

Università degli Studi di Bari

Dipartimento di Informatica



**Tesi di Laurea Triennale in Informatica e Tecnologie per la
Produzione del Software**

Progettazione sistema SPIR

Autore:

Antonio Ricciardi

Relatore:

Giovanni Dimauro

Correlatori:

Rosalia Maglietta

Carla Cherubini

Anno Accademico 2022/2023

Indice

1	Introduzione	3
1.1	Stato dell'arte	3
1.1.1	Elaborazione delle immagini	3
1.1.2	Feature Extraction	4
1.1.3	Object Segmentation	4
1.1.4	User Interface	4
1.1.5	SPIR	4
1.2	Obiettivi	4
2	Metodi	5
2.1	Porting del sistema SPIR	5
2.1.1	Riscrittura dell'algoritmo originale (MATLAB) in Python .	5
2.1.2	Utilizzo di Python come linguaggio per backend di SPIR .	8
2.1.3	Estrazione della maschera e della pinna del delfino . . .	8
2.1.4	Estrazione delle feature e salvataggio del dataset	8
2.1.5	Match basato sulle feature estratte da SIFT	8
2.1.6	Architettura del sistema	8
2.2	Approccio basato su OpenCV per l'estrazione delle maschere .	8
2.2.1	Miglioramenti apportati al processo di ritaglio	8
2.2.2	Parallelismo e tempi di esecuzione ridotti	8

2.2.3	Riduzione dello spazio occupato dal dataset di feature estratte	8
2.2.4	Adozione di gRPC per il trasferimento dei dati	8
2.3	Approccio con deep learning per l'estrazione delle maschere	8
2.3.1	Implementazione di una rete neurale basata su U-Net	8
2.3.2	Utilizzo delle maschere per il ritaglio della pinna	8
2.4	Interfaccia utente e interazione	8
2.4.1	Creazione di un'interfaccia per l'interazione con il bac- kend Python	8
2.4.2	Utilizzo di Flutter per sviluppare un'interfaccia multiplat- taforma	8
3	Risultati sperimentali	9
3.1	Valutazione dei risultati	9
3.1.1	Confronto tra i diversi approcci implementati	9
3.1.2	Analisi delle prestazioni e dei risultati ottenuti	9
3.2	Limitazioni e problematiche riscontrate con approccio Deep Lear- ning	9
4	Conclusioni	10
4.1	Riepilogo degli obiettivi raggiunti	10
4.2	Possibili sviluppi futuri	10
5	Codice sorgente	11
5.1	Implementazione dell'algoritmo SPIR	11
5.2	Implementazione della rete neurale U-Net	11
5.3	Codice per l'estrazione delle maschere con OpenCV	11
5.4	Codice per l'interfaccia utente in Flutter	11

Capitolo 1

Introduzione

Fondamentale implementare la fotoid

1.1 Stato dell'arte

1.1.1 Elaborazione delle immagini

La manipolazione e l'elaborazione delle immagini svolgono un ruolo fondamentale in numerosi settori, tra cui la visione artificiale, la grafica computazionale e l'analisi di immagini biomediche. In quest'ambito, gli operatori morfologici e logici giocano un ruolo chiave nella modifica e nell'estrazione di informazioni significative da immagini digitali.

Gli operatori morfologici sono basati sulla teoria della morfologia matematica e vengono utilizzati per la manipolazione di forme geometriche all'interno di un'immagine.

Essi consentono di eseguire operazioni come l'erosione, la dilatazione, l'apertura e la chiusura, che possono essere utilizzate per rimuovere rumore, riempire buchi o separare oggetti connessi.

L'erosione riduce le regioni di colore o intensità nell'immagine, mentre la dilatazione le amplia. L'apertura è una combinazione di erosione seguita da dilata-

zione e viene utilizzata per rimuovere piccoli oggetti o dettagli indesiderati. La chiusura, al contrario, consiste in una dilatazione seguita da erosione ed è utile per riempire buchi o connettere regioni separate. Parallelamente agli operatori morfologici, gli operatori logici sono ampiamente utilizzati per combinare e confrontare immagini. Gli operatori logici fondamentali includono l'AND, l'OR e il NOT, che consentono di creare maschere o combinare informazioni da diverse immagini.

Ad esempio, l'operatore AND può essere utilizzato per identificare le regioni comuni tra due immagini, mentre l'operatore OR può essere impiegato per unire due immagini sovrapposte in una sola immagine risultante.

L'operatore NOT, invece, inverte i valori di pixel di un'immagine, consentendo di creare effetti speciali o evidenziare determinate regioni.

1.1.2 Feature Extraction

1.1.3 Object Segmentation

1.1.4 User Interface

1.1.5 SPIR

1.2 Obiettivi

Capitolo 2

Metodi

2.1 Porting del sistema SPIR

In una prima fase di analisi di SPIR (MATLAB) si nota che l’interazione è confinata a linea di comando (CLI), in ottica di offrire le funzionalità di SPIR a una più ampia fascia di utenza, mantendo anche la possibilità di usufruire del codice sorgente da parte di utenti più esperti si è optato per una implementazione basata su client/server, in particolare slegando le componenti di interazione e logica del sistema si ha la possibilità di utilizzare le tecnologie più appropriate a seconda del contesto, per poi mettere in comunicazione le varie componenti.

2.1.1 Riscrittura dell’algoritmo originale (MATLAB) in Python

Lo sviluppo si è concentrato inizialmente sulla riscrittura delle funzionalità principali di SPIR, come linguaggio è stato scelto Python per i seguenti motivi:

- Vasta gamma di librerie disponibili.
- Utilizzo di OpenCV per l’elaborazione delle immagini.
- Buona diffusione in ambito scientifico.
- Supporto attivo dalla community di sviluppatori.

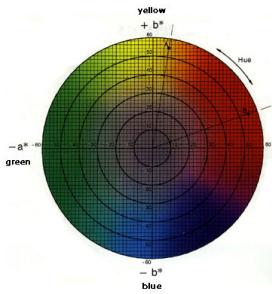


Figura 2.1: CIELAB 2D

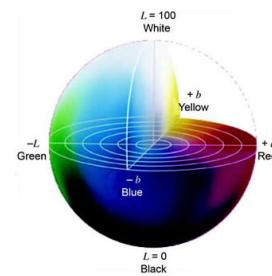


Figura 2.2: CIELAB 3D

La prima funzionalità presa in analisi per il porting è stata l'estrazione delle maschere dalle immagini, di fondamentale importanza al fine di ottimizzare l'area di estrazione delle features, l'estrazione delle maschere viene fatta nel metodo

```
prepare_images_vlfeat.m
```

L'immagine viene convertita nello spazio colore Cielab, è uno spazio colore-opponente con la dimensione L per la luminosità e a e b per le dimensioni colore-opponente, basato sulle coordinate dello spazio colore non lineare compresso CIE XYZ. La luminosità è calcolata usando la radice cubica della luminanza relativa. Lab include tutti i colori percepibili, perciò include completamente i gamut degli spazi colore RGB e CMYK ed è indipendente dal dispositivo che li rappresenta.



Figura 2.3: Immagine originale Figura 2.4: Immagine CIELAB Figura 2.5: Canale B estratto

Come nella versione originale il primo passo è convertire l’immagine da RGB a CIELAB, a differenza di matlab OpenCV utilizza il formato BGR, ovvero il vettore dei colori per ogni pixel è invertito, dall’immagine convertita in spazio colore CIELAB si estraе il canale B dall’immagine questo permette di mettere in evidenza le aree blu dell’immagine così da isolare la parte di mare presente nello sfondo delle foto.

```

1 #Caricamento immagine originale
2 image = cv2.imread("BEN_180713_4.png")
3
4 #Conversione immagine in spazio colore CIELAB
5 im_lab = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
6
7 #Estrazione del canale B
8 b = im_lab[:, :, 2]
9
10 # Applica il metodo di Otsu al canale B
11 thr_b, _ = cv2.threshold(b, 0, 255, cv2.THRESH_BINARY +
                           cv2.THRESH_OTSU)

```

La sogliatura di Otsu, chiamata anche metodo di Otsu, è un algoritmo utilizzato per determinare automaticamente il valore di soglia ottimale per la segmentazione di un’immagine in bianco e nero. L’obiettivo della sogliatura è quello di separare i pixel dell’immagine in due classi, generalmente il primo piano (oggetti di interesse) e lo sfondo.

L’algoritmo di Otsu calcola la soglia ottimale in base all’istogramma dei livelli di grigio dell’immagine. L’istogramma rappresenta la distribuzione dei valori di grigio presenti nell’immagine. L’obiettivo è trovare la soglia che massimizza la varianza tra le due classi (primo piano e sfondo), considerando anche la distribuzione dei pixel di ciascuna classe. In altre parole, si cerca la soglia che massimizza la separazione tra i due gruppi di pixel.

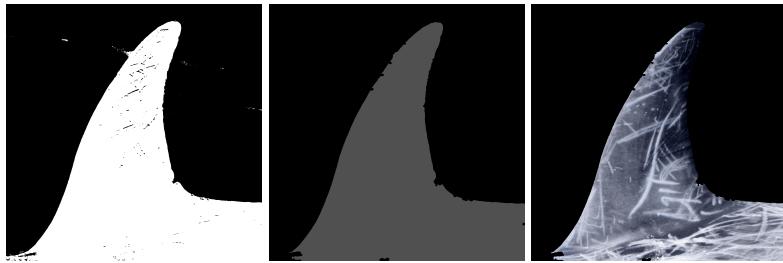


Figura 2.6: Metodo OTSU

Figura 2.7: Operatori morfologici e pulizia

Figura 2.8: Pinna estratta

2.1.2 Utilizzo di Python come linguaggio per backend di SPIR

2.1.3 Estrazione della maschera e della pinna del delfino

2.1.4 Estrazione delle feature e salvataggio del dataset

2.1.5 Match basato sulle feature estratte da SIFT

2.1.6 Architettura del sistema

2.2 Approccio basato su OpenCV per l'estrazione delle maschere

2.2.1 Miglioramenti apportati al processo di ritaglio

2.2.2 Parallelismo e tempi di esecuzione ridotti

2.2.3 Riduzione dello spazio occupato dal dataset di feature estratte

2.2.4 Adozione di gRPC per il trasferimento dei dati

2.3 Approccio con deep learning per l'estrazione delle maschere

2.3.1 Implementazione di una rete neurale basata su U-Net

2.3.2 Utilizzo delle maschere per il ritaglio della pinna

2.4 Interfaccia utente e interazione

2.4.1 Creazione di un'interfaccia per l'interazione con il backend Python

2.4.2 Utilizzo di Flutter per sviluppare un'interfaccia multipiattaforma

Capitolo 3

Risultati sperimentali

3.1 Valutazione dei risultati

3.1.1 Confronto tra i diversi approcci implementati

3.1.2 Analisi delle prestazioni e dei risultati ottenuti

3.2 Limitazioni e problematiche riscontrate con approccio Deep Learning

Capitolo 4

Conclusioni

4.1 Riepilogo degli obiettivi raggiunti

4.2 Possibili sviluppi futuri

Capitolo 5

Codice sorgente

5.1 Implementazione dell'algoritmo SPIR

5.2 Implementazione della rete neurale U-Net

5.3 Codice per l'estrazione delle maschere con OpenCV

5.4 Codice per l'interfaccia utente in Flutter