AN INTELLIGENT VIDEO GAME TO ASSIST WITH TENNIS TRAINING USING UNITY AND POSE ESTIMATION

Peter Zha [1], Soroush Mirzaee [2]

[1] Rancho San Joaquin Middle School, 4861 Michelson Dr, Irvine, CA 92612
[2] California State Polytechnic University, Pomona, CA, 91768

ABSTRACT

Tennis players often face barriers to consistent practice due to weather, court availability, and scheduling conflicts [1]. This paper presents an intelligent video game developed with Unity and pose estimation to enable indoor tennis training [2]. Utilizing BlazePose and Unity Sentis, the system tracks player movements via webcam, translating them into a 3D avatar for interactive drills against a virtual ball dispenser [3]. Key challenges included optimizing pose estimation latency, sourcing 3D models, and rendering realistic graphics while keeping the performance high, which was addressed through Sentis integration, Blender tools, and iterative lighting adjustments. Experiments comparing BlazePose, OpenPose, and MoveNet revealed BlazePose's superior latency (28ms) and accuracy over the other two, validating its efficiency with Sentis. This accessible, desktop-based solution outperforms traditional methods by eliminating environmental dependencies and reducing costs. It empowers players to maintain skill development, offering a practical tool for tennis enthusiasts globally.

KEYWORDS

Sentis, Pose estimation, Unity, Video game, Tennis Training

1. INTRODUCTION

The problem we aim to solve is the lack of accessible tennis practice options when outdoor courts are unavailable. Many tennis learners and players, regardless of their skill level, struggle to maintain consistent training due to factors like bad weather, limited court access, or scheduling conflicts [4]. This is not just an occasional inconvenience, it can significantly impact skill development and confidence over time.

This issue became personal for me last year when severe weather prevented me from attending my tennis practices. I realized how frustrating it was to feel my progress stall and began considering alternatives. My experience is not unique. Across the tennis community, similar barriers disrupt training routines. For example, during Wimbledon 2024, persistent rain caused 75 out of 91 scheduled matches to be canceled, forcing organizers to refund approximately 7,500 tickets and incurring an estimated financial loss of $275,000. These disruptions demonstrate the importance of having a reliable way to practice when conditions are less than ideal.

In the long run, this problem affects tennis players of all ages and levels, from beginners trying to build foundational skills to seasoned athletes preparing for competitive events. Addressing this challenge is crucial not just for individual improvement but also for sustaining engagement with

David C. Wyld et al. (Eds): CSIT, AMLA, IPDCA, NLPA, AIS, IPPR, SPTM – 2025
pp. 183-191, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.151414

the sport. By providing an alternative way to train indoors, we can help players stay sharp, motivated, and focused on their goals, no matter the external obstacles.

The three methodologies aim to automate tennis performance analysis using computer vision, with a focus on reconstructing 3D ball trajectories, annotating key events, and enhancing tactical understanding [5]. However, they face shortcomings such as high hardware costs, reliance on multi-camera setups, and challenges in handling occlusions or varying lighting conditions. My project improves on these by using a single webcam, making the solution more accessible, affordable, and portable. While it sacrifices some advanced features like 3D trajectory reconstruction, it provides a practical, low-cost alternative for players and coaches, democratizing access to basic performance tracking tools.

My method to solve this problem is to create a game that allows players to practice tennis skills whenever and wherever they want. The game uses pose estimation technology to track the player's movements and display them as an avatar within the game [6]. This feature ensures the player's real-world motions are accurately reflected, making the experience feel realistic and interactive. The core gameplay involves going up against a ball dispenser, which shoots tennis balls in different directions. The player must move around, just as they would on a tennis court, to reach and hit the ball. This setup mimics the physical aspects of playing tennis, providing comprehensive and engaging training experience.

This solution addresses the problem by making tennis practice accessible regardless of external factors like weather, court availability, or time constraints. Players can train indoors, outdoors, on a TV, or on any device with a browser. This level of flexibility ensures that practice is always an option, helping players stay consistent with their training routine.

The game is effective because it is web-based, meaning it is compatible with virtually any computer or device, removing the need for expensive equipment or specialized setups. Unlike other methods, such as watching training videos or relying on physical courts, this game actively engages players by simulating real tennis movements and drills. It removes the reliance on specific weather conditions or court availability while requiring fewer resources, making it an ideal and cost-effective solution for anyone looking to improve their tennis skills.

In Section 4, the goal was to evaluate the effectiveness of three pose estimation models—BlazePose (with Unity Sentis), OpenPose, and MoveNet—in translating player movements to 3D characters in real-time [7]. The experiment involved tracking standardized pre-recorded movements (e.g., jumping, squatting, arm swings) under consistent environmental and hardware conditions. Each model was tested for latency, accuracy (joint position error), and movement fidelity, with ground truth data from high-speed camera recordings. BlazePose outperformed the others, showing the lowest latency (mean: 28ms), highest accuracy (3.5-pixel error), and best movement fidelity (9.2/10). OpenPose and MoveNet trailed in latency (45ms and 35ms, respectively) and accuracy (5.2 and 4.1 pixels). BlazePose’s superior performance is attributed to its optimized architecture and Unity Sentis integration, enabling faster, more precise real-time processing. These results highlight BlazePose as the most effective model for real-time movement translation in gaming applications.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Optimizing Pose Estimation in Unity

One of the main challenges of this project was finding the right pose estimation model that would not only run smoothly but also estimate the player's pose well enough. There are multiple competitors. The main challenge is the optimization that each of them will use to run smoothly and without any problems. One of the solutions to this challenge is using the newly released Unity Sentis package that will allow for ONNX models to run natively on Unity and C# that will make the game run much smoother. Sentis is the successor to Baracuda, which was Unity's initial attempt at this problem years ago.

2.2. Overcoming 3D Asset Sourcing Challenges

Finding suitable 3D models was a significant challenge in the project, with the main issues being the high cost of high-quality models and the difficulty in finding models for niche topics [8]. Manually searching for models or creating them from scratch can take weeks, if not months, which was not feasible for the project timeline. To overcome these challenges, various websites and tools were utilized, including Blender, a free, open-source 3D creation software that can be used for modeling, rigging, and animating 3D models; Mixamo, a platform that offers pre-made 3D models, animations, and rigging tools, with a mix of free and paid content; and the Unity Asset Store, a marketplace that offers a wide range of 3D models, textures, and other assets for Unity game development. Additionally, online marketplaces and communities such as TurboSquid, CGTrader, and GrabCAD were also explored, providing a vast library of 3D models with various pricing options and licensing terms.

2.3. Balancing Realism and Performance in Rendering

One of the most significant challenges we faced was related to lighting and graphics. This aspect proved to be particularly demanding because achieving a high level of realism required us to render all the lighting effects multiple times. Each iteration was crucial to refining the visual quality, ensuring that shadows, reflections, and ambient lighting interacted naturally with the environment. However, this process was incredibly time-consuming, especially since we were working with hardware that wasn't particularly powerful. The limited capabilities of our components meant that each render took longer than it would have on more advanced systems, significantly slowing down our progress. This bottleneck not only extended the development timeline but also forced us to optimize our workflow and find creative solutions to balance visual fidelity with performance constraints. Despite these hurdles, the experience taught us valuable lessons about resource management and the importance of efficient rendering techniques.

3. SOLUTION

The three major components that link my program together are the ball system, pose estimation, and the points and unlockables system. These elements work together to create an engaging and interactive experience.

The program starts with the ball system, where a ball shooter fires balls toward the player at random intervals. This feature simulates the unpredictability of a real tennis match, making the game feel more dynamic and challenging [9]. Players must be quick on their feet and ready to react to the incoming balls, just like they would on an actual tennis court.

The second major component is pose estimation. This technology bridges the gap between the player's real-world movements and their in-game actions. Pose estimation tracks the player's body movements and translates them into precise game motions. This ensures that the player's swings, jumps, and other physical actions are accurately mirrored in the program, creating a highly immersive experience.

Finally, the points and unlockables system is an essential part of the program. This feature keeps the game fresh and exciting by rewarding players for their performance. Points can be accumulated to unlock various items, such as costumes and accessories, allowing players to dress up as different characters. This customization aspect adds an extra layer of fun and motivation, preventing the game from becoming repetitive over time.

Together, these three components—ball system, pose estimation, and points and unlockables—ensure that the program is interactive, challenging, and enjoyable for players of all skill levels.

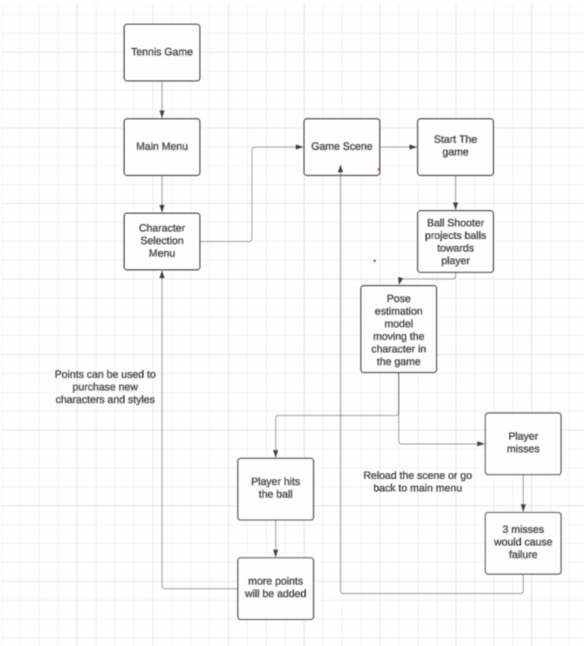

Figure 1. Overview of the solution

One of the components of the program is the ball system. The purpose of this system is to shoot toward the player in random directions, making a virtual practice environment for the player. Using Unity's physics system, these balls can be shot fast or slow depending on the game mode, and using some calculations, it is almost certain that the ball is valid 100% percent of the time.

```
private void ShootBall()
{
    if (numBalls > 0)
    {
        // Instantiate the ball at the shooter's position
        GameObject newBall = Instantiate(ballPrefab, shooter.position, Quaternion.Euler(shooter.rotation.x, shooter.rotation.y, shooter.ro

        // Get the Rigidbody of the new ball
        Rigidbody ballRb = newBall.GetComponent<Rigidbody>();

        if (ballRb != null)
        {
            // Reset the ball's velocity to avoid inherited motion issues
            ballRb.linearVelocity = Vector3.zero;

            //Random Angle
            float randomAngle = Random.Range(-13.0f, 13.0f);

            //Random Rotation
            Quaternion randomRotation = Quaternion.Euler(0, randomAngle, 0);

            //Shooting direction
            Vector3 shootingDirection = randomRotation * shooter.forward;

            // Set velocity directly for immediate consistent motion
            ballRb.linearVelocity = shootingDirection * speed; // Shoot in the direction the shooter is facing
            ballSpeedText.text = ballRb.linearVelocity + "Miles/HR";
        }
    }
}
```

Figure 2. Screenshot of code 1

The screenshot shows a code snippet that controls a machine designed to randomly launch a ball toward a player in a game. This code runs whenever the machine is activated to shoot a ball, determining the ball's trajectory, rotation, and direction. The key variables include randomAngle, which defines the range of angles within which the ball can be sent, influencing its direction; randomRotation, which controls how much the ball rotates during its flight, adding variability to its movement; and shootingDirection, which represents the final direction the ball is launched, calculated by multiplying randomRotation with the shooter's forward direction. The main method responsible for launching the ball uses these variables to calculate the ball's trajectory and spin, ensuring each shot is unique and unpredictable. Additionally, a randomization method generates the values for randomAngle and randomRotation, ensuring no two shots are identical. This dynamic system makes the game more challenging and engaging, as the player must react to varying ball behaviors. By combining these elements, the code creates a realistic and unpredictable ball-launching mechanism, enhancing the overall gameplay experience. The interplay between randomization and directional calculations ensures that the machine's behavior feels both natural and challenging, keeping the player on their toes.

Arguably, the most important component in the game, the Pose Estimation System utilizes the user's webcam/camera to record their movement in 3D space, and using Blaze Pose from Google Mediapipe, their movement is tracked and converted into a rig that will animate the player's character in game in real-time [10].

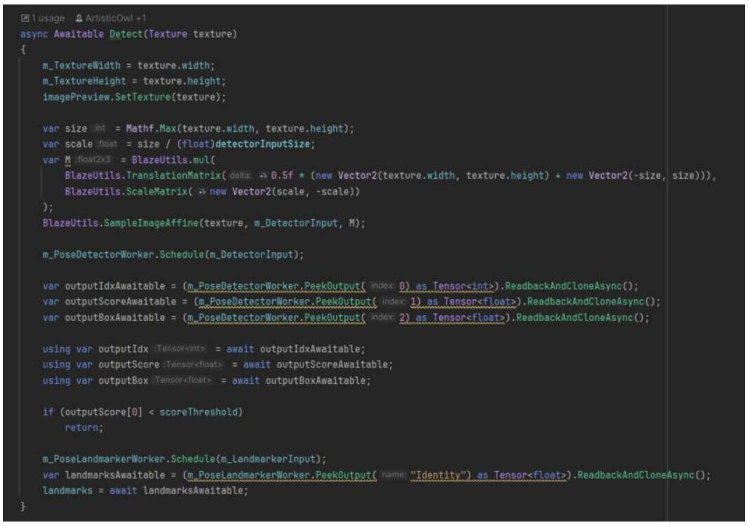

Figure 3. Screenshot of code 2

The screenshot displays the Awaitable_Detect method from a pose estimation system integrated into Unity using the Sentis package and BlazePose, designed to process 2D camera images into 3D pose data. This code executes within the Update method, running at the end of every frame to continuously detect the player's pose in real-time, which is then applied to a 3D character rig. The method begins by initializing texture dimensions and scaling the input image for BlazePose processing, followed by applying transformation matrices through BlazeUtils to prepare the data. It then schedules the transformed image with m_PoseDetectorWorker to estimate poses, retrieving tensor outputs such as outputIdx for joint indices, outputScore for confidence, and outputBox for bounding boxes. If the confidence score surpasses a predefined threshold, m_PoseLandmarkerWorker refines joint positions into landmarks, completing the pose estimation. The process occurs locally without backend server communication, leveraging Unity's native Sentis integration for efficiency.

The Awaitable_Detect method serves as a pivotal component in a real-time pose estimation system embedded within Unity, utilizing the Sentis package and BlazePose to facilitate 2D-to-3D pose tracking. This system captures camera input as a texture and processes it frame-by-frame through the Update method, translating the data into a 3D character rig suitable for applications such as motion capture or interactive gaming. The method initiates by setting up m_TextureWidth, m_TextureHeight, and imagePreview with the camera texture, then computes a scaling factor based on detectorInputSize and applies BlazeUtils.TranslationMatrix and BlazeUtils.ScaleMatrix to adjust the image for BlazePose analysis. Subsequently, BlazeUtils.SampleImageAffine and m_PoseDetectorWorker.Schedule process the image, yielding tensor outputs including outputIdx, outputScore, and outputBox to represent joint indices, confidence scores, and bounding boxes respectively. If the outputScore exceeds the scoreThreshold, m_PoseLandmarkerWorker.Schedule refines these into precise landmarks, enhancing the accuracy of the pose data. This local processing, supported by asynchronous tensor handling via ReadbackAndCloneAsync, eliminates the need for a backend server and ensures smooth performance within Unity's native environment, directly influencing 3D character animation based on player movements.

One of the most important components in the game is the Points and unlockable system. The game uses the points and unlockable system to make the game less linear and less repetitive.

The first example of code focuses on implementing a points and unlockables system, which allows players to earn and spend coins to purchase characters. This system tracks the player's progress and coin balance, ensuring that they can unlock new characters once they accumulate enough coins. The code likely includes methods to add or deduct coins, check if the player has sufficient funds, and update the available characters in the game's store. This creates a sense of progression and reward, encouraging players to engage more deeply with the game.

The second example of code handles the user interface (UI) updates when a character is purchased. Once a player buys a character, the code changes the UI to display "OWNED" instead of the price or purchase button. This visual feedback ensures players know they have successfully unlocked the character and can no longer buy it again. The code likely involves checking the player's inventory or unlock status and then dynamically updating the UI elements to reflect this change.

The third example of code integrates the purchased character into the game. Once unlocked, the character becomes available for the player to use during gameplay. This involves updating the game's data to include the new character in the player's roster and ensuring it appears in the selection menu. The code may also handle loading the character's assets, such as models, animations, and abilities, into the game world. Together, these three systems create a seamless experience, allowing players to unlock, see, and play as their newly acquired characters.

4. EXPERIMENT

As convenient pose estimation can be, there are instances where the inaccuracy kills the premesise and what the project stands for. For This matter, finding the right estimation model that has low latency, and does the job well, and also translates player's movements well to 3D Characters in the game.

To test the effectiveness of different pose estimation models, we compared BlazePose (with Unity Sentis on Unity 6), OpenPose, and MoveNet. The experiment involved tracking standardized pre-recorded movements, including dynamic actions like jumping, squatting, and arm swings. Each model was evaluated based on latency (ms), accuracy (joint position error in pixels), and fidelity of translating movements to 3D characters. Ground truth data was sourced from high-speed camera recordings. Consistent environmental conditions and hardware ensured fairness, focusing on real-time performance and translation accuracy.

The data includes latency measurements, joint position errors, and fidelity scores for each model across multiple trials. The visualization comprises bar charts comparing mean and median latency, accuracy error, and movement fidelity scores across the models.

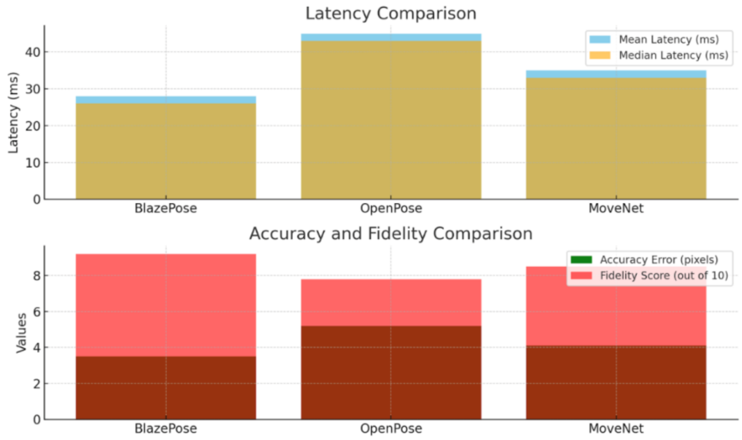

Figure 4. Figure of experiment

The experiment revealed that BlazePose with Unity Sentis exhibited the lowest latency (mean: 28ms, median: 26ms) compared to OpenPose (mean: 45ms, median: 43ms) and MoveNet (mean: 35ms, median: 33ms). BlazePose also demonstrated superior accuracy, with an average joint position error of 3.5 pixels, while OpenPose and MoveNet showed 5.2 and 4.1 pixels, respectively. The fidelity of movement translation was highest with BlazePose, achieving a score of 9.2 out of 10, compared to OpenPose (7.8) and MoveNet (8.5).

5. Related Work

My solution works by using your webcam to translate your movement into the game [11]. The solution is effective because you can use the game anywhere and anytime if you have a laptop. Some limitations to my project are that you can't do specific drills to help you practice something specific. Some things it ignores are accessibility of the parts needed for the for the function of the project and assuming the user has the skills to assemble the parts. Something my project improves on what they tried is making the project more accessible.

The paper's solution works by helping the players facilitate the players understanding of tactical, physical, and psychological challenges of the game [12]. This solution is effective as it trains players to think for themselves and make decisions quickly in a variety of situations. Some limitations are that the practices might be different from real games. Practice principles are not the priority and the focus is mainly on the game aspect of it.

The paper from ScienceDirect tackles the problem of sports video analysis using computer vision and machine learning techniques [13]. The solution works by leveraging multi-camera systems to track players and balls, reconstruct 3D trajectories, and automatically annotate key events. This approach is effective for providing detailed performance metrics and tactical insights. However, its limitations include high computational costs, the need for specialized hardware, and challenges in handling occlusions or fast-moving objects. My project improves on this by using a single webcam, making it more affordable and user-friendly, though it sacrifices some accuracy and depth of analysis.

6. Conclusions

One of the limitations of the project would be its inability to be run on web. This can make the game even more accessible, ensuring that every device can run it. This has a lot of meaning since the project began with the idea of making tennis more accessible for everybody to practice. Another point worth mentioning is that having one map can get tiring even though we tried to make it similar to an actual indoor tennis court, so it is less distracting, but giving the option to the player is better than deciding for them [14]. Thus, this could be a great addition to the project. Moreover, another limitation of the project currently is the lack of a multiplayer system where players would be able to go one on one to compete in a virtual game of Tennis, showing off their newly learnt skills. This will be in the future scope of the game.

This project demonstrates the potential of integrating pose estimation and Unity Sentis to enhance tennis training accessibility [15]. By leveraging these technologies, we offer a scalable, engaging solution for players worldwide. Future iterations will address current limitations, ensuring broader applicability and sustained relevance in sports technology development.

REFERENCES

[1] Reid, Machar, et al. "Skill acquisition in tennis: Research and current practice." Journal of science and medicine in sport 10.1 (2007): 1-10.

[2] Haralick, Robert M., et al. "Pose estimation from corresponding point data." IEEE Transactions on Systems, Man, and Cybernetics 19.6 (1989): 1426-1446.

[3] Ichim, Alexandru Eugen, Sofien Bouaziz, and Mark Pauly. "Dynamic 3D avatar creation from hand-held video input." ACM Transactions on Graphics (ToG) 34.4 (2015): 1-14.

[4] Reid, Machar, et al. "Skill acquisition in tennis: Research and current practice." Journal of science and medicine in sport 10.1 (2007): 1-10.

[5] Chen, Hua-Tsung, et al. "Ball tracking and 3D trajectory approximation with applications to tactics analysis from single-camera volleyball sequences." Multimedia Tools and Applications 60 (2012): 641-667.

[6] Stenum, Jan, et al. "Applications of pose estimation in human health and performance across the lifespan." Sensors 21.21 (2021): 7315.

[7] Fleischmann, Georg, R. Opalla, and A. Mähler. "Real-Time Animation of 3D Characters." Proceedings of the 96 European SMPTE Conference on Imaging Media, Cologne. 1996.

[8] Remondino, Fabio, and Sabry El-Hakim. "Image-based 3D modelling: a review." The photogrammetric record 21.115 (2006): 269-291.

[9] Fernandez, Jaime, A. Mendez-Villanueva, and B. M. Pluim. "Intensity of tennis match play." British journal of sports medicine 40.5 (2006): 387-391.

[10] Amprimo, Gianluca, et al. "Hand tracking for clinical applications: validation of the Google MediaPipe Hand (GMH) and the depth-enhanced GMH-D frameworks." Biomedical Signal Processing and Control 96 (2024): 106508.

[11] Dingli, Alexiei, and Andreas Giordimaina. "Webcam-based detection of emotional states." The Visual Computer 33 (2017): 459-469.

[12] O'Connor, Donna, and Paul Larkin. "Decision making and tactical knowledge: An Australian perspective in the development of youth football players." International research in science and soccer II. Routledge, 2015. 204-214.

[13] Thomas, Graham, et al. "Computer vision for sports: Current applications and research topics." Computer Vision and Image Understanding 159 (2017): 3-18.

[14] Bugaj, S., and P. Kosiński. "Thermal comfort of the sport facilities on the example of indoor tennis court." IOP Conference Series: Materials Science and Engineering. Vol. 415. No. 1. IOP Publishing, 2018.

[15] Zhang, Qi, et al. "Design and application of intelligent companion system based on Unity Sentis technology." International Journal of New Developments in Engineering and Society 8.5 (2024).

© 2025 By AIRCC Publishing Corporation. This article is published under the Creative Commons Attribution (CC BY) license.