Alma Mater Studiorum
Università di Bologna

Scuola di Ingegneria e Architettura

Corso di laurea magistrale in Ingegneria Informatica

SISTEMA DI VISIONE PER L'ISPEZIONE DELLA FRUTTA

Progetto in "Computer Vision and Image Processing M"

github.com/bobcorn/fruits-inspector

Docente:
Prof. Luigi Di Stefano

Studente:
Marco Rossini

Anno Accademico 2019/2020

Indice

Abstract

Il presente report illustra i ragionamenti e le scelte adottate per la realizzazione di un sistema di visione in grado di individuare i difetti e le imperfezioni presenti sulla superficie dei frutti, allo scopo di permetterne una corretta classificazione.

1. Prima parte

1.1. Segmentazione dell'immagine

I requisiti relativi alla prima parte richiedono che i frutti siano analizzati al fine di individuarne il **contorno** ed i **difetti** presenti sulla superficie. Il primo passo è stato pertanto costituito dal separare il frutto dallo sfondo, effettuando una **binarizzazione** dell'immagine in scala di grigi. Per effettuare la binarizzazione, si è provato ad utilizzare l'algoritmo di Otsu, al fine di garantire robustezza ad eventuali variazioni di illuminazione, ma l'utilizzo di questa strategia si è rivelato non soddisfacente, poiché la soglia automatica individuata dall'algoritmo si è rivelata generalmente più elevata del dovuto, comportando l'esclusione indesiderata di alcune porzioni del frutto. Al fine di evitare l'utilizzo di una soglia costante per tutte le immagini (individuata in un valore intorno a 30), per non rendere il sistema eccessivamente rigido, si è scelto di calcolare tale soglia adottando un **approccio dinamico**, sfruttando la seguente **relazione** tra i pixel di ogni immagine analizzata:

$$soglia = \frac{moda + k \cdot mediana}{2}$$

dove *moda* è il valore più frequente assunto dai pixel dell'immagine, *mediana* il valore mediano assunto dai pixel dell'immagine, e *k* un valore costante (configurabile a seconda dello scenario applicativo). L'utilizzo di questa formula (con $k = 0.5$) ha permesso di separare correttamente i frutti dallo sfondo, garantendo una maggiore flessibilità ad eventuali piccole variazioni di luminosità rispetto all'utilizzo di una soglia prefissata. Ottenuta l'immagine binarizzata, si è quindi effettuato un **labeling delle componenti connesse**, e si è ottenuta la maschera relativa al frutto selezionando la **componente connessa avente area maggiore**. La maschera del frutto ottenuta in questo modo, tuttavia, non è risultata ancora definitiva, poiché su di essa sono talvolta risultate presenti alcune **lacune**, ossia porzioni di pixel riconosciute erroneamente come sfondo. Per ottenere la maschera corretta, si è pertanto

scelto di eliminare tali lacune adottando un approccio **flood fill** (sfruttando l'apposita funzione messa a disposizione dalla libreria OpenCV), grazie al quale è stato possibile colmarle. Si è ottenuta in questo modo la maschera definitiva corrispondente alla superficie del frutto, della quale è stato possibile individuare il contorno (mediante l'apposita funzione OpenCV), soddisfando così il **primo requisito** relativo al **delineamento del contorno** del frutto.

Fig. 1.1a – Immagine originale.

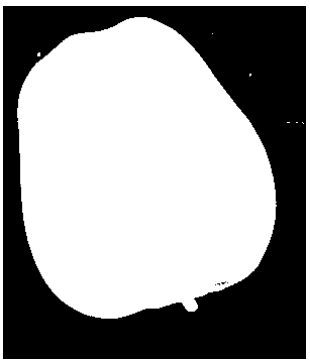

Fig. 1.1b – Immagine binarizzata.

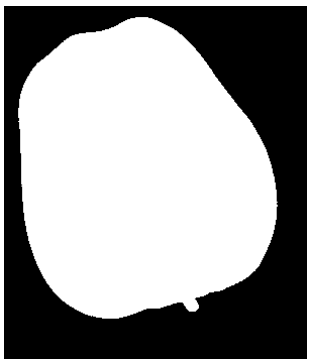

Fig. 1.1c – Isolamento della componente avente area maggiore (frutto) e suo riempimento.

Fig. 1.1d – Immagine originale con delineamento del contorno del frutto.

1.2. Individuazione dei difetti

Per soddisfare il **secondo requisito**, relativo all'individuazione dei **difetti** presenti della superficie, si è scelto di adottare un'approccio basato sul **riconoscimento dei contorni**, sfruttando le brusche variazioni d'intensità presenti in prossimità dei difetti stessi. Utilizzando la maschera del frutto, si sono selezionati i corrispondenti pixel dell'immagine in scala di grigi originale, ottenendo in questo modo il frutto in scala di grigi separato correttamente dallo sfondo. Si è quindi effettuato su di esso uno **smoothing** mediante un **filtro bilaterale**, che ha permesso di uniformare la superficie del frutto (eliminando il rumore), ma **preservando i contorni** degli eventuali difetti presenti su di essa. A questo punto, è stato possibile procedere con l'applicazione di un **edge detector di Canny**, che ha consentito di ottenere i contorni relativi ai difetti presenti sulla superficie. Tra questi contorni, tuttavia, è stato naturalmente rilevato anche il **contorno più esterno**, relativo al frutto stesso, che non risulta però di interesse. Per **escluderlo**, si è pertanto scelto di invertire la maschera del frutto rilevata inizialmente (ottenendo in questo modo la maschera relativa allo sfondo), di dilatarla, e quindi di sottrarla al risultato dell'edge detection. In questo modo, è stato possibile isolare i soli contorni dei difetti presenti sulla superficie del frutto. Per consolidare tali contorni, si è effettuata un'operazione di **chiusura** per un kernel grande, che ha consentito di renderli molto più evidenti. Ottenuti i difetti consolidati in questo modo, si è quindi effettuato un **labeling delle componenti connesse**, che ha permesso di iterare su ogni componente-difetto individuato al fine di **evidenziarne la posizione** sull'immagine originale a colori, soddisfando così anche il **secondo requisito**. Per garantire robustezza verso il rilevamento di eventuali falsi positivi nella fase di edge detection, si è scelto di imporre una **soglia minima per l'area** di ogni componente-difetto rilevato, al di sotto della quale esso è stato ignorato.

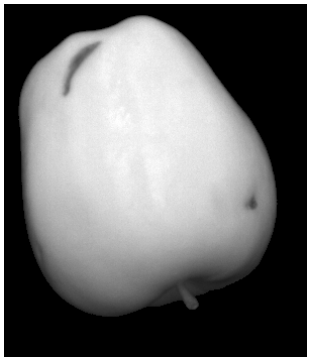

Fig. 1.2a – Frutto isolato in scala di grigi.

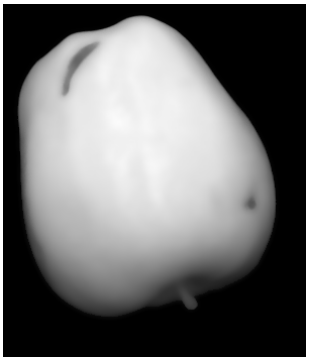

Fig. 1.2b – Frutto isolato in scala di grigi con smoothing mediante un filtro bilaterale.

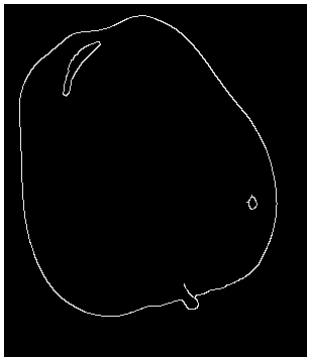

Fig. 1.2c – Contorni ottenuti a seguito dell'applicazione dell'edge detector di Canny.

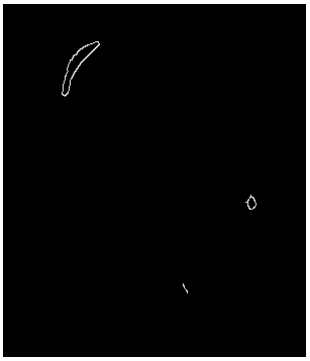

Fig. 1.2d – Contorni con esclusione del contorno esterno sfruttando la maschera dello sfondo.

Fig. 1.2e – Contorni consolidati mediante un'operazione di chiusura.

Fig. 1.2f – Immagine originale con filtraggio ed evidenziazione dei difetti individuati.

2. Seconda parte

2.1. Segmentazione dell'immagine

I requisiti relativi alla seconda parte richiedono che siano individuate, possibilmente senza falsi positivi, le **macchie** di colore marrone-rossastro presenti sui frutti, al fine di permetterne una corretta classificazione. Per soddisfare questo requisito, si è effettuato un primo **passo di binarizzazione analogo** a quello descritto nella prima parte (al fine di ottenere la maschera relativa al frutto), con l'unica differenza data dall'utilizzo di un **diverso valore del parametro** k nella formula per il calcolo della soglia di binarizzazione (si è scelto $k = 1$). Al termine di questa fase si è pertanto ottenuta, in maniera analoga alla prima parte, la maschera definitiva corrispondente alla superficie del frutto da analizzare.

2.2. Individuazione delle macchie

Per procedere all'individuazione delle macchie si è utilizzata, analogamente alla prima parte, la maschera del frutto per selezionare i corrispondenti pixel dell'immagine originale, ma scegliendo questa volta l'immagine a colori. La ricerca delle macchie è infatti effettuata **sulla base del colore**, poiché il contrasto presente sull'immagine in bianco e nero non è sufficiente per permettere un riconoscimento basato su tecniche di binarizzazione o di riconoscimento dei contorni. Ottenuto in questo modo il frutto separato correttamente dallo sfondo nello spazio di colore RGB, si è scelto di convertirlo nello **spazio di colore L*a*b***, che risulta maggiormente indicato per questo tipo di esigenza. Lo spazio L*a*b* infatti, diversamente dallo spazio RGB, codifica i colori separando l'informazione relativa alla loro **luminosità** (dimensione **L***) da quella relativa alla loro **tonalità** (dimensioni **a* e b***), facendo in modo che le variazioni dei valori numerici nella codifica di ogni colore rispecchino in maniera fedele le stesse variazioni visive percepite dall'occhio umano. Nel confronto tra colori, pertanto, la **nozione di distanza** tra i loro valori risulterà in questo spazio **maggiormente significativa** rispetto alla loro distanza nello spazio RGB. Convertita l'immagine in questo modo, si è quindi proceduto ad analizzare i colori dell'immagine tentando **due metodi alternativi**.

2.2.1. Primo metodo: quantizzazione dei colori

Il primo metodo esplorato ha assunto come ipotesi che non fossero disponibili ulteriori informazioni relative ai frutti in aggiunta alle immagini da analizzare. Si è pertanto cercato di individuare un approccio che risultasse completamente **autonomo ed automatico** nell'individuazione delle macchie, **senza** che risultasse necessario fornire al sistema **informazioni di supporto aggiuntive** relative alle macchie stesse (ad eccezione di una sola, molto semplice e generica, come si vedrà successivamente). Per questa strategia, si è scelto di considerare l'immagine di ogni frutto come fondamentalmente costituita da **tre colori dominanti**: il nero dello **sfondo**, il verde-giallo del **frutto** ed il marrone-rosso delle **macchie**. L'idea alla base di questa strategia prevede che, **quantizzando** in qualche modo i pixel di ogni immagine in queste tre

macro-categorie, e **discriminando** in qualche modo quale delle tre costituisca la categoria di interesse, risulti teoricamente possibile **isolare le macchie** semplicemente selezionando i pixel dell'immagine appartenenti alla corrispondente categoria. Per realizzare la quantizzazione, si è scelto di utilizzare l'algoritmo di **clustering K-means** (sfruttando l'apposita funzione messa a disposizione dalla libreria Scikit-learn) imponendo $k = 3$ (per ottenere, appunto, i tre colori dominanti cercati), e fornendo in ingresso all'algoritmo il valore di ogni pixel dell'immagine. Sfruttando la separazione tra l'informazione relativa alla luminosità e quella relativa alla tonalità di colore, messa a disposizione dallo spazio di colore L*a*b*, si però scelto di fornire in ingresso all'algoritmo non le triple corrispondenti ad ogni dimensione dello spazio di colore, ma le doppie relative alle **sole dimensioni a* e b***, ossia quelle relative alla tonalità, con l'obiettivo di rendere il clustering il più possibile indipendente dalle **variazioni di luminosità** eventualmente presenti nell'immagine, e di rendere inoltre la **nozione di distanza** utilizzata dall'algoritmo più significativa (dovendo elaborare uno spazio avente dimensionalità inferiore).[1] Effettuata questa operazione, è stato possibile ottenere la suddivisione nelle **tre categorie di colore**, codificate come la **coppia di informazioni** costituita dai tre **centroidi di ogni cluster** e dalla **maschera di labels** relative ad ogni cluster. Si è reso quindi necessario discriminare quale dei tre cluster corrispondesse al colore relativo alle macchie. Osservando la coppia di dimensioni a* e b* per ognuno dei tre risultati restituiti dall'algoritmo di clustering, è risultato semplice escludere preliminarmente quello relativo allo **sfondo nero**, poiché tali dimensioni assumevano sempre un valore molto prossimo a 128 (ossia, il valore associato al nero per le dimensioni a* e b* nella codifica OpenCV). Escluso facilmente lo sfondo in questo modo, non è risultato tuttavia altrettanto immediato effettuare la discriminazione sugli altri due cluster. Per realizzare tale distinzione, si è scelto di fornire in ingresso al sistema un **parametro costante** di riferimento, pari ad una tripletta RGB[2] di una **tonalità di marrone scuro**,

[1] la motivazione è legata alla cosiddetta "curse of dimensionality".

[2] si è scelto di utilizzare una tripletta RGB per una maggiore intuitività del parametro, ma sarebbe stato possibile inserire anche una tripletta nello spazio L*a*b* (o anche direttamente le sole componenti a* e b*).

ossia un colore molto simile a quello delle macchie (si è scelta la tripletta [71, 56, 27], ma il parametro è configurabile). La tripletta RGB è stata quindi convertita nella corrispondente tripletta L*a*b*, ed è stata calcolata la distanza (di Manhattan) tra le sue componenti a* e b* e ciascuno dei due centroidi dei cluster rimanenti da classificare. É stato quindi possibile discriminare il **cluster relativo alle macchie** come quello avente **distanza minima dalla tonalità di marrone scuro**. Individuato il cluster di interesse in questo modo, è stato possibile selezionare i pixel relativi dall'immagine originale sfruttando la maschera delle labels restituita dall'algoritmo di clustering. Tali componenti, corrispondenti alle macchie, sono state quindi colmate da eventuali **lacune** in esse presenti secondo un approccio **flood fill**, e sono state scartate eventuali componenti aventi area inferiore ad una certa soglia minima. Le **posizioni delle macchie** individuate sono state infine evidenziate sull'immagine originale, soddisfando così il **requisito** richiesto.

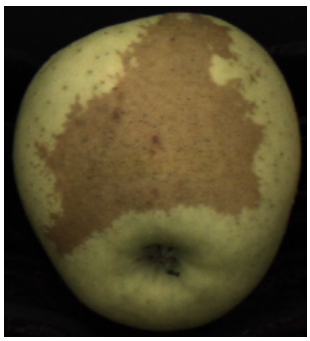

Fig. 2.2.1a – Immagine originale.

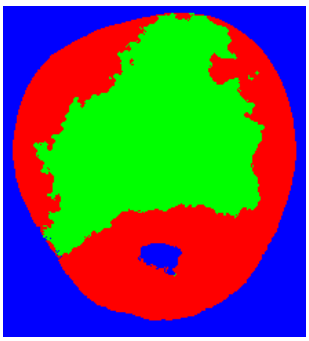

Fig. 2.2.1b – Risultato del clustering mediante l'algoritmo K-means.

Fig. 2.2.1c – Isolamento della macchia sfruttando la relativa maschera.

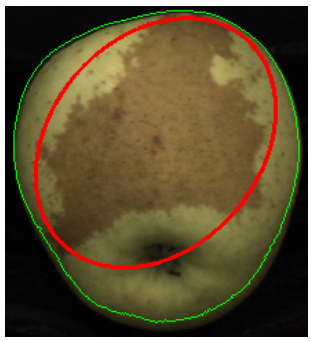

Fig. 2.2.1d – Immagine originale con filtraggio ed evidenziazione delle macchie individuate.

2.2.2. Secondo metodo: distanza di Mahalanobis

Il secondo metodo esplorato, alternativo al primo, ha assunto come ipotesi che fossero invece disponibili **informazioni aggiuntive** relative ai frutti in aggiunta alle immagini da analizzare, ossia le informazioni relative alle caratteristiche delle loro macchie. L'idea di fondo di questa strategia prevede che, estraendo preliminarmente alcuni **campioni delle macchie** dalle immagini da analizzare, e fornendole in ingresso al sistema come **esempi di training**, risulti teoricamente possibile configurare il sistema affinché possa calcolare la distanza (di Mahalanobis) tra gli esempi stessi ed i pixel dell'immagine, permettendo di determinare quali pixel corrispondano alle macchie filtrando in

base ad una **distanza massima**. Questo metodo presenta pertanto come svantaggio la "scomodità" di dover effettuare un'**operazione preliminare di training** del sistema, ma il vantaggio di garantire, almeno teoricamente, un **livello di attendibilità più fondato**. Per applicare questo metodo, si è pertanto provveduto ad estrarre dalle immagini disponibili un insieme di campioni relativi alle macchie (5 per ogni immagine), sotto forma di piccole immagini di dimensione variabile, avendo cura di selezionare porzioni appartenenti solo ed esclusivamente alle macchie. Si è quindi reso necessario consolidare le molteplici informazioni fornite dai diversi campioni in un'**informazione unica**, tale da tenere conto dei contributi di ogni singolo campione, e allo stesso tempo di esibire le informazioni necessarie al calcolo della distanza scelta per questo confronto, ossia la **distanza di Mahalanobis**. Si è quindi provveduto ad iterare su ogni immagine campione, convertendola nello spazio di colore L*a*b*, e calcolando, per ciascuna di esse, il **valore medio** assunto dalla coppia di dimensioni **a*** e **b***[3] relativa ad ogni pixel (necessario per il calcolo della tonalità di **colore di riferimento**) e la loro **matrice di covarianza** (necessaria per il **calcolo della distanza** di Mahalanobis). Si è quindi ottenuta la tonalità di colore di riferimento calcolando la **media dei valori medi** memorizzati per ogni immagine. Si sono quindi ottenute in questo modo tutte le informazioni necessarie per il calcolo della distanza di Mahalanobis. Iterando su ogni pixel dell'immagine da analizzare, e calcolando la distanza tra il valore assunto dalla relativa coppia di dimensioni a* e b* e la tonalità di colore di riferimento, si è quindi classificato come **pixel relativo alle macchie** ogni pixel avente **distanza inferiore ad una certa soglia** massima (specificata pari a 1.5). Unendo i pixel filtrati in questo modo, ed effettuando un **labeling delle componenti connesse**, si sono pertanto individuate le componenti corrispondenti alle macchie. Tali componenti, analogamente al primo metodo, sono state quindi colmate da eventuali **lacune** in esse presenti secondo un approccio **flood fill**, e sono state scartate eventuali componenti aventi area inferiore ad una certa soglia minima. Le posizioni delle macchie individuate in questo modo sono state infine evidenziate sull'immagine originale, soddisfando così il requisito richiesto.

[3] si è esclusa la dimensione L* per le stesse ragioni descritte nel metodo precedente.

Fig. 2.2.2a – Campioni estratti relativi alle macchie presenti nella prima immagine.

Fig. 2.2.2b – Campioni estratti relativi alle macchie presenti nella seconda immagine.

Fig. 2.2.2c – Colore medio di riferimento rilevato (utilizzato in coppia alla matrice di covarianza).

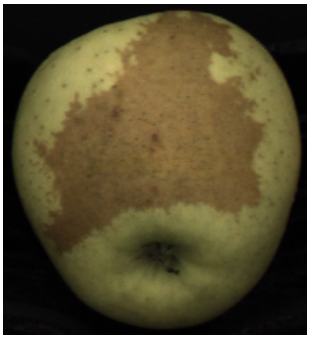

Fig. 2.2.2d – Immagine originale.

Fig. 2.2.2e – Isolamento della macchia sulla base della distanza di Mahalanobis dai campioni estratti.

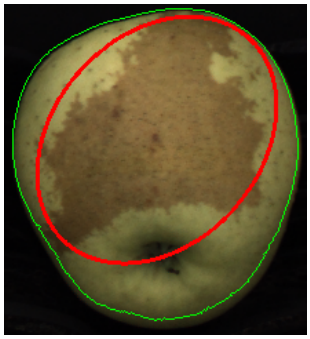

Fig. 2.2.2f – Immagine originale con filtraggio ed evidenziazione delle macchie individuate.

3. Terza parte (facoltativa)

3.1. Segmentazione dell'immagine

I requisiti relativi alla terza parte richiedono che le immagini fornite siano analizzate al fine di **separare correttamente** ogni frutto dallo sfondo (rimuovendo eventuali **oggetti distrattori**) e di individuare il **difetto** presente in una sola di esse. Per soddisfare questo requisito, si è effettuato nuovamente un primo **passo di binarizzazione analogo** a quello descritto per le parti precedenti (al fine di ottenere la maschera relativa al frutto), con alcune **piccole differenze**. La prima differenza è che si è scelto, in questo caso, di effettuare preliminarmente un'**equalizzazione dell'istogramma** dell'immagine in scala di grigi, la quale ha permesso di **migliorare il contrasto** favorendo ottimamente il processo di binarizzazione. La seconda differenza è stata la scelta di un **diverso valore del parametro** k nella formula per il calcolo della soglia di binarizzazione (si è scelto $k = 2.3$). Al termine di questa fase si è pertanto ottenuta una prima maschera relativa al primo piano, ma, diversamente dalla prima e dalla seconda parte, non è stato possibile considerare tale maschera come definitiva, poiché sono talvolta risultati compresi in essa anche alcuni **contatti con oggetti distrattori** presenti nell'immagine oltre al frutto (come, ad esempio, parti del nastro trasportatore o adesivi).

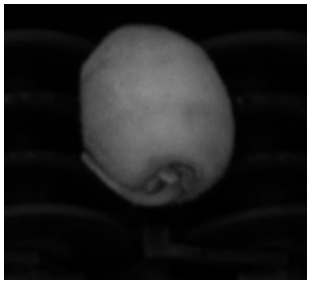

Fig. 3.1a – Immagine originale in scala di grigi.

Fig. 3.1b – Immagine binarizzata senza equalizzazione preliminare.

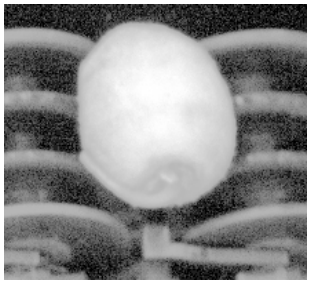

Fig. 3.1c – Immagine originale in scala di grigi equalizzata.

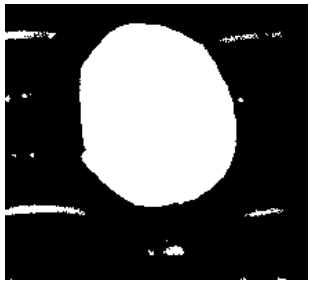

Fig. 3.1d – Immagine binarizzata con equalizzazione preliminare.

3.2. Gestione dei distrattori

Per ottenere la maschera definitiva, si è reso necessario **separare la componente relativa al solo frutto dalle componenti di disturbo**, "ritagliando" la macro-componente lungo alcuni segmenti. Per realizzare questa separazione, si è scelto di adottare un approccio basato sul calcolo dell'**inviluppo convesso** delle componente, e sull'analisi dei **difetti di convessità** (entrambi ottenuti mediante funzioni apposite, messe a disposizione dalla libreria OpenCV). Ottenute le zone di non convessità dall'inviluppo convesso, insieme ai relativi punti giacenti sul contorno della componente a distanza massima dall'inviluppo convesso (anch'essi restituiti dall'apposita funzione OpenCV) per ogni zona di non convessità, sarebbe idealmente possibile **congiungere con una linea** ciascuno di questi punti con il suo vicino più vicino, al fine di poter separare correttamente la componente connessa. In realtà, ciò non risulta praticabile, poiché i punti di difetto rilevati in questo modo risultano in numero maggiore rispetto a quelli necessari (ossia, quelli in prossimità di un reale contatto), a causa di alcune non convessità "spurie" presenti sul contorno della componente. Al fine di poter applicare con successo questa strategia, si è reso pertanto necessario effettuare un **filtraggio preliminare al tracciamento delle linee**, in modo tale da escludere i punti non corrispondenti a reali situazioni di contatto. Tra le possibili scelte di filtraggio percorribili, si è osservato come, analizzando una piccola finestra di pixel centrata su ogni punto di non convessità, per i punti "spuri" si rilevi una quantità di pixel bianchi e neri quasi equa, mentre per i punti di reale contatto si rilevi una quantità di pixel bianchi molto superiore rispetto a quelli neri. Ciò è intuitivamente dovuto al fatto che una finestra centrata sui punti spuri catturi per metà lo sfondo, e per metà il contributo di un oggetto, mentre una finestra centrata sui punti di reale contatto catturi una porzione di sfondo (minore), e un contributo (maggiore) non più dovuto ad un singolo oggetto, ma a due oggetti ravvicinati. In virtù di questa considerazione, si è applicato un filtraggio all'insieme totale dei punti di difetto rilevati, imponendo una **soglia minima sul rapporto tra pixel bianchi e neri** (si è scelto un valore pari a 1.35). Ottenuti i punti filtrati, si è reso necessario accoppiarli in funzione della loro distanza, affinché ogni punto venisse congiunto con il suo vicino più

vicino, ma evitando di ricongiungere punti già precedentemente congiunti. Per risolvere questo **problema di assegnamento**, si è scelto di utilizzare l'**algoritmo Ungherese** (o algoritmo di Kuhn–Munkres), assumendo come costo la distanza euclidea tra tutte le possibili coppie di punti, e minimizzando il costo totale. Le coppie assegnate in questo modo sono state quindi tutte congiunte tra di loro. Il tracciamento delle linee avvenuto secondo questa strategia, effettuato sull'immagine binarizzata, ha consentito di **separare con successo i frutti dai distrattori**, che risultavano prima identificati come connessi. Realizzata questa separazione, si è effettuato un **labeling delle componenti connesse**, e si è ottenuta la maschera relativa al frutto selezionando la componente avente area maggiore, della quale è stato possibile individuare il contorno (mediante l'apposita funzione OpenCV), soddisfando così il **primo requisito** relativo al **delineamento del contorno** del frutto.

Fig. 3.2a – Immagine originale.

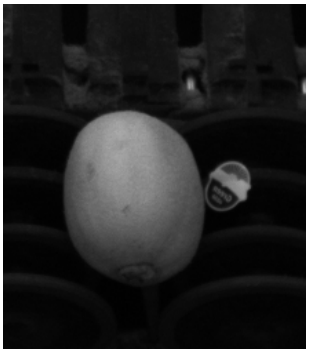

Fig. 3.2b – Immagine originale in scala di grigi.

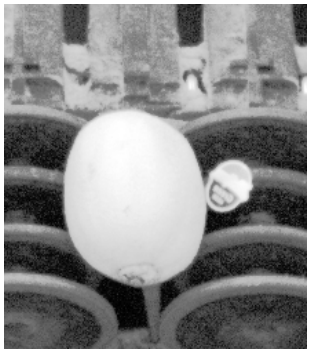

Fig. 3.2c – Immagine originale in scala di grigi equalizzata.

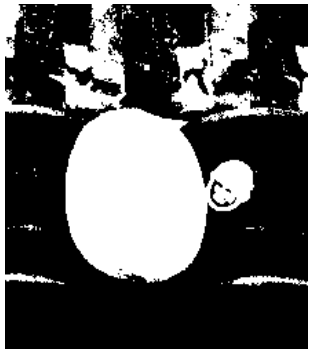

Fig. 3.2d – Immagine binarizzata con equalizzazione preliminare.

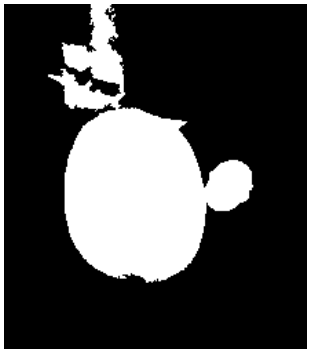

Fig. 3.2e – Isolamento della componente avente area maggiore e suo riempimento.

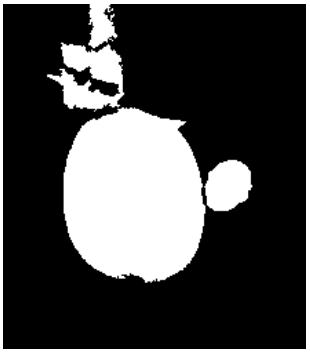

Fig. 3.2f – Separazione della componente-frutto dalle componenti relative ad oggetti di disturbo.

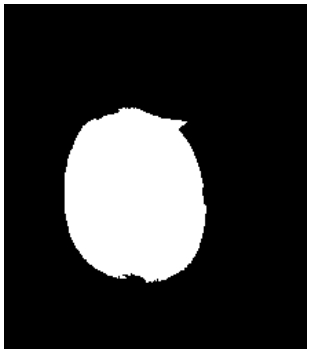

Fig. 3.2g – Isolamento della componente avente area maggiore (frutto).

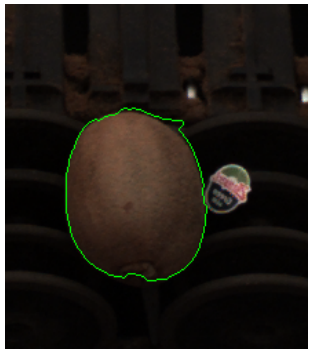

Fig. 3.2h – Immagine originale con delineamento del contorno del frutto.

3.3. Individuazione dei difetti

Per soddisfare il **secondo requisito**, relativo all'individuazione dei **difetti** presenti della superficie di uno dei frutti, si è scelto di adottare un'approccio basato sul **riconoscimento dei contorni**, analogo in tutto e per tutto a quello descritto nella prima parte, con l'unica differenza data da una scelta leggermente diversa dei parametri di configurazione dell'**edge detector di Canny**. Ottenuti i **difetti consolidati** in questo modo, si è quindi effettuato un **labeling delle componenti connesse**, che ha permesso di iterare su ogni componente-difetto individuato al fine di **evidenziarne la posizione** sull'immagine originale a colori, soddisfando così anche il **secondo requisito**. Per garantire robustezza verso il rilevamento di eventuali falsi positivi nella fase di edge detection, si è scelto di imporre una **soglia minima per l'area** di ogni componente-difetto rilevato, al di sotto della quale esso è stato ignorato.

Conclusioni

Complessivamente, i risultati ottenuti sulle immagini di test a disposizione sono risultati in linea con gli obiettivi. I requisiti relativi alla **prima parte** possono ritenersi soddisfatti con successo, e non sono emerse criticità rilevanti che dovrebbero far presupporre errori di analisi anche su immagini nuove. I requisiti relativi alla **seconda parte** sono invece risultati soddisfatti in maniera molto precisa sulla seconda immagine fornita, e in maniera meno precisa sulla prima. Per entrambi i metodi esplorati, infatti, le macchie presenti sulla superficie del frutto nella prima immagine sono state individuate includendo un numero molto elevato di **falsi positivi**, che ha causato il rilevamento di aree più estese di quelle reali. Questo effetto indesiderato è risultato molto simile per entrambi i metodi, nonostante la diversità delle strategie adottate. Per migliorare questo risultato si potrebbe valutare, nel primo metodo, l'utilizzo di un **algoritmo di clustering differente**, eventualmente basato su una **nozione di distanza più significativa** rispetto alla distanza euclidea adottata dall'algoritmo K-means (come, ad esempio, la distanza di Mahalanobis). Per il secondo metodo, si potrebbe invece valutare la selezione di un insieme di **campioni maggiormente rappresentativi**, eventualmente estratti da un numero molto elevato di esempi, al fine di migliorare l'affidabilità della funzione di calcolo della distanza di Mahalanobis.

Fig. 4.1a – Immagine originale.

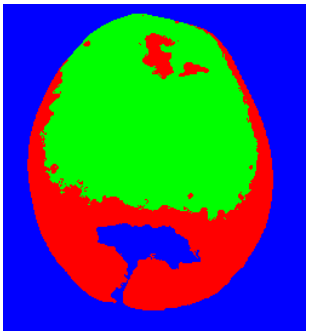

Fig. 4.1b – Risultato del clustering mediante l'algoritmo K-means.

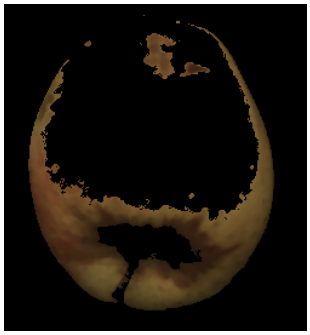

Fig. 4.1c – Isolamento della macchia sfruttando la relativa maschera (con presenza di falsi positivi).

Fig. 4.1d – Immagine originale con filtraggio ed evidenziazione delle macchie individuate (con sovrastima).

Fig. 4.1e – Immagine originale.

Fig. 4.1f – Colore medio di riferimento rilevato (utilizzato in coppia alla matrice di covarianza).

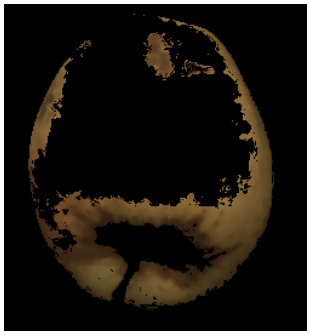

Fig. 4.1g – Isolamento della maschera sulla base della distanza di Mahalanobis dai campioni estratti (con presenza di falsi positivi).

Fig. 4.1h – Immagine originale con filtraggio ed evidenziazione delle macchie individuate (con sovrastima).

I requisiti relativi alla **terza parte**, infine, possono ritenersi soddisfatti in maniera coerente con gli obiettivi, ma, diversamente dalla prima parte, il contesto presuppone in questo caso il verificarsi di situazioni più imprevedibili, per cui il margine di errore potrebbe risultare più significativo. In particolare, relativamente alla separazione ed esclusione dei **distrattori**, le immagini di test a disposizione (poiché in numero molto limitato) non permettono di prevederne il livello di robustezza anche su immagini nuove. É tuttavia ammissibile che la strategia implementata possa incorrere nel rilevamento di **falsi positivi** che sfuggano al filtraggio, causando il collegamento di punti "spuri" che potrebbero separare in maniera erronea il frutto dagli oggetti di disturbo. Al fine di scongiurare tale eventualità, potrebbe rendersi necessario ripensare la strategia di filtraggio dei punti di difetto, sostituendola con una basata su caratteristiche differenti, oppure approssimando il contorno. Infine, l'aspetto legato alle **performance** non è stato preso in esame, in quanto non richiesto, pertanto il software presenta sicuramente ampi margini di ottimizzazione. Nonostante ciò, esso risulta comunque piuttosto **rapido nell'esecuzione** della **prima** e della **terza parte**, con tempi di elaborazione[4] medi per ogni immagine, rispettivamente, di 0.028 e 0.025 secondi, che consentono l'analisi, rispettivamente, di circa **36 e 40 immagini al secondo**. In virtù delle performance rilevate, assumendo di apportare possibili ottimizzazioni, risulta ragionevole contemplare un loro eventuale utilizzo anche

[4] le performance riportate sono state calcolate utilizzando un processore Intel Core i5 Dual-Core 2,7 GHz.

in scenari real-time. Si riscontrano invece **performance molto più limitate** nell'esecuzione della **seconda parte**, dove i tempi di elaborazione medi per ogni immagine, per il **primo** ed il **secondo metodo**, sono, rispettivamente, di circa 0.235 e 0.140 secondi, che consentono l'analisi, rispettivamente, di circa **0.4 e 0.7 immagini al secondo**. Tali inferiori valori di performance sono causati dall'**onerosità**, rispettivamente, dell'operazione di **clustering** e di calcolo della **distanza di Mahalanobis**, e non risulta pertanto possibile, in questo caso, contemplare un loro eventuale utilizzo anche in scenari real-time.