Real-time Hand Pose Tracking for Gaming Applications

Rain Juhl
Stanford University
rjuhl@stanford.edu

Abstract

This paper builds on hand segmentation technologies, such as Google's Mediapipe [3], to recognize hand poses in a fraction of a millisecond. The methodology deviates from traditional approaches that use cropped images fed into a CNN [2] to classify images by exploring feature vectors composed of hand articulation points in conjunction with a relatively small MLP. This leads to faster training and inference times. Furthermore, the paper explores how to track poses through time and space and map them to predefined motions by subsampling the palm locations and comparing them against previously crafted score maps. Together, these technologies allow hand pose and motion to be tracked in real-time for gaming applications. A demo of its use can be found here.

1. Introduction

Hand-tracking technology has a plethora of real-world applications, ranging from live ASL translation to teleoperation to healthcare. One of this technology's most captivating applications lies in human-computer interaction, particularly for gaming. This technology enables the development of games where players use their hand movements and poses to control the games' core mechanics. While prevalent in virtual reality (VR) games, this concept can be extended to traditional gaming using a computer's camera and sophisticated image processing techniques.

However, the implementation of this technology in gaming is not without challenges. Accurately identifying the hand, its pose, and its motion in real-time, with high precision, is a complex task. Other potential barriers include the cost and the time required to implement and train models. Developing robust deep neural network models necessitates substantial data, training time, and financial investment–all of which are critical factors in determining a project's feasibility.

To address these challenges, this paper presents the development of a game demo inspired by the "Avatar: The Last Airbender" series, where characters perform element-bending actions through specific body and hand movements. This fictional universe provides an ideal backdrop for a game that interprets player interactions through hand gestures. The demo processes hand pose data to associate it with an elemental action and then maps the motion to a corresponding moveset. Players can execute various elemental attacks, blocks, or special moves using distinct hand poses and motions. Initially supporting fire and earth elements, the system is designed to be expansively adaptable for additional moves or different game concepts.

Specifically, the game needs the following characteristics. It must find and classify hand poses with a high degree of accuracy. While a hand pose is held, it needs to track it and then be able to identify different paths from each other through a score. Finally, it needs to do this all in real time on a CPU. This requires processing each frame within 33 milliseconds to achieve over 30 frames per second (FPS). The demo accomplishes this through a pipeline that extracts hand articulation points using Google's Mediapipe. Next, it inputs this data into a compact MLP with 10,435 parameters and employs a predefined score map to evaluate different motion paths.

2. Related Works

2.1. MobileNet

CNNs have established themselves as formidable tools for image segmentation and classification. However, their computational demands, particularly for convolution operations, can be a significant hurdle for real-time applications on CPUs. MobileNet, a model designed to address this challenge, introduces the innovative concept of depth-wise separable convolutions, which supplant traditional convolutional layers. This approach bifurcates the convolution into a lightweight depth-wise operation for spatial filtering and a 1x1 convolution for feature generation, substantially reducing computational complexity [1].

Subsequent iterations, MobileNetV2 and MobileNetV3, have built upon this foundation. MobileNetV3 represents the apex of this series to date, integrating the most effective techniques from the developers' research. It uses an

inverted residual structure paired with linear bottlenecking for more efficient implementation and leverages the intuition that the necessary information lies in the bottleneck layers and can be distorted by the nonlinearities of the network. Additionally, the integration of lightweight attention mechanisms propels MobileNetV3 to the forefront of performance in its class [2].

Overall, MobileNet outperformed other similar models in calcification, object detection, and segmentation tasks. Despite these advancements, MobileNetV3 grapples with limitations when deployed in ultra-high-speed contexts. For instance, with over 2 million parameters, MobileNet's inference time can exceed 43 milliseconds, insufficient for tasks requiring 30 FPS, such as hand pose detection, without even accounting for motion analysis. Moreover, while these models offer impressive capabilities in classification, object detection, and segmentation, they demand considerable fine-tuning on specific datasets and entail substantial training costs, posing additional challenges for real-time applications.

2.2. Google MediaPipe Hand Tracking

Google researchers have developed a component within MediaPipe, known as MediaPipe Hands, which is adept at on-device, real-time hand tracking. The architecture of this model is discussed in their paper "MediaPipe Hands: On-device Real-time Hand Tracking." [3] The model processes images in two steps: first locating the palm and then finding 21 hand articulations for the full hand.

In the first phase, the focus on palm detection simplifies the task for the model, as locating the palm is less complex than identifying the entire hand. This approach reduces the number of anchors required by a factor of five. An encoder-decoder feature extractor is utilized in this step to enhance context awareness, enabling the model to effectively distinguish between hands in the foreground and background [3].

The second phase leverages the output from the initial step, cropping the first output to predict twenty-one 2.5D points, determine handedness, and assess the likelihood of hand presence. It does this using convolutional operations for the feature extraction [3].

The performance metrics of the model are impressive, boasting a precision rate of 95.7% and a mean squared error, normalized by palm size, of 13%. Notably, the model achieves an inference time of 7.5 milliseconds on mobile devices, with a reduced model attaining an even more remarkable 1.1 millisecond [3]. This efficiency facilitates fast and accurate hand articulation predictions in real time.

While MediaPipe Hands significantly enhances hand pose detection, it doesn't inherently classify hand poses nor is it readily fine-tuned. Nonetheless, its utility as a feature extractor is undeniable. By channeling its output into a simplistic classifier, such as a small MLP, it can maintain rapid inference speeds. This synergy ensures that image processing can comfortably exceed 30 FPS.

3. Methodology

To address the intricate task of detecting hands, classifying their poses, and tracking their movements, a methodical three-step process is employed. In the initial stage, each frame undergoes analysis using Google's MediaPipe Hand to detect the presence of a hand. Upon successful hand detection, the subsequent step involves transforming the output into a feature vector, which is then processed by a four-layer multilayer perceptron (MLP). This MLP is tasked with classifying the hand's pose into one of three categories: earth, fire, or no pose.

Following pose classification, if an earth or fire pose is identified, the system records the motion details for subsequent analysis. This motion data becomes crucial when there is a transition from an earth or fire pose to another pose. At this juncture, the accumulated motion data is evaluated against a score map, which draws inspiration from template filtering techniques [5]. This evaluation generates a score for each predefined motion template. When the highest score surpasses a predetermined threshold, the system triggers a corresponding move.

3.1. The MLP Model

3.1.1 Model Design

The hand pose classifier is an MLP model with 10K parameters. It consists of 4 feed-forward linear layers, three of which are followed by a ReLU activation layer. The initial input layer is designed to accommodate 63 features, which encapsulate the spatial information of the hand. Specifically, these features are the x, y, and z coordinates corresponding to the 21 joint articulations captured by MediaPipe, effectively representing the hand's three-dimensional structure. The final layers output three values that correspond to the probability of each of the three classes respectively.

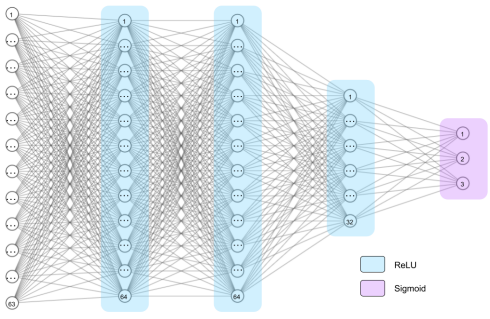

Figure 1. Hand Pose Classifier

3.1.2 Training the Model

The model was trained on a processed version of the ASL(American Sign Language) Alphabet Dataset [4]. Co-opting ASL letters for game hand poses opened a vast amount of data that was already collected. The ASL sign for X became the sign for fire and the sign for E became earth. The remainder of the dataset was labeled as no pose, which includes the other letters, nothing, and random hands. To prepare the data for training, each hand image was processed through MediaPipe to extract 63-dimensional feature vectors, capturing the x, y, and z coordinates of the 21 hand articulations. To enhance the model's resilience to various hand orientations and sizes, the data underwent normalization and augmentation. Specifically, random 3D rotations (up to a maximum of 45 degrees along any axis) were applied ensuring robustness against rotational variances. Twelve augmented data points were created from each fire and earth labeled data point and 1 from the rest. Note that the data was split into 70%, 15%, and 15% for the train, validation, and test set respectively. The model was then trained for 100 epochs with Adam as an optimizer, a multiplicative learning rate schedule, and weighted cross-entropy loss. The weights were 2.0 for fire and earth and 1.0 for no pose. This weighting strategy was designed to bias the model toward predicting elemental poses, aligning with the game's interactive dynamics, where recognizing an elemental pose is more critical than identifying the absence of a pose, especially given the predominance of "no pose" data in the training set.

3.2. Score Maps

The game requires the definition of specific motions corresponding to each move, necessitating a mechanism to recognize and classify these motion paths. Template matching, a technique traditionally utilized in image analysis, provides a foundation for this functionality [5]. This method involves convolving a template image with a target image to assign scores to various regions, subsequently using a threshold score to identify the presence of the template's pattern.

Adapting this concept for motion path recognition, the game translates motion data into a quasi-image format, allowing the application of template matching principles. Each predefined motion in the game is represented as a template in this space. As a player performs a motion, it is transformed into this format and compared against the set of templates using a convolution-like operation. A score is generated for each comparison, reflecting the degree of similarity between the player's motion and the predefined templates. When a player's motion generates a score that exceeds a predetermined threshold for a specific template, the game recognizes this as the execution of the corresponding move.

3.2.1 Generation

The first step in scoring the motion is to create the score maps. The game has a built-in function to take line drawings and convert them into score maps. The line drawing is resized w.r.t the aspect ratio to the score map size and then binarized. From there a Gaussian blur is applied. The new image is then normalized and the center of mass (COM) is computed. Finally, all black pixels are converted to the negative number such that the entire image will sum to zero. This penalizes extra unnecessary movement in a path. The final image and the COM is saved as a score map. Figure 2 has three examples of score maps that are used in the game demo.

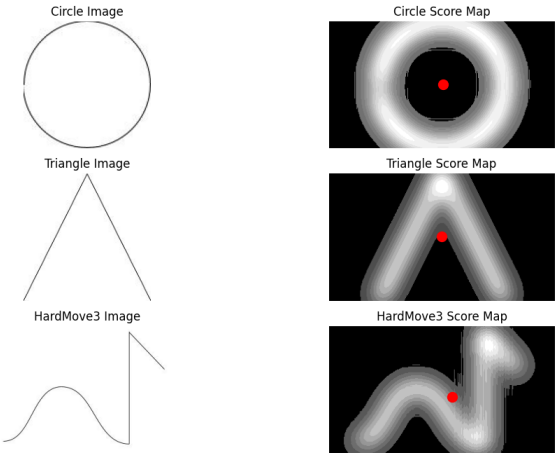

Figure 2. Generated Score Maps

3.2.2 Evaluation

To accurately compare a player's motion with the established score maps, the extraction and transformation of the motion into a comparable image format that is scale-invariant and resilient to translation, is crucial. To accomplish this, a motion vector of xy-coordinates is first converted to a binary image. It is then cropped and resized to the score map size w.r.t its aspect ratio. To ensure the motion will overlap properly with the score map, its COM is computed and translated such that its COM overlaps with the COM of the score map. The two are multiplied together and summed for a score for each score map. If the highest score surpasses a predefined tolerance threshold, the corresponding motion path is identified as the player's action.

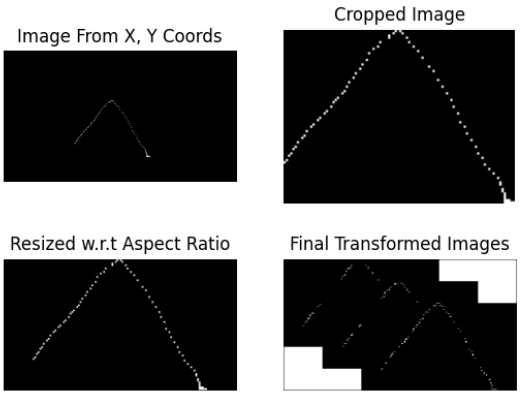

Figure 3. Coordinates to Image Steps for Evaluation
Input comes real hand movement

4. Results

Preliminary qualitative feedback from user trials suggests that the demo offers a natural and intuitive user experience, with users able to consistently execute moves after a period of acclimatization. Participants appreciated the novel gaming experience, affirming that the core mechanics were well-implemented and engaging. However, they also noted occasional misclassifications of hand poses or motions, an observation corroborated by personal experiences with the demo. These instances, while not predominant, highlight areas for refinement. The overall positive reception underscores the demo's potential as a viable proof of concept for integrating hand motion as a central gameplay element; however, it's essential to examine game components both qualitatively and quantitatively.

4.1. Hand Pose Classifier Results

During training, the MLP achieved a loss of 0.8 and a balanced accuracy greater than 99.99%. However, to understand how it generalizes, we must look at how the model performed on the test set. The accuracy for each class is in Table 1.

Class	Test Set Accuracy
Fire	0.9984
Earth	0.9997
Other	0.9997

Table 1. Hand Pose Model Class Accuracy (Test Set)

As evident by the table, the model does an exemplary job of predicting all classes. Furthermore, the model boasts an inference time of **0.0084ms** on CPU Intel i-7, which can be easily processed in the 34ms window. In practical applications, the model performs commendably. To further refine the user experience and mitigate abrupt changes in predicted classes, implementing a smoothing mechanism proves beneficial. Stability and accuracy in real-time class predictions are enhanced by requiring a sequence of more than 10 consecutive predictions in the same class before acknowledging a switch.

4.2. Score Map Results

To assess the efficacy of the score maps within the system, a foundational test was conducted focusing on two of the three movements incorporated in the demo: a circular motion and a triangular motion without the base. This test involved scoring a perfect circle and an ideal triangle without the base against the corresponding score maps to verify the accuracy and reliability of the scoring pipeline.

Shape	Circle Score	Triangle Score	Move Three Score
Circle	1800	-21	209
Triangle	209	1474	312

Table 2. Hand Pose Model Class Accuracy (Test Set)

The test results for the circle and triangle inputs underscore the effectiveness of the score maps in distinguishing between different motions. Specifically, the circle input yielded a score approximately 9 times higher on the circle score map than on any other; while the triangle input achieved a score roughly 5 times higher on the triangle score map compared to the next highest score. These results affirm that, under optimal conditions, the score maps excel in accurately identifying and differentiating motions.

Additionally, visual representations corroborate the system's efficacy. In these tests, users' motions were not only displayed but also scored in real time, akin to the scenario depicted in Figure 3, providing an interactive and interpretable feedback loop. This approach enabled users to understand and adapt to the scoring system, with the feedback on motion capture being predominantly positive. Such visual and numerical feedback mechanisms enhance the transparency and user engagement of the system, further validating the score maps' utility in real-time applications.

5. Conclusion

The game demo successfully establishes a foundational framework for building interactive game experiences based on hand gestures. Qualitative results from user testing show that users can effectively interact with the game using hand gestures and that the experience is positive. In the future,

more user testing and a structured survey would be beneficial to improve the game demo and overall experience. Moreover, visual representations of the image-scoring process show that motion paths are typically scored reasonably.

Quantitatively, the system demonstrates high inference speed, with classification and detection accuracy surpassing 99%. This allows for precise gameplay with minimal need for result smoothing. Targeted tests on score maps show a clear distinction between intended and alternative motions, with a significant performance differential indicating the system's effectiveness in interpreting hand poses and motions.

Although the game demo shows an effective base blueprint, there are still many areas to explore for improvement. To better understand the performance of the models, future work should focus on expanding metrics for quantifying the performance of score maps. A standardized, expressive, and easy-to-generate metric would allow for many variations to be compared. Another promising direction is exploring neural methods for mapping the motions to scores. This was the original approach, but with a lack of time and data to build a powerful model, score maps were utilized instead. Finally, the hand pose classification MLP model was used because it is quick and easily trainable, but the choice and design are relatively arbitrary. Exploring more architectures, further data augmentation, and testing regularization techniques would make it even more accurate and improve the game experience.

This research underscores the feasibility of real-time hand motion and pose processing on a CPU for game interactions, offering a glimpse into the potential for developing a new genre of gesture-based games. Future work can expand this prototype into a comprehensive gaming experience, further exploring the intersection of machine learning, computer vision, and interactive entertainment.

6. Contrubutions

6.1. Rain Juhl (rjuhl)

- Did all the research
- Wrote the paper
- Coded the Demo
- Recorded the Demo
- Code can be found here
- https://github.com/Rjuhl/AvatarGameDemo

6.2. Kathryn Garcia and Muki Kozikoglu

- Provided user feedback on the game demo

6.3. Kim Swensen and ChatGPT

- Helped with editing the paper. To clarify, Rain Juhl wrote the whole paper but both ChatGPT and Kim Swensen helped fix grammar and re-write for clarity.

References

[1] Bo Chen-Dmitry Kalenichenko Weijun Wang-Tobias Weyand Marco Andreetto Hartwig Adam Andrew G. Howard, Menglong Zhu. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *Google Inc*, 2017.

[2] Grace Chu Liang-Chieh Chen Bo Chen-Mingxing Tan Weijun Wang Yukun Zhu Ruoming Pang Vijay Vasudevan Quoc V. Le Hartwig Adam Andrew Howard, Mark Sandler. Searching for mobilenetv3. *Google AI*, 2019.

[3] Andrey Vakunov Andrei Tkachenka George Sung Chuo-Ling Chang Matthias Grundmann Fan Zhang, Valentin Bazarevsky. Mediapipe hands: On-device real-time hand tracking. *Google Research*, 2020.

[4] Debashish Sau. Asl(american sign language) alphabet dataset, 2022. ASL images for classification.

[5] Adaptive Vision. Template matching, 2017. Template Matching Doc Article.