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

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* o *elemento strutturante*, 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.

1.3 OPERAZIONI MORFOLOGICHE

2

BINARIZZAZIONE DI IMMAGINI

3

TEMPLATE MATCHING

4

TEXT DETECTION

4.1 INTRODUZIONE

4.2 EAST

4.3 ALGORITMO PROPOSTO

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 di Bologna