Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Tesi di Laurea

LIBRERIA PYTHON PER L'OCR DI DOCUMENTI D'IDENTITÀ

PYTHON LIBRARY TO PERFORM OCR ON IDENTITY DOCUMENTS

ALESSIO FALAI

Relatore: *Maria Cecilia Verri*

Anno Accademico 2018-2019

INDICE

ELENCO DELLE FIGURE

"If a machine is expected to be infallible, it cannot also be intelligent"
— Alan Turing

INTRODUZIONE

Questa tesi vuole essere un riassunto delle attività svolte in un tirocinio curriculare presso l'azienda QI-LAB di Firenze nel periodo *marzo - giugno 2019*. L'offerta di tirocinio prevedeva l'apporto di migliorie, in termini di accuratezza ed efficienza, riguardanti una libreria per il riconoscimento ottico dei caratteri di alcuni campi di documenti d'identità, attualmente solo italiani. La libreria, denominata QI-OCR, era inizialmente il risultato del lavoro compiuto da un precedente tirocinante, *Emilio Cecchini*, che si era occupato dell'implementazione del framework di base, sul quale ho avuto il piacere di lavorare io stesso. QI-OCR viene distribuita come pacchetto *pip* ed è stata scritta con linguaggio Python, in versione *3.6.8*, anche se compatibile con la versione minor successiva, ovvero la *3.7.x*, nonchè l'ultima versione stabile rilasciata, nella data in cui mi trovo a scrivere questo testo. QI-OCR prevedeva la suddivisione in 4 moduli principali:

1. **Preprocessing**: Si occupa della localizzazione del documento all'interno dell'immagine, del ritaglio dei contorni e del raddrizzamento, tramite l'algoritmo di template matching *SIFT*.

2. **Docfields**: Si occupa di effettuare un ritaglio statico, mediante coordinate predeterminate, dei campi d'interesse del documento già centrato.

3. **OCR**: Si occupa di effettuare una binarizzazione dei campi dell'immagine mediante un algoritmo di sogliatura globale *ad-hoc* e di fornire le immagini prodotte in input a un motore di OCR.

4. **Postprocessing**: Si occupa di applicare tecniche di analisi sintattica e semantica sui risultati ritornati dal software OCR.

Il lavoro da me svolto riguarda il miglioramento della libreria QI-OCR, mediante tecniche che verranno descritte successivamente. A tal proposito, la tesi è suddivisa in cinque capitoli. Il capitolo 1 presenta una descrizione dei prerequisiti matematici necessari a comprendere correttamente le spiegazioni dei metodi presenti nei capitoli successivi. Il capitolo 2 riguarda lo studio e l'implementazione di tecniche di binarizzazione e

segmentazione di immagini, ovvero soluzioni necessarie per la "pulizia" di una generica immagine contenente testo, che permettano di ottenere idealmente il solo testo di colore nero e tutto il resto di colore bianco. Il capitolo 3 riguarda invece lo studio di algoritmi di segmentazione basata dal riscontro sul modello, in cui vengono introdotti metodi alternativi al ben noto SIFT per l'individuazione della posizione di un documento all'interno di un'immagine. Il capitolo 4 introduce l'applicazione di tecniche avanzate di *text-detection* con lo scopo di colmare problemi derivanti da documenti che non rispettano alcun tipo di template prefissato, come ad esempio la vecchia carta d'identità cartacea italiana, che risulta comunque rilasciabile, in alternativa alla nuova carta d'identità elettronica italiana, in casi di estrema esigenza o per tutti i cittadini che non possono recarsi per motivi di handicap presso il municipio. Infine, il capitolo 5 descrive ulteriori migliorie e innovazioni introdotte, sia per quanto riguarda il lato sviluppo/distribuzione (containerizzazione, benchmarks, ...), sia per quanto riguarda l'esperienza utente (OCR parametrizzabile sui campi del documento, implementazione di diversi motori di OCR, ...).

1

ALCUNI PREREQUISITI MATEMATICI NELL'AMBITO DELLA COMPUTER VISION

1.1 IMMAGINI DIGITALI

Un'immagine digitale I con risoluzione $m \times n$ è una matrice di valori interi di dimensione $n \times m$, che può essere matematicamente interpretata come una funzione semplice (nè iniettiva nè suriettiva) $I\colon \mathbb{N} \times \mathbb{N} \supseteq X \to \mathbb{P}^c$, che si occupa di mappare una coppia ordinata $(u, v) \in \mathbb{N} \times \mathbb{N}$, con $u, v \in \mathbb{N}$, in una *c-upla* $p = (p_1, p_2, \ldots, p_c)$, in cui ciascun valore p_i di p rappresenta il valore del pixel associato nel canale i, con $0 \leqslant p_i \leqslant 2^b - 1$, dove b è il numero fissato di bit utilizzati per rappresentare ciascun pixel e 2^b indica il massimo numero di colori o di livelli di grigio utilizzabili. Per esempio, ipotizzando di utilizzare canali colore standard, nel caso in cui si lavori con immagini in scala di grigi si avrà $c = 1$, mentre nel caso in cui si lavori con immagini *RGB* si avrà $c = 3$ ($c = 4$ per immagini *RGBA* o *CMYK*). Inoltre, supponendo di utilizzare una profondità pari a $b = 8$ sarà possibile specificare interi compresi tra 0 e 255 per i valori di ciascun pixel. Infine, indichiamo la risoluzione dell'immagine come la coppia $(m, n) \in \mathbb{N} \times \mathbb{N}\colon (n, m) = \max_{(x,y)\in X}(x, y)$.

1.2 CONVOLUZIONE

La *convoluzione* è un'operazione alla base dell'*image processing*, grazie alla quale è possibile andare ad analizzare ed eventualmente accentuare o alleviare diversi aspetti di una data immagine, come la sfocatura, la nitidezza, i contorni e molto altro. L'operazione, che viene descritta nella forma discreta, prende dunque in input due immagini digitali, ovvero l'immagine originale I (normalmente un singolo canale) e l'immagine K, denominata *kernel*, e intuitivamente si occupa di far scorrere la matrice K sull'immagine I, generalmente a partire dal bordo in alto a sinistra,

effettuando una somma pesata dei valori di I dati dalle proiezioni delle posizioni correntemente analizzate dalla matrice K, in cui i pesi sono dati proprio dagli elementi di K. Solitamente la dimensione della matrice K è molto minore della dimensione dell'immagine originale e spesso K è una matrice quadrata $n \times n$, con n dispari. Più formalmente, l'operazione di *convoluzione* $I * K$ in un punto (i, j) è data da:

$$\begin{aligned} I^*(i,j) &= \sum_{x=-n}^{n} \sum_{y=-n}^{n} (I(i-x, j-y) \cdot K(x,y)) = \\ &= \sum_{x=1}^{n} \sum_{y=1}^{n} (I(i + \left\lceil \frac{n}{2} \right\rceil - x, j + \left\lceil \frac{n}{2} \right\rceil - y) \cdot K(x,y)) \end{aligned} \tag{1.1}$$

Nel caso in cui nella formula 1.1 i segni - e + venissero invertiti si otterrebbe la formulazione della *correlazione* $I \otimes K$ in un punto (i, j):

$$\begin{aligned} I^{\otimes}(i,j) &= \sum_{x=-n}^{n} \sum_{y=-n}^{n} (I(i+x, j+y) \cdot K(x,y)) = \\ &= \sum_{x=1}^{n} \sum_{y=1}^{n} (I(i - \left\lceil \frac{n}{2} \right\rceil + x, j - \left\lceil \frac{n}{2} \right\rceil + y) \cdot K(x,y)) \end{aligned} \tag{1.2}$$

Convoluzione e *correlazione* possono risultare equivalenti se per passare da un'operazione all'altra si effettua un ribaltamento orizzontale e verticale del filtro (graficamente K viene ruotata di 180°). Dunque le due operazioni risultano identiche nel caso in cui la matrice K sia simmetrica rispetto ai due assi. In particolare, si preferisce utilizzare la convoluzione quando sono necessarie le proprietà commutativa, associativa e distributiva. Inoltre, dato che spesso il risultato delle operazioni descritte viene utilizzato come valore di intensità di pixel, tale risultato viene normalizzato, dividendolo per la somma dei pesi del filtro. La complessità computazionale delle operazioni, data ad esempio un'immagine di dimensione $m \times m$ e un kernel di dimensione $n \times n$, è piuttosto elevata, dato che richiede un numero di moltiplicazioni pari a $n^2 \cdot m^2$ e altrettante somme. Di seguito un esempio di convoluzione e correlazione:

$$I = \begin{array}{c} \\ i \end{array} \begin{array}{c} \begin{array}{ccccc} & & j & & \\ & & \cdots & & \\ & 30 & 28 & 32 & \\ \vdots & 27 & 26 & 10 & \vdots \\ & 29 & 22 & 18 & \\ & & \cdots & & \end{array} \end{array}, K = \begin{pmatrix} 4 & -2 & 1 \\ -1 & 5 & -3 \\ -6 & 0 & 4 \end{pmatrix}$$

Calcoliamo $I * K$ nella posizione (i, j) come

$$I^*(i,j) = 18 \cdot 4 - 22 \cdot 2 + 29 \cdot 1 - 10 \cdot 1 + 26 \cdot 5 - 27 \cdot 3 - 32 \cdot 6 + 28 \cdot 0 + 30 \cdot 4 = 24,$$

mentre calcoliamo $I \otimes K$ nella posizione (i, j) come

$$I^{\otimes}(i,j) = 30 \cdot 4 - 28 \cdot 2 + 32 \cdot 1 - 27 \cdot 1 + 26 \cdot 5 - 10 \cdot 3 - 29 \cdot 6 + 22 \cdot 0 + 18 \cdot 4 = 67$$

Un problema che può risultare evidente riguarda il modo di trattare i bordi, nel caso in cui l'intorno di un pixel non sia disponibile. Per ovviare a tale problema esistono diverse tecniche valide, come ipotizzare che i pixel non disponibili abbiano intensità zero, oppure prolungare i pixel di bordo supponendo intensità costante in quelli non disponibili.
Alcuni esempi di filtri classici:

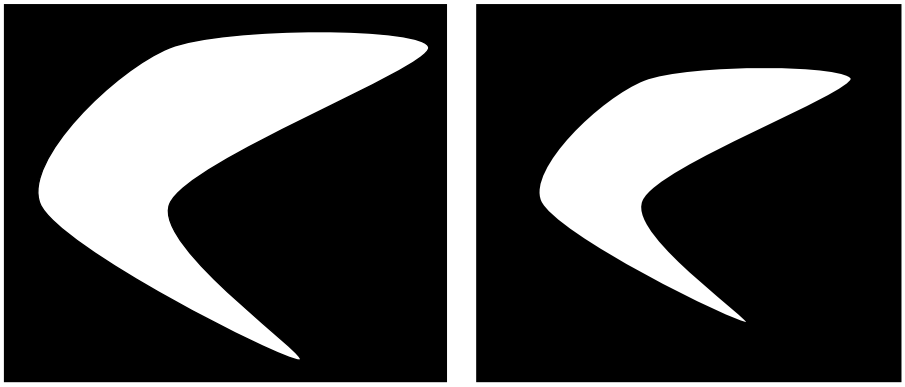
- Sfocatura (filtro di *Gauss*): $\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$
- Nitidezza: $\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$
- Contorni (filtro di *Laplace*): $\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$

1.3 ELEMENTI DI MORFOLOGIA

1.3.1 *Basi della matematica morfologica*

La matematica morfologica è una collezione di operatori, basata sulla teoria degli insiemi e definita su una struttura astratta[1]. Il suo obiettivo è quello di analizzare le forme e gli oggetti di un'immagine digitale, attraverso trasformazioni quali erosione, dilatazione, apertura, chiusura e altre. Nella loro forma originaria le operazioni morfologiche prevedono l'utilizzo di un'*immagine binaria* a singolo canale, composta da pixel che possono assumere solo due valori, 1 e 0, bianco e nero, *foreground*

1 La struttura astratta alla quale si fa riferimento è un *reticolo* infinito, un'estensione della teoria degli insiemi di Minkowski.



(a) Prima (dopo) (b) Dopo (prima)

Figura 1: Esempio di erosione (dilatazione) morfologica con elemento strutturante rettangolare

e *background*, oltre che di un *elemento strutturante* (*SE*), anch'esso sotto forma di immagine binaria a singolo canale, generalmente di dimensione molto minore di quella dell'immagine originale. Per correttezza, da ora in poi, in questa sezione, considereremo un'immagine binaria I anche come un insieme di punti Q_I contenente tutte le coppie (u, v) di pixel *foreground* di I, ovvero tali che $I(u, v) = 1$ e considereremo i termini in **grassetto** come coppie ordinate $(u, v) \in \mathbb{N} \times \mathbb{N}$.

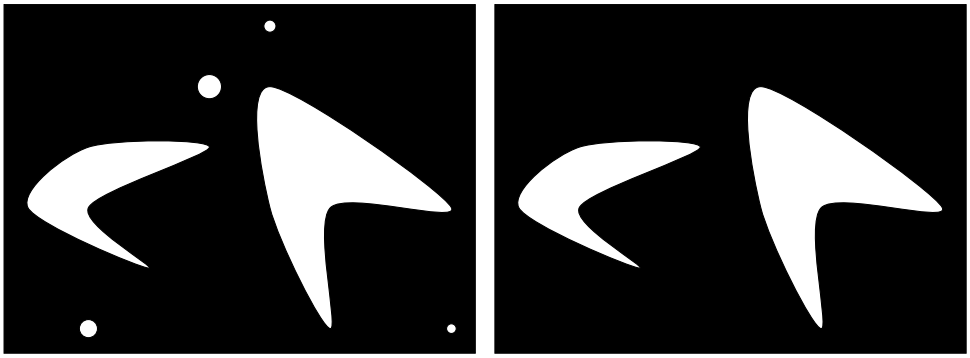
Di seguito una descrizione degli operatori morfologici di base:
Il risultato dell'operazione di *erosione* di un'immagine mostra dove lo SE viene contenuto negli oggetti dell'immagine, ovvero, intuitivamente permette di rimuovere pixel dal contorno di *componenti connesse*[2] di colore bianco, in quantità dipendente dalla grandezza dello SE. In formule:

$$[\epsilon_{Q_{SE}}(I)](\mathbf{x}) = \bigwedge_{\mathbf{s} \in Q_{SE}} (I(\mathbf{x} + \mathbf{s})) \tag{1.3}$$

Il risultato dell'operazione di *dilatazione* di un'immagine mostra dove lo SE tocca gli oggetti dell'immagine, ovvero, intuitivamente permette di aggiungere pixel al contorno di componenti connesse di colore bianco, in quantità dipendente dalla grandezza dello SE. In formule:

$$[\delta_{Q_{SE}}(I)](\mathbf{x}) = \bigvee_{\mathbf{s} \in Q_{SE}} (I(\mathbf{x} + \mathbf{s})) \tag{1.4}$$

2 Una *componente connessa C* di un insieme di punti *foreground* P di un'immagine binaria I è un insieme $C \subseteq P$ tale che $\forall \mathbf{p}_1, \mathbf{p}_n \in C$ esiste un percorso $\mathbf{p}_1, \ldots, \mathbf{p}_n$ in cui ogni $\mathbf{p}_i$ è *k-vicino* (*4-vicino* o *8-vicino*) a $\mathbf{p}_{i-1}$ e $\neg\exists \mathbf{q} \notin P$ *k-adiacente* a un punto di P. [7]



(a) Prima (b) Dopo

Figura 2: Esempio di apertura morfologica con elemento strutturante circolare

Le operazioni di *apertura* (*chiusura*) sono invece la combinazione in sequenza di erosione (dilatazione) e dilatazione (erosione) dello SE con la propria *riflessione*[3]. Queste due operazioni vengono molto utilizzate per rimuovere "sporco" (*noise*) dalle immagini e per "chiudere" piccoli "buchi" all'interno di oggetti che si trovano nel *foreground*. In formule:

$$\gamma_{Q_{SE}}(I) = \delta_{\check{Q}_{SE}}(\epsilon_{Q_{SE}}(I)), \tag{1.5}$$

$$\phi_{Q_{SE}}(I) = \epsilon_{\check{Q}_{SE}}(\delta_{Q_{SE}}(I)) \tag{1.6}$$

1.3.2 *Ricostruzione morfologica*

La *ricostruzione* è un'operazione molto utile nella matematica morfologica, che viene spesso presentata come un insieme di operatori *geodetici*, che consentono di estrarre le componenti connesse dell'immagine originale, marcate da un'altra immagine *marker*. Dunque, gli operatori di *erosione* e *dilatazione* possono essere ridefiniti in senso *geodetico*. In particolare, consideriamo al solito un'immagine binaria *I* (*mask*), definita dal suo insieme di punti *foreground* Q_I, un elemento strutturante (SE), che definisce la connettività da rispettare, e un'immagine *marker F* tale che $Q_F \subseteq Q_I$, che permette di definire quali componenti connesse devono subire i ri-

3 La *riflessione* di un insieme di punti P di un'immagine binaria I rispetto all'origine viene definita come $\check{P} = \{(-u, -v) : (u, v) \in P\}$.

sultati delle operazioni morfologiche. A questo punto, possiamo definire l'*erosione condizionale* come:

$$[\epsilon_{Q_{SE}}(F)]_I^{(0)}(\mathbf{x}) = [\epsilon_{Q_{SE}}(F)](\mathbf{x}) \vee I(\mathbf{x}) \tag{1.7}$$

e la *dilatazione condizionale* come:

$$[\delta_{Q_{SE}}(F)]_I^{(0)}(x) = [\delta_{Q_{SE}}(F)](\mathbf{x}) \wedge I(\mathbf{x}) \tag{1.8}$$

Le varianti *geodetiche* vengono definite in modo iterativo a partire da quelle *condizionali*. Dunque, si avrà che l'*erosione geodetica* può essere espressa come:

$$[\epsilon_{Q_{SE}}(F)]_I^{(i)}(\mathbf{x}) = [\epsilon_{Q_{SE}}([\epsilon_{Q_{SE}}(F)]_I^{(i-1)}(\mathbf{x}))]_I^{(0)}(\mathbf{x}), i = 1, 2, 3, \ldots, \tag{1.9}$$

mentre la *dilatazione geodetica*:

$$[\delta_{Q_{SE}}(F)]_I^{(i)}(\mathbf{x}) = [\delta_{Q_{SE}}([\delta_{Q_{SE}}(F)]_I^{(i-1)}(\mathbf{x}))]_I^{(0)}(\mathbf{x}), i = 1, 2, 3, \ldots \tag{1.10}$$

Concettualmente, le iterazioni sopra definite potrebbero continuare indefinitamente, ma per ragioni pratiche si sceglie di farle terminare per un intero n tale che nessun cambiamento avvenga $\forall n' > n$. L'output stabile $R_I^{\epsilon}(Q_F)) = [\epsilon_{Q_{SE}}(F)]_I^{(n)}(\mathbf{x})$ e $R_I^{\delta}(F)) = [\delta_{Q_{SE}}(F)]_I^{(n)}(\mathbf{x})$ così definito viene denominato rispettivamente *erosione* e *dilatazione* con *ricostruzione*.

Definiamo adesso le operazioni di *apertura* e *chiusura* con *ricostruzione*: L'operazione di apertura permette di rimuovere oggetti in base allo SE usato, ma ha il limite di non preservare le altre strutture, che vengono comunque modificate. Questo limite viene completamente superato dall'operatore di *apertura con ricostruzione*, che opera nel seguente modo: l'immagine viene erosa con lo SE come per l'apertura normale, ma invece di far seguire una dilatazione, si opera una ricostruzione usando l'immagine originale come *mask*, e l'immagine erosa come *marker*. L'operazione di *chiusura con ricostruzione* opera in modo analogo. In formule:

$$\tilde{\gamma}_R(I) = \gamma_R^{(n)}(I) = R_I^{\delta}[\epsilon^{(n)}(I)], \tag{1.11}$$

$$\tilde{\phi}_R(I) = \phi_R^{(n)}(I) = R_I^{\epsilon}[\delta^{(n)}(I)] \tag{1.12}$$

1.3.3 *Altri operatori*

Gli operatori fin'ora descritti permettono di estrarre caratteristiche importanti all'interno delle immagini, ma, ad esempio, consentono difficilemente di effettuare un'equalizzazione di parametri di sfondo, oppure di estrarre piccoli particolari d'interesse, oppure ancora di rilevare i contorni di un oggetto. Per ovviare a queste problematiche, si può ricorrere all'utilizzo di trasformazioni più specifiche, derivate da quelle di base. In particolare, possiamo definire l'operatore *top-hat*, nelle due versioni *bianca* e *nera*, rispettivamente:

$$g^{w}_{Q_{SE}}(I) = I - \gamma_{Q_{SE}}(I), \tag{1.13}$$

$$g^{b}_{Q_{SE}}(I) = \phi_{Q_{SE}}(I) - I \tag{1.14}$$

Entrambi ritornano un'immagine contenente gli elementi dell'immagine originale che sono più piccoli dello SE. Il primo ritorna inoltre solo gli elementi più *luminosi* degli oggetti a loro vicini, mentre il secondo solo gli elementi più *scuri* degli oggetti a loro vicini. Questo ci fa capire che un buon utilizzo per questo operatore riguarda la correzione di condizioni di luce non uniforme all'interno di un'immagine, per consentire una separazione tra *foreground* e *background* più marcata, grazie al conseguente incremento di contrasto.

Altri operatori morfologici sono quello *hit-or-miss*, che consente di estrarre pixel i cui vicini hanno una specifica configurazione (*pattern*), in base allo SE utilizzato; oppure quello del *gradiente*, che consiste nella differenza tra dilatazione e erosione di un'immagine, e restituisce, ad esempio, i contorni di oggetti.

2

BINARIZZAZIONE DI IMMAGINI

2.1 INTRODUZIONE

Nel campo dell'*image processing* ricopre un ruolo fondamentale la possibilità di distinguere diversi oggetti, forme e contorni presenti nell'immagine in analisi. Per far ciò è possibile ricorrere a svariate metodologie, che sono strettamente dipendenti dal contesto in cui si intende operare. Per esempio, nel caso del *riconoscimento ottico dei caratteri* (OCR), le operazioni che andranno descritte assumono importanza assoluta, dato che questo tipo di applicazione, per operare al meglio, ha spesso bisogno di ricevere in input un'immagine in bianco e nero, in cui lo sfondo è di colore bianco e il testo è di colore nero. Gli argomenti affrontati, nel caso di un'applicazione OCR, saranno soprattutto utili nel caso in cui il motore utilizzato sia *open-source* (vedi *Tesseract*), ovvero nel caso in cui un buon *pre-processing* dell'immagine possa fare la differenza. In questo articolo andremo quindi ad analizzare gli *algoritmi* più ricorrenti nell'ambito del *thresholding* di immagini e nella sezione 2.5 affronteremo un approccio specificamente studiato per i casi d'uso della libreria QI-OCR.

Da qui in avanti ipotizzeremo di lavorare con immagini originali in *scala di grigi* con profondità 8 bit, anche se i metodi descritti possono facilmente essere generalizzati per immagini con più di un canale colore.

2.2 SOGLIATURA GLOBALE

È il più semplice metodo di *thresholding*, che prevede la scelta di un determinato valore di soglia, compreso tra 0 e 255. L'algoritmo di sogliatura prevede quindi la scansione, pixel per pixel, dell'immagine e, nel caso in cui il valore del pixel esaminato sia minore del valore di soglia, tale pixel assumerà valore zero, che corrisponde al colore nero. Altrimenti, nel caso in cui il valore del pixel esaminato sia maggiore o uguale del valore di

soglia, tale pixel assumerà valore 255, che corrisponde al colore bianco. Risulta facile intuire quale possa essere il problema principale di questo tipo di algoritmo di sogliatura, ovvero la scelta del valore di soglia. Nel caso in cui il contesto in cui si opera sia relativamente statico, la *sogliatura globale* risulta comunque l'opzione più consigliata, per la sua facilità di implementazione. In particolare, il metodo descritto risulta adatto se le immagini in analisi presentano all'incirca caratteristiche uniformi in termini di illuminazione e contrasto. Se così non fosse, sarebbe necessario valutare l'implementazione di algoritmi leggermente più complessi.

Algoritmo 1 Sogliatura globale

```
1:  function GLOBAL-THRESHOLDING
2:      image ← input image
3:      thresh ← threshold value
4:      for i in ROWS(image) do
5:          for j in COLUMNS(image) do
6:              if image[i][j] ⩾ thresh then
7:                  image[i][j] ← 255
8:              else
9:                  image[i][j] ← 0
10:     return image
```

2.3 SOGLIATURA ADATTATIVA (O LOCALE)

Questo metodo di *thresholding* consente di sopperire alle mancanze della sogliatura globale, nel caso in cui le immagini in analisi presentino illuminazione e/o contrasto non uniforme. In particolare, questa tecnica dinamica computa automaticamente differenti valori di soglia per diverse aree dell'immagine. Dunque, l'immagine viene suddivisa in tante *sotto-immagini*, abbastanza piccole da poter ipotizzare che in ciascuna illuminazione e contrasto siano sufficientemente uniformi. Una volta partizionata l'immagine, un valore di soglia viene calcolato per ciascuna *finestra*. Il calcolo di ogni valore di soglia dipende dall'implementazione specifica dell'algoritmo, ma genericamente si ricorre all'utilizzo, per ogni sotto-immagine, di semplici operatori statistici, come la media, la mediana o la media fra massimo e minimo.

Alcune tecniche di sogliatura locale efficaci sono implementate da algoritmi quali quello di *Niblack*[9] e *Sauvola*[10]. In particolare, l'algoritmo di *Niblack* prevede l'utilizzo delle metriche di media e deviazione standard,

per una specifica finestra centrata su ciascun pixel dell'immagine. La formula utilizzata per calcolare un generico valore di soglia è data da:

$$T(x, y) = m(x, y) + k \cdot s(x, y), \tag{2.1}$$

dove $T(x, y)$ indica il valore di soglia per il pixel in posizione (x, y), $m(x, y)$ ($s(x, y)$) indica la media (deviazione standard) dei valori dei pixel vicini a quello centrato sulla finestra corrente e k è un coefficiente, che può essere determinato empiricamente[1].

L'utilizzo di parametri configurabili è molto comune per algoritmi di questo tipo, in quanto questi risultano essere fortemente dipendenti dal contesto di utilizzo. Un esempio pratico riguarda la libreria *OpenCV*, che nella funzione ADAPTIVETHRESHOLD consente di selezionare un valore c, che viene sottratto da ciascun valore di soglia di ogni sotto-immagine. Per esempio, nel caso in cui il valore di soglia venga calcolato utilizzando la media $mean$, il limite scelto non sarà esattamente $mean$, ma $mean - c$.

Purtroppo, il problema principale della sogliatura adattativa è che tende a valorizzare tutti gli elementi presenti nell'immagine, non consentendo di differenziare al meglio le componenti d'interesse da quelle che invece si vorrebbero scartare.

Algoritmo 2 Sogliatura adattativa

```
1: function LOCAL-THRESHOLDING
2:     image ← input image
3:     for pixel in image do
4:         thresh ← STATISTICAL-OPERATOR(NEIGHBORS(PIXEL))
5:         if pixel ⩾ thresh then
6:             pixel ← 255
7:         else
8:             pixel ← 0
9:     return image
```

1 *Niblack* consiglia l'impostazione del parametro k al valore -0.2.

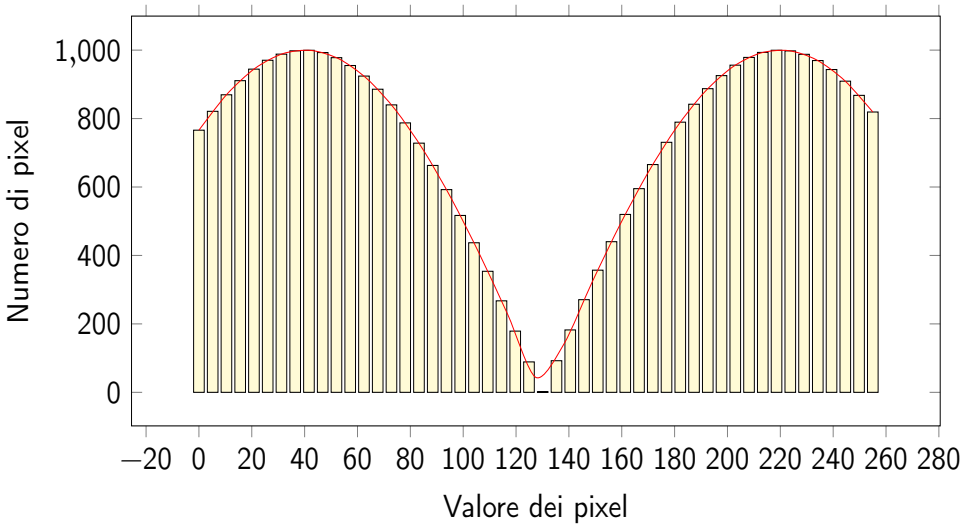


Figura 3: Esempio di immagine con istogramma bimodale

2.4 SOGLIATURA DI OTSU

Questo tipo di sogliatura utilizza tecniche di analisi dell'*istogramma*[2] dell'immagine ed è particolarmente adatto per immagini *bimodali* (vedi 3), ovvero per immagini che presentano istogrammi con una netta separazione tra due picchi principali. La binarizzazione di *Otsu* si occupa proprio di trovare la "valle" di scissione fra tali picchi, che risulta essere proprio il valore di soglia ottimale per l'applicazione di un'operazione di sogliatura globale.

Nel caso in cui l'immagine in analisi non presenti esattamente due massimi locali nella funzione istogramma, il procedimento di Otsu potrebbe però portare a risultati indesiderati.

2.5 ALGORITMO DI SOGLIATURA PROPOSTO

2 L'*istogramma* h di un'immagine I in scala di grigi, con valori d'intensità $\in [0, K-1]$ e definiti nell'immagine della funzione I, è una funzione tale che $h(i)$ è uguale al numero di pixel di I con valore di intensità i, per $0 \leqslant i < K$. Più formalmente $h(i) = |(u, v) : I(u, v) = i|$.

3

TEMPLATE MATCHING

4

TEXT DETECTION

5

ULTERIORI MIGLIORIE

5.1 MOTORI OCR

5.2 BENCHMARKS

5.3 CONTAINERIZZAZIONE CON DOCKER

5.4 CHIAMATE PARAMETRIZZABILI SUI CAMPI DEL DOCUMENTO

5.5 POSTPROCESSING

CONCLUSIONI E SVILUPPI FUTURI

RINGRAZIAMENTI

BIBLIOGRAFIA

[1] Wilhelm Burger, Mark J. Burge - *Digital Image Processing - An Algorithmic Introduction Using Java, Second Edition*

[2] Raffaele Cappelli - *Fondamenti di Elaborazione di Immagini - Operazioni sulle immagini* - Università degli Studi di Bologna

[3] Wikipedia - *https://it.wikipedia.org*

[4] Guocheng Wang, Yiwen Wang, Hui Li, Xuanqi Chen, Haitao Lu, Yanpeng Ma, Chun Peng, Yijun Wang, Linyao Tang - *Morphological Background Detection and Illumination Normalization of Text Image with Poor Lighting* - PLOS ONE

[5] OpenCV documentation - *https://docs.opencv.org*

[6] Simone Parca - *Studio e sviluppo di algoritmi per l'analisi di immagini della pelle acquisite tramite un sensore capacitivo* - Università degli Studi di Bologna

[7] Francesco Tortorella - *Elementi di geometria delle immagini digitali binarie* - Università degli studi di Cassino e del Lazio Meridionale (Cited on page 12.)

[8] *Advanced morphological image processing* - University of Groningen

[9] W. Niblack - *An introduction to Digital Image Processing* - Prentice-Hall (Cited on page 18.)

[10] J. Sauvola, M. Pietikainen - *Adaptive document image binarization* - Pattern Recognition 33(2), pp. 225-236 (Cited on page 18.)