Image & Video Analysis project

Social distancing measure

Lorenzo De Luca e Manuel Natale Sapia

Abstract

In questo progetto l'obiettivo è stato quello di implementare un software che permetta di individuare il distanziamento sociale tra le persone. In particolare usando OpenPose, che estrae uno scheletro 2D delle persone rilevate dalla webcam/video preregistrato, vengono estratte le coordinate relative ai punti del piede destro e del piede sinistro per individuare il posizionamento di ogni persona all'interno della scena e, supponendo che il posizionamento dei piedi individuati siano sullo stesso piano 3D, usando l'omografia, si effettua una trasformazione delle coordinate trovate rappresentandole in una *bird eye view*. In questo modo è stato possibile calcolare la distanza tra ogni persona e nel caso in cui questa distanza sia inferiore a 2 metri un messaggio di avviso, visivo e sonoro, indica il mancato rispetto della distanza interpersonale.

1 Introduzione

Nella lotta contro il Covid-19, il distanziamento sociale ha dimostrato di essere una misura molto efficace per rallentare la diffusione del virus in assenza di soluzioni definitive.

Per contribuire a garantire il protocollo di distanziamento sociale, basandoci sull'idea del lavoro svolto da Landing AI [1], abbiamo sviluppato un software per il rilevamento del distanziamento sociale che è in grado di rilevare se le persone si tengono a distanza di sicurezza l'una dall'altra, analizzando i flussi video di una telecamera in tempo reale o da registrazioni preesistenti.

In particolare viene evidenziato con un allarme sia visivo che sonoro se una o più persone si trovano ad un distanziamento inferiore alla distanza minima accettabile.

Per implementare quanto appena descritto, abbiamo usato Python 3, e abbiamo strutturato il codice, reperibile su Github [2], in tre file principali:

- *socialDistance.py*: che contiene tutta la parte di importazione delle librerie di OpenPose, la definizione delle costanti e delle variabili globali, l'apertura del flusso video e la chiamata ai metodi ausiliari necessari alla gestione dei vari meccanismi con il quale ogni frame viene elaborato.

- *birdViewFunction.py*: che contiene i metodi principali per la creazione e la gestione della *bird view* e per la gestione della *normal view*.

- *utils.py*: che contiene vari metodi necessari alla creazione della *Region Of Interest* e all'estrazione dei punti dei piedi dal vettore generato da OpenPose.

In questo documento illustriamo il lavoro svolto, descrivendo i vari passaggi effettuati.

2 OpenPose

Per il rilevamento delle persone è stato utilizzato OpenPose, un software largamente sviluppato dal CMU-Perceptual-Computing-Lab [3]. Il funzionamento di Open Pose si basa sull'individuazione di punti caratteristici del corpo umano, associando ad ogni specifica parte del corpo un punto di riferimento e collegandoli tra loro con un certo criterio per creare lo *skeleton*. I due metodi principali per l'individuazione dei punti sono COCO e BODY_25, visibili in Figura 1, ed in particolare nel nostro elaborato abbiamo scelto di utilizzare BODY_25.

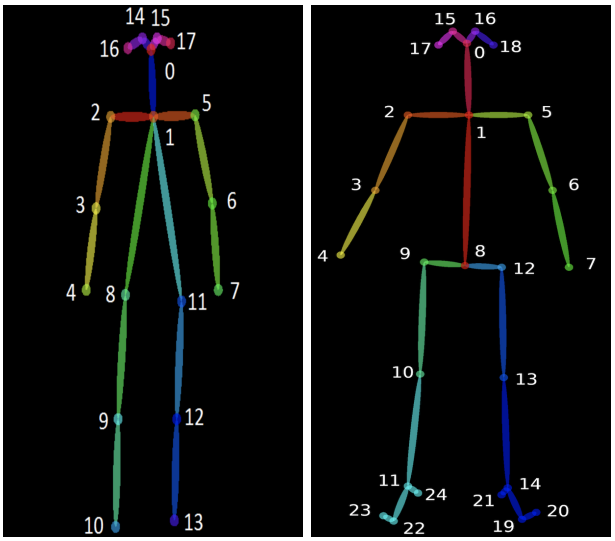

Figure 1: Pose Output Format COCO (left) and BODY_25(right)

2.1 OpenPose nel nostro elaborato

Nel nostro lavoro, per poter calcolare il distanziamento sociale tra le persone, abbiamo supposto che il posizionamento dei piedi delle persone individuate all'interno del video fosse sullo stesso piano 3D, ovvero un pavimento non in pendenza. Quindi, sfruttando il funzionamento di OpenPose con BODY_25, abbiamo considerato i punti 19 e 22, corrispondenti rispettivamente al piede sinistro e al piede destro, e dopo aver trovato il punto medio tra questi, abbiamo potuto individuare dove una persona fosse all'interno della scena.

Nel nostro codice quanto appena descritto viene salvato nel vettore dinamico `humansFeetPoints`, che presenta per ogni persona individuata nella scena una coppia di coordinate x,y che si riferiscono al piede sinistro e una seconda coppia di coordinate che corrisponde al piede destro.

3 Calibrazione

I video che il software è in grado di analizzare possono essere presi da un vista prospettica arbitraria. Dunque, per effettuare un calcolo corretto dell'omografia, che trasforma la *normal view* in una *bird view* (cioè una rappresentazione dall'alto in basso di una scena), affinché la calibrazione avvenga correttamente è necessario posizionare una scacchiera nel primo frame. Questo ci consente di sfruttare il metodo di openCV `findChessboardCorners()` che permette di rilevare i vertici di una scacchiera con un determinato numero di righe e colonne all'interno dell'immagine oppure nel caso in cui la scacchiera non venga rilevata per motivi di luce ambientale o di distanza dall'obiettivo della fotocamera, possiamo immettere manualmente i quattro vertici (in alto a sinistra, in alto a destra, in basso a destra e in basso a sinistra).
Il rilevamento dei vertici ci consente di capire quale sia effettivamente la vista prospettica della scena ripresa, in questo modo possiamo andare ad individuare la regione di interesse sulla quale possiamo fare la mappatura ai rispettivi vertici della *bird view*. I concetti appena presentati sono trattati teoricamente in modo più accurato nel successivo sotto paragrafo 3.1.
L'implementazione di quanto appena detto viene sviluppata tramite la funzione `get_camera_prospective()`, che sfruttando la funzione di openCV `getPerspectiveTransform()`

ci consente di creare la matrice di trasformazione M e la matrice di trasformazione inversa M^{-1}.

Una considerazione importante riguardante la regione di interesse da mappare riguarda il fatto che una porzione di questa regione potrebbe capitare su un lato di una parete, pertanto potrebbe generare degli errori. Tuttavia abbiamo considerato che in un normale scenario, nessuna persona (nessun piede) potrà mai camminare su quella parte di piano, per tanto la suddetta porzione viene comunque rappresentata nella *bird view*, ma nonostante ciò non verrà mai visualizzata nessuna persona in quel tratto.
Un altro fattore analizzato riguarda il ridimensionamento in una regione più piccola della ROI, infatti in una vista prospettica, i punti individuati vicino al punto di fuga o in una posizione molto distante dalla scacchiera (base di partenza tramite la quale viene fatta la trasformazione), rischiano di distorcere la corretta individuazione della distanza tra due persone.

Infine, l'individuazione della scacchiera ci permette di stimare anche il fattore di scala della *bird view*, ovvero di capire quanti pixel corrispondono ad un lato della scacchiera, in modo da poter stimare in metri la distanza interpersonale. Questo viene fatto con il metodo di openCV `perspectiveTransform()`, il quale genera i punti che ci permettono di definire la distanza di riferimento `d_tresh`.

3.1 Tecnica di trasformazione da *normal view* a *bird view*

La distorsione prospettica è una distorsione geometrica che avviene nelle riprese video che viola la percezione visiva dello spazio 3D alludendo ad una falsa posizione degli oggetti all'interno della scena. Per questo motivo per un'analisi più accurata sul distanziamento sociale, si ricorre ad una correzione della distorsione, riportandoci in una condizione di vista dall'alto (*bird view*).

La trasformazione proiettiva avviene grazie alla matrice di omografia M, la quale consente una trasformazione dei pixel della *normal view* distorta alla *bird view* tramite l'equazione

$$p_2 = Mp_1$$

dove p_1 e p_2 rappresentano i pixel dell'immagine prima e dopo la trasformazione, con

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

In base alla presenza o assenza di informazioni sui parametri della trasformazione prospettica e degli oggetti osservati, si possono applicare due metodi principali per la stima della matrice omografica; nel nostro caso è stato applicato il metodo di *'Stima della matrice omografica mediante le informazioni sull'orientamento angolare della fotocamera e distanza dal piano di visualizzazione'*, che si basa sulla trasformazione della distorsione prospettica in una *bird view*, ovvero si basa sull'idea di posizionare una videocamera virtuale in modo che il suo asse ottico sia perpendicolare al piano visualizzato. La formulazione geometrica del problema possiamo vederla in Figura 2 e il calcolo della matrice omografica si basa sull'individuazione dell'angolo θ_1 che si forma tra l'asse ottico della videocamera reale (RC) e quello della videocamera virtuale (VC) (per maggiori dettagli vedi [5]).

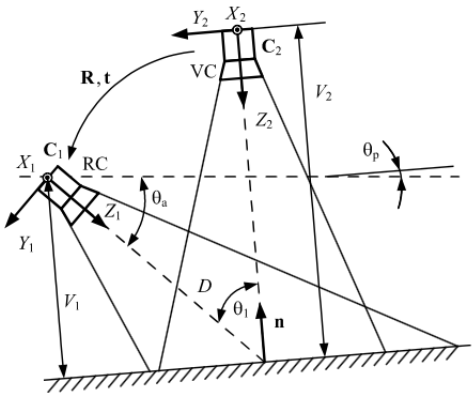

Figure 2: Posizione reciproca di telecamera reale e virtuale.

4 Bird View

All'interno della *bird view* verranno disegnati i punti, correttamente traslati in base alla matrice di omografia trovata, della media tra la posizione del piede sinistro e destro di ogni persona e, quando all'interno della scena saranno presenti almeno 2 persone, verranno anche disegnate delle circonferenze che rappresentano la distanza che ciascuna persona deve mantenere dalle altre affinché sia rispettata la distanza sociale e infatti, finché questa non verrà violata le circonferenze saranno di colore verde mentre quando verrà violata saranno di colore rosso.

Per rendere lo spostamento dei punti visualizzati, corrispondenti alle persone nella scena, più lineare è stato realizzato un meccanismo a finestra scorrevole in cui si vanno a memorizzare 7 posizioni per ogni persona (6 passate e 1 attuale) e ne viene calcolata la media ad ogni frame. Questo meccanismo è stato implementato utilizzando una struttura dati di tipo `deque()` la quale permette agevolmente di effettuare inserimenti in testa e rimozioni in coda.

La funzione che ci consente di rappresentare i punti sulla *bird view* è `plot_points_on_bird_eye_view()`, mentre ciò che riguarda il calcolo della distanza tra una coppia di persone viene gestito dalla funzione `find_pairwise_distance()`. Infine `plot_distance_circle_nodes()` ci consente di rappresentare e di gestire la variazione di colore delle circonferenze rappresentati la distanza interpersonale.

5 Normal View

Nella *normal view* viene rappresentata la scena originale, in cui ad ogni persona viene sovrapposto lo *skeleton* di OpenPose e le linee di distanza tra ogni coppia di persone. Le distanze in metri tra le coordinate presenti in questa vista vengono calcolate utilizzando le coordinate precedentemente trovate della *bird view*; questo è stato possibile sfruttando la matrice di trasformazione inversa $M^{-}1$.
Successivamente, sfruttando le coordinate e le distanze trovate, si disegnano, grazie alla funzione `plot_lines_between_person()`, delle linee tra ogni coppia di persone e si stampa a schermo la relativa distanza numerica in metri. Per enfatizzare il distanziamento sociale, come precedentemente fatto nella *bird view*, le linee sono di colore rosso se la distanza sociale non è rispettata, viceversa saranno di colore verde. Inoltre, in questa vista, il mancato rispetto del distanziamento sociale include un 'WARNING' visivo, accompagnato da un segnale sonoro.

6 Esempi di Output

In figura 3 possiamo vedere un frame nel caso in cui la distanza sociale sia rispettata: si nota che nella *bird view* le circonferenze che delimitano la distanza interpersonale non si intersecano e sono di colore verde, indicatore del fatto che le persone nella scena non sono in 'WARNING'.

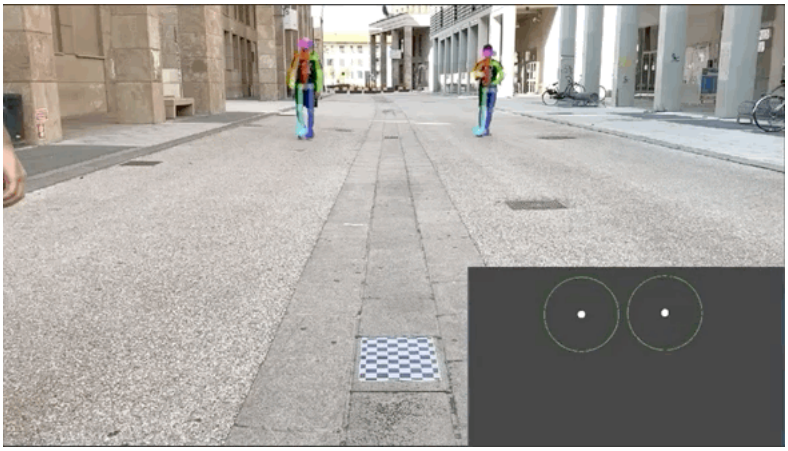

Figure 3: Frame con distanza sociale rispettata

Viceversa in figura 4 è rappresentato un frame in cui la distanza sociale tra due persone della scena non sia rispettata, quindi le circonferenze di distanza

interpersonale sono intersecate e di colore rosso, dunque le persone sono in 'WARNING'.

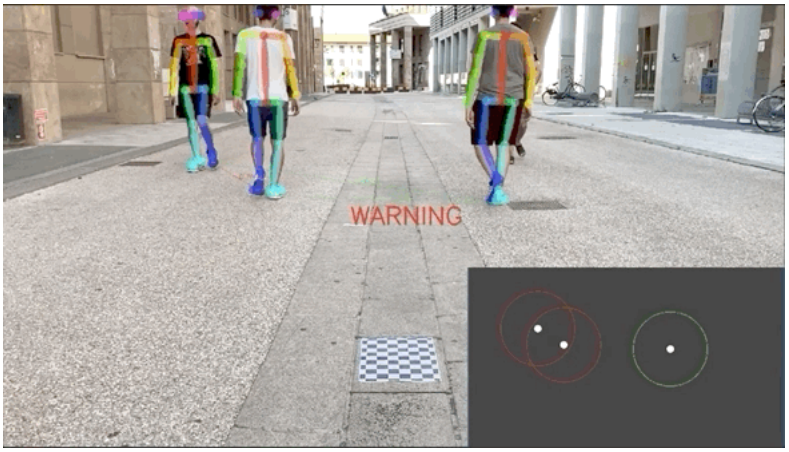

Figure 4: Frame con distanza sociale non rispettata 'WARNING'

Alcuni risultati video, per capire meglio la dinamica di funzionamento del lavoro svolto, sono disponibili al link [6].

References

[1] Landing AI Creates an AI Tool to Help Customers Monitor Social Distancing in the Workplace, Apr 2020 https://landing.ai/landing-ai-creates-an-ai-tool-to-help-customers-monitor-social-distancing-in-the-workplace/.

[2] GitHub Repository of the project code https://github.com/loredeluca/Document-Recognition.

[3] OpenPose https://github.com/CMU-Perceptual-Computing-Lab/openpose.

[4] PoseEstimation https://github.com/ildoonet/tf-pose-estimation.

[5] "Bird's Eye View Transformation Technique in Photogrammetric Problem of Object Size Measuring at Low-altitude Photography", I.S. Kholopov, Atlantis Press, 2017.

[6] Google Drive folder with results https://drive.google.com/drive/folders/1X1h2gBOS2y37HF5bY90luw8SgPwK6XUW?usp=sharing.