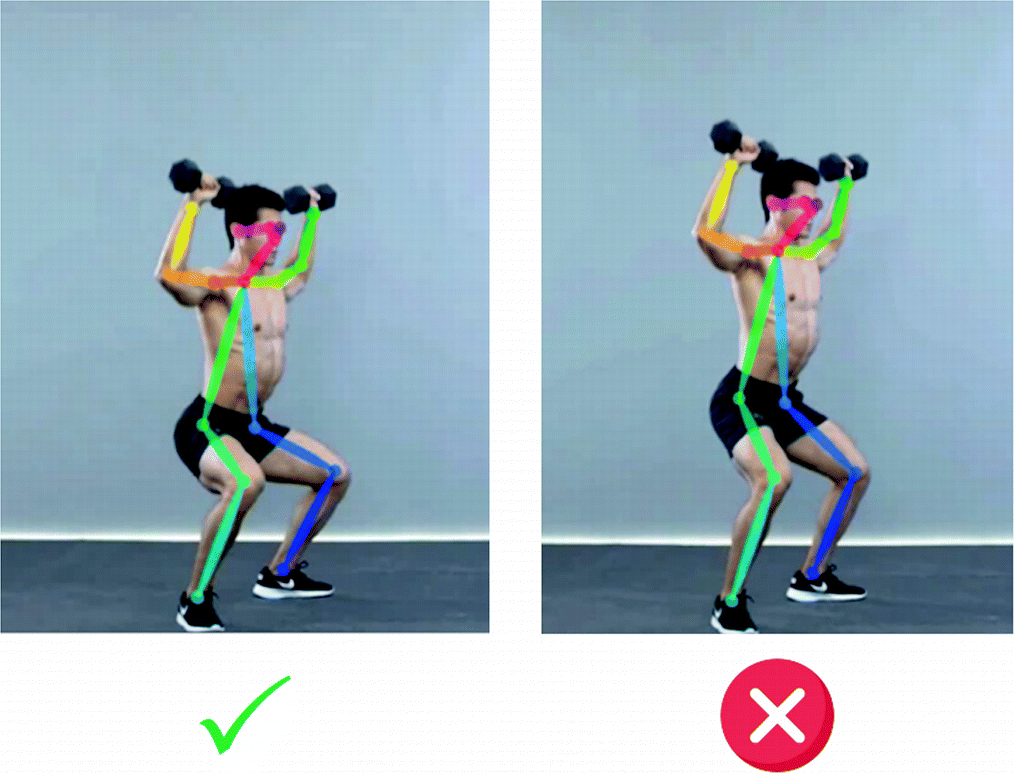
2021

**A cura di**

|  |  |
| --- | --- |
| **Mangione Luigi** | **Mat: 699620** |
| **Marzano Savino** | **Mat: 699921** |

<https://github.com/Savino-M/TrainerEstimation>

TRAINER ESTIMATION



**1.0 INTRODUZIONE**

Trainer estimation è un’applicazione pensata per aiutare chi svolge attività fisica, in casa o in palestra, per monitorare la propria sessione di allenamento. È infatti, utile per tenere conto delle ripetizioni svolte di un esercizio e per verificarne il corretto svolgimento, pertanto, vengono contate solamente le ripetizioni eseguite correttamente.aa

**2.0 UTILIZZO**

L’interazione avviene quasi tutta vocalmente: l’utente infatti può scegliere l’esercizio da eseguire, dettando il nome dell’esercizio o scrivendolo. Per il resto, TrainerEstimation comunica con la voce: conta, ad esempio, le ripetizioni o guida l’utente nell’eseguire correttamente l’esercizio.

**3.0 PROGETTAZIONE E REALIZZAZIONE**

Nella fase di progettazione si sono definiti innanzitutto quali siano i casi d’uso del programma ed eventuali risorse, come un modello di esempio per ogni esercizio e il modo in cui stabilire se l’esercizio viene eseguito correttamente o meno.

Per fare ciò, si è deciso in primo luogo di utilizzare come linguaggio di sviluppo, Python. Questo perché esistono numerose librerie, utili per le funzionalità del software. Si sono infatti utilizzate diverse librerie, ognuna con uno scopo ben preciso e dedita ad una fase specifica dell’esecuzione. Le due librerie principali utilizzate sono OpenCV e MediaPipe.

* OpenCV viene utilizzata per acquisire le immagini da webcam ed eventualmente modificarle;
* MediaPipe invece, è una libreria realizzata da Google, sviluppata in Python e utile per il rilevamento della posizione di una persona: è infatti in grado di rilevare fino a 33 punti fisici diversi.

Per riconoscere la posizione di partenza, è stato creato un classificatore apposito per esercizio. Se quindi l’utente viene rilevato in posizione iniziale, si passa alla fase successiva.

Da questo momento il programma inizia a registrare le azioni dell’utente, ricavare le coordinate dei landmarks frame per frame e creare un dataframe, che viene costantemente aggiornato. Una volta che l’utente torna in posizione di partenza, viene effettuato un confronto tra il dataframe creato in ‘live’ e quello preesistente, riferito ad un’esecuzione eseguita correttamente.

Il confronto viene effettuato tramite DTW, una tecnica utile per confrontare sequenze temporali. Questo tipo di funzione restituisce una certa distanza, la quale, se minore della soglia impostata, reputa la ripetizione corretta e ne incrementa quindi il numero.

Il codice è stato strutturato in modo da essere il più generale possibile, è infatti molto semplice aggiungere un nuovo esercizio da supportare. Basta creare una nuova cartella contenente i file che servono all’esecuzione.

In particolare, servirà inserire:

* Un classificatore in grado di rilevare la posizione di partenza dell’esercizio;
* un dataframe che rappresenterà l’esempio di esecuzione corretta dell’esercizio;
* un video di esempio (di esecuzione corretta)

Aggiunti questi tre file alla cartella delle risorse, basterà semplicemente definire una soglia di accettazione per l’esecuzione.

**3.1 BUILD DATAFRAME**

Per far sì che l’applicazione funzioni, è necessario un dataframe. Crearlo però è complesso, ecco perché si è creato uno script “di utilità”, BuildDataFrame. Questo ha lo scopo di analizzare un video e crearne un dataframe. Solitamente questo tipo di strutture dati sono in formato json, ma per rendere più intuitiva la sua visualizzazione, nel caso in cui ce ne sia bisogno, viene convertito in un file Excel.

Utilizzare questo script, quindi, ha semplificato molto il lavoro. Soprattutto nel caso in cui si volessero aggiungere dei nuovi esercizi.

**3.2 BASICS**

Per guidare l’utente nell’utilizzo del software, ma anche per aiutarlo nello svolgimento dell’esercizio, abbiamo creato lo script Basics. Questo ha il compito di analizzare il video della posizione iniziale, in base all’esercizio scelto dall’utente, ricavare quindi i landmarks e “disegnarli” a schermo. A questo punto, l’utente, in maniera molto intuitiva e semplice, si posiziona in maniera corretta. Questo perché, oltre alle linee guida, vengono mostrati anche i landmarks dell’utente stimati in tempo reale.

**3.3 THREAD RILEVAMENTO**

Come si evince dal nome, questo script è un Thread, che lavora quindi in concorrenza con **Interfaccia**. Mentre quest’ultimo script ha il solo scopo di mostrare le linee guida all’utente, calcolate da Basics e i landmarks rilevati dalla webcam, ThreadRilevamento svolge la componente principale dell’applicazione. Essendo sempre in esecuzione, si occupa di rilevare frame per frame se l’utente è posizionato nella posizione di partenza, tenendo ovviamente conto dell’esercizio scelto. Affinché avvenga, viene utilizzato il classificatore, che, ad ogni frame, verifica se l’utente è in posizione o meno, in tal caso, si passa alla fase successiva. In particolare, si aspetta che l’utente “si muova” per poi tornare in posizione di partenza, ovvero quando avrà eseguito una ripetizione dell’esercizio. Fino a questo momento, le coordinate dei landmarks, rilevate frame per frame, vengono inserite in un dataframe volatile. Una volta eseguita la ripetizione, questo dataframe viene confrontato con quello già presente, corrispondente cioè ad una corretta esecuzione dell’esercizio. Qui, viene restituito un valore, una distanza, che se minore di una soglia, ritiene la ripetizione eseguita correttamente, incrementandone il numero, che viene dettato. Fatto ciò, il dataframe volatile viene svuotato. Questo procedimento avviene per ogni ripetizione eseguita, strategia utile che consente di non appesantire la dimensione del dataframe “live” e quindi non caricare ulteriormente la CPU.

**3.4 DYNAMIC TIME WARPING**

Il DTW viene utilizzato nella fase di confronto dei dataframe e quindi nella valutazione dell’esecuzione. Un dataframe non è altro che una tabella dati eterogenea bidimensionale, a dimensione variabile, con gli assi etichettati. Nel nostro specifico caso, una riga rappresenta un frame, mentre le colonne rappresentano le coordinate dei landmarks in quel frame. Per creare un dataframe e gestirlo, ci viene in aiuto la libreria Pandas.

Abbiamo utilizzato il DTW come tecnica di confronto, perché è utile per calcolare la similarità o la distanza tra due strutture dati, o serie temporali, con lunghezze differenti. Abbiamo utilizzato una libreria esterna, fastdtw: dati in input due strutture dati, siano esse monodimensionali o bidimensionali, restituisce la distanza fra esse.

**CLASSIFICATORE**

**4.0 CONCLUSIONI**

L’applicazione si dimostra funzionare egregiamente su pc, unica piattaforma per ora supportata dall’app, anche se, come verificato, la maggior parte degli utenti avrebbe preferito una sua implementazione su smartphone, per ovvie ragioni di portabilità. È stato inoltre effettuato un test per valutarne le prestazioni in termini di fps, e con una media di 18 frame al secondo, l’app risulta utilizzabile e non particolarmente fastidiosa.

Da sottolineare inoltre, che non avendo a disposizione una webcam interna, sono state utilizzate delle webcam esterne, connesse tramite Wi-Fi, che forse hanno leggermente appesantito il carico su CPU, già abbastanza grande di suo. Questo è dovuto a diversi fattori, al fatto che la componente principale del programma viene eseguita su Thread, in quanto l’altra parte si occupa di disegnare la posizione inziale. Inoltre, la libreria utilizzata, MediaPipe, lavora su CPU. E quindi rende molto pesante il carico su questo componente hardware. Senza contare il fatto che il confronto dell’esecuzione, utilizzato il DTW, richiede molte risorse computazionali. Riassumendo, tutto il programma pesa sulla CPU.