**Alma Mater Studiorum – Università di Bologna**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SCUOLA DI INGEGNERIA E ARCHITETTURA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

**TESI DI LAUREA**

In

Calcolatori Elettronici – T

Creazione applicazione per il tracking

di persone e oscuramento volti.

CANDIDATO: Relatore:

Francesco Luzzi Prof. Stefano Mattoccia

Correlatore:

Alessio Mingozzi

Sessione

Anno Accademico 2020/21

Sommario

[1 Introduzione 3](#_Toc86830681)

[2 Strumenti utilizzati 3](#_Toc86830682)

[2.1 CMake 3](#_Toc86830683)

[2.2 OpenPose 3](#_Toc86830684)

[2.3 OpenCv 3](#_Toc86830685)

[2.4 NumPy 3](#_Toc86830686)

[3 Sviluppo 4](#_Toc86830687)

[3.1 Input 4](#_Toc86830688)

[3.2 Logica 4](#_Toc86830689)

[3.3 Primi output 4](#_Toc86830690)

[3.4 Modifiche dei threshold 4](#_Toc86830691)

[3.5 Nuovi output 4](#_Toc86830692)

[4 Conclusione 4](#_Toc86830693)

# Introduzione

Alla base di questo progetto vi è la necessità di nascondere automaticamente l’identità delle persone presenti all’interno di una fotografia o di un video coprendone il volto.   
Per questo tipo di applicazione non bisogna non riconosce le persone, ma “semplicemente” la posa di ognuna di esse concentrandosi, sulla posizione del volto (naso-collo) così da poterlo oscurare.

L’ approfondimento di questo problema mi ha spinto a creare questa applicazione…

La presentazione sarà suddivisa in x capitoli:  
 -…

# Strumenti utilizzati

Per lo sviluppo di questa applicazione gli strumenti utilizzati sono stati:

## Anaconda

Anaconda è un package manager, un manager di ambienti virtuali, una distribuzione di Python/R improntata alla data-science, con una grande collezione di pacchetti open-source.

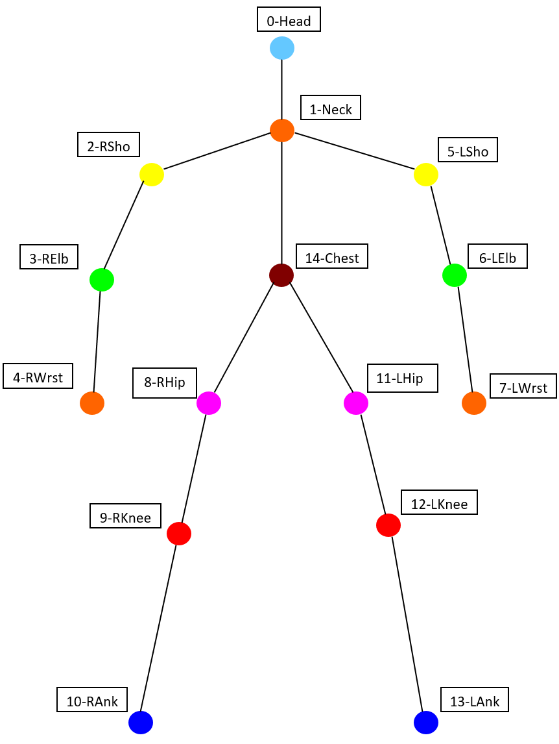
Anaconda è open-source ed è stata una scelta quasi ovvia nel mio caso, la gestione delle versioni e dipendenze dei diversi pacchetti rispetto alla versione di python utilizzata è molto comoda sia dal punto di vista dell’automazione ma anche perché questo permette di non “sporcare” il resto del sistema operativo.

## CMake

CMake è un sistema open-source estendibile che gestisce il processo di compilazione in un Sistema Operativo, in modo indipendente da altri compilatori. Differentemente da altri sistemi cross-platform,

CMake viene utilizzato in combinazione all’ambiente di compilazione nativo per compilare codice, creare librerie, wrappers ed eseguibili, potendo anche sfruttare librerie dinamiche e statiche.  
  
CMake è progettato per supportare gerarchie di directory complesse e applicazioni dipendenti da diverse librerie.

## OpenPose

Sfruttando il machine learning è stata creata una rete neurale che passato come input un’immagine restituirà una mappa di probabilità delle diverse parti del corpo delle persone presenti nell’immagine.  
La rete neurale usata nello specifico utilizza un formato di output dei “keypoints” chiamato MPI

Questa rete restituisce anche mappe di probabilità per le diverse “connessioni” tra i diversi keypoints, molto utili per riconoscere la relazione tra essi(sono della stessa persona o meno).

## OpenCv

Libreria open source molto presente nel panorama della computer vision, sviluppata in C++ e resa disponibile tramite un wrapper anche su python. L’implementazione da me utilizzata è leggermente modificata: è stata compilata con CMake utilizzando il source code “originale” di opencv ed il codice della versione “contrib”, tutto ciò per abilitare/sfruttare l’accelerazione GPU data dai drivers CUDA di NVIDIA™, difatti le GPU sono molto più efficienti, per questo tipo di workload algebrico/matriciale, rispetto alle CPU.

## NumPy

Libreria OpenSource indispensabile per fare operazioni algebriche in python con overhead e durata dell’esecuzione minori.

Di fatto è una libreria in C con un wrapper per python, è molto veloce e performante in quanto il codice utilizzato è precedentemente compilato e non interpretato al momento come python.

# Sviluppo

Immagini del codice, con spiegazione semplice, non troppo nel dettaglio di ciò che sta succedendo.

## Input

Immagini utilizzate per il testing e problemi con questa rete neurale.

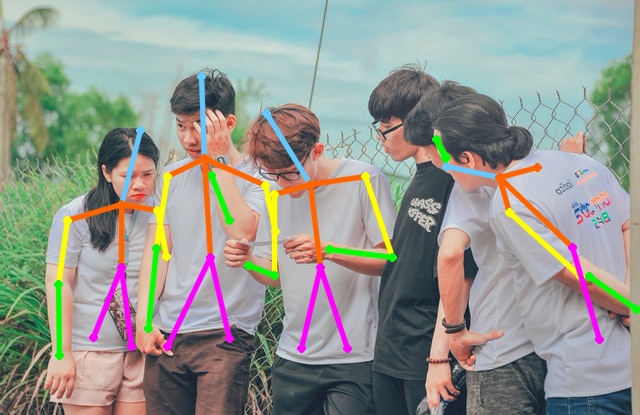
## Logica

Immagini del codice con immagini esplicatorie della logica dietro la probMap, il ritrovamento dei keypoints e l’utilizzo delle mappe di probabilità PAF (Part Affinities) che codificano il grado di associazione tra i diversi keypoints, utili rivendicare diversi keypoints alle persone corrette.

(Spiegazione per quale motivo è sbagliato usare un algoritmo che lega diversi keypoint a seconda della distanza senza basarsi sui PAF)

## Primi output



Immagini di output degli input con spiegazione della problematica su immagine qui di sopra

## Modifiche dei threshold

Spiegazione della modifica dei valori di threshold per la rilevazione dei keypoints, specificando che solitamente i keypoint meglio riconosciuti sono quelli del collo e del naso/viso, quindi anche perdere qualche keypoint dei piendi non ci interessa allo scopo dell’applicazione

## Nuovi output

Nuove immagini di output degli input con commento sui cambiamenti dell’output e sull’aumento delle prestazioni (dati i minori keypoints “inutili” registrati)

Immagine che contiene pavimento, interni, edificio, piastrellato

Descrizione generata automaticamente

Immagine che contiene persona, esterni, inpiedi, gruppo

Descrizione generata automaticamente

# Conclusione

Possibili ambiti di utilizzo, utilità di questa applicazione e possibili migliorie quali, per esempio, ridurre il peso di esecuzione limitandosi ad analizzare la posizione soltanto dei keypoints del viso per avere un framerate migliore.

Luzzi Francesco, ringraziamenti ed addio.