**TRAINER ESTIMATION**

**DOCUMENTAZIONE**

**Progetto per l’esame di sistemi ad agenti, a cura di:**

|  |  |
| --- | --- |
| **Mangione Luigi** | **Mat: 12345** |
| **Marzano Savino** | **Mat: 699921** |

**1.0 INTRODUZIONE**

Trainer estimation è un’applicazione pensata per aiutare chi svolge attività fisica, in casa o in palestra, a monitorare la propria sessione di allenamento. È infatti utile per tenere conto delle ripetizioni svolte di un determinato esercizio e per verificare che lo si abbia eseguito correttamente. Infatti, vengono contate solamente le ripetizioni eseguite correttamente.

**2.0 UTILIZZO**

L’interazione è molto semplice, in quanto l’utente deve solo scegliere quale esercizio eseguire (fra quelli supportati), posizionarsi nella posizione stabilita, e al resto ci pensa TrainerEstimation. Infatti, questo assistente di allenamento conta le ripetizioni e aiuta l’utente nel caso esegua una ripetizione sbagliata. Per una migliore usabilità, le ripetizioni non appaiono a schermo, in quanto per l’utente potrebbe risultare difficile eseguire l’esercizio e allo stesso tempo guardare lo schermo del computer. Il conteggio viene quindi dettato a voce dall’applicazione.

**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 ad esempio 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 innanzitutto di utilizzare come linguaggio di sviluppo, Python. Questo perché esistono numerose librerie, utili per le funzionalità del software. Abbiamo infatti utilizzato 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 ricavare la posizione di partenza, è stato creato un video apposito, dove una persona è posizionata in posizione di partenza per diversi secondi. Non abbiamo potuto utilizzare un’immagine perché, in questa, la libreria non rileva nessun landmark. Dopo aver scansionato quindi il video, si ricavano le coordinate di ogni landmark (punto corporeo) e confrontate con quelle dell’utente, rilevate in tempo reale. Se quindi l’utente viene rilevato in posizione di partenza, allora si passa alla fase successiva. Il programma inizia, da questo momento, 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 una esecuzione eseguita correttamente. Il confronto viene effettuato questa volta, tramite DTW. Tecnica utile per confrontare delle 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 file testuale contenente, riga per riga, le coordinate x e y di ogni landmark (una coppia di righe, quindi, rappresenterà la posizione di un landmark);
* 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, il gioco è fatto. Basterà semplicemente definire una soglia di accettazione per la posizione iniziale e per l’esecuzione.

**3.1 BUILD DATAFRAME**

Come detto, per far sì che l’applicazione funzioni, bisogna avere a disposizione un dataframe di esempio che rappresenti l’esecuzione corretta di un esercizio. Creare un dataframe non è per nulla semplice, ecco perché si è deciso di creare 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 di molto il lavoro. Soprattutto nel caso in cui si vogliano aggiungere ora dei nuovi esercizi. Lo stesso principio viene utilizzato per creare il file testuale contenente le coordinate del landmark in posizione di partenza. Soltanto che, una volta creato il dataframe relativo a questo, i dati vengono poi manualmente ricopiati in un file testuale. Questo perché purtroppo non siamo riusciti a convertire il dataframe in .txt da codice.

**3.2 BASICS**

Per aiutare e quindi 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 la posizione iniziale e “disegnarla” a schermo. Così facendo, all’utente riesce, in maniera molto intuitiva e semplice, a posizionarsi in maniera corretta per l’esecuzione dell’esercizio. Questo perché, oltre alle linee guida, che in breve rappresentano la posizione dei landmarks corretta, vengono mostrati anche i landmarks dell’utente, in tempo reale, riuscendo quindi a capire quasi perfettamente la posizione giusta.

**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, e quindi non avendo mai una fine, si occupa di rilevare frame per frame se l’utente è posizionato nella posizione di partenza, tenendo ovviamente conto dell’esercizio che è stato scelto. Per fare ciò, vengono semplicemente comparate le coordinate dei landmarks del frame con le coordinate presenti all’interno del file txt. Se il risultato, e quindi la distanza tra le coordinate è minore di una certa soglia (stabilita dopo vari test), allora si passa al rilevamento e alla valutazione dell’esecuzione. 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, tutti i landmarks (le loro coordinate) rilevati frame per frame, vengono inseriti in un dataframe volatile. Una volta eseguita la ripetizione, questo dataframe viene confrontato con quello già presente, corrispondente ovvero ad una corretta esecuzione dell’esercizio. Anche qui, viene restituito un valore, una distanza, che se minore di una soglia, diversa da quella utilizzata per la posizione iniziale, ritiene la ripetizione eseguita correttamente, incrementando il numero di ripetizioni, che viene quindi sintetizzato vocalmente. Fatto ciò, il dataframe volatile viene svuotato. Questo procedimento avviene quindi 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 deciso di utilizzare il DTW come tecnica di confronto, perché è appunto utile per calcolare la similarità o la distanza tra due strutture dati, o serie temporali, con lunghezze differenti. Per fare ciò, abbiamo utilizzato una libreria esterna, fastdtw. Questa, infatti, dati in input due strutture dati, siano esse monodimensionali o bidimensionali, restituisce la distanza fra esse.

**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. In parole povere quindi, tutto il programma pesa sulla CPU