci troviamo nella situazione di applicare un operato di sogliatura che riesca ad estrarre i singoli caratteri rispetto lo sfondo. Abbiamo optato per l'utilizzo di un thersholding adattivo, che riesca ad applicare una sogliatura che si adatti alla natura morfologica della immagine. In particolare abbiamo optato per il metodo `adaptiveThreshold()` di OpenCV.

`adaptiveThreshold()` trasforma un'immagine in scala di grigi in un'immagine binaria applicando una soglia adattiva, seguendo le formule: (8)

- **Theshold binario**

$$dst(x,y)\begin{cases} MAX\_VALUE & src(x,y) > T(x,y) \\ 0 & otherwise \end{cases} \tag{3.23}$$

- **Threshold binario invertito**

$$dst(x,y)\begin{cases} 0 & src(x,y) > T(x,y) \\ MAX\_VALUE & otherwise \end{cases} \tag{3.24}$$

Dove $src(x,y)$ è un pixel all'interno della immagine di input, $dst(x,y)$ è un pixel dell'immagine di output, $T(x,y)$ è il threshold calcolato individualmente per ogni pixel e MAX_VALUE è il valore da assegnare ai punti sopra la soglia di threshold.

Un parametro chiave è *adaptiveMethod* con il quale si può scegliere l'algoritmo che implementa il threshold adattivo. Si hanno due approcci:

- **ADAPTIVE_THRESH_MEAN_C** Dove la soglia $T(x, y)$ è il valore medio calcolato su una matrice *blocksize* X *blocksize* - C rispetto al punto (x, y). Quindi con il crescere di blocksize aumenta la matrice su cui calcolare il valore medio.
- **ADAPTIVE_THRESH_GAUSSIAN_C** Dove la soglia $T(x, y)$ è una somma pesata (sfruttando una distribuzione gaussiana) della matrice $blocksizeXblocksize - C$ calcolata sui vicini del punto (x, y)

In entrambi i casi *blocksize* è la distanza del punto (x,y) dai sui vicini e C una costante da sottrarre. Il risultato della sogliatura adattiva varia al variare del algoritmo scelto e anche dai valori *blocksize* e C. Nel nostro caso abbiamo usato l'algoritmo ADAPTIVE_THRESH_MEAN_C con $blocksize = 23$ e $C = 2$.

Passiamo a presentare alcuni risultati su immagini binarie. Nel renderizzare un'immagine binaria sono possibili 2 scelte; la prima riguarda la dimensione della matrice di acquisizione di partenza, la seconda riguarda la tecnica di baseline correction da usare. Cosi possiamo distinguere 4 casi distinti e ne riportiamo un'analisi.

La 4 possibili situazioni sono:

- Immagine binaria ottenuta da una acquisizione dell'intera spalla e baseline correction con ALSS 3.40
- Immagine binaria ottenuta da una acquisizione dell'intera spalla e baseline correction con decomposizione Wavelet 3.41
- Immagine binaria ottenuta da una acquisizione ritagliata sulle scritte e baseline correction con ALSS 3.42
- Immagine binaria ottenuta da una acquisizione ritagliata sulle scritte e baseline correction con decomposizione Wavelet 3.43

Per i casi che riguardano l'intera spalla 3.40 e 3.41, il threshold adattivo ha restituito delle immagini molto simili tra loro, da una analisi visiva risulta facile trovare il DOT. I caratteri sono ben marcati e nitidi, con dei contorni netti senza sbavature. Guardando alle scritte è difficile trovare differenze sostanziali tra le due immagini. Piccole variazioni si possono scorgere sul rendering della texture attorno i loghi *HRS* e *Winter*, ma anche qui si tratta di inezie.

Quindi possiamo concludere che per quando riguarda il risultato del threshold adattivo su acquisizioni della spalla intera non riusciamo ad individuare delle differenze enormi tra le due tecniche di linerizzazione.

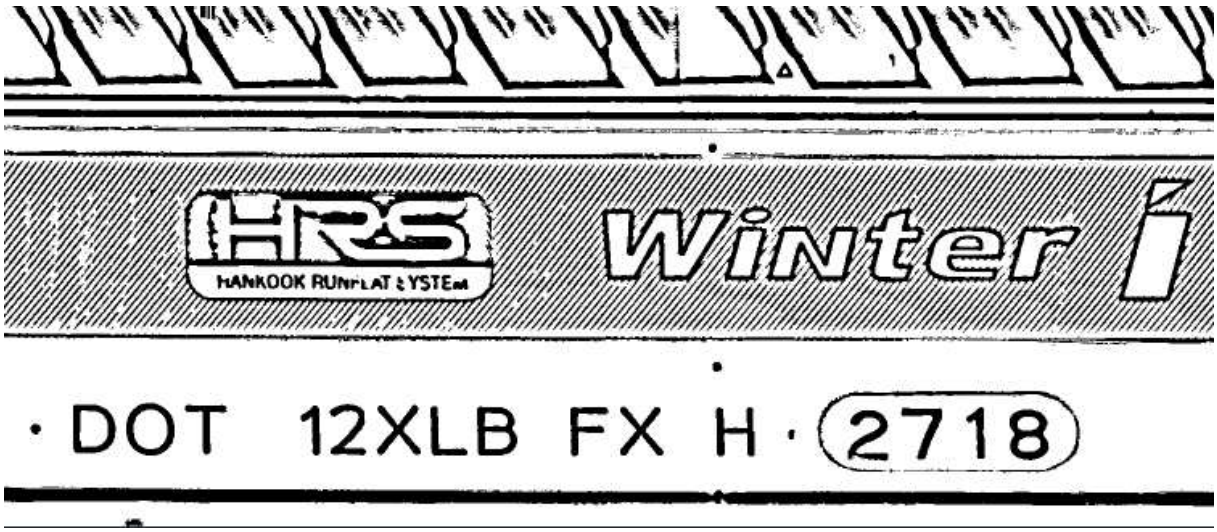


Figura 3.40: Immagine binaria ottenuta da 3.36 (ALSS)

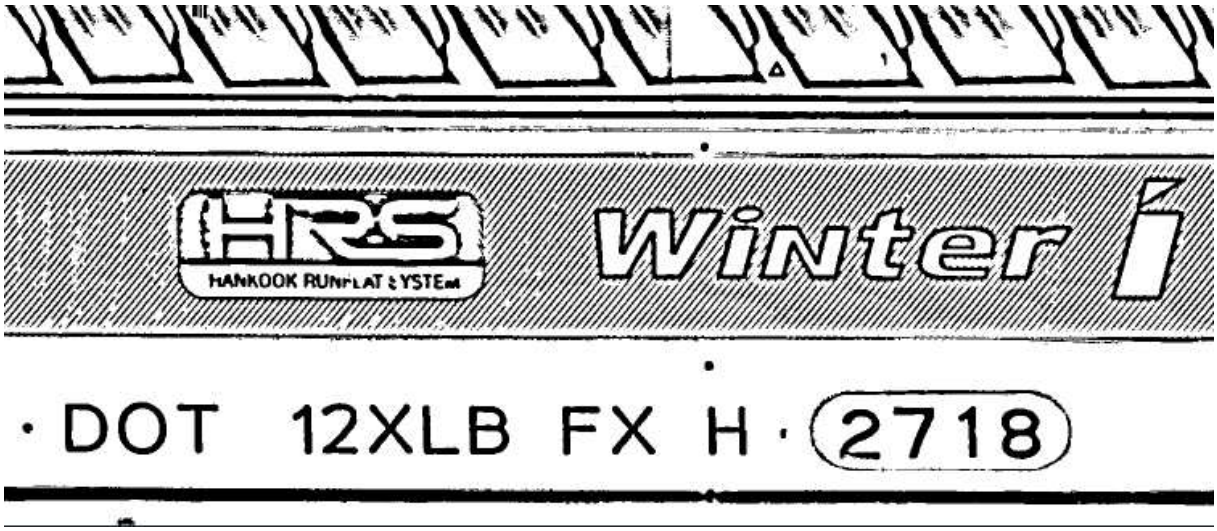


Figura 3.41: Immagine binaria ottenuta da 3.37 (Wavelet)

Mettendo a confronto le immagini binarie ottenute dall'acquisizione delle sole scritte 3.42 e 3.43, tra le due tecniche di linearizzazione non riusciamo ad individuare differenze sostanziali. In entrambe DOT risulta nitido con contorni ben definiti. Non riusciamo a trovare delle differenze che ci possano far dire quale delle due immagini binarie sia la migliore. Quindi, come nel caso precedente, anche con acquisizioni ritagliate non si hanno differenze notevoli tra le due tecniche di linearizzazione.

Il paragone ha esito differente se confrontiamo immagini ottenute della stessa tecnica di linerizzazione, ma variando il tipo di acquisizioni. Riportiamo il confronto tra 3.40 e 3.42. Le differenze riguardano i dettagli nel codice DOT. Notiamo come in 3.40 si hanno caratteri leggermente meno definiti e marcati rispetti a quelli in 3.42. Volendo fare due esempi puntuali. Si può notare come la bacchetta orizzontale del numero 2 nella prima immagine sia molto frastagliata e più fine rispetto a quella nella seconda immagine. Sempre restando sui numeri, la aureola che li circonda risulta più corposa e definita nella seconda foto.

Concludendo questa analisi, affermiamo che a parità di tecniche di linearizzazione non si hanno

differenze sostanziali sulle immagini derivanti, ma si notato differenze più marcate al variare della dimensione dell'acquisizione. Con acquisizioni più strette si ottengo dei risultati migliori. Di conseguenza, per ottenere un DOT più dettagliato e nitido si consiglia di lavorare su acquisizioni stretta sulle sole scritte.

Figura 3.42: Immagine binaria ottenuta da 3.38 (ALSS)

Figura 3.43: Immagine binaria ottenuta da 3.39 (Wavelet)

3.4.4 Trasformazioni morfologiche

Un ulteriore elaborazione riguarda le trasformazioni morfologiche su immagini binarie . Le trasformazioni potrebbero essere applicate cercando di migliorare la resa dei caratteri. Un esempio potrebbe essere quello di volere caratteri più spessi e quindi con le trasformazione di Dilation e Erode si riuscirebbe a gestire lo spessore degli elementi nell'immagine. Oppure si vogliono eliminare dei piccoli graffi attorno le scritte, con operazioni di Closing e Opening riusciamo a gestire molto del rumore indesiderato.

Riportiamo degli esempi, a nostro parere validi, per comprendere quali possono essere alcune delle trasformazioni da applicare.

Come si può notare dalle figure 3.44 3.45 3.46 3.47 3.48 applicando le giuste trasformazioni è possibile modellare l'immagine mettendo in risalto caratteristiche specifiche rispetto ad altro. Un insieme di erosione, opening e closing potrebbe essere molto utile per gestire e rimuove rumore o elementi indesiderati. Nel nostro esempio, una erosione riuscirebbe a rimuovere parzialmente l'aureola che circonda il numero 2718 3.44. D'altro canto, con un coretto utilizzo della dilation si riuscirebbe a metter in risalto gli oggetti di maggior interesse in un'immagine. Un esempio potrebbe essere, voler marcare maggiormente la presenza del codice DOT, rendendo i caratteri più spessi, il tutto potrebbe essere fatto molto facilmente.

Figura 3.44: Erode su 3.42

Figura 3.45: Dilation su 3.42

Figura 3.46: Skeleton su 3.42

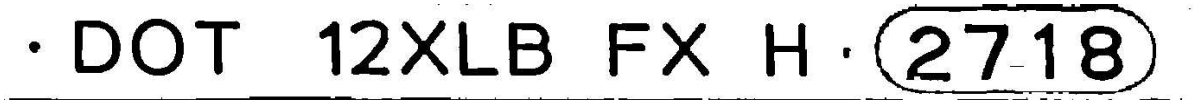


Figura 3.47: Opening su 3.42

Figura 3.48: Closing su 3.42

Inoltre possiamo gestire l'effetto delle trasformazioni personalizzando la matrice di kernel e sul numero di iterazioni sulla immagine. Lavorando sulla sua dimensione riusciamo ad avere risultati diversi, esempi in figura 3.51 e 3.52 dove con una sola iterazione ma con due kernel differenti abbiamo risultati diversi. Possiamo notare come nella seconda figura i caratteri siano più spessi con un effetto di dilatazione maggiore. Nel secondo esempio 3.49 e 3.50 con una erode, abbiamo un comportamento speculare al esempio precedente. Con l'aumentare della dimensione della matrice di kernel aumenta anche l'intensità della trasformazione.

Figura 3.49: Erode su 3.42 matrice di kernel 3x3

Figura 3.50: Erode su 3.42 matrice di kernel 7x7

Figura 3.51: Dilation su 3.42 matrice di kernel 3x3

Figura 3.52: Dilation su 3.42 matrice di kernel 7x7

Una ulteriore elaborazione, potrebbe riguardare la possibilità di direzionare la trasformazione. Con una matrice kernel rettangolare si riesce ad direzionare l'effetto da applicare, un esempio

è riportato in figura 3.53. Usando una matrice con un numero di colonne maggiore rispetto le righe è possibile avere una dilatazione orizzontale.

Figura 3.53: Dilation su 3.42 matrice di kernel 3x7

3.5 OCR

L'individuazione del DOT passa dalla analisi dell'immagine binaria.

Dovendo lavorare con immagini molto più lunga che alta, i modelli OCR non restituiscono dei risultati molto accurati. Quindi per ovviare a questo problema si è deciso di implementare due sistemi di supporto all'OCR distinti tra di loro ;

- **Sistema di tiling** Si basa sul'idea di analizzare tramite OCR piccole porzioni di sotto immagini, facendo scorrere una finestra di dimensioni costante lungo l'intera immagine binaria.
- **Edge detection** L'idea è di individuare all'interno della immagine dei blobs in corrispondenza delle scritte, e solo a quel punto di passare al OCR i soli blobs che rispecchiano delle precise caratteristiche geometriche.

3.5.1 OCR con sistema di tiling

La necessità di spezzare la scansione originale in sotto parti è stata necessaria per migliorare l'accuratezza dei modelli OCR. I modelli di OCR adottati restituiscono risultati non molto accurati se devono lavorare su immagini di dimensioni molto grandi, la loro accuratezza migliora notevolmente se gli vengono date in input immagini dalle dimensioni contenute. Quindi, si è deciso di dare in input ai modelli OCR piccole porzioni successive della immagine originale, utilizzando un sistema basato su tiles (finestre o mattonelle) con stride (passo).

La grandezza di tiles e strides sono state settante dopo alcune considerazioni; consapevoli che la dimensione in larghezza della sigla "DOT" è di circa 300 pixel e sapendo che con l'aumentare delle tiles da iterare, aumenta anche il tempo di elaborazione (il numero di iterazioni è legato alla dimensione delle tiles).

La soluzione trovata è un compromesso. Si è deciso di usare tiles di larghezza pari a 1000 pixel (la altezza non è un problema, dato che ci stiamo concentrando sulla fascia delle solo scritte) e stride di 500 pixel. Avendo scansioni di 16000 pixel in larghezza riusciamo ad iterare nel caso peggiore su 30 tiles (16 con stride dispari e 14 su stride pari).

Lo stride è la metà di una tile, questo perché con lo scorrere del'immagine non rischiamo di perderci informazione utile. Nel caso di stride nullo potrebbero verificarsi situazioni dove DOT si trova esattamente in mezzo a due tiles consecutive. Ciò implica che la scritta non potrebbe essere analizzata correttamente. Perché verrebbe spezzata in due, con una metà nella tile di sinistra e l'altra nella tile di destra. Il problema si risolve facilmente con una stride di 500 pixel. Considerando sempre il caso sfortunato di prima, con questo stride si riuscirebbe ad avere una tile in mezzo che riuscirebbe ad incorporare perfettamente l'intera scritta DOT.

In figura 3.54 è sintetizzato l'intero funzionamento. A, B e C sono esempi di tiles, S è lo stride tra una finestra e la successiva.

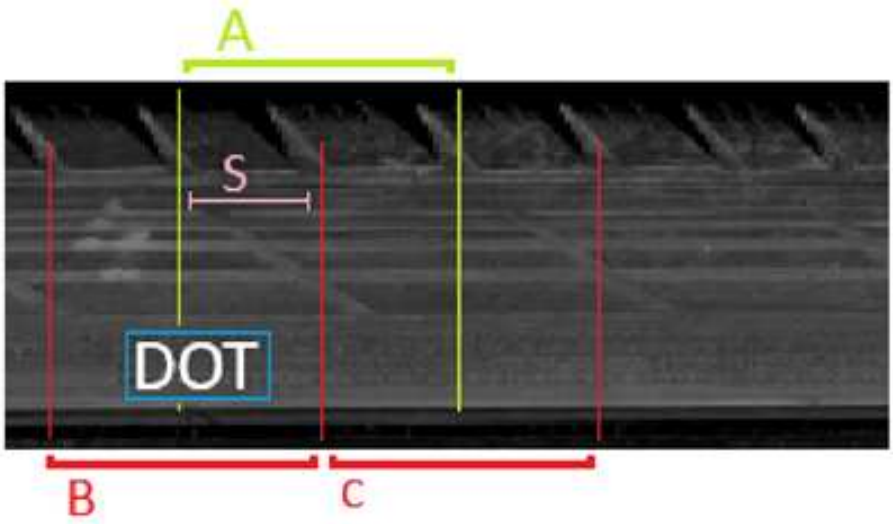


Figura 3.54: Sintesi sistema di tiling

Il numero di tiles in cui verrà divisa l'immagine è fortemente legato alla posizione della sigla DOT. Nel coso di immagine con DOT all' estrema sinistra, dopo poche iterazioni il sistema individuerà la tile fortunata. Il caso peggiore si verifica se la scritta DOT si trova alla estrema destra, qui sarà necessario scorrere l'intera immagine aumentando notevolmente il tempo di elaborazione. Quindi, si può affermare che la situazione desiderabile è riuscire ad acquisire lo pneumatico in maniera da avere il DOT ad inizio scansione, ma questo resta al momento una situazione utopica perché sarebbe necessario conoscere a priori la posizione del DOT.

Di conseguenza possiamo dire che il numero di iterazioni di tiles sull'immagine è casuale, legato se non altro all'orientamento con cui lo pneumatico arriva in postazione di acquisizione.

Questo approccio legato a tile, potrebbe introdurre una situazione spiacevole. La situazione da evitare sarebbe quella in cui, sfortunatamente, l'acquisizione dello pneumatico inizia sul DOT. Cosi il DOT risulterebbe spezzato in due sulla scansione finale. La porzione finale del DOT si troverebbe ad inizio scansione e la restante parte in coda alla scansione **??**. Per ovviare a questo fastidioso problema, si è deciso di acquisire più profili di quelli necessari per una acquisizione completa. Infatti, una scansione completa si otterrebbe con soli 14960 profili, ma per evitare la problematica di un DOT spezzato si è deciso di catturare 16000 profili. Aumentare il numero di profili ci consente una maggiore sicurezza. Se anche iniziamo acquisendo un DOT spezzato, siamo sicuri che riusciremo a acquisire il DOT completo a fine ciclo. Certo, catturare profili in più implica aumentare il tempo di acquisizione, ma il tempo aggiunto resta comunque contenuto quindi sacrificabile per l'enorme vantaggio ottenuto.

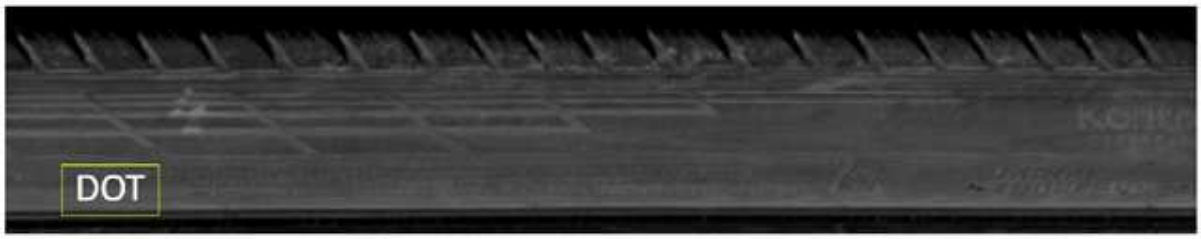


Figura 3.55: Caso di DOT ad inizio scansione

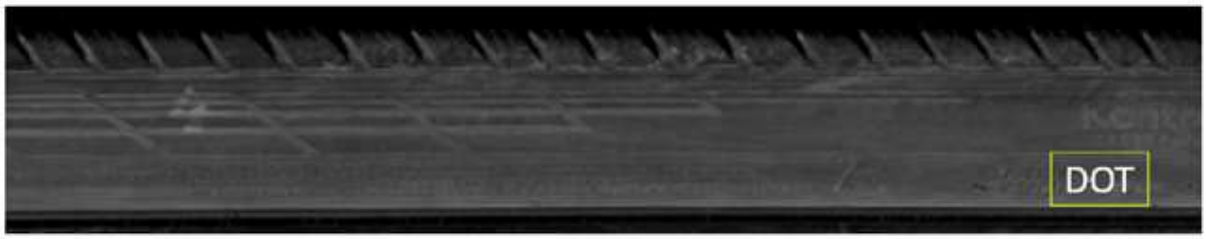


Figura 3.56: Caso di DOT a fine scansione

Figura 3.57: Caso di DOT spezzato a metà

Nella prima figura 3.55 vi è un esempio di DOT ad inizio scansione (caso migliore), ci saranno poche iterazioni. Nella seconda figura 3.56 il caso di DOT a fine scansione (caso peggiore), ci saranno molte iterazioni. Ultima figura 3.57, caso con DOT spezzato dalla scansione e A rappresenta la porzione di scansione acquisita in più.

Il sistema di tiling procede nel seguente modo: ogni finestra estratta viene data in pasto al modello OCR cosi che possa comprendere se vi è presente la scritta “DOT”. Nel momento che viene individuata la finestra contente “DOT”, il sistema di scorrimento si ferma e ritorna il testo decifrato e le coordinate della sua bounding box. Il testo decifrato sarà utile per comprendere se la scritta individuata è esattamente quella che cercavamo, la bounding box sarà utile per individuare un punto come origine per mappare le restanti scritte dello pneumatico.

A seguire saranno dettagliati i passaggi chiave del tiling utilizzato:

- La dimensione delle tiles è impostata a 1000 pixel con stride di 500, come già spiegato in precedenza. L’analisi della scansione inizia da sinistra verso destra. Il software inizia a sezionare l’immagine nelle varie tiles.
- I modelli OCR utilizzati sono Tesseract e EasyOCR. Entrambi restituiranno il testo decifrato e i punti necessari per disegnare una bounding box attorno alla scritta. Nello specifico Tesseract restituisce le coordinate X e Y del punto in alto a sinistra della BB con larghezza e altezza con la quale sarà facile calcolare i restanti 3 punti. EasyOCR ritorna le coordinati X e Y dei 4 angoli della BB.

  Da entrambi gli OCR è possibile ottenere le coordinate del angolo in alto e sinistra della BB all’interno della tile. Successivamente saranno calcolate le coordinate assolute all’interno dell’immagine originale, conoscendo in numero della tile vincente.
- Ogni tile viene analizzata e successivamente viene fatto un check sul testo decifrato, se il check è positivo si ferma immediatamente la scansione andando a disegnare la

boundingbox. La condizione del check è molto semplice, viene controllato che il testo estratto sia uguale alla stringa “DOT” (maiuscolo senza spazi).

- Il sistema si ferma al primo DOT incontrato, perché su uno pneumatico sarà presente un solo codice DOT. Quindi, sarebbe inutile continuare a cercarne un altro sapendo che si tratterebbe sempre dello stesso codice, ed inoltre una singola acquisizione copre una sola volta l’intero pneumatico.

In conclusione con questo sistema di analisi riusciamo ad estrarre coordinate relative a assolute del DOT in una specifica scansione, sapendo che la presenza del DOT è univoca e non cambia.

Esempio

Da sistema di tiling, otteniamo come primo risultato la tile contenente il DOT. La tile incorpora il DOT e altri elementi, questo era facilmente intuibile sapendo che stiamo gestendo finestre da 1000 pixel. In questo caso ci troviamo il DOT presente sulla destra, con altre tre sigle alfanumeriche 3.58.

Figura 3.58: Tiling OCR: tile con DOT

Quindi il modello OCR si troverà a dover controllare la totalità delle scritte nella tile, e ad estrarre la sola scritta DOT con la propria bounding box. Di conseguenza di risultato ottenuto 3.64.

Figura 3.59: Tiling OCR: risultato finale (EasyOCR)

Si può notare come la bounding box disegnata accerchia correttamente solamente il DOT.

Da questo esempio possiamo notare che il sistema di tiling risponde perfettamente alle nostre esigenze, risulta un sistema utile come supporto ai modelli OCR, per individuare il DOT e a determinare la sua posizione all’interno dalla immagine originale.

3.5.2 OCR con edge detection

Questo secondo metodo, sfrutta un’idea molto semplice. Riuscire ad individuare regioni(blobs) ben distinte dal resto della immagine associate direttamente alle varie scritta. Per decifrare correttamente il DOT, si danno in input al modello OCR la successione dei blobs trovati

(saranno estratti sfruttando la hierarchy[10][11] restituita dal metodo `findContours()` di OpenCV). Quindi il modello lavorerà con porzioni di immagini contenute, migliorando notevolmente la sua accuratezza. Nel momento in cui avrà trovato DOT sarà possibile costruire un sistema di riferimento partendo dalla posizione della blob all'interno dell'immagine.

Introduciamo una panoramica sui principali approcci di edge detection con una particolare attenzione all'algoritmo di Canny.

Edge detection

Nella visione l'edge detection è una tecnica di elaborazione che consente di individuare i bordi di oggetti all'interno di immagini. Funziona rilevando discontinuità nella luminosità. Il rilevamento dei bordi consente all'utente di concentrarsi in porzioni dell'immagine, riducendo la quantità di dati da analizzare, preservando le proprietà strutturali dell'immagine.

Un edge (o bordo) è un cambiamento improvviso di intensità, una discontinuità nella luminosità o nel contrasto dell'immagine. Solitamente i bordi si verificano al confine di due regioni. Vengono individuati per aiutare l'analisi di immagini digitali: (28)

- Possono estrarre caratteristiche importanti di un'immagine come angoli, linee e curve.
- Possono fornire forti indizi visivi che possono aiutare il processo di riconoscimento di oggetti o pattern.
- Possono ridurre le informazioni non necessarie nell'immagine preservando la struttura dell'immagine.

La edge detection passa attraverso l'utilizzo di maschere. Al cambiare della maschera variano i bordi individuati. Quindi, possiamo estrarre specifici bordi applicando una particolare maschera. Dunque, in base alla maschera abbiamo diversi operatori:

Operatore Prewitt l'operatore Prewitt viene utilizzato per il rilevamento di bordi orizzontali e verticali.

- **Bordi verticali** La maschera utilizzata determina bordi verticali. Questo perché la maschera ha la colonna centrale di soli zeri. Quindi utilizzare questa maschera, ci darà i bordi verticali in un'immagine.
  Matrice per rilevamento bordi verticali: $\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$
- **Bordi Orizzontali** La maschera utilizzata determina bordi orizzontali. La maschera ha la riga centrale di soli zeri. Quando si utilizzerà questa maschera saranno messi in rilievo i bordi orizzontali.

[10] hierarchy è un vettore contenente informazioni sulla topologia dell'immagine. Ha tanti elementi quanti sono i contorni presenti nel immagine. Il vettore organizza i vari contorni creando una gerarchia tra loro.

[11] Contorni successivi e precedenti si trovano sullo stesso livello gerarchico, invece contorni genitori e figli si troveranno su livelli gerarchici susseguenti.

Matrice per rilevamento bordi orizzontali: $\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$

Operatore Sobel L'operatore Sobel è molto simile all'operatore Prewitt. Come l'operatore Prewitt, anche l'operatore Sobel viene utilizzato per rilevare bordi verticali e orizzontali. La differenza principale è che nell'operatore Sobel i coefficienti delle maschere non sono fissi e possono essere regolati secondo le proprie esigenze.

- **Bordi verticali**

  Esempio di una matrice per rilevamento bordi verticali: $\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$

  Questa maschera funziona esattamente come la maschera verticale dell'operatore Prewitt. C'è solo una differenza: ha i valori "2" e "-2" al centro della prima e della terza colonna. Quando applicata su un'immagine, questa maschera evidenzierà i bordi verticali.

- **Bordi orizzontali**

  Esempio di una matrice per rilevamento bordi orizzontali: $\begin{pmatrix} -1 & 2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$

  La maschera sopra troverà i bordi orizzontali e ha la colonna centrale di soli zeri. Quando si utilizzerà questa maschera su un'immagine, verranno risaltati bordi di direzione orizzontali. L'unica differenza rispetto la matrice di Prewitt sono gli elementi centrali della prima e della terza riga.

bussola Robinson Utilizza una maschera chiamata Bussola di Robinson. La caratteristica principale è che può avere 8 orientamenti a ricordare il comportamento di una bussola. Riesce ad estrarre bordi rispetto le 8 direzioni.

Non esiste una maschera fissa, ma presa una maschera questa viene ruotata tenendo in considerazione l'orientamento dei bordi che si vogliono trovare. Facciamo un esempio.

$$NordOvest: \begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -10 & 0 \end{pmatrix} \quad Nord: \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \quad NordEst: \begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

$$Ovest: \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \quad Est: \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

$$SudOvest: \begin{pmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{pmatrix} \quad Sud: \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \quad SudEst: \begin{pmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{pmatrix}$$

Come si può vedere, la direzione lungo la quale individuare i bordi fa capo alla posizione degli 0 all'interno della matrice.

bussola di Kirsh La bussola di Kirsh è un operatore non lineare che riesce a determinare bordi lungo direzioni prestabilite. Prende il nome da Russell A. Kirsch.

Come per il caso Robinson abbiamo 8 maschere che hanno caratteristiche specifiche per ogni direzione.

$$NordOvest: \begin{pmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{pmatrix} \quad Nord: \begin{pmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{pmatrix} \quad NordEst: \begin{pmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{pmatrix}$$

$$Ovest: \begin{pmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{pmatrix} \quad Est: \begin{pmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{pmatrix}$$

$$SudOvest: \begin{pmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{pmatrix} \quad Sud: \begin{pmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix} \quad SudEst: \begin{pmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix}$$

Operatore Laplaciano Laplaciano determina bordi sfruttando evidenti discontinuità di intensità sulla scala di grigi. Di conseguenza, questa operazione produce immagini con bordi su vari livelli di grigio (introducendo molte discontinuità) su sfondo scuro.

Inoltre si posso distinguere due approcci distinti: Laplaciano positivo e Laplaciano negativo. Non possiamo applicare sia l'operatore Laplaciano positivo che quello negativo sulla stessa immagine. Dobbiamo applicarne solo uno alla volta. La cosa da ricordare è che se si applica l'operatore Laplaciano positivo, dobbiamo sottrarre l'immagine risultante dall'immagine originale per ottenere un'immagine più nitida. Allo stesso modo, se si applica l'operatore Laplaciano negativo, dobbiamo aggiungere l'immagine risultante all'immagine originale per ottenere l'immagine più nitida

Canny L'algoritmo proposto da Canny e si basa su immagini in scala di grigio. Pertanto, il prerequisito è convertire l'immagine originale in scala di grigi prima di applicare l'algoritmo. L'algoritmo, ad oggi, è considerato il miglior algoritmo di rilevamento dei bordi per immagini influenzate da rumore e può essere suddiviso in cinque diversi passaggi: (21)

- **Filtro Gaussiano** All'immagine viene applicato un filtro Gaussiano di dimensioni $(2k+1)x(2k+1)$:

$$G(i,j) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{(i-k-1)^2+(j-k-1)^2}{2\sigma^2}} \tag{3.25}$$

dove $1 \leqslant i, J \leqslant 2k+1$). Questo passaggio sfoca leggermente l'immagine per ridurre gli effetti del rumore sul rilevamento dei bordi e per prevenire falsi rilevamenti. Maggiore è la dimensione del kernel gaussiano, minore è la sensibilità al rumore. D'altro canto, l'errore di localizzazione dei bordi è leggermente maggiore all'aumentare della dimensione del kernel del filtro gaussiano, poiché l'effetto di sfocatura è più forte. Un kernel 5×5 (cioè k = 2) è una buona dimensione per la maggior parte dei casi, ma varia anche a seconda delle situazioni specifiche.

- **Calcolo gradiente** Il secondo passo consiste nel calcolare il gradiente dell'immagine smussata. Inizialmente, viene applicato un operatore che ritorni le derivate prime lungo le direzioni orizzontali I_x e verticali I_y. Il gradiente sull'intensità dell'immagine è determinata come:

$$I = \sqrt{{I_x}^2 + {I_y}^2} \tag{3.26}$$

  Un bordo in un'immagine può puntare in una varietà di direzioni, e l'algoritmo di Canny cerca di individuare bordi orizzontali, verticali e diagonali. A questo scopo, l'angolo del gradiente viene determinato come:

$$\theta = tan^{-1}\frac{I_y}{I_X} \tag{3.27}$$

  ed è arrotondato a uno dei quattro angoli che rappresentano la verticale, l'orizzontale e le due direzioni diagonali ($0\circ$, $45\circ$, $90\circ$ e $135\circ$)

- **Soppressione non massimale** Il terzo passo consiste nell'applicazione di una tecnica di assottigliamento dei bordi. Ovvero, la soppressione non massimale mira a sopprimere i valori del gradiente impostandoli a 0, tranne i massimi locali che indicano un cambiamento più marcato dell'intensità di grigio. Per fare ciò, l'algoritmo controlla se per ciascun pixel diverso da zero, il gradiente ha un valore maggiore rispetto ai sui vicini lungo la direzione del gradiente; in tal caso, il pixel viene mantenuto invariato, altrimenti viene impostato a 0.

- **Threshold con doppia soglia** Dopo l'applicazione della soppressione non massima, i pixel dei bordi rimanenti forniscono una rappresentazione più accurata dei bordi reali. Tuttavia, sono ancora presenti alcuni pixel sui bordi indesiderati, causati dal rumore. Quindi è necessario filtrare i pixel con un gradiente debole preservando solo i pixel con valori di gradiente elevato. Ciò si ottiene selezionando e applicando due valori di soglia, alta e bassa. Se il valore dell'intensità del gradiente di un pixel è superiore al valore della soglia alta, il pixel viene contrassegnato come pixel del bordo . Se il valore dell'intensità del gradiente di un pixel del bordo è inferiore al valore di soglia superiore e maggiore del valore di soglia inferiore, il pixel viene contrassegnato come pixel del bordo debole. Se il valore dell'intensità del gradiente è inferiore al valore della soglia bassa, il pixel viene soppresso. È ovvio che quando i valori di soglia sono più alti, l'immagine risultante presenta meno falsi bordi ma maggiori spazi tra i bordi stessi.

- **Isteresi** Infine, il quinto passo dell'algoritmo del bordo Canny è il tracciamento del bordo mediante isteresi, per rimuovere gli spazi tra i segmenti del bordo e portare tutti i pixel del bordo finale allo stesso valore di intensità. Tutti i pixel contrassegnati come pixel del bordo forte nel quarto passaggio dell'algoritmo, saranno sicuramente coinvolti nell'immagine del bordo finale. Un pixel del bordo debole, invece, viene mantenuto solo quando riempie uno spazio tra i pixel del bordo forte.

  In particolare, per tracciare le connessioni dei bordi per ogni pixel del bordo debole, l'algoritmo considera una porzione 3×3 dell'immagine centrata su di esso e controlla se tra gli otto pixel che circondano il pixel del bordo debole centrale, c'è almeno un pixel del bordo forte: in in tal caso, il pixel del bordo debole viene identificato come

uno da preservare e il suo valore viene modificato nel valore del pixel del bordo forte.

Esempio

Come per il sistema di tiling, riportiamo un esempio. In questo caso, il sistema applica più operati all'immagine binaria. Quindi riportiamo i risultatati dei vari operatori impiegati.

Figura 3.60: Edge OCR: immagine di partenza

Partendo da 3.60, è applicata una trasformazione di Dilation per incrementare lo spessore dei caratteri, il risultato è un insieme di blobs con la quale è facile individuare le regione di ogni scritta 3.61.

Figura 3.61: Edge OCR: dilation

Gli stessi blobs saranno elaborati per estrarne i contorni (edge detection), quindi tramite l'algoritmo di Canny si riesce ad ottenere l'immagine in figura 3.62. I contorni, appena estrapolati, sono disegnati in verde sull'immagine binaria.

Figura 3.62: Edge OCR: contours detection

Una volta riusciti a determinare i contorni dei blobs, è possibile estrarre le regioni attorno le scritte che rispettano determinati criteri geometrici. Ovvero tutte la regioni che rispettano l'altezza e larghezza della scritta DOT. Il risultato in 3.63.

Figura 3.63: Edge OCR: region detection

In fine, il modello OCR dovrà lavorare soltanto con le regioni delimitate dalle boundingbox che rispettano i vincoli geometrici. Nello specifico manterrà e analizzerà solo i rettangoli attorno il DOT.

Figura 3.64: Edge OCR: risultato finale (EasyOCR)

Anche questo sistema, basato su edge detection, risulta essere un approccio valido come supporto ai modelli OCR. Partendo da un'immagine bianca e nera si riesce a determinare il DOT e la sua bounding box in modo che sia facile estrarne la posizione.

3.6 Risultati

Per verificare la capacità del software, l'abbiamo testato su 20 scansioni diverse. Le scansioni riguardano 10 pneumatici distinti, dove per ognuno sono stati acquisiti entrambi i lati. Tra i due lati esiste un differenza per la sigla DOT: considerando il pneumatico montato sul mezzo, il lato esterno rivolto verso la strada ha un DOT con le cifre settimana e anno di produzione. Invece il lato interno è privo di queste ultime cifre. Quindi, avendo a disposizione soli 10 pneumatici si hanno 2 acquisizioni per ognuno, lato esterno ed interno. Abbiamo potuto usare entrambi i lati senza compromessi, dato che eravamo interessati solamente ad individuare la sigla di 3 caratteri DOT, e questa resta identica sia sul lato esterno che interno.

Riportiamo i risultati in 4 tabelle, separati per tipologia di OCR e per sistema di supporto OCR impiegati. Quindi avremo:

- Risultati con Tesseract su tiling ed edge system
- Risultati con EasyOCR su tiling ed edge system

Su ogni tabella è presente il testo decodificato, inoltre sono riportate le coordinate in X e Y del punto in alto a sinistra della bounding box che racchiude DOT all'interno dell'immagine originale.

Passiamo a confrontare i risultati ottenuti con Tesseract e EasyOCR.

3.6.1 Risultati con Tesseract

Per confrontare i risultati ottenuti con l'utilizzo di Tesseract abbiamo 2 tabelle 3.6.1. In particolare vogliamo trovare delle differenze tra il tiling system e l'edge system, differenze con la quale è possibile determinare il sistema con maggiore accuratezza e precisione.

Subito notiamo che per tutte le scansioni analizzate si è riusciti ad individuare correttamente la scritta DOT, quindi possiamo dire che in questo caso entrambi i sistemi hanno dimostrato di possedere un ottima accuratezza.

Il discorso cambia leggermente per quanto riguarda la precisione nel individuare il punto in alto a sinistra della bounding box (terza e qurta colonna). Qui troviamo piccole differenze. Utilizzando il sistema basato sui bordi, notiamo come i punti sono sempre spostati leggermente di qualche pixel più in alto e a sinistra (parliamo mediamente di 5 pixel in X e circa 3 pixel in Y) rispetto al sistema di tiling.

Le differenze riscontrate restano irrisorie e non introducono una disparità marcata. Quindi possiamo dire che i due sistemi hanno un ottima precisione dato che riescono ad individuare delle coordinate pressoché equivalenti per le nostre scansioni.

Risultati con Tesseract (tiling system)			
N.	Testo	Top Left X [pixel]	Top Left Y [pixel]
1	"DOT"	10718	243
2	"DOT"	1629	243
3	"DOT"	9254	253
4	"DOT"	9908	253
5	"DOT"	5792	236
6	"DOT"	6541	254
7	"DOT"	10274	254
8	"DOT"	3217	238
9	"DOT"	8711	237
10	"DOT"	1553	237
11	"DOT"	12047	268
12	"DOT"	14508	264
13	"DOT"	1175	264
14	"DOT"	13678	271
15	"DOT"	4992	270
16	"DOT"	3202	242
17	"DOT"	12436	243
18	"DOT"	8059	250
19	"DOT"	12687	250
20	"DOT"	4445	267

Risultati con Tesseract (edge system)			
N.	Testo	Top Left X [pixel]	Top Left Y [pixel]
1	"DOT"	10716	240
2	"DOT"	1627	240
3	"DOT"	9252	250
4	"DOT"	9907	250
5	"DOT"	5789	233
6	"DOT"	6540	251
7	"DOT"	10271	251
8	"DOT"	3215	235
9	"DOT"	8711	240
10	"DOT"	1551	234
11	"DOT"	12052	265
12	"DOT"	14506	261
13	"DOT"	1172	261
14	"DOT"	13675	268
15	"DOT"	4989	267
16	"DOT"	3203	239
17	"DOT"	12434	240
18	"DOT"	8056	247
19	"DOT"	12685	247
20	"DOT"	4443	264

3.6.2 Risultati EasyOCR

Come nel paragrafo precedente, riportiamo i risultati ottenuti da EasyOCR. Anche qui abbiamo un ottima accuratezza, come si può notare dalle tabelle 3.6.2 si hanno risultati corretti per il testo decifrato, sia con il tiling system che con edge system. Un altra somiglianza con i risultati precedenti, è la leggera differenza, di qualche pixel, nelle coordinate del punto in alto e sinistra della boundng box. Si possono notare delle differenze di circa 2 pixel per le coordinate in X e Y, resta comunque una differenza meno marcata rispetto al caso precedente. Anche qui, abbiamo il sistema di tiling con punti più in alto e a sinistra rispetto all'edge system.

Risultati con EasyOCR (tiling system)			
N.	Testo	Top Left X [pixel]	Top Left Y [pixel]
1	"DOT"	10712	234
2	"DOT"	1623	234
3	"DOT"	9247	244
4	"DOT"	9904	244
5	"DOT"	5785	227
6	"DOT"	6536	245
7	"DOT"	10268	245
8	"DOT"	3210	229
9	"DOT"	8704	230
10	"DOT"	1546	228
11	"DOT"	12046	259
12	"DOT"	14501	255
13	"DOT"	1170	255
14	"DOT"	13672	262
15	"DOT"	4986	261
16	"DOT"	3194	233
17	"DOT"	12439	234
18	"DOT"	8052	241
19	"DOT"	12681	241
20	"DOT"	4438	258

Risultati con EasyOCR (edge system)			
N.	Testo	Top Left X [pixel]	Top Left Y [pixel]
1	"DOT"	10712	232
2	"DOT"	1625	232
3	"DOT"	9249	242
4	"DOT"	9905	242
5	"DOT"	5785	224
6	"DOT"	6537	243
7	"DOT"	10269	243
8	"DOT"	3211	226
9	"DOT"	8705	227
10	"DOT"	1548	228
11	"DOT"	12047	259
12	"DOT"	14502	253
13	"DOT"	1169	253
14	"DOT"	13671	255
15	"DOT"	4986	258
16	"DOT"	3198	232
17	"DOT"	12431	232
18	"DOT"	8053	239
19	"DOT"	12683	240
20	"DOT"	4439	256

Guardando i risultati raccolti si può affermare che i sistemi di tiling e l'edge system sono equivalenti, entrambi ci restituiscono risultati con elevata accuratezza e precisione. Per quanto riguarda i pixel di differenza, non riscontrano un vero impedimento. Su delle immagini con delle dimensioni pari alle nostre scansioni, quei pochi pixel di indecisione non sono assolutamente un ostacolo e non compromettono la precisione del risultato.

Una considerazione può essere fatta sui due modelli OCR, Tesseract e EasyOCR. Entrambi hanno lavorato benissimo su immagini binarie, decifrando alla perfezione DOT, ma il primo riporta delle bounding box leggermente più larghe rispetto alle bounding box del secondo. Anche qui parliamo di pochi pixel (in media 4 pixel) e si può affermare che la differenza non è per nulla penalizzante.

Un ultima discriminante è il tempo di esecuzione. I tempi di esecuzione sono stati misurati considerando la sola esecuzione della porzione di codice legata al OCR. Quindi nelle tempistiche non sono considerate le operazione di lettura acquisizione e rendering dell'immagine binaria. Inoltre, i tempi sono stati misurati campionando l'esecuzione degli script più volte sulle 20 acquisizioni, per estraniare quanto più possibile qualsiasi errore dovuto alle condizioni momentanee del PC usato.

- **Tesseract con tiling** ha avuto un tempo medio 11,650 secondi, un minimo di 1,605 secondi e massimo di 21,714 secondi
- **EasyOCR con tiling** ha avuto un tempo medio 30,770 secondi, un minimo di 6,002 secondi e massimo di 48,339 secondi
- **Tesseract con edge system** ha avuto un tempo medio 1,495 secondi, un minimo di 0.990 secondi e massimo di 3,293 secondi
- **EasyOCR con edge system** ha avuto un tempo medio 1,994 secondi, un minimo di 1,394 secondi e massimo di 5,087 secondi

Immediatamente si può notare come il sistema di tiling ha tempistiche di molto superiori al edge system, sia con Tesseract che con EasyOCR. Questo è una differenza valida che pone i due sistemi su livelli distinti. Nonostante entrambi ottengono pressoché gli stessi risultati, hanno un marcata differenza in fatto di tempo di esecuzione, il che ci spinge a preferire il sistema basato sull'individuazione di bordi.

In più, nel tiling system, si nota un grande divario tra il tempo minimo e massimo. Il sistema nel caso migliore (con DOT ad inizio immagine) conserva un tempo competitivo ma diventa pressoché inutilizzabile nel caso peggiore (con DOT a fine immagine) con un tempo fuori portata.

Il vantaggio temporale riscontrato con l'edge system è facilmente spiegabile con il fatto che il modello di OCR non deve intervenire su tutte le regioni trovate, ma solo su quelle che rispettano dei criteri imposti (criteri legati alle dimensioni di DOT), così da semplificare l'esecuzione e ridurre di molto il tempo d'esecuzione.

In conclusione, la scelta sul modello OCR non crea grandi problemi, entrambi riescono benissimo ad individuare DOT e bounding box. Discorso diverso per i sistemi di supporto al OCR, qui c'è una netta differenza in fatto di tempo di esecuzione. Il sistema basato sulla ricerca di bordi impiega mediamente molti meno secondi per restituire un risultato, rispetto il sistema di tiling. Quindi, per velocizzare l'elaborazione si consiglia l'utilizzo del edge system.

Capitolo 4

Implementazioni future

Con questo progetto si è riusciti a determinare con precisione la posizione della scritta DOT all'interno di un pneumatico, partendo da una acquisizione ad alta risoluzione della propria spalla laterale. Riuscire a determinare le corrette coordinate era il primo passo, per la futura implementazione di un sistema di riferimento con il quale sarebbe possibile determinare la posizione corretta di qualsiasi altra scritta.

4.0.1 Applicazioni future

Un volta determinate le coordinate delle scritte, verrà facile applicare un OCR. Basterà, partendo dal punto origine, specificare le regioni che racchiudono specifiche sigle o scritte. Una volta ottenuta la zona d'interessa, sarà possibile passarla in input all'OCR per decifrare il proprio testo.

Quindi, determinare con precisione il punto di coordinate zero (posizione DOT), diventa un problema chiave per una corretta mappatura dell'intera spalla e per un OCR accurato.

Il sistema di mappature dovrà tener conto della circolarità della scansione. Non si possono determinare coordinate senza considerare che la ruota scansionata è circolare e che la posizione del DOT non sarà sempre la stessa. Per l'implementazione di un sistema di riferimento si consiglia di lavorare su settori circolari. Ovvero, le coordinate di origine corrisponderebbero a grado 0° e l'intero pneumatico sarebbe coperto in 360°, detto ciò si potrebbero riportare le posizioni delle singole scritte come settori angolari distanti dallo 0°.

Elenco delle figure

Bibliografia

[1] *Aditya mahajan,* ***EasyOCR: A Comprehensive Guide****,* *https://medium.com/@adityamahajan.work/easyocr-a-comprehensive-guide-5ff1cb850168*.

[2] *Eletra sensing tecnology,* ***encoder incrementali*** *https://www.eltra.it/assets/Uploads/EncoderPedia-immagini-e-file/PDF/ENCODER-INCREMENTALI-PRINCIPIO-FUNZIONAMENTO.pdf*.

[3] *Jaided, ai* ***EasyOCR documentation****, https://www.jaided.ai/easyocr/documentation/*.

[4] *Jaided, github* ***EasyOCR****, https://github.com/JaidedAI/EasyOCR*.

[5] *Kubler group,* ***kubler encoders****,* *https://www.kuebler.com/it/prodotti/misurazione/encoders/product-finder/dettagli-del-prodotto/5825*.

[6] *Mech mind,* ***Mech-Eye 3D Laser Profile User Manual*** *https://docs.mech-mind.net/en/eye-3d-profiler/latest/getting-started/getting-started.html*.

[7] *Numpy,* ***Numpy API references****, https://numpy.org/doc/stable/reference/*.

[8] *Open source computer vision* ***OpenCV: OpenCV Modules****, https://docs.opencv.org/4.x/index.html*.

[9] *Pytesseract, github* ***Python Tesseract****, https://github.com/h/pytesseract*.

[10] *Scikit image* ***scikit image's documentation****, https://scikit-image.org/docs/stable/*.

[11] *Scikit-learn,* ***scikit learn imputate****, https://scikit-learn.org/stable/modules/impute.html*.

[12] *Scikit-learn* ***scikit learn****, https://scikit-learn.org/stable/api/index.html*.

[13] *Tesseract, github* ***Tesseract OCR****, https://github.com/tesseract-ocr/tesseract*.

[14] ***Come leggere le dimensioni di uno pneumatico*** *https://cerchigomme.it/leggere-un-pneumatico*.

[15] ***Encoder a quadratura*** *https://www.riccardomonti.eu/Arduino/esercizi/ENCODER/encoder.html*.

[16] ***How to read DOT number*** *https://www.chapelhilltire.com/how-to-read-your-tires-dot-number/*.

[17] E. Bemporad,
Tecniche dianalisi strumentali e statistiche
https://www.unirc.it/documentazione/formazione/materiale/Lezione_1.pdf.

[18] Bruce, Batchelor, and Whelan,
Intelligent Vision Systems for indusrty
isbn 3540199691, springer; 1997th edition (april 30, 1997).

[19] Giesko, Zbrowski, and Czajka,
Laser profilometer for surface inspection and profile measurement
https://bibliotekanauki.pl/articles/257675.pdf, 2007.

[20] Kuric, Klarák, Sága, Císar, Hajdučík, and Wiecek,
Analysis of the Possibilities of Tire-Defect Inspection Based on Unsupervised Learning and Deep Learning
https://www.mdpi.com/1424-8220/21/21/7073, 2021.

[21] W. McIlhagga,
The Canny edge detector revisited
doi https://doi.org/10.1007/s11263-010-0392-0, 2024.

[22] Mithe, Indalkar, and Divekar, ***Optical Character Recognition***, *issn 2277-3878, 2013.*

[23] D. Pelliccia,
Two methods for baseline correction of spectral data
https://nirpyresearch.com/two-methods-baseline-correction-spectral-data/, 2024.

[24] D. Pelliccia,
Wavelet denoising of spectra
https://nirpyresearch.com/wavelet-denoising-spectra/, 2023.

[25] J. Peng, S. Peng, A. Jiang, J. Wei, C. Li, and J. Tan,
Asymmetric least squares for multiple spectra baseline correction *doi https://doi.org/10.1016/j.aca.2010.08.033, 2010.*

[26] R. Smith, *Google inc.*
An Overview of the Tesseract OCR Engine
doi https://doi.org/10.1109/ICDAR.2007.4376991, 2007.

[27] A. Sorgonà, *Bio.m.a.a., università mediterranea di reggio calabria,*
Introduzione alla analisi delle immagini
http://limeacademy.uniroma3.it/wp-content/uploads/2019/04/12-Analisi-di-immagine.pdf.

[28] S. B. Verma,
Edge detection and line detection in Image processing, *M.Tech (Computer Science) 2nd Semester, MTCS-202 (C).*