Predicting Spatial Data of Virtual Reality Hand Tracking Joints with Recurrent Neural Networks*

Gytis Dokšas[1,*,†] and Tomas Blažauskas[1]

[1] *Kaunas University of Technology, Kaunas, Lithuania*

Abstract

Hand tracking in virtual reality (VR) is crucial for enhancing user experience and interaction. However, current hand tracking technologies face challenges such as occlusions due to other devices, lighting variations and lack of direct sensor data access for developers. The study explores the use of recurrent neural networks (RNN), particularly long short-term memory (LSTM) models, to predict spatial data of VR hand tracking joints. A predictive model was created to infer hand movements using tracked data from previous frames, improving robustness and precision. Experimental results demonstrate that even small architecture LSTM models achieve a mean Euclidean distance error of 2.5 millimeters for position predictions and 3.66 degrees for rotation predictions with an inference time of 2.2 milliseconds for both hands. The system can be used to enhance VR interaction without compromising performance.

Keywords

Recurrent neural networks, hand tracking, virtual reality

1. Introduction

Virtual reality (VR) has been growing for the past years – the global VR market was value at approximately 15.9 billion USD in 2024 and it is planned to reach 89.5 billion USD by 2033 [1] which shows a compound annual growth rate for 21.2% during the period. This growth can be explained by advancements in hardware and increase of VR supporting software, making VR more affordable and accessible for users. The range of VR applications is also expanding as it is being used in various industries [2, 3, 4]: automotive, healthcare, retail, tourism, real estate, architecture, interior design, education, recruitment, entertainment, sports and art. During technology analysis it was noticed that Meta Quest 2 system is still the most popular – Steam platforms' 2025 January hardware usage survey data [5] suggests that Oculus Quest 2 is used by 31.67% of users, followed by Meta Quest 3 (23.26%), Valve Index HMD (13.17%), Oculus Rift S (5.65%) and Meta Quest 3S (4.85%).

Even though advancements are made, current VR hand tracking systems still face some challenges [6]. One of major issues is the interference caused by additional devices like haptic gloves. Such devices enhance tactile feedback but can obstruct the headset's cameras, leading to tracking disruptions. Lighting conditions can also impact tracking accuracy. Insufficient or excessive lighting can cause the system to lose tracking of hand movements, resulting in less immersive user experience. Additionally, it is hard to override default hand tracking implementations due to lack of access to direct sensor data from virtual reality headsets. Such access is blocked due to privacy policies and security concerns. Instead, software development kits are provided to allow access to information. Meta CTO Andrew Bosworth suggests that in the future such access might be available [7]. Until then, these problems could be solved by additionally predicting hand joint spatial data

*IVUS 2025: Information Society and University Studies 2025, May 15, Kaunas, Lithuania

[1] Corresponding author.

[†] These authors contributed equally.

gytis.doksas@ktu.edu (G. Dokšas); tomas.blazauskas@ktu.lt (T. Blažauskas)

0009-0004-2250-0767 (G. Dokšas); 0000-0003-2858-328X (T. Blažauskas)

using artificial intelligence-based algorithms.

In this study an approach is provided to enhance VR hand tracking by implementing a predictive model using recurrent neural networks (RNN), particularly long short-term memory (LSTM) networks to enhance tracking when only headsets are used with no controllers or additional devices on hands. Spatial hand joint information, available through SDKs, is tracked for set number of frames and used to predict hand movements in the future. This allows mitigation of tracking interruption caused by hand obstructions and suboptimal lighting conditions. By training the RNN on sequences of hand movement data, the system learns temporal patterns, enabling it to generate accurate prediction even when visual tracking is compromised in virtual reality devices. The approach seeks to improve the robustness of virtual reality hand tracking in VR environments, providing users with more seamless and immersive experience, with minimal impact on the performance. The solution is designed to work with OpenXR based hand tracking, ensuring broad compatibility across multiple VR devices and enhancing its interoperability and availability for developers and users.

2. Related Work

The problem of human motion prediction has been studied by other researchers, particularly in scenarios where limb positions must be inferred over time. Many recent works rely on recurrent neural networks (RNNs) due to their ability to model sequential dependencies, making them well suited for tracking human joint movements. However, existing research primarily focuses on predicting general limb positions rather than detailed hand tracking without focusing on performance, which introduces differences in problem scope and requirements.

Tang et al. proposed a recurrent attention network (RAN) to improve human pose prediction by dynamically focusing on the most relevant spatial-temporal features in motion sequences [8]. This study aimed to enhance pose estimation accuracy by integrating self-attention mechanisms within an RNN based framework. Compared to traditional RNN models that process input sequences uniformly, RAN selectively assigns more weight to the most important frames, allowing models to better capture motion dependencies over time. This method demonstrated improvements in predicting human limb positions but was mainly applied to whole body motion rather than detailed finger tracking. Additionally, even though this study uses real time inference constraints (operating at 30 frames per second), it is insufficient for VR hand tracking applications as it requires lower latency. Furthermore, rotational predictions were not included, limiting method's applicability for hand tracking modelling in VR.

Zhang et al. introduced a physics-informed motion prediction model by integrating Euler-Lagrange equations with recurrent neural network architecture [9]. Study focused on improving long-term motion prediction accuracy by encoding physical constraints to ensure realistic human movement dynamics and trajectories. This approach improved stability and smoothness of predicted motion paths, especially in rapid or complex movements. Compared to current study, it was tested on whole body tracking tasks where minor inaccuracies did not significantly impact realism of movements. In contrast, VR hand tracking requires precise joint level prediction where even millimeter level errors can lead to unnatural movements. Added computational complexity increases inference times, making the approach less suitable for VR applications.

Study by Liu et al. explored a hybrid approach combining recurrent neural networks with Unscented Kalman Filter (UKF) for motion prediction [10]. The UKF was used to refine RNN generated predictions by adding sequential state estimation, which reduced motion noise and smoothed the tracking trajectories. Most significant improvements were noticed when handling erratic motion sequences, making it well suited for human motion tracking. However, the focus was put on predicting upper body joint movements with no direct real-time inference constraints.

Cui et al. proposed a graph based recurrent model for 3D human motion prediction using fully connected spatio-temporal graphs to model dependencies between joints [11]. This approach outperforms traditional RNN models. Study showed robust performance in full body pose forecasting tasks using sports and physical activity datasets. However, the study did not consider rotational

predictions and focused only on position prediction which is insufficient for VR hand tracking since it requires both positional and rotational data to successfully visualize tracked hands. Additionally, graph models are larger and have increased inference delays.

Gamate et al. introduced a hybrid classical-regressive kinematics model designed for continuous 3D hand trajectory prediction during ballistic movements in VR environments [12]. Their approach uses interpretable kinematic equations with regression based coefficient interpolation instead of using deep learning based methods. The authors used OptiTrack motion capture system (8 cameras with 7 markers on participants) in combination with Oculus Quest headsets and Oculus Touch controllers. Participants were provided with 4 VR activities: custom VR task of pointing at specific location and three popular VR games. The tracking setup in the study relies on external body markers and motion capture through the OptiTrack system, focusing on wrist-level positional data rather than fine hand joint tracking. Oculus Quest was used as part of VR interface and not as the main tool for data collection. Additionally, evaluation was done offline during trajectory reconstruction. The proposed solution focuses on using only headset tracking data provided by SDK and is designed for real-time prediction during tracking loss in VR hand tracking scenarios.

Meta's documentation suggests that it is possible to use Fast Motion Mode to allow high-frequency tracking to enhance hand tracking performance in VR [13]. Meta implements predictive algorithms to interpolate motion between frames which allow tracking at 90-120 frames per second. However, the inner implementation and predictive model details are not available for the public. Additionally, even when such predictive models are used, the issues persist if occlusions or poor lighting conditions are present as hand tracking is lost since it requires visibility for most of the joints for algorithms to work. The research suggests that there are possibilities to improve both hand tracking robustness, availability and performance.

3. Methodology

A system was created to perform real-time hand tracking in virtual reality using RNN architectures – gated recurrent unit (GRU) and long short-term memory (LSTM). It contains multiple subsystems: data recording, visualization and prediction inference. To achieve this a hybrid approach with combination of Unity with OpenXR package for VR interaction and visualization is joined with Python based PyTorch framework, used in deep learning models. A comparison of game engines and available packages was performed before choosing the technologies. Unity 2022.3.20f1 was selected due to larger VR device support and wider asset store, compared to Unreal Engine. OpenXR 1.10.0 is used in combination with XR Hands 1.4.1 package. These technologies were selected due to the largest amount of supported VR devices and widely available documentation. Since Unity uses primarily C# while prediction models are implemented in Python, a communication bridge was required. Meta Quest 2 headset was used for development and testing while connected to a computer via cable.

3.1. Integrating Python based prediction with Unity

To establish efficient communication bridge between Unity's C# environment and Python based deep learning models, several approached were tested and evaluated. Each method was assessed based on latency, reliability and ease of integration. The primary goal was to achieve low-latency bidirectional communication, making sure hand tracking predictions could be integrated with Unity VR hand tracking data. Experimentation suggests that Python.NET (pythonnet) package is the most suitable for high performance as it provides the lowest roundtrip latency.

For benchmarking, a test environment was created where a communication method was executed 1000 times. Each execution involved sending a fixed-size message from C# to Python, processing the message in Python by implementing an artificial 10 millisecond delay simulating model inference, sending the response back to C# and recording the round-trip latency using Stopwatch class in C#. The values are averaged, and the first 10 messages are ignored to account for initialization delays. Python Scripting Unity package, user datagram protocol (UDP) sockets, transmission control

protocol (TCP) sockets, Google remote procedure call (gRPC) and Python.NET (pythonnet) were tested. The results are provided in the table below (see Table 1).

Table 1
Comparison of communication technologies between C# and Python systems

Technology	Round-trip latency (ms)	Comments
Python Scripting package	-	Unavailable at runtime
UDP sockets	1.271	4096-byte packages used, can be unreliable, requires additional mechanisms
TCP sockets	1.725	4096-byte packages used, reliable but slower transmission
gRPC	1.866	Open-source, designed to work on multiple platforms
Python.NET	0.008	Allows to directly call embedded Python in Unity, extremely low inference time

Python Scripting package was not tested as it does not allow communication at runtime – it only works for scene setup. UDP sockets provided good results with the second lowest average round-trip latency of 1.27 milliseconds, but it uses an unreliable protocol for data transfer meaning that additional safety mechanisms need to be implemented. TCP sockets have a reliable transmission stream but work slower due to using connection-oriented protocol. gRPC, while having a possibility to seamlessly integrate multiple platforms, had the slowest round-trip latency of 1.87 milliseconds. Python.NET performed especially well since it provides direct bridge between Python and .NET environments providing latency lower than 1 ms (a limit of C# Stopwatch class). This communication method is used throughout the system.

3.2. Data collection and visualization

To create a representative dataset a recording system was implemented. It records data from 26 hand joints, including wrist, palm and individual finger segments (metacarpals, proximal, intermediate, distal joints and fingertips) which results in a total of 676 recorded features per both hands. Each joint information contains position (3D vector), rotation (4D quaternion), linear velocity (3D vector) and angular velocity (3D vector) which was recorded at maximum available tracking frequency while ensuring optimal lighting conditions. In case hand tracking stops during recording, the frames lacking information are removed. The input data was saved as three separate datasets using device space (default VR tracking space), transformed local space using hierarchical tree representation of tracked hand from Unity and transformed space which tracks local distance and orientation of each joint relative to wrist.

Visualization subsystem includes showing the default currently tracked hands, visualization of recorded data and predicted hand joint positions and rotations. Examples of tracked hands and singular predicted hand are provided below (see Figure 1). The user has a possibility to visualize already recorded data by visualizing the recorded hand movements in all available data spaces.

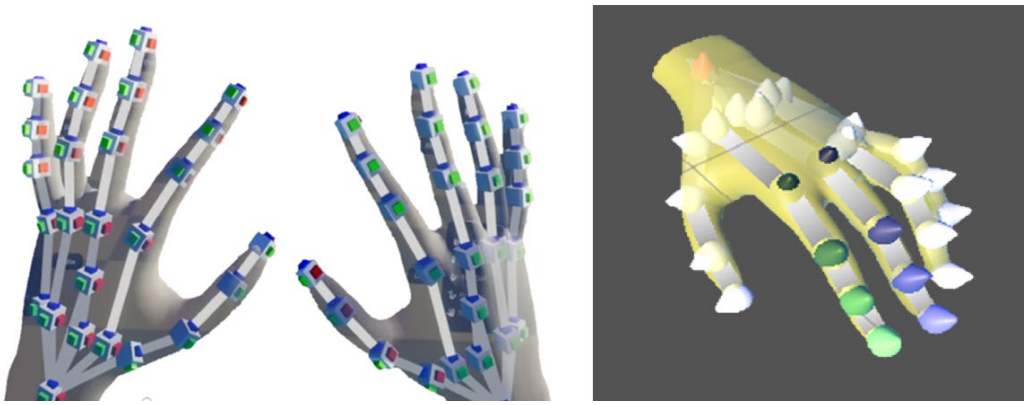

Figure 1: Hand tracking visualization (left) and predicted hand visualization (right)

Visualization allows easier understanding of predicted data and debugging during the development since it is hard to understand spatial data and transformations otherwise.

3.3. Prediction implementation

A dynamic system was developed to allow easy experimentation and modification of used parameters. After similar work analysis it was decided to implement gated recurrent unit (GRU) and long short-term memory (LSTM) architectures. LSTM networks are well suited for capturing long-term dependencies in sequential data due to their gating mechanisms which help mitigate vanishing gradient issues. This makes LSTM networks particularly effective for tracking complex temporal dependencies in motion data. However, their increased number of parameters and computational overhead makes them slightly slower in inference. GRU provides simplified architecture with fewer parameters, combining forget and input gates into a single update gate. This results in faster training and inference times, making GRU a more computationally efficient choice. Additionally, attention mechanisms were implemented for both architectures. These mechanisms allow focusing on relevant time steps, handling long-term dependencies more effectively.

The data was split into training, validation and test datasets (60%/20%/20%). To enhance training process and improve accuracy of hand joint predictions several optimization techniques and custom implementations were made. AdamW optimizer was used as primary optimization method as it introduces decoupled weight decay, which prevents excessive parameter updates and improves generalization. ReduceLROnPlateu scheduler was used to dynamically adjust the learning rate based on validation performance. If the validation loss remains unchanged for 3 epochs, the learning rate was reduced by factor of four. This, combined with early stopping, ensures learning without stagnation or overfitting.

The prediction results are compared on two main criteria – inference performance time and quality of predictions. The performance was evaluated by performing a prediction and detecting its inference time. The quality of predictions during training was evaluated using custom loss functions and by using custom metrics for position and rotation features. A custom loss function was implemented which allowed combining multiple losses per feature. To gather positional loss a mean Euclidean distance loss and hierarchical distance losses were implemented. Euclidean distance loss calculates mean distances between predicted and validation data while hierarchical loss penalizes deviations in distances between connected nearby joints (fingers). Regarding metrics, for positional accuracy root mean square error (RMSE) – due to it being sensitive to large deviations, mean absolute error (MAE) – more stable measure for overall performance, mean Euclidean distance (MED) – computes geometric distance and provides spatially meaningful error measure, maximum Euclidean distance, temporal jerk – measures motion smoothness, and specific joint (wrist and index fingertip) mean Euclidean distance metrics were used. For rotational predictions, mean angular difference – computes average angular deviation between actual and predicted values and geodesic loss – measures shortest rotational distance along a unit sphere were implemented.

3.4. Experimentation

The study experimentation phase focuses on evaluating different model architectures and parameters. The results are provided for prediction of single frame ahead in collected tracking data. This approach was chosen because tracking interruptions typically last for only a few frames, especially at maximum allowed tracking frequencies. Input and output representations and training parameters are modified during tests to determine optimal combinations for hand tracking prediction. Several key variables were changed and compared, including data spaces, model architectures, input and output features used in predictions and usage of additional methods.
In total 44 experiments were conducted using 9800 pre-recorded frames. Sequential parameter optimization was used – one parameter is iteratively tuned at a time while keeping the previously optimized parameters fixed. Experimentation was done in a specific order of modifications: dataset space used (device, local hierarchical and local wrist), architecture changes (LSTM, GRU, attention mechanism, hidden state size and model changes), input features used (only position, only position and rotation and all features), output features used (only position, only rotation, both position and rotation), followed by other training parameter changes (batch size changes, weighted specific joints and weight decay usage) in provided order.

Tracking space experiments were performed using both LSTM and GRU architectures. In both cases, results suggest that the most optimal dataset uses transformed hierarchical local space. This can be explained by the fact that used architectures are capable of learning the relationships between neighboring joints well. Architecture comparison experiments suggest that medium sized LSTM models (512 hidden units with 2 layers) outperform GRU models in most cases while having higher inference times. The findings align with theoretical expectations as LSTM models are designed to better capture long-term dependencies compared to GRU. Introduction of attention mechanisms for both LSTM and GRU provide degraded accuracy – results suggest that currently used 15 frame window horizon is short enough to work well without such mechanisms as adding them implements unnecessary complexity to the models. Additionally, recorded datasets have some abrupt movements (for better generalization) which in attention mechanism case can reduce effectiveness. Input and output feature experimentation suggests that most accurate position predictions are gathered when predicting both position and rotation while using the same features as inputs. Results show that higher rotation prediction accuracy is gathered when input is both position and rotation while output is only rotation. This behavior is explained by the fact that hierarchical local positions inherently change when the parent joint is rotated, additional contextual information provides better results. Regarding other parameter changes, the best batch size balance is struck when 512 elements are used. Experiments demonstrate that better accuracy is achieved when specific joint weights in loss function calculations are used in combination with weight decay.

The results are recorded in comparative tables, highlighting the best performing configuration and metrics as bolded text. The best combinations and results from each experiment are combined into a single comparative table each (see Table 2 and Table 3). Input and output features in the table represent a binary feature selection vector using one-hot encoding where first position represents position being used, second position represents rotation usage, third and fourth position represent linear and angular velocity usage. Comparative table contains results where a separate model is trained for each hand while metrics are provided for left hand only (both hands present similar metrics).

Table 2
Features used in each experiment

No.	Input features used	Output features used	Architecture (hidden, layers)	Batch size	Position loss	Rotation loss	Weight decay
1	(1, 1, 1, 1)	(1,1,0,0)	LSTM (512, 2)	512	MED and Finger Hierarchy	Quaternion Angle	FALSE
2	(1, 1, 1, 1)	(1,1,0,0)	LSTM (512, 2)	512	MED and Finger Hierarchy	Quaternion Angle	FALSE

3	(1, 1, 1, 1)	(1,1,0,0)	GRU (256, 2)	512	MED and Finger Hierarchy	Quaternion Angle	FALSE
4	(1, 1, 1, 1)	(1,1,0,0)	GRU (1024, 2)	512	MED and Finger Hierarchy	Quaternion Angle	FALSE
5	(1, 1, 0, 0)	(1,1,0,0)	LSTM (512, 2)	512	MED and Finger Hierarchy	Quaternion Angle	FALSE
6	(1, 1, 1, 0)	(1,1,0,0)	LSTM (512, 2)	512	MED and Finger Hierarchy	Quaternion Angle	FALSE
7	(1, 1, 0, 0)	(1,1,0,0)	GRU (1024, 2)	512	MED and Finger Hierarchy	Quaternion Angle	FALSE
8	(1, 1, 1, 0)	(1,1,0,0)	GRU (1024, 2)	512	MED and Finger Hierarchy	Quaternion Angle	FALSE
9	(1, 1, 0, 0)	(1,1,0,0)	LSTM (512, 2)	512	MED and Finger Hierarchy	Quaternion Angle	FALSE
10	**(1, 1, 0, 0)**	**(0,1,0,0)**	**LSTM (512, 2)**	**512**	-	**Quaternion Angle**	**FALSE**
11	**(1, 1, 0, 0)**	**(1,1,0,0)**	**LSTM (512, 2)**	**256**	**MED and Finger Hierarchy**	**Quaternion Angle**	**0,001**
12	(1, 1, 0, 0)	(1,1,0,0)	LSTM (512, 2)	512	MED and Finger Hierarchy	Quaternion Angle	0,001
13	(1, 1, 0, 0)	(1,1,0,0)	LSTM (512, 2)	1024	MED and Finger Hierarchy	Quaternion Angle	0,001

Table 3
Metrics gathered for specific experiments

No.	RMSE	MAE	MED	Max ED	Temp. Jerk	Wrist MED	Index tip MED	Mean angle diff.	Geodesic loss	Training time	Infer. Time (ms)
1	0.0046	0.0023	0.0046	0.242	0.037	0.0026	0.0056	5.49	0.096	120.4	0.9
2	0.0046	0.0023	0.0046	0.242	0.037	0.0026	0.0056	5.49	0.096	120.4	0.9
3	0.0063	0.0037	0.0074	0.308	0.042	0.0059	0.0082	6.73	0.118	174.3	0.8
4	0.0071	0.0036	0.0072	0.509	0.038	0.0044	0.0088	7.48	0.130	305.9	2.1
5	0.0029	0.0015	0.0031	0.067	0.036	0.0009	0.0042	5.64	0.098	124.3	0.1
6	0.0034	0.0018	0.0037	0.161	0.036	0.0019	0.0046	5.81	0.101	168.0	0.1
7	0.0033	0.0018	0.0036	0.062	0.036	0.0019	0.0051	7.51	0.131	259.8	2.0
8	0.0043	0.0024	0.0047	0.301	0.037	0.0032	0.0054	6.84	0.119	262.2	2.0
9	0.0029	0.0015	0.0031	0.067	0.036	0.0009	0.0042	5.64	0.098	124.3	1.0
10	-	-	-	-	-	-	-	**3.67**	**0.064**	**68.4**	1.1
11	**0.0024**	**0.0013**	**0.0025**	**0.060**	0.036	**0.0007**	**0.0034**	4.44	0.077	137.1	1.1
12	0.0028	0.0015	0.0030	0.068	0.036	0.0010	0.0041	5.33	0.093	188.7	1.1
13	0.0036	0.0020	0.0040	0.072	**0.035**	0.0016	0.0049	8.09	0.141	192.6	1.2

The findings provide insights into the most effective architectural choices, input strategies and best training optimization for real-time hand motion predictions in VR environments. Regarding position predictions, the best results were gathered using position and rotation as input while predicting both position and rotation using LSTM network with 2 layers and 512 hidden units per layer with weight decay of 0.001. For rotation predictions, it was noticed that a model which used position and rotation as input and predicts only rotation works better. The same architecture provided the best results. Best performing loss functions were mean Euclidean loss, finger hierarchical distance loss and quaternion angle loss.

4. Conclusions

The analysis of Meta Quest 2 tracking system revealed that, while it supports controller-free hand tracking, it encounters limitations when additional devices such as haptic gloves are used, when parts of the hand are occluded or when lighting conditions are suboptimal. In terms of real-time communication between C# and Python, the most effective approach was found to be Python.NET, as it enables ultra-low latency (below 1 millisecond), ensuring smooth integration between Unity and

PyTorch based prediction models.

Experimental results show that the most accurate position prediction was achieved using hierarchical local space data representation, input sequences of the previous 15 frames, and an LSTM model with 2 layers and 512 neurons per layer. The batch size of 256, combined with Mean Euclidean Distance and hierarchical finger distance loss functions, along with quaternion angular loss produced a 2.5 millimeter mean Euclidean error for joint positions. Similarly, for rotation predictions, the best results were obtained under the same input conditions but with a batch size of 512, predicting only rotation while applying quaternion angular loss and no weight decay, resulting in a 3.66-degree rotational error. These findings indicate that even small recurrent neural network models, combined with custom weighting and loss strategies, can significantly improve hand tracking robustness and precision in VR environments without compromising the performance.

Additional improvements could be achieved by expanding the recorded dataset to include more hand positions and rotations near the outer bounds of the tracking space in combination with data augmentation techniques to enhance generalization. More advanced hyperparameter optimization strategies beyond sequential parameter optimization could further improve model accuracy. Extending the prediction to multiple future frames, rather than single-frame outputs may improve tracking continuity during rapid movements.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] Virtual Reality Market Size, Share, Trends and Forecast by Device Type, Technology, Component, Application, and Region, 2025-2033. Key statistics. IMARC Group, 2024. URL: https://www.imarcgroup.com/virtual-reality-market.

[2] S. Thompson, VR Applications: Key Industries Already Using Virtual Reality. VirtualSpeech, 2024. URL: https://virtualspeech.com/blog/vr-applications.

[3] K. D. Singh, Top 6 VR Applications Transforming Industries in 2024. Magineu Journals, 2024. URL: https://www.magineu.com/journals/top-6-vr-applications-transforming-industries-2024/.

[4] 18 Transformative Ways Industries Are Leveraging AR and VR. Forbes Technology Council, 2024. URL: https://www.forbes.com/councils/forbestechcouncil/2024/05/17/18-transformative-ways-industries-are-leveraging-ar-and-vr/.

[5] Steam Hardware & Software Survey: January 2025. Valve Corporation, 2025. URL: https://store.steampowered.com/hwsurvey.

[6] Unity Hand Tracking Troubleshooting and Limitations. Meta Platforms, Inc., 2025. URL: https://developers.meta.com/horizon/documentation/unity/unity-handtracking-troubleshooting-limitations/.

[7] Meta Will “Keep Looking At” Giving Quest Developers Raw Camera Access. David Heaney, 2024. URL: https://www.uploadvr.com/metas-cto-thoughts-quest-developers-raw-camera-access.

[8] J. Tang, J. Wang, J.-F. Hu, Predicting human poses via recurrent attention network, Visual Intelligence 1 (2023). doi:10.1007/s44267-023-00020-z.

[9] Y. Li, H. Zhou, X. Sun, Incorporating physics principles for precise human motion prediction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. doi:10.1109/CVPR.2024.10484216.

[10] A. Kumar, M. Patel, Dynamic model informed human motion prediction based on unscented Kalman filter, IEEE Transactions on Pattern Analysis and Machine Intelligence 47 (2025). doi:10.1109/TPAMI.2025.9790329.

[11] L. Chen, Z. Wu, Learning dynamic relationships for 3D human motion prediction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. doi:10.1109/CVPR.2024.9157765.

[12] N. M. Gamage, D. Ishtaweera, M. Weigel, A. Withana, So Predictable! Continuous 3D Hand Trajectory Prediction in Virtual Reality, *Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology (UIST)*, ACM, Virtual Event, USA, 2021, pp. 332–343. doi:10.1145/3472749.3474753.

[13] Unity Fast Motion Mode. Meta Platforms, Inc., 2025. URL: https://developers.meta.com/horizon/documentation/unity/fast-motion-mode