DanCe-V

Team C2: Danny Cui, Rex Kim, Akul Singh
18-500 Capstone Design, Spring 2025
Electrical and Computer Engineering Department
Carnegie Mellon University

Product Pitch

With the rise of short, trendy dance challenges on TikTok and Instagram Reels, casual users need a simple, fun, and accessible way to practice these dances without expensive motion capture setups. Our goal this semester was to develop a Unity-based dance coaching tool that uses only a webcam and a computer, making dance learning more engaging and approachable to a wider audience that may not have the time or resources to dedicate to serious dance instruction. Our solution: an intuitive tool for casual dancers who want immediate, real-time feedback on their movements. The application focuses on providing accurate, real-time guidance while maintaining the playful and creative spirit that drives viral dance content.

The important use-case requirements were real-time performance, accurate pose tracking, clear user feedback, and intuitive 3D visualization. To meet these needs, we engineered a system that extracts 2D pose landmarks from MediaPipe, maps them into 3D avatar motion in Unity, and provides real-time scoring and feedback. Developing this system required overcoming significant technical challenges: implementing and testing multiple comparison algorithms (Dynamic Time Warping, Euclidean distance, and Cosine similarity), recreating realistic 3D humanoid movements from noisy 2D inputs, and maintaining low latency for real-time interaction.

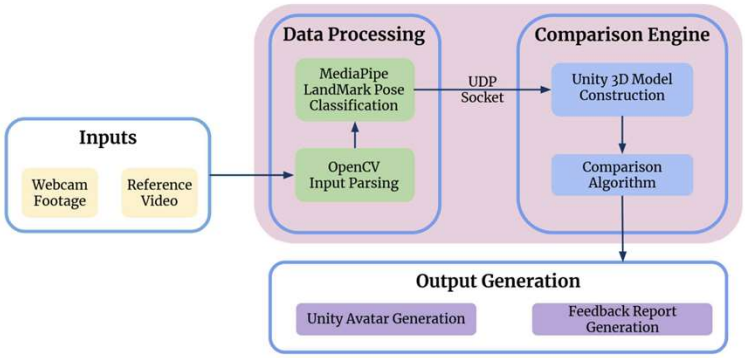
System Architecture

DanCe-V employs a three-layer architecture consisting of:

1. Data Acquisition Layer: Captures and processes raw video data.
2. Analysis Layer: Performs comparative analysis between user and reference movements.
3. Presentation Layer: Delivers feedback through visual representations and reports

The complete system operates through the following sequence of operations: 1. The user selects a reference dance video from the library or uploads their own. 2. If new, the reference video is processed through OpenCV and MediaPipe to extract landmark data. 3. When practice begins, the user's webcam activates and begins processing their movements in real-time. 4. Normalized pose data from both sources is compared using the single-frame algorithm. 5. Real-time feedback is provided through the Unity visualization. 6. Throughout the session, movement data is recorded for post-session analysis. 7. Upon completion, the multi-frame DTW analysis runs, generating the detailed feedback report. 8. Results are stored in the user's profile to track long term progress.

This architecture enables DanCe-V to provide both immediate guidance during practice and deeper insights afterward, creating a comprehensive learning experience that approaches the benefits of personal instruction while requiring only standard consumer hardware.



Conclusions & Additional Information

Our system successfully achieved the primary goals we set at the beginning of the semester: to create an accessible, intuitive, and real-time dance coaching tool using only a webcam and a computer. Through careful algorithm selection, system optimization, and extensive testing, we met all major performance requirements, including processing speed, tracking fidelity, scoring accuracy, and real-time feedback responsiveness.

There are several promising directions for further development. Future work could focus on optimizing a full-body physics model to more accurately recreate natural human motion in the avatar, especially for complex and dynamic dances. Enhancing the robustness of depth estimation would allow better recognition of intricate forward and backward movements. Additional feedback types could be implemented by utilizing custom mesh deformations or visual cues to highlight incorrect body parts in real time. Expanding the system to support mobile platforms would make dance training even more accessible. Multiplayer or collaborative dance modes could also be introduced to enhance engagement. Together, these improvements would significantly expand the system's versatility and create new opportunities for both casual and serious dancers.

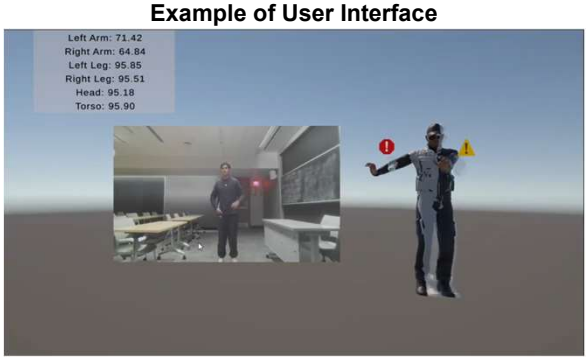
System Description

On the Unity side, we developed a complete user interface that includes a scoreboard, a main menu, and a reference video selection screen. Users select a reference dance video (.mp4), which is then sent to the Python backend for landmark extraction and comparison processing. After analysis, Python returns results such as matching scores and pose deviation feedback to Unity, where it is displayed in real time alongside the reference and user avatars.

The avatars in Unity are rigged 3D humanoid models driven by pose landmark data. We employed physics-based movement using inverse kinematics, joint physics, lerps (linear interpolation), and quaternions to model realistic joint rotations and transitions. Special attention was given to depth modeling, with mathematical conversions applied to reconstruct 3D poses from 2D MediaPipe landmarks. Depth estimation techniques were implemented to infer the missing z-axis data for more natural movement and improved accuracy in avatar posture.

The backend comparison engine, implemented in Python, underwent several algorithms for flexibility and testing. It mainly consists of a FastDTW algorithm with a sliding window approach for detecting time sequence alignments as well as timing differences between the reference video and the live user webcam footage

Example of User Interface



System Evaluation

A two-pronged approach was utilized when evaluating the system to ensure that it met both design specifications as well as use-case requirements. For the design specifications, we focused on testing the UDP packet transmission rate as well as the Unity FPS during operation.

Regarding the use-case requirements, the system was tested comprehensively for both false positive and false negative feedback. In addition, users were asked to learn unfamiliar dances on the system, which helped us verify the usability aspect of the product.

Design Trade-Offs (Unity Engine):

Design Aspect	Technique Used	Tradeoff Rationale
Positional Smoothing	**Gradual interpolation of joint positions** using Linear Interpolation (Lerp)	**Prioritizes smooth transitions** over exact pose replication **to reduce jitter from noisy input**
Rotational Stability	**Smooth blending of directional rotations** using Quaternions (rotations in 3D), Spherical Interpolation (Slerp)	**Maintains fluid joint orientation** while sacrificing **some responsiveness to sudden pose shifts**
Anatomical Direction	Manual computation of joint-based body axes based on CV landmarks	**Ensures realistic body alignment** but requires careful directional vector modeling. Provides **consistent root and torso orientation**, but **loses global frame-of-reference awareness**
Blend Control (FK/IK)	Weighted blending between computed, tracked rotations using Forward Kinematics (FK) and Inverse Kinematics (IK)	**Smooths out rotational jitter** but **dampens finer user-intended motion**, uses inverse kinematics for natural joint articulation

https://course.ece.cmu.edu/~ece500/projects/s25-teamc2/

Electrical & Computer ENGINEERING
Carnegie Mellon