Tools recognition

Elaborazione delle immagini

(2022/2023)

Matlab

Indice:

Matlab

Lo scopo del progetto

Lo scopo del progetto è «realizzare un'applicazione che, data un'immagine, permetta di riconoscere una serie di utensili posizionati su un piano».

Per soddisfare la richiesta abbiamo creato un dataset da circa **460 immagini** su **8 sfondi diversi**, dove ogni oggetto è stato fotografato con **due o tre angolazioni** diverse per garantire varie prospettive dell'oggetto stesso.
Alla fine siamo arrivati alle **10 classi** degli utensili con una classe, chiamata «unknown», che trattiene tutti oggetti sconosciuti al classificatore.

Da notare che circa **un terzo** del dataset è stato riempito con le immagini create nell'Adobe Photoshop, risolvendo la mancanza degli utensili necessari dei componenti del gruppo.

Dataset:

Classe	N° forme oggetto	N° immagini
Avvitatore	13	34
Forbici	15	30
Brugola	20	19
Cacciavite	16	28
Chiave inglese	22	31
Martello	21	47
Taglierino	16	45
Pinze	11	41
Sega	15	42
Coltello	16	16
unknown	75	130
Totale	240	463

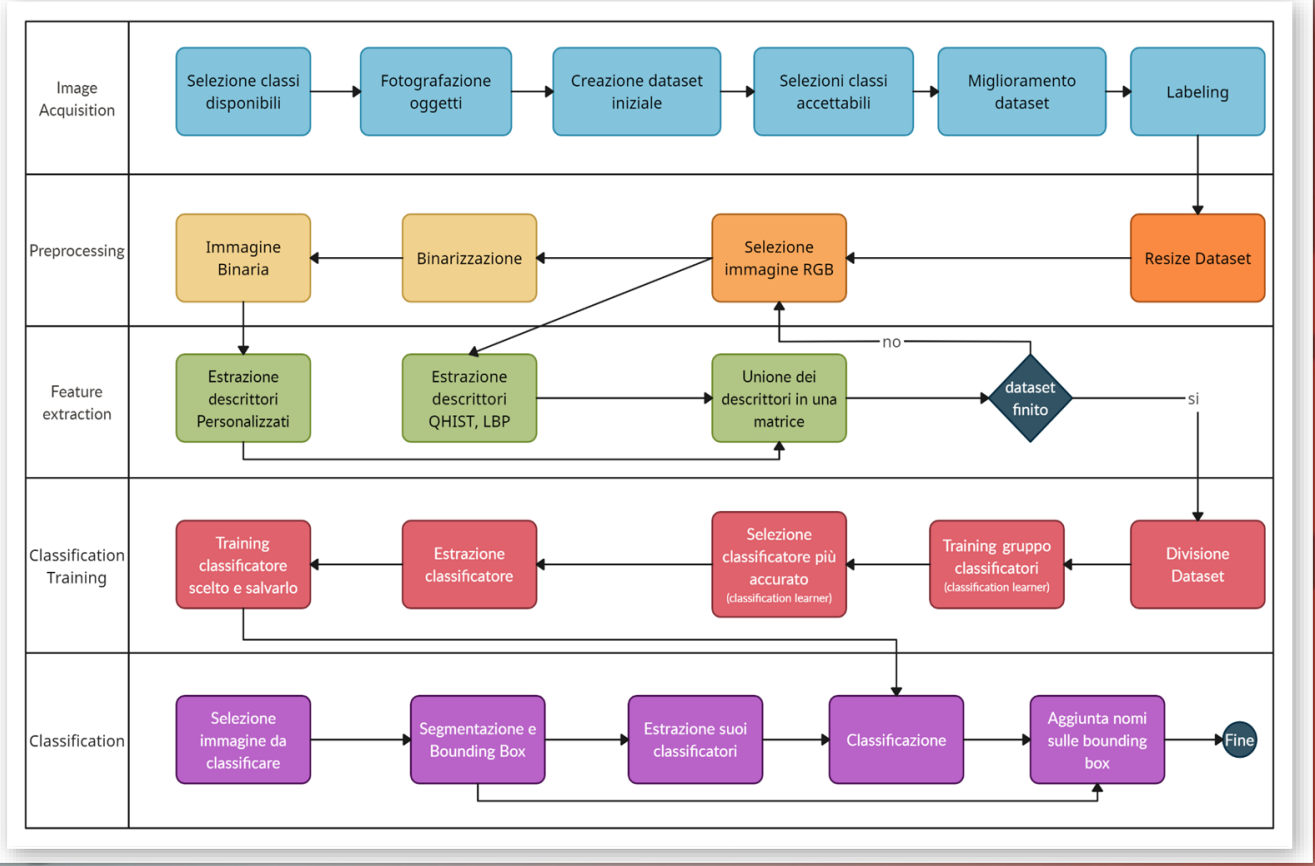
Matlab

Le assunzioni e le problematiche del progetto

Le assunzioni	Le problematiche
Si assume che non sono presenti le immagini con un grande **cambiamento di luce** (es: buio).	Il dataset è composto dalle immagini con le **risoluzioni diverse**, perciò nelle successive fasi queste verranno ridimensionate sotto le medesime scale.
Si assume che le immagini presenti sono state fotografate su uno **sfondo piano** (es: la tavola di colore omogeneo, di legno, le piastrelle del pavimento) **con e senza dei riflessi di luce**.	In fase di binarizzazione abbiamo riscontrato il problema delle **fughe delle piastrelle** (uno dei nostri sfondi), difficilmente cancellabili perché spesso eliminavano dei piccoli oggetti in esame, come le brugole.
Si assume che le immagini sono state fotografate a circa **50 cm** di distanza dall'oggetto in esame.	In fase di feature extraction abbiamo riscontrato la difficoltà di **contare il numero delle maniglie** dell'oggetto (es: se le pinze sono aperte, allora conta bene, altrimenti le considera un oggetto con una sola maniglia, perché la binarizzazione usa **gli elementi strutturali troppo grandi** per vedere la piccola distanza tra le due maniglie).
Si assume che nel dataset le immagini sono abbastanza **nitide**, senza rumori (anche se viene gestito il caso delle immagini rumorose).	
Si assume che nelle immagini con tanti oggetti, questi **non si sovrappongono** e sono abbastanza **distanti** sia tra di loro che dal bordo (circa 5 cm).	In fase di classificazione alcuni oggetti vengono **confusi tra di loro** (es: le forbici e le pinze, i cacciaviti ed i coltelli), a causa della loro **simile forma** o della **lettura parziale** degli oggetti (es: la lama del coltello spesso non viene letta, rimane solo il manico).

Matlab

La pipeline



Matlab

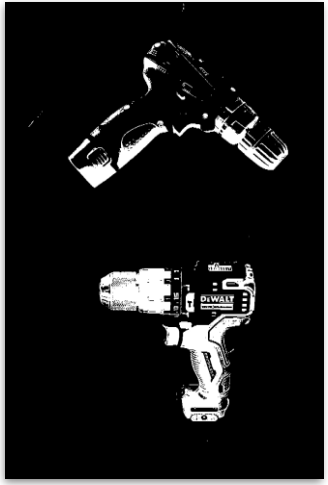
Il pre-processing

Il primo passo del pre-proccesing è il ridimensionamento dell'immagine, quindi portiamo l'immagine alla risoluzione **1400xN** (mantenendo la scala) e trasformiamo l'immagine ai livelli di grigio. Successivamente si applicano i seguenti passaggi:

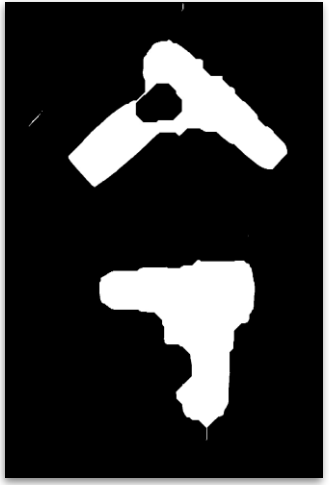
Trasformazioni top-hat

Gamma correction

Inversione dell'immagine



Operazioni morfologiche



L'idea principale è **minimizzare la quantità dei dettagli** dello sfondo per rendere meglio distinguibile la differenza tra lo sfondo e l'oggetto in esame.

Matlab

Il pre-processing (con ulteriore miglioramento)

Per gestire la presenza del rumore nel dataset, si aggiunge **il filtro mediano** sulla matrice 3x3.

Immagine **con il rumore** "salt & pepper" se **non viene applicato** il filtro

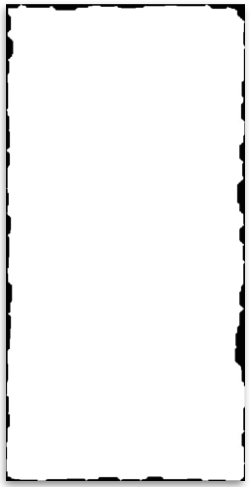


Immagine **con il rumore** "salt & pepper" se **viene applicato** il filtro

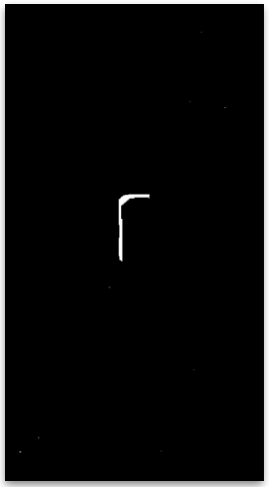
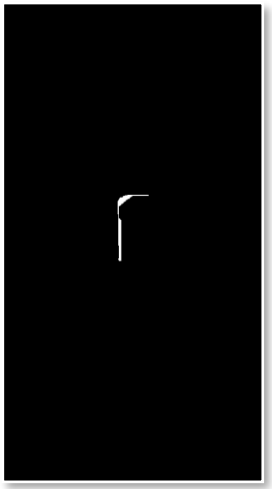


Immagine **senza il rumore** "salt & pepper" se **viene applicato** il filtro



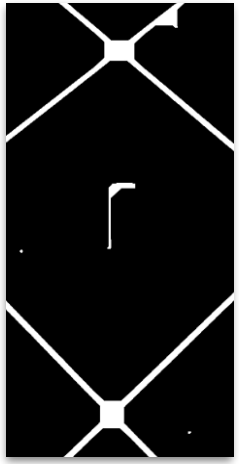
Da notare che il filtro mediano **corrompe leggermente i contorni** degli oggetti in esame nel caso dell'**immagine nitida**.

Matlab

Il pre-processing (le versioni scartate)

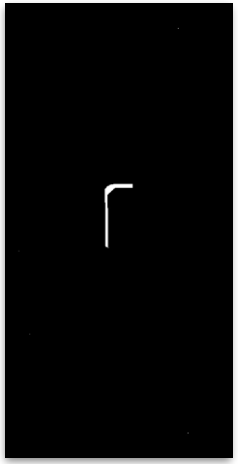
- Il filtro gaussiano;
- Il metodo di Otsu;
- L'operatore di Sobel;
- Le operazioni morfologiche;

- La trasformazione da RGB a Lab, applicando adapthisteq al canale della luminosità;
- Il filtro mediano su 3x3;
- Il filtro gaussiano;
- Il metodo di Otsu;
- L'operatore di Sobel;
- Le operazioni morfologiche;



- L'incremento delle intensità dei pixel tramite imadjust;
- Deviazione standard dell'immagine tramite stdfilt;
- Il metodo di Otsu;
- L'operatore di Sobel;
- Le operazioni morfologiche;

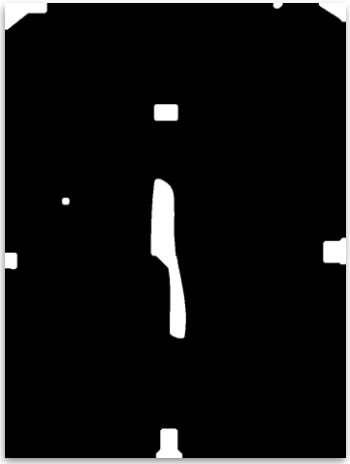
Risultato finale:



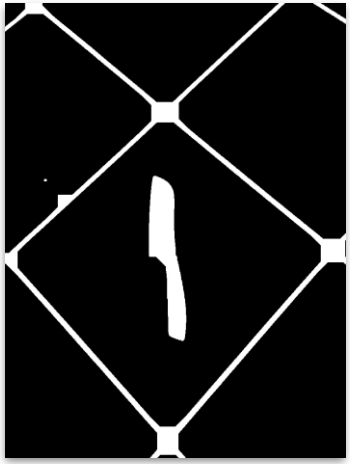
Matlab

Il pre-processing (le versioni scartate)

- Il filtro gaussiano;
- Il metodo di Otsu;
- L'operatore di Sobel;
- Le operazioni morfologiche;

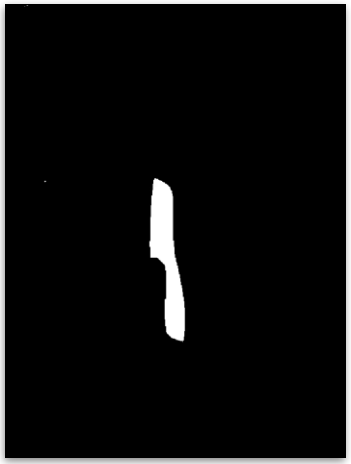


- La trasformazione da RGB a Lab, applicando adapthisteq al canale della luminosità;
- Il filtro mediano su 3x3;
- Il filtro gaussiano;
- Il metodo di Otsu;
- L'operatore di Sobel;
- Le operazioni morfologiche;



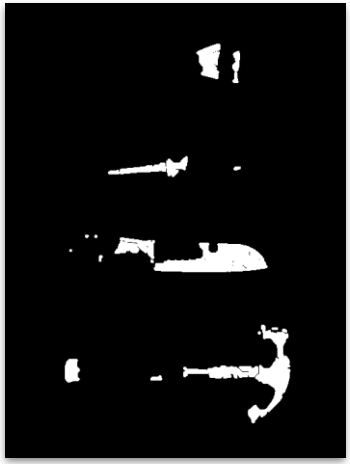
- L'incremento delle intensità dei pixel tramite imadjust;
- Deviazione standard dell'immagine tramite stdfilt;
- Il metodo di Otsu;
- L'operatore di Sobel;
- Le operazioni morfologiche;

Risultato finale:

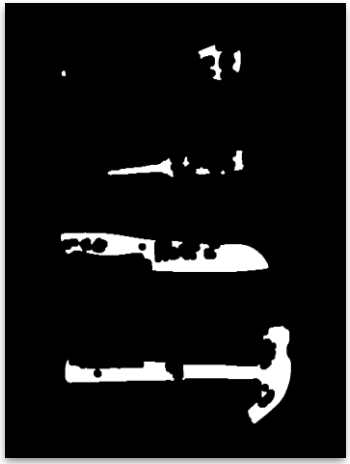


Matlab

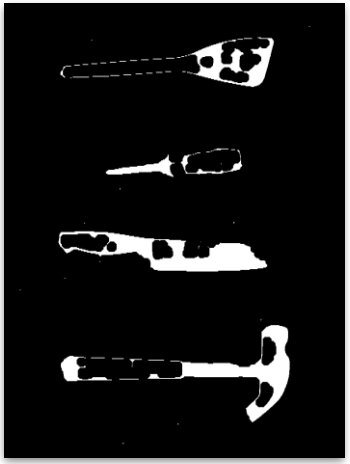
Il pre-processing (le versioni scartate)



- Il filtro gaussiano;
- Il metodo di Otsu;
- L'operatore di Sobel;
- Le operazioni morfologiche;

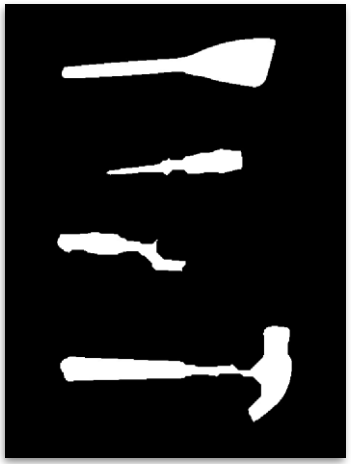


- La trasformazione da RGB a Lab, applicando adapthisteq al canale della luminosità;
- Il filtro mediano su 3x3;
- Il filtro gaussiano;
- Il metodo di Otsu;
- L'operatore di Sobel;
- Le operazioni morfologiche;



- L'incremento delle intensità dei pixel tramite imadjust;
- Deviazione standard dell'immagine tramite stdfilt;
- Il metodo di Otsu;
- L'operatore di Sobel;
- Le operazioni morfologiche;

Risultato finale:



Matlab

La segmentazione

Grazie alla binarizzazione e alle nostre assunzioni è stato possibile segmentare le immagini in modo alquanto diretto.

Per estrarre un oggetto alla volta bisogna seguire quindi questi 3 passaggi:

- Binarizzazione
- Labeling
- Covering

Esempio:

Matlab

Binarizzazione

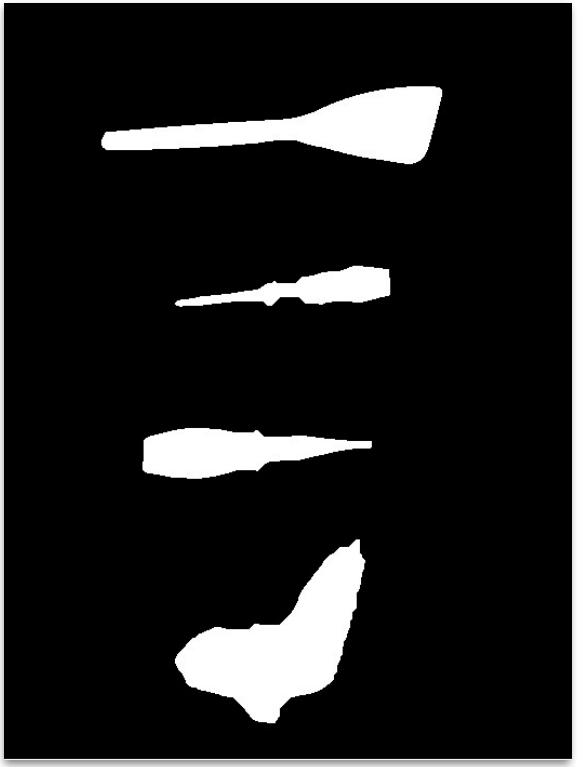
Questo passo, in maggior parte spiegato prima, ha richiesto una sola piccola aggiunta:

- togliere gli oggetti non desiderati.

Per questo abbiamo semplicemente eliminato le parti mantenute dalla binarizzazione con area troppo piccola. Nel nostro caso — 1500 pixel di area.

Dopo un po' di prove, abbiamo notato che **1500 è una soglia piuttosto buona**, permettendoci di mantenere anche gli oggetti più piccoli, come le brugole, ma allo stesso tempo eliminando una gran parte dei rumori rimasti dopo la binarizzazione.

Esempio:



Matlab

Labeling

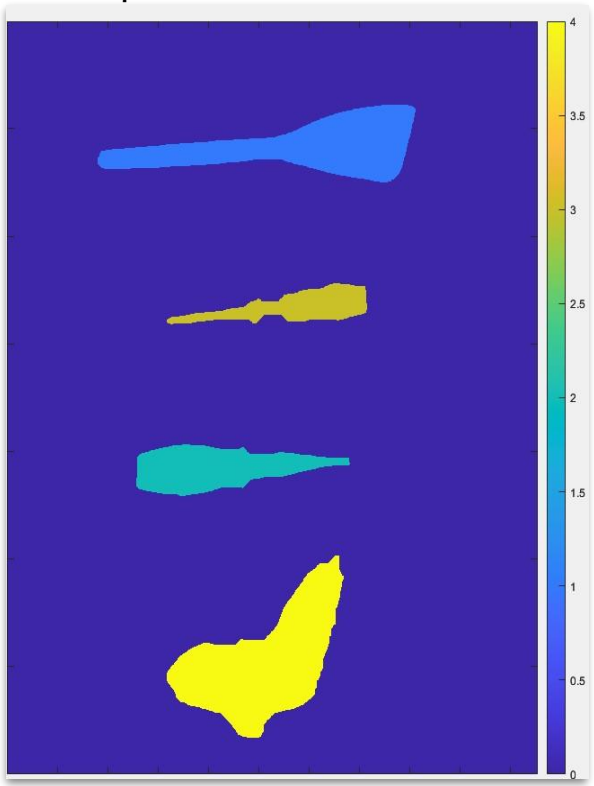
Dopo aver binarizzato l'immagine e rimosso gli oggetti piccoli, possiamo identificare ogni oggetto grazie alla **labeling delle componenti connesse**.

Questa ci permette di identificare ogni oggetto e anche di trovare quanti ne abbiamo trovati, indicato dall'etichetta più grande, quindi il massimo della matrice delle labels.

Ora possiamo trovare **le bounding box di ogni oggetto**, prendendo per ogni etichetta le seguenti 4 informazioni:

- La posizione del pixel più a sinistra.
- La posizione del pixel più a destra.
- La posizione del pixel più in alto.
- La posizione del pixel più in basso.

Esempio:

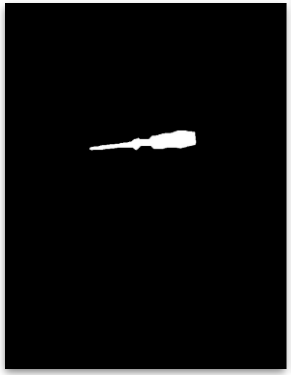


Matlab

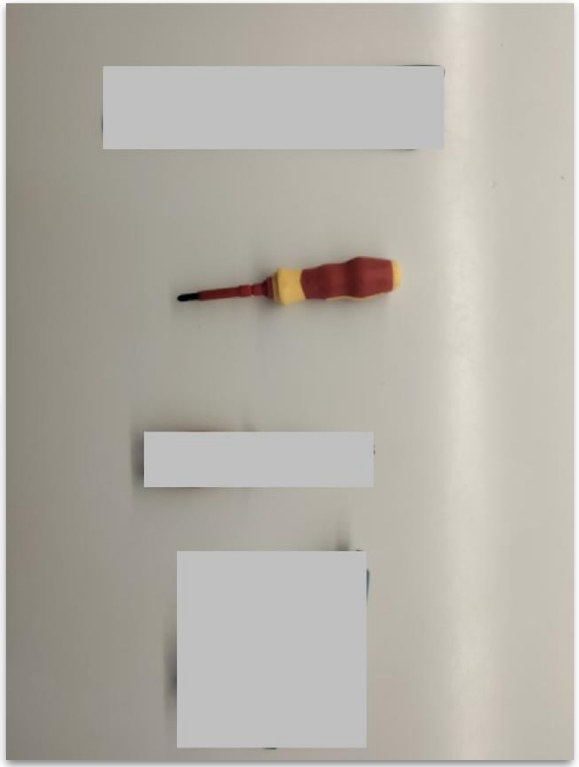
Covering

Una volta che abbiamo le bounding box per tutti gli oggetti dell'immagine, possiamo far sì di avere in evidenza solo uno di essi.
Per fare questo, **copriamo l'area delle bounding box** di tutti gli oggetti tranne quello che ci serve al momento in grigio, un colore scelto perché tra gli sfondi usati non viene evidenziato e non disturba troppo gli descrittori.

Questo metodo di segmentazione è uno tra quelli possibili. L'abbiamo scelto perché permette di rispettare le assunzioni esposte in precedenza, mantenendo la scala dell'oggetto per avere descrittori coerenti.



Esempio:



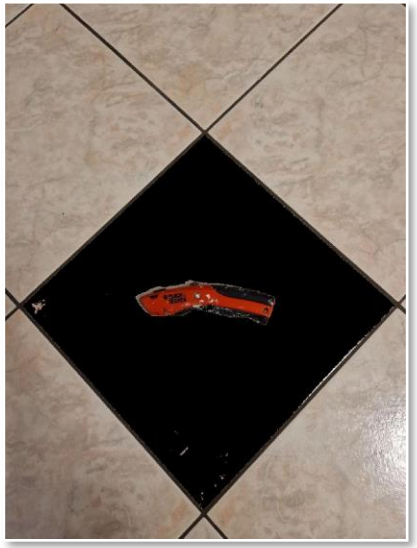
Matlab

Segmentazione Region Growing (la versione scartata)

Abbiamo provato un altro metodo di segmentazione inizialmente, usando la **region growing**. Purtroppo, oltre ad essere un processo alquanto pesante, alcuni dei nostri sfondi non permettevano il suo utilizzo.

A destra possiamo vedere un esempio, dove anche se l'oggetto viene selezionato alquanto bene, le fughe **bloccano la crescita dell'area**.

Dato che le immagini su questo tipo di sfondo non sono molto uniformi, **selezionare seed statiche è inutile** e cercare automaticamente delle posizioni ci porta ad un lavoro troppo impegnativo che è stato **risolto con la binarizzazione**.



Matlab

Il feature extraction

Dopo un periodo di ricerca, abbiamo deciso che il metodo migliore per distinguere tra loro le classi di attrezzi scelte è la loro **forma** ed alcune feature relative ad essa.

In congiunzione alle feature relative alla forma, abbiamo usato **LBP** per estrarre degli descrittori relativi alla texture e **QHIST** per la valutazione dell'istogramma, permettendoci di alzare di molto la precisione nella classificazione.

Per avere delle informazioni coerenti per la forma dell'oggetto, per primo lo ruotiamo, portandoci, quando possibile in una **posizione orizzontale**. In congiunzione, portiamo sempre la risoluzione a **700x700** a causa dell'impossibilità di mantenere una risoluzione uguale nella produzione del dataset.

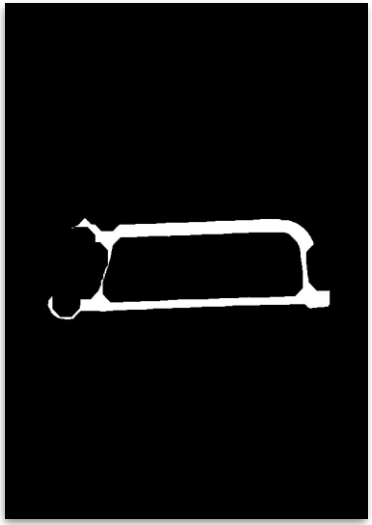
Feature create
L'area
La proporzione altezza/larghezza
Un sottoinsieme delle proiezioni
L'appartenenza all'oggetto del suo centro di gravità
La proporzione tra l'area convessa minima dell'oggetto e la sua area attuale
Il numero dei buchi
Il numero delle maniglie
Un insieme di distanze tra gli edge dell'oggetto e il centro di gravità

Matlab

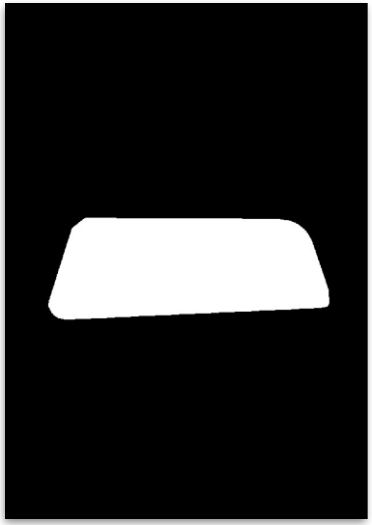
Feature (Solidity AreaConvessa/Area)

Per estrarre questi descrittori per un oggetto, facciamo i seguenti passaggi:

Binarizzazione



Area convessa min



Esempio proporzione

Sega	4.1
Avvitatore	1.3
Pinze	1.5
Taglierino	1.1
Chiave inglese	1.5
Cacciavite	1.9
Martello	2.3
Forbici	2.2
Brugola	2.6
Coltello	2.4

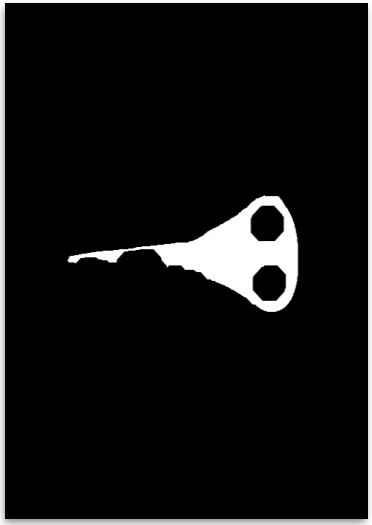
Per ottenere la solidità, divido l'area convessa minima per l'area dell'oggetto binario.

Matlab

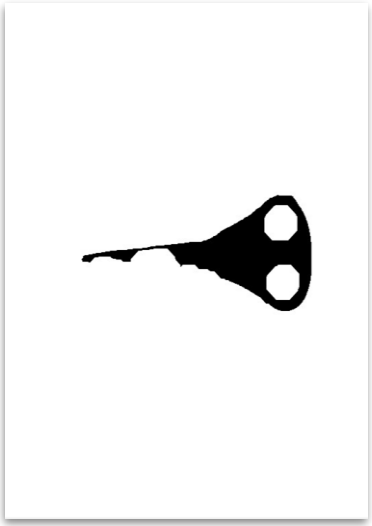
Feature (Numero buchi)

Per estrarre quanti buchi ha un oggetto, facciamo i seguenti passaggi:

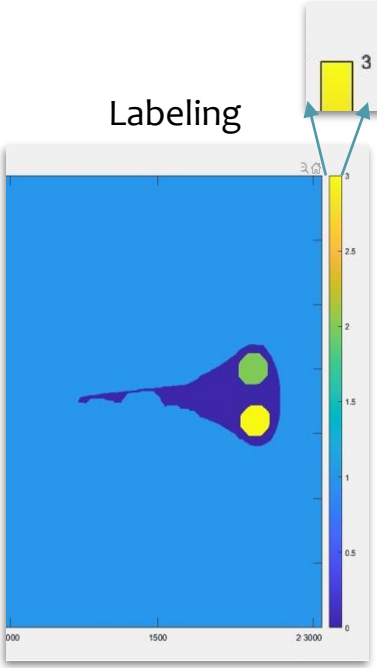
Binarizzazione



Inversione



Labeling



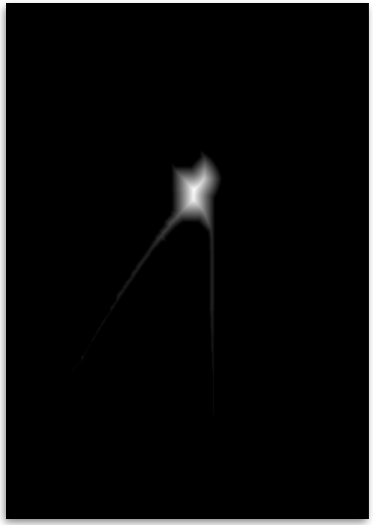
Dopo la labeling dell'inverso della binarizzazione, otteniamo che sono trovati tanti oggetti quanti **il numero di buchi più uno per lo sfondo**.

Matlab

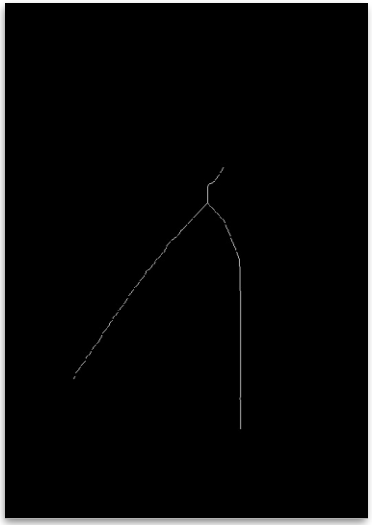
Feature (Numero maniglie)

Per estrarre quante maniglie ha un oggetto, facciamo i seguenti passaggi dopo la binarizzazione:

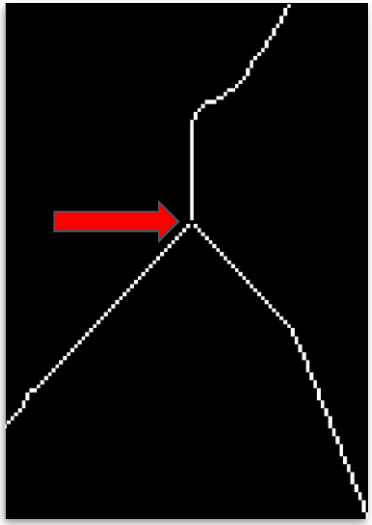
Trasformata a distanza



Scheletrizzazione



Rimozione pixel branching



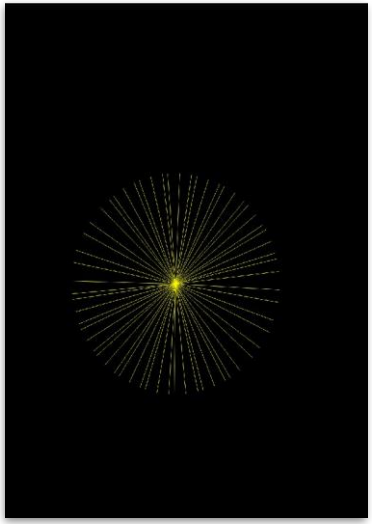
Dopo questi passi, calcoliamo **l'area**, quindi lunghezza dato che abbiamo larghezza di 1px, **delle varie componenti connesse**.
Quando abbiamo uno di questi **più grande della metà dell'altezza**, lo consideriamo come se fosse una maniglia.

Matlab

Feature (Distanze dal centro di gravità)

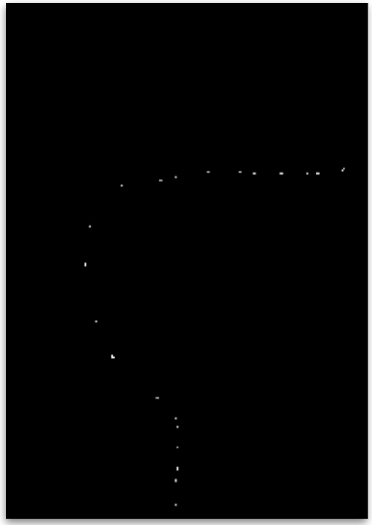
Per estrarre questi descrittori per un oggetto, facciamo i seguenti passaggi dopo la binarizzazione:

Creo linee di intersezione



Sottraggo agli edge le linee

Estraggo i punti di intersezione



Le **60** linee sono sempre lunghe 300 pixel e partono dal centro di gravità.
Infine per **60 punti di intersezione** calcoliamo la **distanza** tra di esso ed il centroide dell'oggetto.

Matlab

La classificazione

In quest'ultima fase, una volta ricevuta in input un'immagine, essa **viene segmentata** e per **ogni suo oggetto** viene creata **una bounding box**. A questo punto **si calcolano i descrittori** per ogni oggetto e si controlla nell'albero di decisione **a cosa coincidono**.
Infine come output otteniamo l'immagine in cui ogni suo oggetto ha la **propria bounding box** con il **nome della classe di appartenenza**.

Tra tutti classificatori che abbiamo provato, i migliori risultati sono stati dati da questi tre tipi di classificatori, tra cui **il migliore è stato il terzo**:

- Albero di decisione
- Classificatore K-NN
- Ensemble bagged trees

Matlab

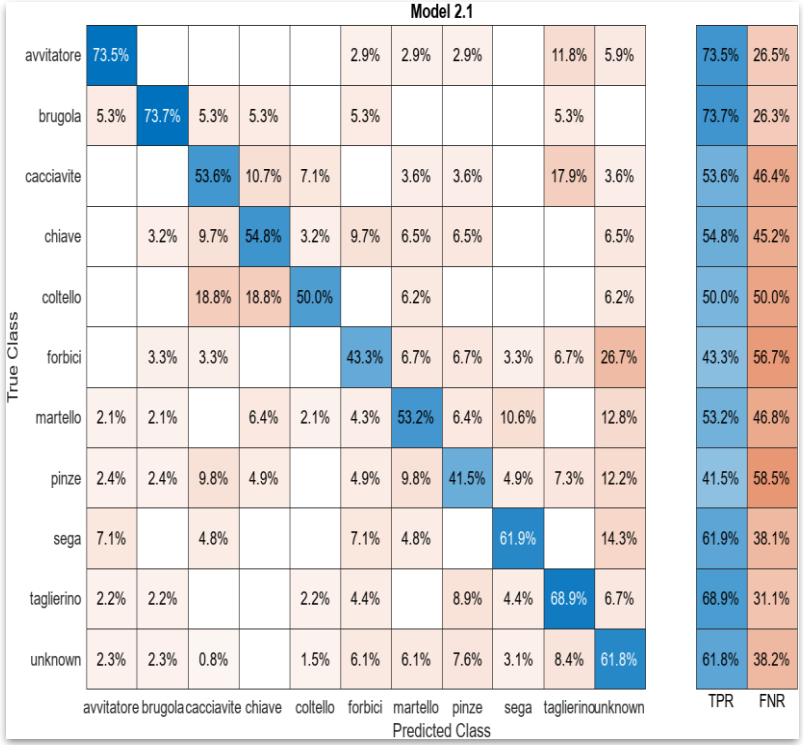
L’albero di decisione

Gli alberi di decisione (o **decision trees**) sono una rappresentazione dell’apprendimento del classificatore con **N variabili in input** e **M variabili in output**, che identificano una decisione.

Il processo di decisione è rappresentato da un **albero logico rovesciato** e consiste in una **serie di test**, dividendosi in due o più diramazioni, finché non si arriva alla decisione finale (nel nostro caso è la classe a cui appartiene l’oggetto).

Nell’immagine di destra, che rappresenta la **matrice di confusione** con **true positive** e **false negative** di questo specifico classificatore, si vedono i risultati della divisione della nostra popolazione iniziale nelle varie classi attraverso l’utilizzo delle features estratte.

Esempio: l’accuratezza del modello = 58.6%



Matlab

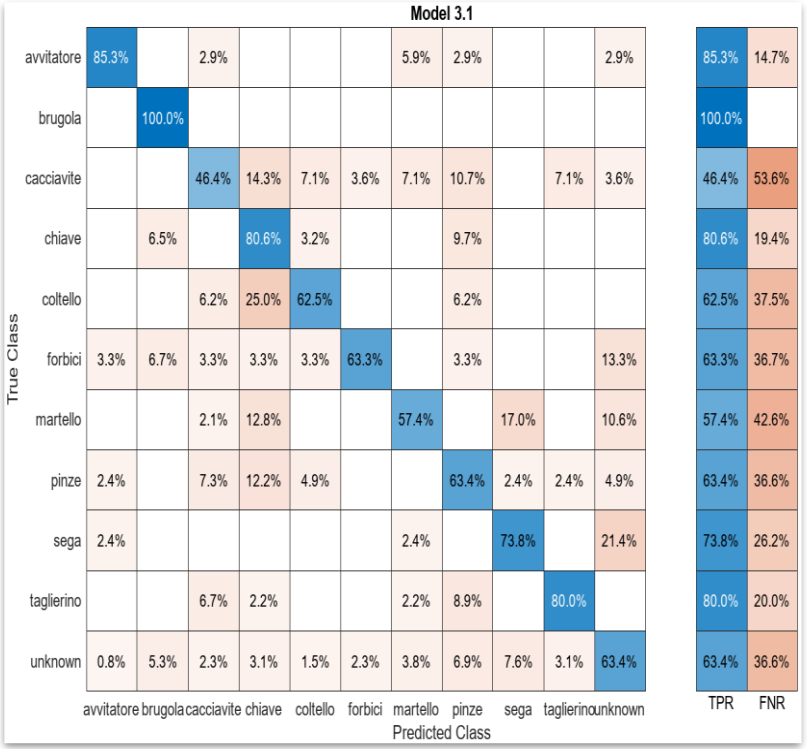
Il classificatore K-NN

Il classificatore K-NN (o **k-nearest neighbors**) è un classificatore di apprendimento supervisionato non parametrico, che **utilizza la prossimità** per effettuare le classificazioni, basandosi sulle **caratteristiche degli oggetti vicini a quello considerato**.

In pratica l'input è costituito da **esempi di addestramento più vicini** nella distribuzione spaziale, mentre l'output è **l'appartenenza ad una classe**.

Nell'immagine di destra, che rappresenta la **matrice di confusione** con **true positive** e **false negative** di questo specifico classificatore, **K-NN fine** (a 1 vicino), si può notare la **sensibilità ai valori anomali** del K-NN (es: i cacciaviti), il che significa che una sola istanza anomala può influenzare significativamente sulla previsione per gli esempi circostanti.

Esempio: l'accuratezza del modello = 68.5%



Matlab

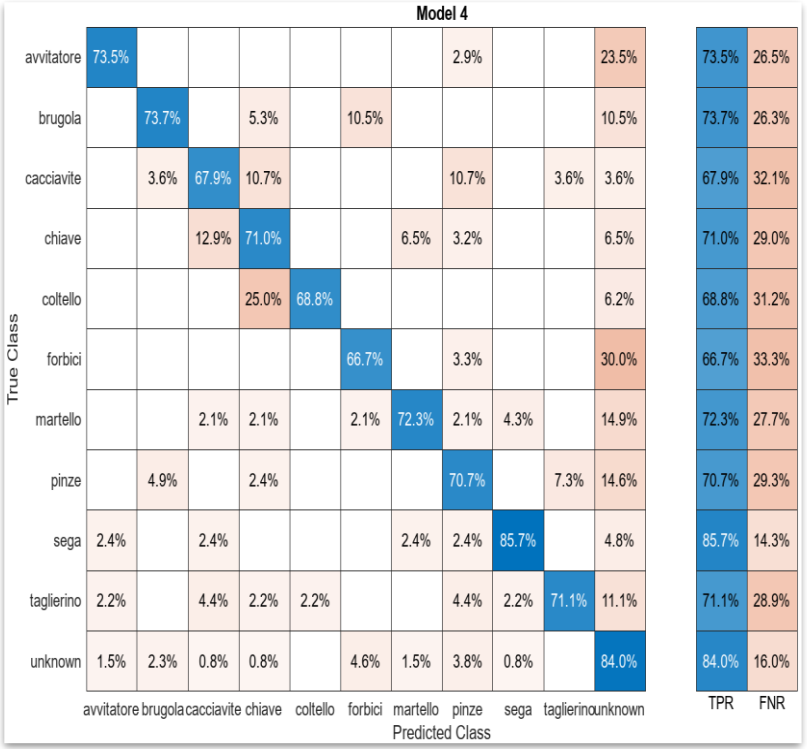
Ensemble bagged trees

Il classificatore **ensemble bagged trees** si basa sul concetto di addestramento, che richiama l'utilizzo di **diversi classificatori**, uniti in un certo modo per riuscire a **massimizzare le prestazioni**, usando i punti di forza di ognuno e limitando le debolezze dei singoli.

Ogni singolo classificatore conta allo stesso modo e abbia lo stesso peso, addentrandosi su una **porzione casuale di dati** in input tra cui alcuni possono comparire in più modelli, mentre altri non comparire mai (**la tecnica del campionamento casuale con rimpiazzo**). In questo modo viene risolto l'overfitting e vengono aumentate le capacità predittive.

Quindi nel nostro caso specifico il classificatore ensemble di tipo bagged tree utilizza un solo modello, **l'albero di decisione**, che però viene **addestrato su training set diversi**. Per questo motivo i risultati ottenuti sono migliori rispetto a quelli ottenuti con i classificatori utilizzati in precedenza.

Esempio: l'accuratezza del modello = 75.9%



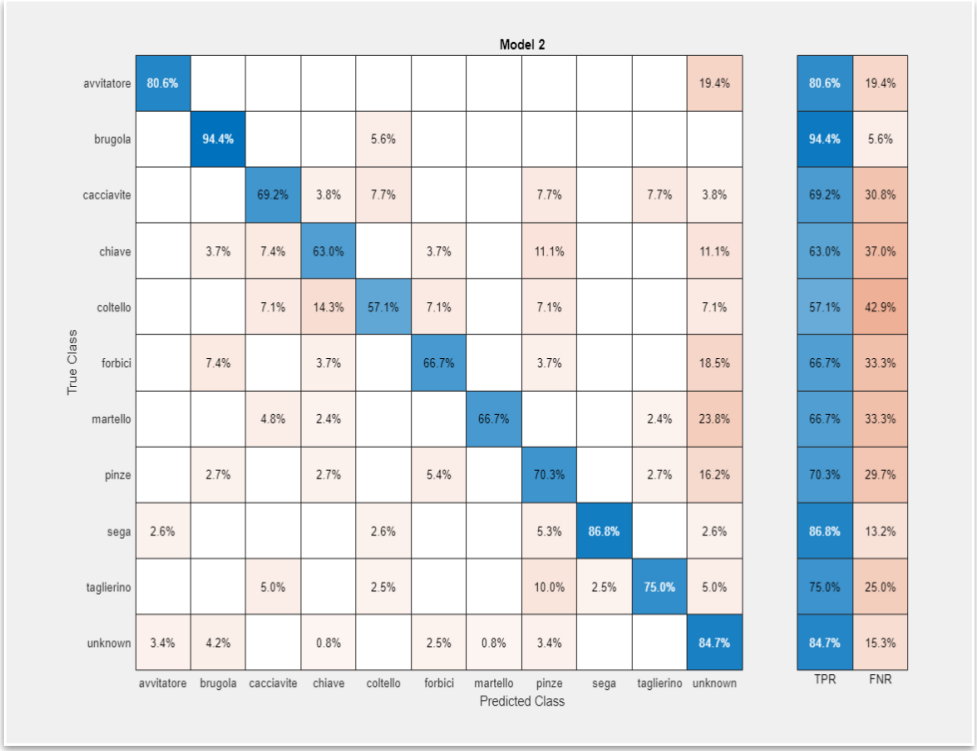
I risultati di Classification Learner

Le seguenti immagini ci mostrano i risultati finali che abbiamo ottenuto attraverso la classificazione del nostro dataset, usando Bagged Trees.

Nell'immagine di destra vediamo la **matrice di confusione** creata dalle classi di oggetti che abbiamo utilizzato con affianco una tabella che mostra **true positive** e **false negative**.

L'accuratezza della validation è **75-80%**.

Testando sul **10% del dataset**, tenuto a parte, **70-75%**.



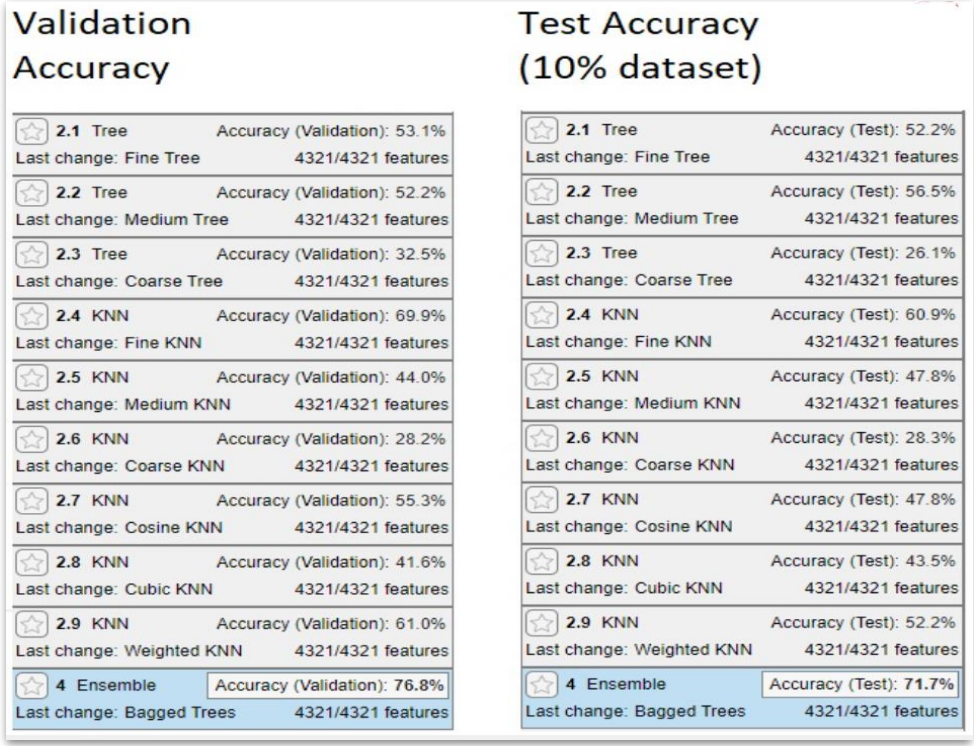
Matlab

I risultati di Classification Learner

Nell'immagine di destra vediamo la tabella che mostra **l'accuratezza dei classificatori** che abbiamo usato.

Gli **alberi di decisione**, sono **molto stabili** nei loro risultati, anche per gli oggetti non presenti nel dataset.

K-NN, seppur sembri promettente, in questo caso lo abbiamo trovato **meno stabile degli alberi**. Anche se sembra presentare risultati migliori dai test, controllando con gli oggetti esterni si è presentato scadente.



Matlab

La conclusione e discussione dei risultati

Ci troviamo infine con un progetto che abbiamo trovato **soddisfacente** e **molto interessante** da implementare.

Seppure abbiamo fatto degli errori, iniziando dalla creazione del dataset fino alle problematiche legate al codice, abbiamo **mitigato** quanto possibile e **imparato** dai nostri errori, simulando un **ambiente più reale** con dei risultati **molto promettenti**.

Nonostante tutto, siamo arrivati ad avere un'accuratezza di **oltre 70%**, sia in validation che testando su una parte separata del dataset. I risultati sono simili anche testando su immagini esterne, con elementi nuovi o in condizioni non presenti nel dataset.

Matlab

Progetto fatto da:

Nome/Cognome	Matricola	Dettagli	Contributo
Dan Litviniuc	869329	Formazione del dataset Segmentazione Feature Extraction Classificazione Creazione App Testing	45%
Oleksandra Golub	856706	Formazione del dataset Arricchimento del dataset Binarizzazione Testing	35%
Lorenzo Cino	866120	Formazione del dataset Classificazione Testing	20%

Matlab