Introduzione ad OpenCV in C++

OpenCV

OpenCV è una libreria open-source, scritta in C, per lo streaming video real-time, ovvero la **Computer Vision** e gira sotto Windows, Linux e MacOS X.

La libreria permette una semplice gestione di immagini trattandole come “***matrici di pixel*“,** alle quali è possibile accedervi in maniera molto semplice e rapida. Prima di cimentarci nella vera e propria **Computer Vision**, che è un ambito di una vastità non indifferente, diamo una sguardo a quella che si può definire l’**Image Processing**, cioè l’**Elaborazione delle Immagini** attraverso la libreria in questione.

L’**Elaborazione delle Immagini** è la base della **Computer Vision**. Ci permette di andare a modificare/migliorare le immagini o sequenze di immagini, cioè video, in modo tale che le successive operazioni siano svolte con maggiore semplicità: può quindi essere considerata una sorta di ***pre-processing.*** Vediamo qualche esempio.

Supponiamo che, data in input l’immagine successiva, cioè un’immagine contenente alcune monete, la si voglia elaborare per effettuare quella che viene definita una **Object Recognition**, cioè il riconoscimento degli oggetti presente nell’immagine o nella sequenza di immagini.

Come si può notare molto facilmente, è presente del **rumore** su questa immagine, che

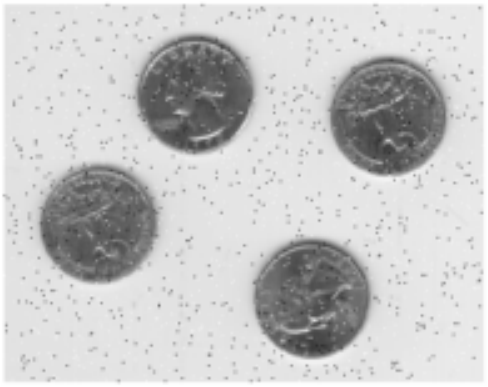


Immagine con presenza di rumore

e
pu
ò
re
nd
er
e
in
ef
fi
ca
ce
qu
al
si
as
i
su
cc
es
si
va
an
al
is
i
de
ll
’i
mm
ag
in
e.

Attraverso l’applicazione di **tecniche** di Elaborazione delle Immagini, arriveremo a migliorare quest’immagine in modo tale da facilitare tutte le successive analisi ed elaborazione

delle immagini, fino ad eliminare completamente il rumore e ottenere un’immagine pulita.



Immagine senza rumore

Come effettuare le operazioni di pulizia dell’immagine, dicattura di caratt

eristiche dell'immagine, cioè le **features** dell'immagine, e tante altre operazio

ni
,
sa
ra
nn
o
de
sc
ri
tt
e
ed
ar
go
me
nt
at
e
ne
gl
i
ar
ti
co
li
su
cc
es
si
vi
di
qu
es
ta
ru
br
ic

a
ch
e
te
rm
in
er
à
co
n
la
cr
ea
zi
on
e
di
un
ve
ro
e
pr
op
ri
o
Ob
je
ct
Re
co
gn
it
io
n.

Tutto questo sarà permesso dalla libreria che andremo ad introdurre in questa sezione, cioè **OpenCV.**



C++

La prima operazione da effettuare è l'installazione della libreria: per quanto riguarda **Windows**, l'installazione ha bisogno di **Visual Studio** per il corretto funzionamento, mentre per quanto riguarda **Linux** abbiamo bisogno di avviare questo link.

Una volta avviato e terminata l'istallazione, prepariamo l'**ambiente di lavoro**, cioè come deve essere creata la directory per l'esecuzione del file ***.cpp***.

Nella directory devono essere presente i seguenti file:

- **CMakeLists.txt**
- **codice.cpp**

Il **CMakeLists.txt** deve essere strutturato in un certo modo, in modo tale da permettere la compilazione del file **codice.cpp**

```
cmake_minimum_required(VERSION 2.8)
project(Prova)
find_package(OpenCV REQUIRED)
add_executable(codice.exe codice.cpp)
target_link_libraries(codice.exe ${OpenCV_LIBS})
```

L'argomento di ***project()*** corrisponde al nome della directory nella quale vogliamo compilare il codice.

Adesso passiamo alla parte di codice vera e propria: andremo a visualizzare il codice completo e andremo a commentare ogni singola riga in modo tale da essere il più possibile chiari.

```
#include <opencv2/opencv.hpp>
#include <iostream> //namespace per utilizzare tutte le funzioni di OpenCV
using namespace cv;
using namespace std; int main(int argc, char **argv)
{ //Apriamo l'immagire utilizzando imread() //Mat corrisponde alla matrice di pixel che forma l'immagine Mat immagineOriginale = imread(argv[1]); //Attraverso l'attributo data andiamo a verificare se l'immagine //è stata aperta correttamente if(immagineOriginale.data == NULL) { cerr << "Error open image" << endl; return(-1); } //namedWindow() costruisce la finestra dove poi imshow() //mostrerà l'immagine namedWindow("Immagine Originale", CV_WINDOW_AUTOSIZE); imshow("Immagine Originale", immagineOriginale); //Supponiamo che si voglia fare un resize dell'Immagine //Andiamo ad utilizzare la funzione resize() //Cloniamo prima l'immagine originale in un'altra matrice Mat immagineResize = immagineOriginale.clone(); resize(immagineResize, immagineOriginale, Size(640, 480)); namedWindow("Immagine Resize", CV_WINDOW_AUTOSIZE); imshow("Immagine Resize", immagineOriginale); //Supponiamo di voler ottenere le dimensioni dell'immagine //Gli attributi rows e cols ci permettono di conoscere //il numero di righe e il numero di colonne dell'immagine cout << "Numero di righe dell'immagine: " << immagineOriginale.rows << endl; cout << "Numero di colonne dell'immagine: " << immagineOriginale.cols << endl; //Supponiamo di voler convertire l'immagine da RGB a Scala di Grigi //Usiamo la funzione cvtColor() Mat immagineGray = immagineOriginale.clone(); cvtColor(immagineGray, immagineGray, CV_BGR2GRAY); namedWindow("Immagine Gray", CV_WINDOW_AUTOSIZE); imshow("Immagine Gray", immagineGray); waitKey(0); return(0);
}
```

- Il comando ***waitKey(0)*** permette di mettere in attesa il programma finché non si premerà un tasto. Se si volesse aprire direttamente l’immagine in Scala di Grigi, basta

inserire uno **0** come secondo argomento nell'***imread()*** **.**

Ricordiamo che OpenCV permette di aprire senza alcun problema i formati immagine: ***.jpg, .jpeg*** o ***.png.*** Non funziona con le **GIF**.

Al termine della scrittura di **codice.cpp** andiamo a compilare il tutto aprendo il terminale nella directory dove sono contenuti i due file ed eseguire i seguenti passi:

- **cmake .**
- **make**
- **./codice.exe <path_image>**

PYTHON

Per quanto riguarda **Python**, le operazioni non sono risultano molto più semplici ma, ovviamente, anche più veloci. L'installazione della Libreria si effettua tramite il comando ***pip3,*** per quanto riguarda ovviamente **Python3**.

- ***pip3 install opencv-python***

Automaticamente verrà installata l'ultima versione disponibile attualmente di OpenCV. Passiamo al codice.

```
import cv2
import sys # Apertura immagine a colori
Image = cv2.imread(sys.argv[1])
if Image is None: print("Errore Apretura Immagine")
sys.exit(0) cv2.imshow("Immagine Originale", Image) #
Conversione immagine da RGB in Scala di Grigi
GrayScale = cv2.cvtColor(Image, cv2.COLOR_BGR2GRAY)
cv2.imshow("Immagine GrayScale", GrayScale) # Stampa delle
informazioni dell'immagine
print("Numero di righe immagine: ", Image.shape[0])
print("Numero di colonne immagine: ", Image.shape[1])
cv2.waitKey(0)
```

Ovviamente tale codice non merita ulteriori speigazione dato che le funzioni usate sono le stesse descritte nella parte di

codice in C++, con la differenza che sono utilizzate in Python.

Nel prossimo articolo

Nel prossimo articolo andremo ad approfondire il tutto: entreremo nei dettagli implementativi del tipo **Mat in C++**, andremo a spiegare come si accede ai singoli pixel della Mat e le differenze tra immagini in **RGB** o **Scala di Grigi (GrayScale)**, quindi introdurremo cosa si intende per **Spazio di Colore di un immagine**.

Per qualsiasi informazione o ulteriore chiarimento non esitate e commentate!

https://italiancoders.it/introduzione-ad-opencv-in-c/