

UNIVERSITÀ DI SIENA

1240

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE E
SCIENZE MATEMATICHE

Corso di Laurea in
Ingegneria informatica e dell'informazione

La difficoltà di estrarre informazioni
da immagini digitali: un caso di studio

Relatore:
Prof.ssa Brunetti Sara

Candidato:
Criscuolo Dario

Anno Accademico 2021-2022

Indice

1 Introduzione	2
1.1 Perché utilizzare delle immagini?	3
2 Nozioni preliminari	5
2.1 Rappresentazione di un'immagine digitale	5
2.2 Cosa è la segmentazione	6
2.3 OpenCv	7
2.3.1 Basic thresholding operation	8
2.3.2 Basic thresholding operation applicato a un'immagine	10
3 Possibili soluzioni	13
3.1 Simple Background Remover	13
3.1.1 Introduzione	13
3.1.2 Quando va usato	14
3.1.3 Calcolo a mano	16
3.1.4 Soluzione attraverso algoritmo	18
3.1.5 Confronto tra calcolo a mano e algoritmo	21
3.2 Background Remover with Noises	22
3.2.1 Introduzione	22
3.2.2 Quando va usato	22
3.2.3 Calcolo a mano	24
3.2.4 Soluzione attraverso algoritmo	26
3.2.5 Confronto tra calcolo a mano e algoritmo	29
3.3 Basic Thresholding Operations	30
3.3.1 Introduzione	30
3.3.2 Quando va usato	30
3.3.3 Calcolo a mano	31
3.3.4 Soluzione attraverso algoritmo	33
3.3.5 Confronto tra calcolo a mano e algoritmo	36
4 Limiti nell'uso degli algoritmi	37
4.1 Il problema della luminosità	37
4.1.1 Di cosa si tratta	37
4.1.2 Possibili soluzioni	39
4.2 Il problema della qualità delle immagini	41
4.2.1 Di cosa si tratta	41
4.2.2 Possibili soluzioni	42

4.3	Il problema delle immagini con oggetti in parte coperti	43
4.3.1	Di cosa si tratta	43
4.3.2	Possibili soluzioni	44
5	Conclusioni e possibilità future	45
6	Appendice	47
6.1	Calcoli dello pseudocodice degli algoritmi di misurazione	47
6.1.1	Calcoli dello pseudocodice dell'Algoritmo 2	47
6.1.2	Calcoli dello pseudocodice dell'Algoritmo 4	48
6.1.3	Calcoli dello pseudocodice dell'Algoritmo 6	49
	bibliografia	50

Capitolo 1

Introduzione

Computer e algoritmi sono diventati utili in tutti gli aspetti della nostra vita quotidiana, e questi utilizzano spesso le immagini. Molti settori infatti impiegano la fotografia, strumento immediato e alla portata di tutti, per effettuare computazioni e misurazioni. Nella falegnameria ad esempio, data una fotografia o un'immagine, è necessario segmentarla ottenendo solo la porzione necessaria dell'immagine (in una fotografia di alcuni tronchi nella foresta l'obiettivo è estrarre solo la porzione di tronchi o legname, togliendo così lo sfondo) su cui poi verrà effettuata la misura (ad esempio il calcolo del volume di tale legname). Purtroppo uno dei problemi principali è la qualità di tali immagini, che nella maggior parte dei casi, non è abbastanza elevata da permettere una perfetta distinzione degli oggetti. Ne consegue quindi, che estrarre dati e misure dalle immagini, è un task difficile, ma d'interesse rilevante nell'area della *Image Analysis* e in generale della *Computer Science*.

L'obiettivo di questa tesi di laurea è quello di fornire un metodo per l'estrazione di una parte specifica di un'immagine, anche di bassa qualità, ricavando successivamente la grandezza reale dell'oggetto che è scopo di analisi. Nel concreto prenderò delle immagini dove sono presenti tronchi visti da davanti all'interno di uno sfondo generico, per poi estrarre con la segmentazione l'area dei tronchi e ottenere così il volume totale dei tronchi all'interno della fotografia.

1.1 Perché utilizzare delle immagini?

Il lettore si potrebbe domandare perché ho scelto di trattare le immagini come tema fondamentale nel mio elaborato. La facilità con cui posso fare una fotografia a una grande quantità di oggetti e la vastità dei dispositivi che effettuano fotografie ha portato a un'esplosione nel volume dei dati sotto forma d'immagini. Ad esempio, quante app sul nostro smartphone utilizzano la fotocamera? Basti pensare ai social e abbiamo già un esempio rilevante. Date delle fotografie, per un ingegnere informatico nasce l'interesse di analizzare tali immagini e, conseguentemente, estrarre dati da esse per comprendere informazioni e successivamente prendere decisioni.

Un esempio pratico di estrazione dati da immagini è quello di *Google Lens*[5]. Uno dei compiti che può svolgere questo strumento è: data una fotografia di un fiore nella natura o di un animale nel suo habitat naturale, l'algoritmo di *Google Lens* è in grado d'identificare la specie della pianta o dell'animale in questione, aiutando l'identificazione dell'oggetto.

Un altro esempio di estrazione dati da immagini viene dalla società *Apple*. La multinazionale ha sviluppato un'applicazione [1] capace di misurare le dimensioni di un oggetto semplicemente ponendolo davanti alla fotocamera.

Tali tecnologie devono anche fronteggiare il problema della qualità delle immagini, che nella maggior parte dei casi, non è abbastanza elevata da permettere una perfetta distinzione degli oggetti. Nasce quindi la necessità di trovare un metodo in grado di fronteggiare il problema della risoluzione delle immagini. A tal proposito si candida in maniera preponderante la tecnica della segmentazione: tale tecnica rappresenta la costruzione di multiple immagini, a partire da quella iniziale, le quali evidenziano parti diverse, il suo foreground e il suo background.

Le implementazioni della tecnica di segmentazione vengono prese dalla libreria *Open-Cv* [7], la quale ha già implementato una propria soluzione sull'estrazione di una determinata figura da un'immagine. Tale libreria possiede molti metodi e implementazioni, i quali spesso fin troppo complessi, perciò nella mia ricerca ho evidenziato e semplificato alcuni algoritmi, così da rendere più fruibile la risoluzione del mio problema. In conclusione, nella maggior parte dei casi gli algoritmi modificati risolvono il problema dell'identificazione degli oggetti ed eseguono una misurazione precisa degli stessi, anche se in altri casi hanno evidenziato alcune limitazioni.

Qui segue una breve descrizione di ogni capitolo dopo l'introduzione.

- Capitolo 3: **Nozioni preliminari.** In questa sezione viene spiegato quali strumenti ho utilizzato per approcciarmi al problema: la segmentazione e la misurazione di un oggetto in un'immagine.
- Capitolo 4: **Possibili soluzioni.** In questa sezione vengono spiegati i vari metodi con cui possiamo affrontare il problema (sia la segmentazione che la misurazione) e i loro pro e contro. Inoltre segue un confronto tra la misurazione ottenuta con calcoli cosiddetti "a mano" e la misurazione ottenuta tramite l'algoritmo.
- Capitolo 5: **Limiti degli algoritmi.** In questa sezione vengono descritti i vari limiti che ogni algoritmo possiede a seconda dei vari casi pratici.
- Capitolo 6: **Conclusioni e possibilità future.** In questa sezione traggo le conclusioni del mio lavoro e suggerisco possibili tecnologie che nel futuro migliorerebbero l'utilizzo di questi algoritmi.

Capitolo 2

Nozioni preliminari

2.1 Rappresentazione di un'immagine digitale

Inizio definendo cosa sia e come viene rappresentata un'immagine digitale: un'immagine digitale è rappresentata da una matrice di punti, detti anche pixel, ciascuno con quantità finite e discrete di rappresentazione numerica: la sua intensità o livello di grigio, e le sue coordinate spaziali denotate con x , y . Queste informazioni sono necessarie per permettere agli algoritmi di segmentazione e di misurazione di eseguire correttamente i loro processi. Altre informazioni come il formato dell'immagine non sono necessarie: oggi vengono principalmente utilizzati *.jpg* e *.png* ed entrambi possono essere utilizzati negli algoritmi sotto riportati senza alcun problema (dato che le immagini che vengono utilizzate sono viste come matrici di pixel, tale forma non è influenzata dal formato). I parametri essenziali di cui abbiamo bisogno per la nostra ricerca sono:

- Le sue dimensioni, cioè il numero di pixel per lunghezza e altezza;
- La tipologia di sfondo, cioè se lo sfondo è chiaro o scuro;
- Il parametro di traduzione di grandezza da immagine a realtà, che permette di mappare la dimensione reale di un oggetto all'interno dell'immagine.

2.2 Cosa è la segmentazione

La segmentazione dell’immagine è uno dei passaggi più importanti nell’estrazione delle informazioni da un’immagine. Il suo scopo è dividere un’immagine in più parti: gli oggetti al suo interno e l’area esterna che li circonda. Esistono principalmente due tipologie di segmentazione: segmentazione completa e segmentazione semplice. La prima prende come ulteriore input le possibili figure geometriche degli oggetti che possono apparire nell’immagine per poi dividere l’immagine in foreground (gli oggetti rilevati) e background (l’area esterna). Per ottenere una segmentazione completa la cooperazione con livelli di elaborazione più elevati che utilizzano una conoscenza specifica del dominio del problema è necessaria. La segmentazione semplice, invece, non necessita di ulteriori input, l’immagine è divisa in regioni separate che sono omogenee rispetto a una proprietà scelta dall’utente, come luminosità, colore, riflettività, trama, ecc. Tale elaborazione è indipendente dal contesto; non viene utilizzato alcun modello correlato agli oggetti e non necessita nessuna conoscenza di quanto rappresentato nell’immagine.[9] Nel mio caso, dato che le immagini sono comunemente composte da oggetti di colore contrastante, situati su uno sfondo più o meno uniforme può bastare una segmentazione semplice.

2.3 OpenCv

Come anticipato nell'Introduzione, nella mia ricerca è stato fondamentale l'utilizzo di OpenCV.

OpenCV [7] (acronimo in lingua inglese di Open Source Computer Vision Library) è una libreria software libera originariamente sviluppata dalla società *Intel*, nel centro di ricerca in Russia di Nižnij Novgorod (il creatore della libreria). Successivamente fu poi mantenuto da Willow Garage, fallita nel 2014 e ora è gestita da Itseez - quest'ultima una divisione della società Intel.

Il linguaggio di programmazione utilizzato per lo sviluppo della libreria è principalmente il C++ (un linguaggio di programmazione general purpose), ma è possibile interfacciarsi anche attraverso il C, Python e Java. Io utilizzerò sia algoritmi in linguaggio C++ che in linguaggio Python.

OpenCV è open-source e può essere utilizzato su macchine con vari sistemi operativi come Windows, Linux e MacOS X. Le funzionalità principali che OpenCV offre sono:

- Algoritmi di elaborazione dati di video;
- Algoritmi di elaborazione dati d'immagini;
- Algoritmi di strutturazione gerarchica di dati.

Tra le varie funzionalità di cui dispone OpenCV io ho deciso di utilizzare *Basic thresholding operation*[4] per la gestione d'immagini, trattandole come matrici di pixel. Nella sezione successiva spiegherò nel dettaglio tale funzionalità.



Figura 2.1: Logo OpenCV

2.3.1 Basic thresholding operation

Basic thresholding operation [4] è un algoritmo sviluppato da OpenCV ed è considerato il metodo più semplice per segmentare un'immagine. L'algoritmo prende la gradazione di luminosità dei pixel dell'immagine $f(i,j)$ (dove i e j rappresentano la posizione del singolo pixel sugli assi x e y) e il Threshold deciso T . Poi compara la gradazione di luminosità di ogni pixel con il Threshold, se $f(i,j) \geq T$ allora il pixel verrà assegnato al foreground, altrimenti verrà assegnato al background. Nella Figura 2.2 presento un esempio di tale funzione.



Figura 2.2: Esempio di applicazione di segmentazione

La segmentazione ha la funzione di differenziare a quale dei due gruppi assegnare i pixel all'interno di un'immagine: se al foreground o background. Questo si effettua attraverso una comparazione tra l'intensità di ogni pixel e attraverso una soglia decisa dall'utente. Una volta stabilito a quale gruppo assegnare ogni singolo pixel, si assegna alla posizione dei pixel appartenenti al gruppo in foreground il colore bianco, mentre al background il colore nero.

Per esempio se applicassi la segmentazione alla Figura 2.3 (tutta di colore grigio scuro) essa essendo di un colore più vicino al nero diventerebbe tutta background, mentre se prendessi la Figura 2.4, essendo di colore più chiaro verrebbe interamente categorizzata come foreground. Quindi questo l'algoritmo sfrutta la luminosità dei vari pixel per distinguere tra foreground e background.



Figura 2.3: Grigio scuro

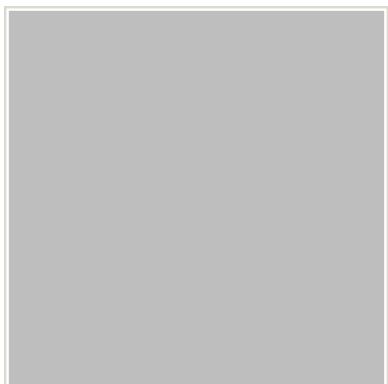


Figura 2.4: Grigio chiaro

2.3.2 Basic thresholding operation applicato a un'immagine

In generale, l'algoritmo di basic thresholding operation si applica a tutte le tipologie d'immagini, ma è interessante appronfondire come esso opera nel concreto quando viene data in input un'immagine.

Prendiamo come esempio quello presente nella Risorsa [4] dove ci sono due immagini: la Figura 2.5 e la Figura 2.6. La prima presenta una piramide vista dall'alto e la seconda lateralmente. Nella prima immagine la punta della piramide, essendo più in alto è più luminosa e quindi verrà assegnata al foreground, mentre la parte scura ed esterna andrà nel background.

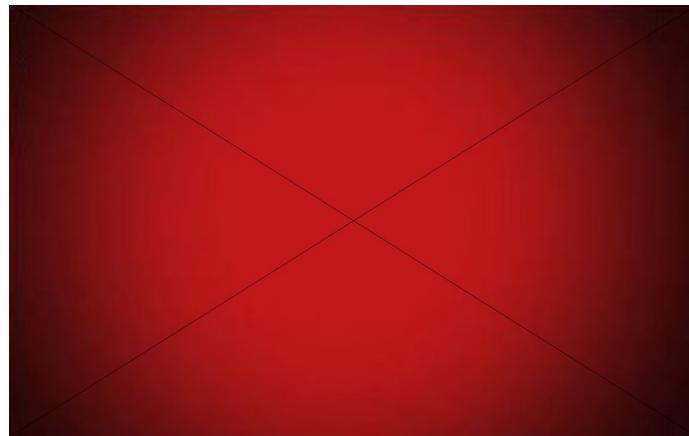


Figura 2.5: Piramide vista dall'alto

Se potessi vedere lateralmente l'immagine avrei la Figura 2.6. In questa immagine notiamo in rosso l'oggetto (la piramide) visto lateralmente, e una linea blu, la quale divide l'oggetto rispetto alla soglia scelta dall'utente. Data la soglia, sopra di essa avrò il foreground che l'utente desidera selezionare, mentre sotto alla soglia dell'oggetto avrò il background.

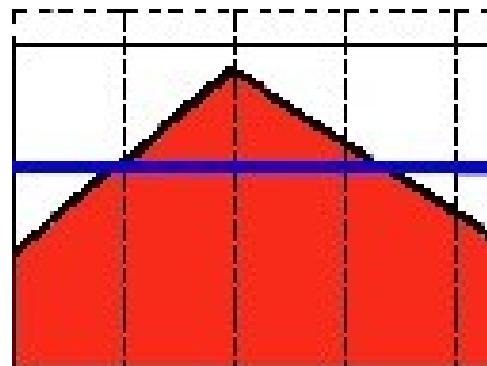


Figura 2.6: Piramide vista lateralmente

L'utente applicando l'algoritmo otterrebbe come output in foreground la Figura 2.7 se vista dall'alto e la Figura 2.8 vista lateralmente.

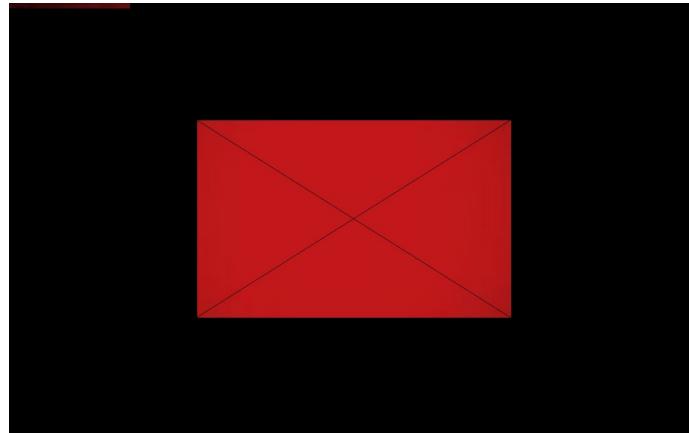


Figura 2.7: Piramide vista dall'alto edove c'è solo il foreground

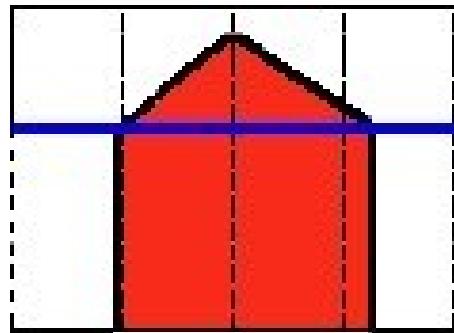


Figura 2.8: Piramide vista dal lato dove c'è solo il Foreground

Viceversa succede se l'attenzione dell'utente è sul background: otterrei la Figura 2.9 dall'alto e la Figura 2.10 lateralmente.

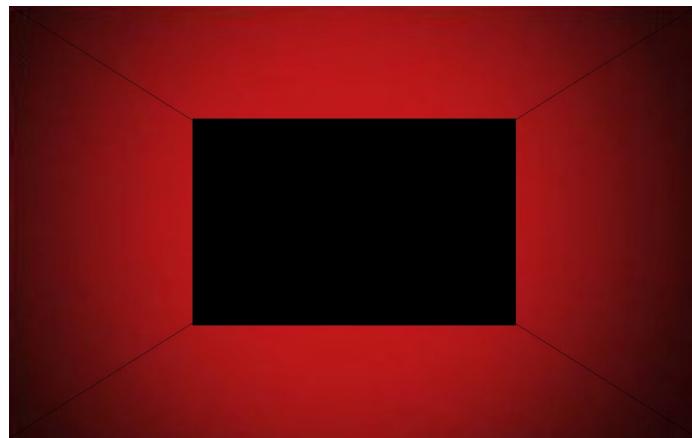


Figura 2.9: Piramide vista dall'alto dove c'è solo il Background

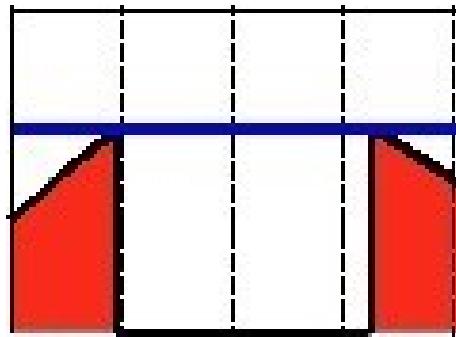


Figura 2.10: Piramide vista dal lato dove c'è solo il Background

Anche se l'esempio è semplice, da tale immagine si può comprendere come funziona questa libreria di OpenCV. Nelle prossime sezioni mi concentrerò sugli algoritmi di segmentazione presenti su OpenCV.

Capitolo 3

Possibili soluzioni

In questa sezione mi concentrerò sulle soluzioni presenti su OpenCV per la segmentazione. Tra tali soluzioni presenterò tre algoritmi [3, 6, 4]:

- Simple Background Remover;
- Background Remover with Noises;
- Basic Thresholding Operations.

L'algoritmo Simple Background Remover è il più semplice, infatti funziona solo in alcuni casi specifici. L'algoritmo Background Remover with Noises è una versione migliorata e generalizzata dell'algoritmo Simple background remover. Infine l'algoritmo Basic Thresholding Operations funziona pressoché sempre e permette di personalizzare e parametrizzare la segmentazione manualmente.

Dopo aver applicato uno tra i seguenti algoritmi di segmentazione inserirò i dati ottenuti in un algoritmo di calcolo del volume totale. Questo prima assegna a ogni pixel nell'immagine un valore in millimetri nella realtà e poi calcola il volume totale dei tronchi come somma del totale di pixel.

Nella prossima sezione userò come esempi delle foto fatte da me o prese dal web.

3.1 Simple Background Remover

3.1.1 Introduzione

Il primo algoritmo presentato è basato sulla Risorsa [3]. Tale algoritmo è diviso in tre fasi principali:

1. Nel primo passo una funzione trasforma i colori dell'immagine in sfumature di grigio.

2. Successivamente una funzione trasforma tutti i pixel sopra alla soglia scelta in bianco (il foreground), e sotto la soglia in nero (il background), utilizzando le sfumature di grigio per semplificare il processo.
3. Infine un'altra funzione inverte i colori dell'immagine così da ottenere il foreground in bianco e il background in nero.

3.1.2 Quando va usato

Questo primo algoritmo va usato su immagini dove sono presenti oggetti (nel mio caso tronchi) di piccole dimensioni e di colore uniforme, ciò è necessario perché altrimenti l'algoritmo non riesce a identificare correttamente i bordi dei vari tronchi. Se per esempio lo applicassi alla Figura 3.12 otterrei un'immagine confusa (Figura 3.1) dove parti del tronco vengono confuse come background e quindi colorate di nero.



Figura 3.1: Risultato se applico alla Figura 3.12 l'Algoritmo 1

Un'altra limitazione di tale algoritmo è lo sfondo dei tronchi il quale deve essere nero per il corretto funzionamento dell'algoritmo. Se lo sfondo fosse bianco si confonderebbe con il foreground, rendendo scorretto il calcolo dell'area interna, dato che l'algoritmo utilizza il numero totale di pixel bianchi presenti nell'immagine per calcolare l'area. Per esempio se applicassi l'algoritmo alla Figura 3.19 otterrei la Figura 3.3, la quale presenta sia l'interno dei tronchi, che la parte esterna dell'immagine di colore bianco. Ciò sarebbe un problema dato che l'algoritmo utilizza il numero totale di pixel bianchi presenti nell'immagine per calcolare l'area ottenendo quindi un risultato sbagliato.



Figura 3.2: Esempio d'immagine dove l'Algoritmo 1 non funziona



Figura 3.3: Risultato se applico alla Figura 3.2 l'Algoritmo 1

3.1.3 Calcolo a mano

Per dimostrare l'utilità di questo algoritmo, applico alla Figura 3.4¹ questo metodo. Prima di passare all'applicazione dell'algoritmo, calcolo manualmente il volume dei tronchi all'interno dell'immagine, per poi confrontare i risultati ottenuti dall'algoritmo con quelli ottenuti manualmente.



Figura 3.4: Esempio 1 d'immagine con tronchi

I dati che ho a disposizione sono il diametro del tronco evidenziato (30 cm) e la lunghezza media dei tronchi (2m).



Figura 3.5: Esempio con evidenziato il tronco di cui la grandezza reale è conosciuta

¹<https://www.google.com/search?q=tronchi+images>

L'area approssimata del tronco cerchiato sarà:

$$A = \pi * r^2$$

e il volume, dato l'area dei tronchi e la lunghezza di essi, sarà:

$$V = A * l$$

Partiamo dal calcolo dell'area del tronco evidenziato in rosso. In questo caso ottengo un'area pari a $0.07m^2$:

$$A = 3.1416 * 0.15^2 = 0.07m^2.$$

Ora calcolo l'area di tutti i tronchi.

All'interno dell'immagine sono presenti 40 tronchi, 18 dei quali non appaiono nella loro interezza nella foto. Dei restanti 22 tronchi approssimo la loro area come del tronco evidenziato in precedenza, poiché i tronchi hanno diametri simili, ottenendo:

$$A = 22 * 0.07 = 1.54m^2.$$

Degli altri 18 tronchi prendo il 50% dell' area perché risultano tagliati nella foto:

$$A = 18 * 0.07 * 0.5 = 0.63m^2.$$

Quindi l'area totale si può approssimare con:

$$A = 0.63 + 1.54 = 2.17m^2.$$

Infine moltiplico l'area totale per la lunghezza media dei tronchi ottenendo il volume totale:

$$V = 2.17 * 2 = 4.34m^3$$

3.1.4 Soluzione attraverso algoritmo

In questa sottosezione applico il primo metodo per il calcolo del volume dei tronchi della Figura 3.4. Per comprendere le fasi dell'algoritmo presento lo pseudocodice.

Algorithm 1 Algoritmo 1 - Segmentazione

```
1: immagine  $\leftarrow$  filepath(posizione_immagine)
2: immagine_grigio  $\leftarrow$  CvtColor(immagine)
3: immagine_finale_invertita  $\leftarrow$  Applythreshold(immagine_grigio)
4: immagine_finale  $\leftarrow$  Invert(immagine_finale_invertita)
```

Presento una spiegazione delle funzioni utilizzate dall'algoritmo:

- **CvtColor**: la funzione trasforma i colori dell'immagine in sfumature di grigio ottenendo la Figura 3.6.
- **Applythreshold**: la funzione fa diventare bianchi tutti i pixel sopra alla soglia scelta (il foreground), e sotto la soglia (il background) in nero, utilizziamo le sfumature di grigio per semplificare il processo.
- **Invert**: la funzione sottrae da 255 dall'immagine così invertendone i colori ottenendo la Figura 3.7



Figura 3.6: Esempio trasformato in sfumature di grigio applicando la funzione CvtColor



Figura 3.7: Esempio con applicata segmentazione

Adesso dato che la segmentazione è conclusa procedo con la misurazione che consiste nel calcolo del volume. Data la Figura 3.7, parto dal calcolo dell'area del tronco cerchiato in rosso in 3.9.

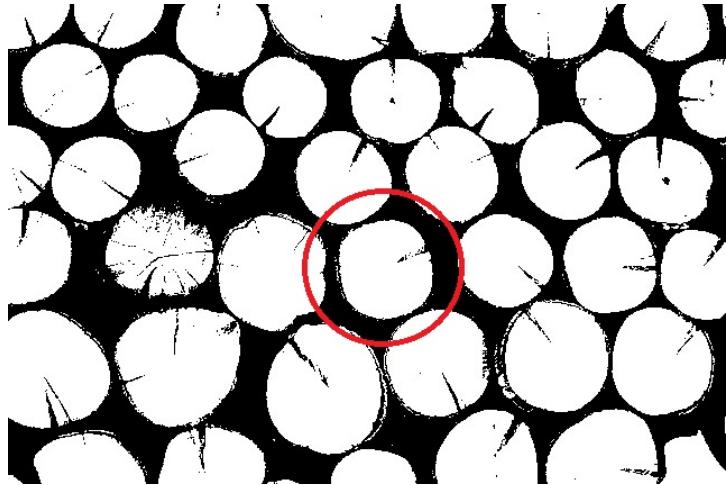


Figura 3.8: Esempio segmentato con evidenziato il tronco di cui la grandezza reale è conosciuta.



Figura 3.9: Esempio tagliato per avere solo parte di cui conosco la grandezza reale

Inserisco la Figura 3.7 e la Figura 3.9 nell'algoritmo 2 che procederà alla misurazione.

Algorithm 2 Algoritmo 1 - Misurazione

```
1: immagine_segmentata  $\leftarrow$  filepath(posizione_imagine_finale)
2: immagine_segmentata_cerchiorosso  $\leftarrow$  filepath(posizione_imagine_cerchiorosso)
3: hh, ww  $\leftarrow$  dimensioni(immagine_segmentata_cerchiorosso)
4: hh1, ww1  $\leftarrow$  dimensioni(immagine_segmentata)
5: pixel_per_metro  $\leftarrow$  funzione1(immagine_segmentata_cerchiorosso, area_data)
6: area_finale  $\leftarrow$  funzione2(pixel_per_metro, area_data, immagine_segmentata)
7: volume_finale  $\leftarrow$  area_finale * lunghezza_data
```

Presento una spiegazione delle funzioni utilizzate dall'algoritmo:

- **funzione1**: la funzione calcola il numero di pixel per metro come rapporto tra l'area data e il numero totale di pixel del foreground.
- **funzione2**: la funzione calcola l'area finale come prodotto tra il totale dei pixel e i pixel per metro.

Nell'appendice riporto i calcoli con i relativi risultati, ottengo come area totale dei tronchi $2.23m^2$ e come volume $4.46m^3$.

3.1.5 Confronto tra calcolo a mano e algoritmo

Confronto i risultati ottenuti utilizzando il calcolo a mano con quelli ottenuti attraverso algoritmo. Dal calcolo a mano ho ottenuto un volume pari a $4.34m^3$, mentre attraverso l'algoritmo il risultato è pari a $4.46m^3$. I due risultati si discostano tra di loro per un fattore del 3%, questo può considerarsi un risultato apprezzabile in quanto i due valori sono vicini. Ai fini di una migliore comprensione preciso che durante il calcolo "manuale" ho semplificato la geometria dei tronchi, trattandoli come cilindri perfetti. Si può dedurre che probabilmente l'algoritmo, che tiene conto di ogni minima variazione nella geometria dei vari tronchi, sia più preciso rispetto al calcolo "manuale" fatto da me. Se considerassi infine il fattore tempo, il calcolo a mano mi ha impiegato 5 minuti, invece attraverso l'algoritmo sono stati necessari solo 10 secondi.

Ricordo che questo algoritmo ha le limitazioni presentate all'inizio della sezione, cioè la necessità di analizzare oggetti di piccole dimensioni e di colore uniforme, inoltre lo sfondo deve essere necessariamente nero.

3.2 Background Remover with Noises

3.2.1 Introduzione

Il secondo algoritmo presentato è basato sulla Risorsa [6]. Tale algoritmo è diviso in quattro fasi principali:

1. Inizio creando il *Threshold* data la soglia decisa dall'utente.
2. Nel secondo passo una funzione fa diventare tutti i pixel sopra alla soglia scelta (il foreground) in bianco, e sotto la soglia (il background) in nero.
3. Successivamente una funzione elimina tutti i pixel bianchi che sono circondati da pixel neri(elimina i pixel bianchi isolati).
4. Infine un'altra funzione inverte i colori dell'immagine così da ottenere in foreground in bianco e il background in nero.

Rispetto all'algoritmo precedente ha la funzione di eliminare le imperfezioni dei tronchi all'interno della fotografia.

3.2.2 Quando va usato

Questo secondo algoritmo va usato su immagini molto dettagliate, in particolare permette di lavorare con grandi variazioni di colore all'interno degli oggetti (nel mio caso tronchi). Se per esempio lo applicassi alla Figura 3.4 otterrei un'immagine confusa (Figura 3.10) dato che la funzione che rimuove le imperfezioni seleziona erroneamente parte dei bordi dei tronchi.

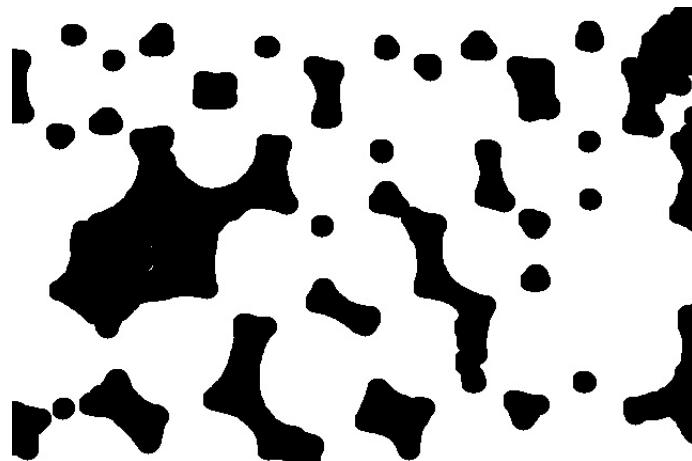


Figura 3.10: Risultato se applico alla Figura 3.4 l'Algoritmo 3

Un'altra limitazione di questo algoritmo è la quantità di tronchi, se per esempio lo applicassi alla Figura 3.19, la quale presenta tronchi troppo piccoli e molto vicini tra loro otterrei la Figura 3.11.



Figura 3.11: Risultato se applico alla Figura 3.19 l'Algoritmo 3

3.2.3 Calcolo a mano

Per dimostrare l'utilità di questo algoritmo, applico alla Figura 3.12² questo metodo. Prima di passare all'applicazione dell'algoritmo, calcolo manualmente il volume dei tronchi all'interno dell'immagine, per poi confrontare i risultati ottenuti dall'algoritmo con quelli ottenuti manualmente.



Figura 3.12: Esempio 2

I dati che ho a disposizione sono il diametro del tronco evidenziato (2 m) e la lunghezza media dei tronchi (15m).



Figura 3.13: Esempio con evidenziato il tronco di cui la grandezza reale è conosciuta

²<https://www.google.com/search?q=tronchi+images>

L'area approssimata del tronco cerchiato sarà:

$$A = \pi * r^2$$

e il volume, dato l'area dei tronchi e la lunghezza di essi, sarà:

$$V = A * l$$

Partiamo dal calcolo dell'area del tronco evidenziato in rosso. In questo caso ottengo un'area pari a $3.14m^2$

$$A = 3.14 * 1^2 = 3.14m^2.$$

Ora calcolo l'area di tutti i tronchi.

All'interno dell'immagine sono presenti solo due tronchi di area molto simile tra loro. L'area totale sarà l'area trovata in precedenza moltiplicata per due:

$$A = 2 * 3.14 = 6.28m^2$$

Infine moltiplico l'area totale per la lunghezza media dei tronchi ottenendo come volume totale:

$$V = 6.28 * 15 = 94.2m^3$$

3.2.4 Soluzione attraverso algoritmo

In questa sottosezione applico il secondo metodo per il calcolo del volume dei tronchi della Figura 3.12. Per comprendere le fasi dell'algoritmo presento lo pseudocodice.

Algorithm 3 Algoritmo 2 - Segmentazione

```
1: immagine  $\leftarrow$  filepath(immagine)
2: soglia  $\leftarrow$  Vector(soglia_decisa, soglia_decisa, soglia_decisa)
3: immagine_segmentata  $\leftarrow$  inRange(immagine, soglia)
4: immagine_pulita  $\leftarrow$  morphologyEx(immagine)
5: immagine_finale  $\leftarrow$  Invert(immagine_pulita)
```

Presento una spiegazione delle funzioni utilizzate dall'algoritmo:

- **Vector**: la funzione crea il *Threshold* data la soglia decisa dall'utente.
- **inRange**: la funzione fa diventare bianchi tutti i pixel sopra alla soglia scelta (il foreground), e sotto la soglia neri (il background).
- **morphologyEx**: la funzione elimina tutti i pixel bianchi che sono circondati da pixel neri.
- **Invert**: la funzione sottrae da 255 dall'immagine così invertendone i colori ottenendo la Figura 3.7



Figura 3.14: Esempio a cui ho applicato la segmentazione con la funzione *inRange*

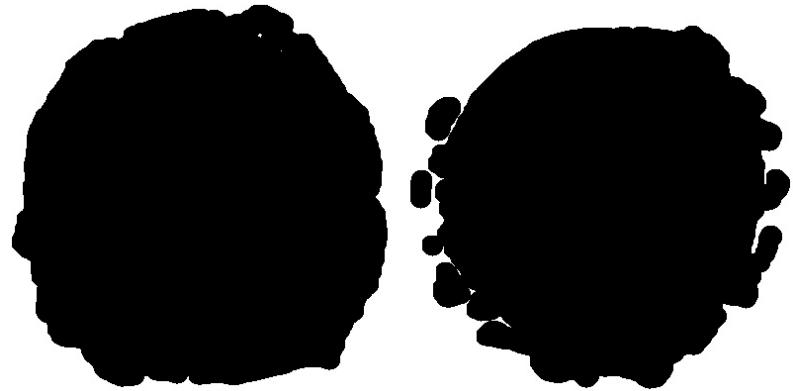


Figura 3.15: Esempio segmentato dove ho tolto gli spazi vuoti applicando morphologyEx

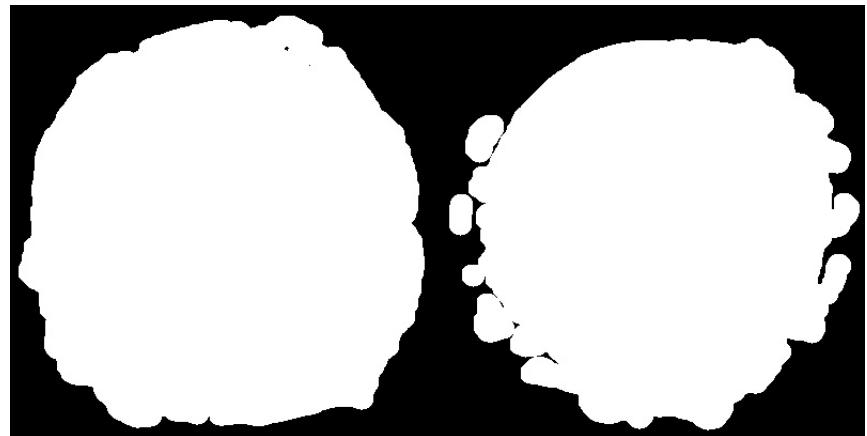


Figura 3.16: Esempio segmentato e con inversione dei colori applicando la funzione Invert

Adesso dato che la segmentazione è conclusa procedo con la misurazione che consiste nel calcolo del volume. Data la Figura 3.16, parto dal calcolo dell'area del tronco cerchiato in rosso in 3.17.

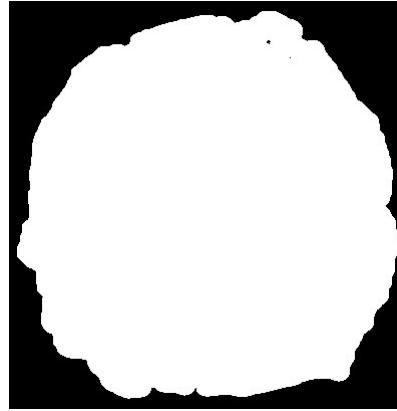


Figura 3.17: Esempio tagliato per avere solo parte di cui so la grandezza reale

Inserisco la Figura 3.16 e la Figura 3.17 nell'algoritmo 4 che procederà alla misurazione.

Algorithm 4 Algoritmo 2 - Misurazione

```

1: immagine_segmentata  $\leftarrow$  filepath(posizione_imagine_finale)
2: immagine_segmentata_cerchiorosso  $\leftarrow$  filepath(posizione_imagine_cerchiorosso)
3: hh, ww  $\leftarrow$  dimensioni(immagine_segmentata_cerchiorosso)
4: hh1, ww1  $\leftarrow$  dimensioni(immagine_segmentata)
5: pixel_per_metro  $\leftarrow$  funzione1(immagine_segmentata_cerchiorosso, area_data)
6: area_finale  $\leftarrow$  funzione2(pixel_per_metro, area_data, immagine_segmentata)
7: volume_finale  $\leftarrow$  area_finale * lunghezza_datta

```

Presento una spiegazione delle funzioni utilizzate dall'algoritmo:

- **funzione1:** la funzione calcola il numero di pixel per metro come rapporto tra l'area data e il numero totale di pixel del foreground.
- **funzione2:** la funzione calcola l'area finale come prodotto tra il totale dei pixel e i pixel per metro.

Nell'appendice riporto i calcoli con i relativi risultati, e ottengo come area totale dei tronchi $5.94m^2$ e come volume $89.06m^3$.

3.2.5 Confronto tra calcolo a mano e algoritmo

Confronto i risultati ottenuti utilizzando il calcolo "a mano" con quelli ottenuti attraverso algoritmo. Dal calcolo a mano ho ottenuto un volume pari a $94.2m^3$, mentre attraverso l'algoritmo il risultato è pari a $89.06m^3$.

Ai fini di una migliore comprensione preciso che durante il calcolo "manuale" ho semplificato la geometria dei tronchi, trattandoli come cilindri perfetti. Si può dedurre che probabilmente l'algoritmo, che tiene conto di ogni minima variazione nella geometria dei vari tronchi, sia più preciso rispetto al calcolo "manuale" fatto da me. I due risultati si discostano tra di loro per un fattore del 6%, questo può considerarsi un risultato apprezzabile in quanto i due valori sono vicini. Se considerassi infine il fattore tempo, il calcolo a mano mi ha impiegato 2 minuti, invece attraverso l'algoritmo sono stati necessari solo 10 secondi.

Ricordo che questo algoritmo ha le limitazioni presentate all'inizio della sezione, cioè funziona per immagini con oggetti di grandi dimensioni e distanziati tra loro.

3.3 Basic Thresholding Operations

3.3.1 Introduzione

Il terzo algoritmo presentato è basato sulla Risorsa [4]. Tale algoritmo è diviso in quattro fasi principali:

1. Nel primo passo una funzione crea una finestra con cui potrò decidere manualmente la soglia della segmentazione.
2. Successivamente una funzione trasforma i colori dell'immagine in sfumature di grigio.
3. Successivamente una funzione trasforma tutti i pixel sopra alla soglia scelta (il foreground) in bianco, e sotto la soglia (il background) in nero, utilizzando le sfumature di grigio per semplificare il processo.
4. Infine un'altra funzione applica le impostazioni scelte nella finestra tramite le barre di scorrimento ottenendo la Figura.

Questo algoritmo rispetto ai precedenti è più User Friendly, dato che permette all'utente di selezionare la soglia di segmentazione attraverso una barra di scorrimento. Inoltre attraverso una funzione di segmentazione più complessa permette una migliore segmentazione, la quale si adatta a qualsiasi dimensione dei tronchi all'interno della fotografia.

3.3.2 Quando va usato

Per utilizzare questo algoritmo basta che nell'immagine scelta il background non sia dello stesso colore del foreground. Quindi questo algoritmo è applicabile sia alla Figura 3.4, che alla Figura 3.12 e alla Figura 3.19. Inoltre questo algoritmo è applicabile sia a immagini con sfondo chiaro che con sfondo scuro.

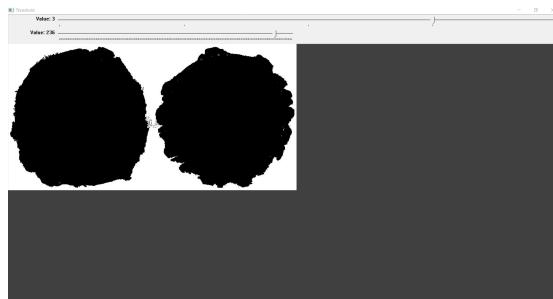


Figura 3.18: Finestra usata per segmentare le immagini con 5

3.3.3 Calcolo a mano

Per dimostrare l'utilità di questo algoritmo, applico alla Figura 3.19³ questo metodo. Prima di passare all'applicazione dell'algoritmo, calcolo manualmente il volume dei tronchi all'interno dell'immagine, per poi confrontare i risultati ottenuti dall'algoritmo con quelli ottenuti manualmente.



Figura 3.19: Esempio 3

I dati che abbiamo a disposizione sono il diametro del tronco evidenziato (40 cm) e la lunghezza media dei tronchi (90 cm).



Figura 3.20: Esempio con evidenziato il tronco di cui la grandezza reale è conosciuta

³Foto fatta da me in Trentino in data 8/25/22

L'area approssimata del tronco cerchiato sarà:

$$A = \pi * r^2$$

e il volume, data l'area dei tronchi e la lunghezza di essi, sarà:

$$V = A * l$$

Partiamo dal calcolo dell'area del tronco evidenziato in rosso. In questo caso ottengo un'area pari a $0.1257m^2$.

$$A = 3.1416 * 0.2^2 = 0.1257m^2$$

Ora calcolo l'area di tutti i tronchi: all'interno dell' immagine sono presenti 121 tronchi, 61 di area molto simile al tronco di area data, 22 leggermente più grandi e 38 più piccole rispetto a quella data. Tale ipotesi è stata formulata sulla base delle misurazioni, effettuate tramite un righello, di ogni singolo tronco riportato nell'immagine. L'area totale sarà:

$$A = 0.125664 * 61 + 0.125664 * 38 * 0.4 + 0.125664 * 22 * 1.2 = 12.89m^2$$

Infine moltiplico l'area totale per la lunghezza media dei tronchi ottenendo il volume totale:

$$V = 12.89 * 0.9 = 11.6m^3$$

3.3.4 Soluzione attraverso algoritmo

In questa sottosezione applico il terzo metodo per il calcolo del volume dei tronchi della Figura 3.19. Per comprendere le fasi dell'algoritmo presento lo pseudocodice.

Algorithm 5 Algoritmo 3 - Segmentazione

```
1: immagine  $\leftarrow$  filepath(immagine)
2: trackbars  $\leftarrow$  CreateTrackbars()
3: immagine_grigio  $\leftarrow$  CvtColor(immagine)
4: immagine_finale  $\leftarrow$  Threshold(immagine_grigio, trackbars)
```

Presento una spiegazione delle funzioni utilizzate dall'algoritmo:

- **CreateTrackbars**: Crea le sbarre di scorrimento nella finestra 3.18.
- **CvtColor**: la funzione trasforma i colori dell'immagine in sfumature di grigio ottenendo la Figura 3.21.
- **Applythreshold**: la funzione fa diventare di colore bianco tutti i pixel sotto alla soglia scelta, mentre se sopra la soglia li fa diventare di colore nero; utilizziamo le sfumature di grigio per semplificarne il processo.
- **Threshold**: la funzione applica le impostazioni scelte nella finestra all'immagine ottenendo la Figura 3.7.



Figura 3.21: Esempio trasformato in sfumature di grigio applicando la funzione CvtColor

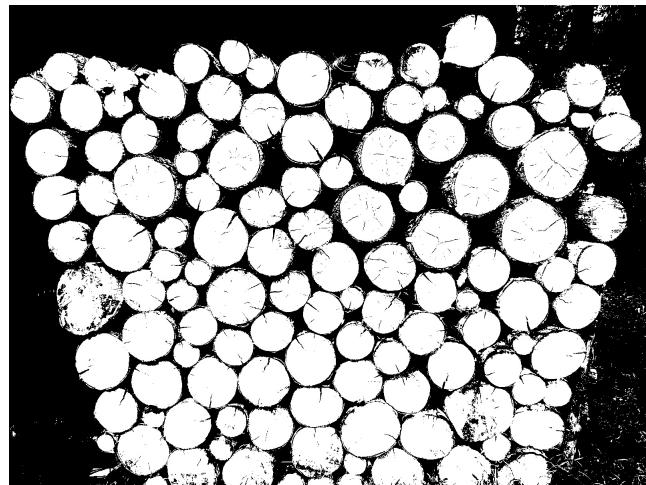


Figura 3.22: Esempio con applicata segmentazione

Adesso dato che la segmentazione è conclusa procedo con la misurazione che consiste nel calcolo del volume. Data la Figura 3.22, parto dal calcolo dell'area del tronco cerchiato in rosso in 3.24.

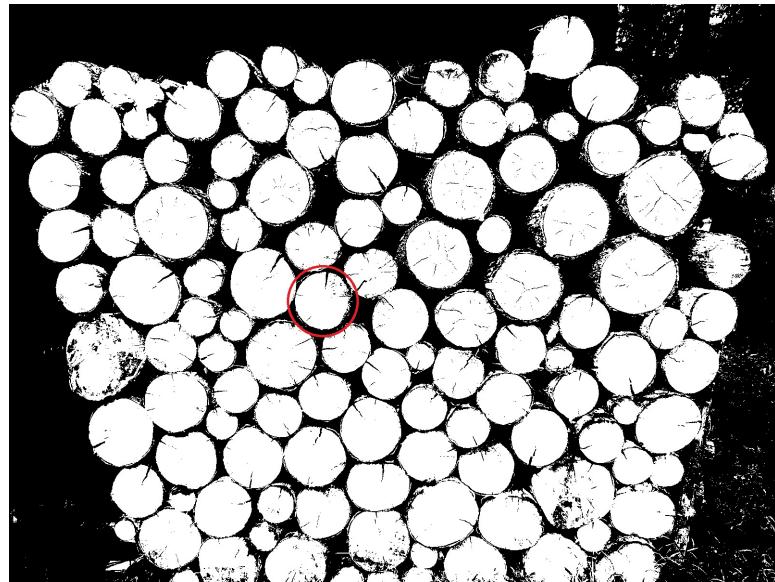


Figura 3.23: Esempio segmentato con evidenziato il tronco di cui la grandezza reale è conosciuta



Figura 3.24: Esempio tagliato per avere solo parte di cui so la grandezza reale

Inserisco la Figura 3.24 e la Figura 3.23 nell'algoritmo 6 che procederà alla misurazione.

Algorithm 6 Algoritmo 3 - Misurazione

```

1: immagine_segmentata  $\leftarrow$  filepath(posizione_immagine_finale)
2: immagine_segmentata_cerchiorosso  $\leftarrow$  filepath(posizione_immagine_cerchiorosso)
3: hh, ww  $\leftarrow$  dimensioni(immagine_segmentata_cerchiorosso)
4: hh1, ww1  $\leftarrow$  dimensioni(immagine_segmentata)
5: pixel_per_metro  $\leftarrow$  funzione1(immagine_segmentata_cerchiorosso, area_data)
6: area_finale  $\leftarrow$  funzione2(pixel_per_metro, area_data, immagine_segmentata)
7: volume_finale  $\leftarrow$  area_finale * lunghezza_data

```

Presento una spiegazione delle funzioni utilizzate dall'algoritmo:

- **funzione1**: la funzione calcola il numero di pixel per metro come rapporto tra l'area data e il numero totale di pixel del foreground.
- **funzione2**: la funzione calcola l'area finale come prodotto tra il totale dei pixel e i pixel per metro.

Nell'appendice riporto i calcoli con i relativi risultati, e ottengo come area totale dei tronchi $12.5m^2$ e come volume $11.27m^3$.

3.3.5 Confronto tra calcolo a mano e algoritmo

Confronto i risultati ottenuti utilizzando il calcolo a mano con quelli ottenuti attraverso algoritmo. Dal calcolo a mano ho ottenuto un volume pari a $11.6m^3$, mentre attraverso l'algoritmo il risultato è pari a $11.27m^3$. Ai fini di una migliore comprensione preciso che durante il calcolo "manuale" ho semplificato la geometria dei tronchi, trattandoli come cilindri perfetti. Si può dedurre che probabilmente l'algoritmo, che tiene conto di ogni minima variazione nella geometria dei vari tronchi, sia più preciso rispetto al calcolo "manuale" fatto da me. I due risultati si discostano tra di loro per un fattore del 2.9%, questo può considerarsi un risultato apprezzabile in quanto i due valori sono vicini. Se considerassi infine il fattore tempo, il calcolo a mano mi ha impiegato 10 minuti, invece attraverso l'algoritmo sono stati necessari solo 30 secondi.

Ricordo che questo algoritmo ha la limitazione presentata all'inizio della sezione, cioè è necessario che l'immagine scelta abbia background di colore diverso rispetto all'oggetto in foreground.

Capitolo 4

Limiti nell'uso degli algoritmi

4.1 Il problema della luminosità

4.1.1 Di cosa si tratta

Gli algoritmi sopra riportati si basano tutti sulla differenza chiaro-scuro tra background e foreground, quindi se uno sfondo presenta sia parti chiare che scure non è possibile applicare accuratamente la segmentazione e conseguente misurazione dell'area dei tronchi dalle foto.

Per esempio nelle Figure 4.1 e 4.2.



Figura 4.1: Esempio con background con parti chiare e parti scure



Figura 4.2: Esempio con background con parti chiare e parti scure

Per esempio se alla Figura 4.1 applico l’Algoritmo 5 con cattura dei colori chiari ottengo la Figura 4.3.



Figura 4.3: Esempio a cui è stata applicata la segmentazione chiaro-scuro

Invece se applico l’Algoritmo 5 con cattura dei colori scuri alla Figura 4.1 ottengo la Figura 4.4.



Figura 4.4: Esempio a cui è stata applicata la segmentazione scuro-chiaro

In entrambi i casi la segmentazione dell’immagine non è avvenuta correttamente e perciò anche il calcolo dell’area dei tronchi risulta errato.

4.1.2 Possibili soluzioni

La soluzione più semplice consiste in ritagliare la foto dalle parti non necessarie. Per esempio la Figura 4.1 diverrebbe come in Figura 4.5.



Figura 4.5: Esempio dove la parte esterna è stata ritagliata con un tool esterno [2]

Questo però renderebbe superfluo il lavoro fatto per catturare solo il foreground e inoltre non è detto che l’utente abbia i mezzi necessari per tale operazione.

Un'altra possibile soluzione è colorare le parti del background che venivano confuse dall'algoritmo per foreground. Per esempio la Figura 4.1 diventerebbe come in Figura 4.6.



Figura 4.6: Esempio dove è stata colorata la parte che rovinava la segmentazione

Anche questa soluzione purtroppo potrebbe non essere possibile da un utente inesperto o potrebbe essere considerato un modo per non affrontare il problema della segmentazione. Ciò che si può concludere da questa limitazione è che l'utente deve necessariamente fare una o più operazioni prima di poter utilizzare gli algoritmi su questa tipologia d'immagini altrimenti i risultati che si otterrebbero non sarebbero significativi.

4.2 Il problema della qualità delle immagini

4.2.1 Di cosa si tratta

Una delle problematiche che hanno ostacolato questo mio lavoro sulle immagini è stata la risoluzione delle fotografie, la quale si è presentata costantemente durante tutto lo sviluppo della tesi.

La bassa qualità delle immagini è legata ai limiti della moderna tecnologia, in particolare dalle fotocamere dei dispositivi comuni, come il telefonino o altri dispositivi che non hanno come obiettivo principale quello di effettuare fotografie. La segmentazione e la misurazione, se effettuate su immagini di bassa qualità risultano inefficienti qualora l'estrazione di dati richiesta sia troppo dettagliata.

Ad esempio se applicassi l'Algoritmo 5 alla Figura 4.7, la quale ha un qualità non elevata, il risultato che otterrei sarebbe lontano dal risultato attuale calcolato a mano.



Figura 4.7: Esempio 1

4.2.2 Possibili soluzioni

Una soluzione possibile può essere quella di utilizzare tool esterni come *Image Upscaler Online* [8] i quali aumentano la qualità dell'immagine (ad esempio in Figura 4.8 si vede come la qualità sia migliorata dalla figura precedente) e successivamente fornire l'immagine migliorata come input all'algoritmo scelto.



Figura 4.8: Immagine a cui ho applicato un *Upscaler*

Purtroppo non è detto che l'utente abbia i mezzi necessari per avere la risoluzione delle proprie immagini sufficientemente alta qualora la richiesta di segmentazione sia troppo specifica. Quindi riguardo questo problema non esiste una vera e propria soluzione, soltanto lo sviluppo tecnologico e la continua ricerca sulle nuove tecnologie permetterà l'attenuazione del problema nel lungo termine.

4.3 Il problema delle immagini con oggetti in parte coperti

4.3.1 Di cosa si tratta

Gli algoritmi sopra riportati si basano tutti sul lavorare sui pixel che contiene un'immagine, quindi se avessi un'immagine con degli oggetti che sono tra la fotocamera e i tronchi questi ultimi non sarebbero interamente identificati e ciò non permetterebbe di contare correttamente il numero di pixel dei tronchi nell'immagine.

Per esempio nella Figura 4.9¹ parte dei tronchi è coperta dall'erba, dando così problemi agli algoritmi di segmentazione.



Figura 4.9: Immagine di tronchi con erba davanti

¹Foto fatta da me in Trentino in data 08/25/2022

4.3.2 Possibili soluzioni

Una possibile soluzione è, teoricamente, con l'aiuto dell'*Intelligenza Artificiale* completare le sagome dei tronchi coperte dall'erba e successivamente applicare l'algoritmo di segmentazione alla nuova immagine ottenuta. Questa soluzione presenta molti problemi, dal più ovvio come il costo che potrebbe avere per l'utente utilizzare un'applicazione d'Intelligenza Artificiale necessaria per questo compito, o prima ancora dalla difficoltà di creare un'applicazione capace di fare ciò, nel caso tale strumento non esistesse o non fosse adatto alla soluzione.



Figura 4.10: Immagine dove i tronchi sono stati completati artificiosamente

Capitolo 5

Conclusioni e possibilità future

Questo studio ha trovato e presentato vari metodi per la soluzione del problema dell'estrazione dei dati da un'immagine. Nello specifico ho analizzato il problema della segmentazione delle immagini e della conseguente misurazione di un oggetto (nello specifico ho effettuato il calcolo del volume di alcuni tronchi presenti in fotografie).

Ho presentato tre algoritmi differenti per complessità e limiti ed ho applicato a ognuno varie immagini così da comprenderne il funzionamento. E' importante tenere presente che questa ricerca, effettuata tramite algoritmi, si è conclusa positivamente, riportando una percentuale di errore minore del 7% rispetto al valore ottenuto mediante calcolo manuale (quest'ultimo con le semplificazioni evidenziate in precedenza). Successivamente ho presentato i vari problemi che si hanno nell'utilizzo delle immagini: la luminosità, la qualità ed eventuali oggetti nel mezzo.

Questo risultato è coerente con l'aspettativa inizialmente espressa nell'elaborato, secondo la quale l'utilizzo delle nuove tecnologie permetterebbe potenzialmente a chiunque di applicare uno degli algoritmi sopra riportati per risolvere tale problema (determinazione del volume di vari oggetti riportati in una fotografia).

Il presente studio integra le nuove tecnologie e la moderna qualità delle immagini. Per quanto ovvio la qualità è determinata dalla risoluzione delle immagini. Tale fattore, tuttavia, rappresenta anche il limite attuale di questi algoritmi. Si ipotizza che in futuro, grazie al miglioramento della tecnologia e conseguentemente all'incremento della risoluzione delle immagini, la risoluzione delle immagini su cui gli algoritmi potranno lavorare sarà tale da aumentare l'accuratezza dei risultati.

Una raccomandazione per ulteriori ricerche future potrebbe essere quella di realizzare uno studio simile, ma che possa essere applicabile a un ordinamento qualsiasi della posizione dei tronchi, cioè che gli oggetti siano disposti in maniera casuale nello spazio di modo che non ci sia bisogno che tutti siano rivolti verso la macchina fotografica.

Come nota finale, ritengo che sarebbe vantaggioso ampliare gli algoritmi sopra presentati per permettere il calcolo del fattore di traduzione di grandezza da immagine a realtà, che consente di mappare la dimensione reale di un oggetto all'interno dell'immagine. In questo modo evitiamo che l'utente perda tempo misurandolo manualmente nella realtà per poi inserirlo nell'algoritmo. Ciò comporterebbe un'ulteriore semplificazione nell'utilizzo degli algoritmi di segmentazione, così da renderli più *User Friendly*.

Capitolo 6

Appendice

6.1 Calcoli dello pseudocodice degli algoritmi di misurazione

6.1.1 Calcoli dello pseudocodice dell'Algoritmo 2

Presento i calcoli che produce l'Algoritmo 2 dati come input le Figure 3.9 e 3.7.

1. Carica la Figura 3.9 e prende le sue dimensioni (altezza=86, larghezza=82)
2. Carica la Figura 3.7 e prende le sue dimensioni(altezza=427, larghezza=640)
3. Prende la media di pixel bianchi della Figura 3.9
4. Calcola il totale di pixel bianchi della Figura 3.9 (5985)
5. Prende da input l'area del tronco circondato in rosso (0.07)
6. Calcola quanto vale un pixel nella realtà (1.17×10^{-5})
7. Prende la media di pixel bianchi della Figura 3.7
8. Calcola il totale di pixel bianchi della Figura 3.7 (190535)
9. Calcola l'area reale della Figura 3.7 (2.2284)
10. Prende da input la lunghezza media dei tronchi (2)
11. Infine calcola il volume totale (4.4568)

6.1.2 Calcoli dello pseudocodice dell'Algoritmo 4

Presento i calcoli che produce l'Algoritmo 4 dati come input le Figure 3.17 e 3.16.

1. Carica la Figura 3.17 e prende le sue dimensioni (altezza=406, larghezza=392)
2. Carica la Figura 3.16 e prende le sue dimensioni (altezza=406, larghezza=800)
3. Prende la media di pixel bianchi della Figura 3.17
4. Calcola il totale di pixel bianchi della Figura 3.17 (116671)
5. Prende da input l'area del tronco circondato in rosso (3.1416)
6. Calcola quanto vale un pixel nella realtà (2.69×10^{-5})
7. Prende la media di pixel bianchi della Figura 3.16
8. Calcola il totale di pixel bianchi della Figura 3.16 (220505)
9. Calcola l'area reale della Figura 3.16 (5.9375)
10. Prende da input la lunghezza media dei tronchi (15)
11. Infine calcola il volume totale (89.06)

6.1.3 Calcoli dello pseudocodice dell'Algoritmo 6

Presento i calcoli che produce l'Algoritmo 6 dati come input le Figure 3.24 e 3.22.

1. Carica la Figura 3.24 e prende le sue dimensioni (altezza=119, larghezza=116)
2. Carica la Figura 3.22 e prende le sue dimensioni (altezza=1200, larghezza=1600)
3. Prende la media di pixel bianchi della Figura 3.24
4. Calcola il totale di pixel bianchi della Figura 3.24 (10885)
5. Prende da input l'area del tronco circondato in rosso (0.1257)
6. Calcola quanto vale un pixel nella realtà (1.1548)
7. Prende la media di pixel bianchi della Figura 3.22
8. Calcola totale di pixel bianchi della Figura 3.22(1084552)
9. Calcola l'area reale della Figura 3.22 (12.5)
10. Prende da input la lunghezza media dei tronchi (0.9)
11. Infine alcola il volume totale (11.2716)

Bibliografia

- [1] Apple app. <https://apps.apple.com/us/app/measure/id138342674>. Ultimo accesso: 08-22-2022.
- [2] Background remover. <https://www.adobe.com/it/express/feature/image-remove-background>. Ultimo accesso: 08-22-2022.
- [3] Basic background remover with opencv. <https://www.freedomvc.com/index.php/2022/01/17/basic-background-remover-with-opencv/>. Ultimo accesso: 08-22-2022.
- [4] Basic thresholding operations main page. https://docs.opencv.org/3.4/db/d8e/tutorial_threshold.html. Ultimo accesso: 08-22-2022.
- [5] Google lens. <https://lens.google/#identify>. Ultimo accesso: 2022-08-22.
- [6] How to remove the background from a picture in opencv. <https://stackoverflow.com/questions/64491530/how-to-remove-the-background-from-a-picture-in-opencv-pythonl>. Ultimo accesso: 08-22-2022.
- [7] Opencv mainpage. <https://docs.opencv.org/3.4/index.html>. Ultimo accesso: 08-22-2022.
- [8] Upscaler. <https://it.aiseesoft.com/image-upscaler/>. Ultimo accesso: 08-22-2022.
- [9] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. 2007. pages=11-249.