Habitus: Boosting Mobile Immersive Content Delivery through Full-body Pose Tracking and Multipath Networking

Anlan Zhang, *University of Southern California;* Chendong Wang, *University of Wisconsin — Madison;* Yuming Hu, *University of Minnesota — Twin Cities;* Ahmad Hassan and Zejun Zhang, *University of Southern California;* Bo Han, *George Mason University;* Feng Qian, *University of Southern California;* Shichang Xu, *Google*

https://www.usenix.org/conference/nsdi24/presentation/zhang-anlan

This paper is included in the Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation.

April 16–18, 2024 • Santa Clara, CA, USA

978-1-939133-39-7

Habitus: Boosting Mobile Immersive Content Delivery through Full-body Pose Tracking and Multipath Networking

Anlan Zhang[†*], Chendong Wang[◇], Yuming Hu[††], Ahmad Hassan[†*],
Zejun Zhang[†*], Bo Han[‡], Feng Qian[†*], Shichang Xu[°]
[†]*University of Southern California,* [††]*University of Minnesota – Twin Cities,*
[‡]*George Mason University,*[◇]*University of Wisconsin – Madison,* [°]*Google*

Abstract

Delivering immersive content such as volumetric videos and virtual/mixed reality requires tremendous network bandwidth. Millimeter Wave (mmWave) radios such as 802.11ad/ay and mmWave 5G can provide multi-Gbps peak bandwidth, making them good candidates. However, mmWave is vulnerable to blockage/mobility and its signal attenuates very fast, posing a major challenge to mobile immersive content delivery systems where viewers are in constant motion and the human body may easily block the line-of-sight.

To overcome this challenge, in this paper, we investigate two under-explored dimensions. First, we use the combination of a viewer's full-body pose and the network information to predict mmWave performance as the viewer exercises six-degree-of-freedom (6-DoF) motion. We apply both offline and online transfer learning to enable the prediction models to react to unseen changes. Second, we jointly use the omnidirectional radio and mmWave radio available on commodity mobile devices to deliver immersive data. We integrate the above two features into a user-space software framework called Habitus, and demonstrate how it can be easily integrated into existing immersive content delivery systems to boost their network performance, which leads to up to 72% of quality-of-experience (QoE) improvement.

1 Introduction

Immersive content, such as virtual/mixed reality (VR/MR) and volumetric videos, allows viewers wearing VR/MR headsets to exercise six-degree-of-freedom (6-DoF) motion (yaw, pitch, roll, X, Y, Z), offering a truly engaging experience [12, 47, 56, 62]. Networked immersive content delivery systems require tremendous network resources (*e.g.,* hundreds Mbps or even Gpbs for high-quality volumetric videos [47, 55, 88]). This poses a major challenge for mobile immersive content delivery systems, which use wireless radios instead of HDMI/USB cables [20, 21] for content delivery.

Recent advances in millimeter wave (mmWave) radio technologies make it feasible to transmit immersive content at a multi-Gbps data rate. mmWave protocols such as 802.11ad [65] (802.11ay [43] in the future) and mmWave 5G [49] have been commercialized on commodity mobile devices. Despite its high data rate, compared to omnidirectional radio, mmWave radio is much more vulnerable to blockage/mobility and its signal attenuates much faster [65]. This creates a major issue for immersive applications where viewers are in constant motion and the human body may easily block the line-of-sight. To overcome this issue, existing systems take three categories of approaches.

• *Improving the PHY layer*. Numerous studies have been conducted on the mmWave radio in general, such as improving MIMO [42] and beamforming [79, 81].

• *Enhancing line-of-sight (LoS)*. Some off-the-shelf commercial products [23] mount the mmWave radio on top of a VR headset to avoid blockages.

• *Using specialized equipment*. Some prior research [27] proposes to deploy multiple reflectors paired with a custom PHY protocol design to improve VR performance over mmWave.

The above approaches help but all have limitations. Engineering the PHY layer alone is inadequate to handle, *e.g.,* the throughput fluctuation incurred by viewers' fast motion. Mounting the radio overhead may still incur frequent non-line-of-sight (NLoS) blockages (*e.g.,* when the viewer looks up/down, raises arms, or passes through an obstacle). Adding reflectors to combat NLoS raises the deployment bar and is incompatible with commodity mmWave protocols. Even in the absence of blockage, highly dynamic user mobility can still cause significant performance drops of mmWave [28, 71, 74].

In this paper, we investigate two complementary, under-explored dimensions to improve the performance of mmWave-based immersive content delivery systems: (1) *full-body-pose guided mmWave throughput prediction* and (2) *joint use of mmWave and omnidirectional radios*. We then integrate them into a holistic middleware framework called Habitus. At a high level, Habitus features a judicious cross-layer design that considers the interplay among viewers' motion, wireless networks, and immersive applications. It creatively leverages features on cheap commodity mobile devices (*e.g.,* dual 802.11ac/ad radios and multi-lens cameras capable of producing stereo images) for affordable high-quality immersive content delivery. Habitus is readily deployable without requiring any change to the existing wireless protocol stack, hardware, or driver. It is orthogonal to and can co-exist with the three categories of solutions described above. The **key challenges** we face include: (1) the dynamics of viewers' motion and mmWave channel incur complex interplay, making accurate throughput prediction difficult; (2) diverse locations and human viewers add more complexity in developing a robust prediction model; even at the same location, the environment

* Work done while at University of Minnesota – Twin Cities.

may change (*e.g.,* a moved chair or a walking spectator); (3) the heterogeneous characteristics of mmWave and omnidirectional radios make their duet difficult.

Full-body-pose Guided mmWave Throughput Prediction (§4). Over a mmWave link, although the throughput fluctuations cannot be completely avoided, they can potentially be predicted to improve the quality-of-experience (QoE) of immersive applications. Habitus utilizes not only the network information, but also viewers' motion to predict mmWave performance. The rationale is that by continuously tracking the 6-DoF motion, an immersive content delivery system can estimate the viewer's future motion trajectory [47, 66, 86], which can then be mapped to the future mmWave performance given the sensitivity of mmWave signal to the physical environment. In particular, we make a new discovery that using the viewer's *full-body pose* (how a person stands, sits, or moves as represented by a set of key points associated with body parts/joints) as features can significantly improve the throughput prediction accuracy, due to the *spatial correlation among body parts* during typical human motion [30]. Motivated by the above, we develop a first-of-its-kind framework that predicts mmWave throughput by jointly leveraging a headset's 6-DoF motion, the viewer's body pose, and network information, through a unified machine learning model. The full-body pose can be captured by a commodity stereo camera conveniently placed, *e.g.,* next to the WiFi AP.

Reacting to Unseen Changes (§5). Using a pre-trained model to predict throughput suffers from a key limitation: it cannot adapt to changes deviating from the training data. We systematically investigate how various types of changes in the immersive streaming context impact the prediction accuracy of a pre-trained model. Based on the insights, we design three orthogonal mechanisms for reacting to different types of unseen changes: (1) *offline transfer learning* handles large changes such as switching to a new location/user; (2) *online transfer learning* updates the model at runtime to tackle smaller changes such as new motion patterns and environmental perturbations; (3) we also leverage the stereo camera to *proactively detect/respond to moving objects* (*e.g.,* a passing person) that affect the mmWave performance.

Joint Use of mmWave and Omnidirectional Radios (§6). Multi-band radio access is a common feature on both mobile devices and WiFi APs. For example, the Asus ROG Phone Series [16] support both 802.11ac and ad. Strategically combining them can boost the network performance for metaverse. We design a lightweight yet effective multipath scheduler for immersive content delivery over mmWave and omnidirectional radios (802.11ad and ac in our prototype). It employs two core design ideas. First, it prioritizes the (low-bandwidth but stable) ac path to guarantee the basic user experience, and opportunistically leverages (high-bandwidth but fluctuating) ad whenever possible. Second, it enhances the mmWave throughput prediction through robust statistical trend analysis [39] to facilitate longer-term throughput forecast.

Implementation (§7). Instead of building a monolithic application, we develop the above features as a generic, user-space middleware framework called Habitus. It offers simple interfaces and data-handling paradigms that are compatible with a wide range of existing immersive applications. It also addresses practical system-level challenges, such as accurate throughput measurement of the highly bursty traffic of immersive content delivery. Habitus consists of 3,541 lines of code (LoC). To demonstrate its efficacy, we develop two immersive apps using its API: one is built from scratch in 5.2K LoC; the other is adapted from a state-of-the-art volumetric streaming system [47] by only modifying 47 LoC.

Datasets (§4, §5) and Evaluation (§8). We thoroughly evaluate Habitus through real-world data and deployment.

- We conduct IRB-approved data collection involving 10 representative motion patterns at 4 representative indoor locations from 3 users. This results in a 21-hour dataset that was used to evaluate Habitus's prediction framework.
- We enhance the above dataset with both static and dynamic environmental changes in a reproducible manner (*e.g.,* using a robotic arm to programmatically inject NLoS, see our demo video [2]), to evaluate Habitus's reaction to changes.
- Using full-body poses reduces 802.11ad throughput prediction error by up to 29% (25%) in MAE (RMSE), compared to using only 6-DoF head motions. This translates to an average QoE improvement of 29% for volumetric content delivery.
- Habitus effectively responds to unseen changes. The offline transfer learning reduces the model training time by 36% to 55% compared to building the model from scratch when switching to a new location or user. The online transfer learning can adapt to a new motion pattern or a typical static environmental change in 32 secs and 15 secs, respectively. By proactively detecting and responding to moving objects, Habitus reduces the volumetric streaming stall by 7%.
- Our multipath solution boosts the average volumetric video quality by 67%, reduces the stall by 64%, and improves the QoE by 72%, compared to using 802.11ad alone. Compared to a recent multipath solution for 802.11ac/ad [71], Habitus reduces the stall by 58% and boosts the quality by 19%.
- We conduct another IRB-approved user trial where we collect 12 viewers' subjective feedback when watching volumetric content. The average ratings for 802.11ad only (basic prediction), 802.11ac+ad (no ad prediction), 802.11ad+ac (basic ad performance prediction), and the full Habitus system (with multipath and full-fledged ad prediction) are 2.67, 2.75, 3.08, and 3.50, respectively (in a 1–5 scale).

Habitus represents to our knowledge a first software framework aiming at optimizing the upper-layer network protocol stack for immersive content delivery (and metaverse applications in general). This paper makes three-fold contributions: the design of the Habitus framework; its implementation, evaluation, and integration into two volumetric content delivery systems; and the release of data [6] (802.11ac/ad performance correlated with full-body motion) and source code [5].

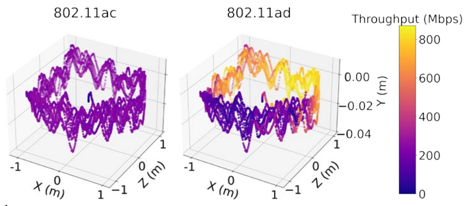


Figure 1: Spatial heatmap of 802.11ac/ad throughput in a room.

2 Background and Motivation

Despite their great potentials on boosting immersive content delivery [47, 55], mmWave signals suffer from increased attenuation, mobility, and blockages [65]. In contrast, 802.11ac operates at the 5GHz band with omnidirectional signal propagation, providing lower but more stable throughput than 802.11ad. In a case study conducted in a personal office (Figure 3), we investigate how the PHY properties of ac and ad affect the QoE of immersive content delivery. Using a smartphone [16] mounted on the user's head and a volumetric video streaming application [47], we measure the QoE while the user walks around the room. Results show that using 802.11ad greatly improves the content quality by 113%, but also hugely increases the video stall by 502% due to its fluctuating throughput under user mobility (Figure 1). Our case study reveals *that ac and ad have distinct network performance due to their complementary PHY properties, and motivates us to strategically combine them to enhance the QoE of immersive content delivery.*

Many studies focus on improving the communication quality of mmWave on the PHY layer [42, 79, 81], while ignoring contextual information for immersive content delivery where a viewer's full body is constantly in motion. On the other side, although solutions with application domain knowledge (*e.g.,* [23, 27]) have shown some effectiveness, they are either incompatible with existing PHY-layer protocols [27] or still suffer from significant LoS blockages [23]. Such a gap motivates us to propose solutions that *judiciously leverage viewer's full-body motion to facilitate mmWave performance forecast, while being compatible to commercial mmWave protocols (802.11ad, mmWave 5G/6G,* etc.*) and easily integrable into diverse immersive applications.*

3 Habitus Overview

Habitus is a software framework enabling immersive content delivery applications to better interact with heterogeneous off-the-shelf wireless networks. It offers two essential features: *accurate runtime mmWave throughput prediction* and *multipath networking* over mmWave (*e.g.,* 802.11 ad) and omnidirectional radio (*e.g.,* 802.11 ac). We assume that the bottleneck is the last-mile radio link(s). Figure 2 shows the workflow of Habitus. As the viewer is watching immersive content, Habitus collects various features in real-time and sends them to an edge node. The edge employs a machine learning model to perform accurate mmWave throughput predictions. The prediction results are then utilized by the application and Habitus's multipath scheduler to determine the appropriate content quality level and how to distribute the content over diverse radio links, respectively. Habitus works on top of the transport layer as a middleware. It is ready for deployment without requiring any changes to the existing wireless protocol stack, hardware, or drivers.

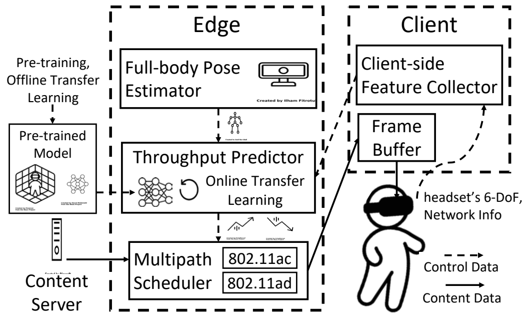


Figure 2: The system architecture of Habitus.

Developing Habitus brings us several challenges: (1) **How to accurately predict mmWave throughput at runtime?** Due to its sensitivity to blockage and mobility, mmWave throughput is difficult to predict, especially under user mobility. (2) **How to ensure the robustness of the prediction models?** A pre-trained model may not be able to handle various unseen changes such as moving to a new location, switching to a different user, or even smaller environmental perturbations. (3) **How to make multipath scheduling efficient and intelligent?** mmWave and omnidirectional radios have their unique natures. How to strategically use these properties to facilitate multipath scheduling is another critical problem.

Habitus tackles the above challenges with 3 core designs.

- **Full-body-pose Guided mmWave Throughput Prediction (§4).** Habitus enhances mmWave throughput prediction by exploiting the viewer's full-body pose, along with the headset's 6-DoF motion and network information, as the features. It employs a cheap stereo camera with off-the-shelf computer vision techniques to capture viewers' poses. We systematically demonstrate the benefit of leveraging body pose through a 21-hour dataset collected at 4 diverse locations.
- **Reacting to Unseen Changes (§5).** Habitus reacts to unseen changes in training data via three orthogonal approaches: offline transfer learning for location/user changes, online transfer learning for new motion patterns and small environmental changes, and proactively detecting/responding to moving objects (*e.g.,* a passing person) affecting mmWave performance.
- **Joint Use of mmWave and Omnidirectional Radio (§6).** Habitus employs the omnidirectional radio that provides stable throughput as a basis. It then opportunistically takes advantage of the fluctuating mmWave radio. It exposes simple, generic interfaces to a wide range of immersive applications.

4 Full-body-pose Guided mmWave Throughput Prediction

Habitus utilizes not only the network information, but also viewers' motion as features for mmWave throughput prediction. It is based on our two observations. (1) The fast signal attenuation and vulnerability to LoS blockages make mmWave throughput highly correlated with the headset's physical position and orientation [65, 83]. Both of them can be predicted from the viewer's historical 6-DoF motion trajectory [47]. (2) Various body parts exhibit spatial correlation [30]. It provides opportunities to enhance the headset motion prediction, which can facilitate mmWave throughput prediction.

4.1 Full-body Pose Estimation

Full-body Pose Representation and Retrieval. Typically, a full-body pose can be represented by a set of key points where each key point corresponds to a joint/part of the human body. We represent the viewer's full-body pose with 15 key points, covering the nose, neck, shoulders, elbows, wrists, hips, knees, and ankles. More details can be found in Appendix A.1.

Some commercial products (*e.g.,* smart suit [17] and body-mounted sensors [22]) allow tracking the full-body pose, but they are expensive and uncomfortable to wear. Habitus instead employs a cheap and easy-to-deploy approach to capture/track the viewer's full-body pose through a stereo camera. It first applies a machine learning model [29, 31, 33, 34, 60, 68] to the RGB frame to estimate the 2D key points of the full-body pose, each comprised of a 2D coordinate and a confidence value w ($0 \leq w \leq 1$). Habitus then maps the 2D key points to 3D space using the depth map [45] that is generated from stereo images, and keeps their confidence values unchanged.

Estimating Missing Key Points. In some cases, for example, when some parts of the body are outside the stereo camera's viewport, we are not able to capture their key points. We observe from our dataset in §4.2 that for 86% of time, the ML model can retrieve at least 10 (out of 15) key points. The 90-th percentile, mean, and median duration of a key point's missing time are 1s, 0.43s, and 0.07s, respectively. This indicates that in most cases, a key point misses for a very short duration. We estimate a missing key point's 3D coordinate on the fly using a combination two approaches: reusing its most recently captured 3D coordinate (when the missing period is short), and linearly extrapolating its coordinate using its historical trajectory (when the missing period is long). The detailed design and evaluation can be found in Appendix A.2.

4.2 Data Collection

We perform a first-of-its-kind study on full-body-pose assisted throughput prediction in real-world settings. We conduct an IRB-approved data collection involving 10 motion patterns at 4 indoor locations from 3 users, resulting in a 21-hour dataset consisting of both the network data and viewers' motion data. This unique dataset is used to evaluate mmWave throughput prediction (§4.3), techniques for tackling unseen changes (§5), and the Habitus system (§8).

4 Indoor Locations. We investigate 4 representative indoor locations with diverse environments (Figure 3). More details can be found in Appendix A.3. **3 Users.** We recruit 3 users

Patterns	Description
S1	The user stands in the center of the room, turning around in a clockwise direction.
S2	The user stands in the center of the room, turning around in a counterclockwise direction.
S3	The user walks around in a clockwise direction.
S4	The user walks around in a counterclockwise direction in a normal speed.
S5	The same as S4, but in a slow speed.
S6	The same as S4, but in a fast speed.
S7	A chair occupies the front place of the access point. The user walks around in a counterclockwise direction.
S8	The same as S3, but the user does not change the orientation of his/her head.
S9	The same as S4, but the user does not change the orientation of his/her head.
S10	The user walks around following the walking trace in S7, but there is no chair.

Table 1: User motion patterns.

with different heights (1.6m, 1.7m, and 1.8m) and genders (1 female and 2 males) to collect data in all the four locations. **10 Motion Patterns.** We consider 10 representative motion patterns when watching immersive contents [47] and summarize them in Table 1. For each motion pattern, we repeat data collection three times.

Dataset Overview. Our dataset consists of 12 {Location, User}-specific sub-datasets, each having 30 (10 motion patterns×3 repeats) data traces. The duration of each data trace is 120 secs and the time granularity of each data point is 1/60 secs. Our dataset consists of not only the network information (throughput and signal strength of both 802.11ac and 802.11ad), but also users' motion information (*i.e.,* 6-DoF motion of users' headsets and users' full-body pose, see §4.1). Across all data traces, the average 802.11ad throughput varies from 275 to 886 Mbps, with the standard deviation ranging from 85 to 358 Mbps. Meanwhile, the average 802.11ac throughput varies from 175 to 378 Mbps, with the standard deviation ranging from 26 to 86 Mbps. The highest throughput of 802.11ad only achieves 2.34× of 802.11ac due to the limitation of our hardware setup.

Hardware Setup. We take *Personal Office* in Figure 3 as an example. **For the edge**, we set up a desktop PC with two network interfaces (NICs) at the corner of the room. It has an Intel Core i9-10900X CPU @ 3.70GHz, an NVIDIA 2080Ti GPU, and 32GB memory. Each NIC is connected to an access point by a 1-Gbps Ethernet cable, one [19] for 802.11ac and the other [13] for 802.11ad. The two[1] APs reside on the floor side-by-side. A stereo camera [24] is installed on the wall and connected to the PC via a USB 3.0 cable. It captures users' motion as RGB-D videos at up to 100 FPS. **For the client device**, users wear a headset [11] for collecting 6-DoF motion of their heads. We mount a smartphone [16] that supports both 802.11ac/ad on the headset to collect network information.

[1]Ideally one access point is able to handle both ac and ad. We use two access points due to the 1-Gbps speed limitation of our Ethernet cables.

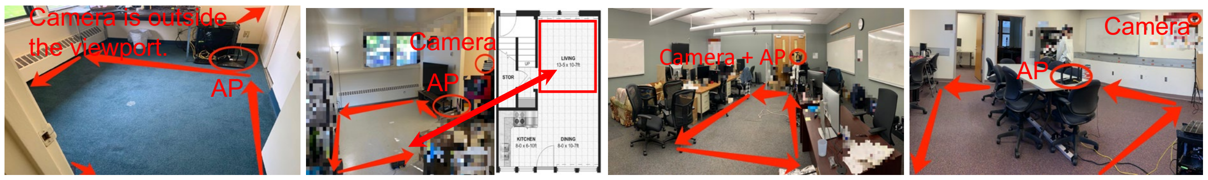


Figure 3: Data collection locations (from left to right: *Personal Office*, *Living Room*, *University Office*, and *Meeting Room*).

We use the same hardware setup for all four locations.

Data Collection Methodology. On the client side, the smartphone establishes two TCP connections with the edge over 802.11ac and 802.11ad, respectively. It performs bulk download from the edge through both paths, and measures the throughput and signal strength (in terms of RSSI). The headset keeps sending the 6-DoF to the smartphone by UDP over 802.11ac. Meanwhile, **on the edge side**, the desktop PC keeps sending the most recent RGB-D frame ID captured by the stereo camera to the smartphone through UDP over 802.11ac for data synchronization purpose (see next).

Data Synchronization and Post-processing. During data collection, the smartphone logs all the information (generated by itself or received from the headset/edge) except the RGB-D video that is recorded at the edge. Since all the devices are in close proximity, the UDP one-way delay is negligible ($\leq$ 2ms) so different pieces of data are properly synchronized. We use zed-openpose [26] (which consumes the RGB-D video) to estimate the user's body pose offline. The pose is synchronized with other information through the RGB-D frame ID, which is recorded by the client at runtime. For key point extraction, we set the input resolution to 320×240 and keep it consistent in our implementation (§7).

4.3 Prediction Methodology and Evaluation

We first formulate our prediction task. Let x_t denote the feature vector at time t, y_t denote the predicted throughput at time t, Δt denote the time granularity, and $\mathbb{M}$ denote the prediction model. Assuming that we perform prediction at time t_0, we have $Y_{t_0,m} = \mathbb{M}(X_{t_0,n})$ where where $X_{t_0,n} = [x_{t_0-(n-1)\times\Delta t}, x_{t_0-(n-2)\times\Delta t}, \ldots, x_{t_0-1\times\Delta t}, x_{t_0}]$ is the feature sequence within a history window (hw) n, and

$$Y_{t_0,m} = \begin{cases} [y_{t_0+m\times\Delta t}] \quad \text{or} \\ [y_{t_0+1\times\Delta t}, y_{t_0+2\times\Delta t}, \ldots, y_{t_0+(m-1)\times\Delta t}, y_{t_0+m\times\Delta t}] \end{cases}$$

can be either a single predicted value after a prediction window (pw) m or a predicted sequence within the pw m where $y_{t_0+i\times\Delta t}$ $(1 \leq i \leq m)$ corresponds to a future timestamp $t_0 + i \times \Delta t$. The hw (pw) in secs is computed as $n \times \Delta t$ ($m \times \Delta t$).

Habitus uses the full-body pose by taking the coordinates and confidence values (§4.1) of its key points as important features to the mmWave throughput prediction models. We investigate 5 different models from recent studies on mmWave throughput prediction [28, 63]. We customize them to our prediction task by tuning the model architecture and the parameters. We list them as follows. **(1) Gradient Boosting Decision Tree (GBDT) [63].** Our *GBDT* model has 100 estimators, bounded by a depth of size 3. It takes $X_{t_0,1}$ as the input and predicts a single throughput value $Y_{t_0,m}$. **(2) Fully-connected Neural Network (BP) and Recurrent Neural Network (RNN) [28].** *BP8* is a fully-connected neural network with 3 hidden layers, each with 40 neurons. It takes $X_{t_0,8}$ as the input and predicts a single throughput value $Y_{t_0,1}$. *RNN8* and *RNN20* have the same network architecture, *i.e.,* a recurrent neural network with 3 hidden layers, each with 8 or 20 neurons. They take $X_{t_0,8}$ and $X_{t_0,20}$ as the input, respectively, and both predict a single throughput value $Y_{t_0,1}$. **(3) Sequence-to-sequence Learning (Seq2Seq) [37, 63, 75, 77].** Our *Seq2Seq* model has a single-layer LSTM encoder-decoder architecture with 128 hidden units. It takes $X_{t_0,n}$ as the input and predicts future throughput sequence $Y_{t_0,m}$ where $m = 2 \times n$.

We use our dataset (§4.2) to train/evaluate the above models {*w/*, *w/o*} the full-body pose as extra features. We perform 10-fold cross-validation for each model on each {Location, User}'s dataset, and quantify their prediction errors by mean absolute error (MAE) and root mean square error (RMSE). For *Seq2Seq*, to fairly compare it with the other models, we only use the value y_{t_0+pw} in its predicted sequence. We use three prediction windows (pw) of {0.5, 1, 2} secs.

Figure 4 shows the average MSE and RMSE for different models across all {Location, User}'s datasets when pw=1 sec. The model trained with (without) full-body pose is denoted as *Model w/ Pose* (*Model*) in Figure 4. We also normalize the prediction error by the average 802.11ad throughput (493 Mbps) of our dataset. We have four observations here. (1) *Seq2Seq w/ Pose* achieves the lowest prediction error: 31 (47) Mbps in MAE (RMSE). (2) Leveraging full-body pose as extra features effectively reduces the prediction error for all the models. The reduction ranges from 5% (*GBDT*) to 29% (*RNN20*) in MAE and 5% (*GBDT*) to 25% (*RNN20*) in RMSE. This quantitatively confirms that leveraging spatial correlation among body parts helps boost the throughput prediction accuracy under viewers' constant motion. (3) Although not shown, the benefit of leveraging full-body pose is similar in both simple (*i.e., Personal Office* and *Living Room*) and complex (*i.e., University Office* and *Meeting Room*) environments. (4) Deep learning models of prior work [28] (*RNN20*, *RNN8*, *BP8*) do not necessarily outperform the non-deep-learning model (*GBDT*). This is likely because those in [28] are designed for limited motion (2-DoF) as opposed to the complex 6-DoF motion in real-world settings. We confirm that the above findings also hold for pw=0.5s and pw=2.0s.

5 Reacting to Unseen Changes

In this section, we investigate how the throughput prediction

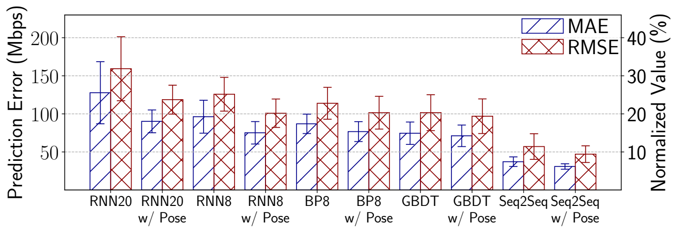


Figure 4: Prediction error of different models {*w/, w/o*} full-body pose as extra features, $pw = 1$ sec.

models developed in §4 react to changes unseen in the training data. This is an important aspect we must consider since the changes are common in practice (*e.g.,* new motion patterns, a chair being moved in the room, another person passing by). We then develop solutions to tackle these changes.

5.1 Measurement Methodology

Recall from §4.3 that we train a model for a particular user's motion patterns in the environment of a particular location. Therefore, a change to any of the above factors may degrade the model's prediction accuracy. We first define these changes. **C1: New Location.** The dimensions, interior design, and furnishings of locations differ vastly, as do the locations of the AP and camera. These variations can significantly impact the behavior of mmWave signal propagation [67,72,81,91]. **C2: New User.** Users differ in their shape and height, resulting in variations of their body poses used by the model. **C3: New Motion Patterns.** Even for the same user at the same location, a new motion pattern may lead to unseen trajectories of the head's motion or the body's pose. **C4: Static Environmental Changes** such as furniture being moved and even small objects being manipulated can affect the mmWave signal propagation [81,91]. **C5: Dynamic Environmental Changes** are similar to **C4** except that the object(s) that perturb the environment are in motion. A representative scenario is that, people as passerby(s) or spectator(s) can temporarily block the LoS and henceforth cause a throughput drop.

We next describe how to measure the impact of the above changes. For a given change C, we construct three datasets $\mathbb{T}_B$, $\mathbb{T}_A$, and $\mathbb{E}^A$. $\mathbb{T}_B$ and $\mathbb{T}_A$ contain the training data *before* and *after* C, respectively; $\mathbb{E}^A$ contains the testing data collected *after* C for evaluating the impact. We use $\mathbb{T}_B$ and $\mathbb{T}_A$ to train two models $\mathbb{M}_B$ and $\mathbb{M}_A$, respectively, using *Seq2seq w/ Pose* with the same hyper-parameters. Next, we test $\mathbb{M}_B$ and $\mathbb{M}_A$ using $\mathbb{E}^A$, and calculate the corresponding MAE as MAE_B^A and MAE_A^A respectively. Then the impact of C on the throughput prediction accuracy is calculated as $\text{MAE}_B^A - \text{MAE}_A^A$. The tradeoff here is accuracy vs. training overhead: MAE_A^A gives the best accuracy but requires retraining the model; MAE_B^A reuses the old model at the cost of degraded accuracy.

We now describe how to construct $\mathbb{T}_B$, $\mathbb{T}_A$, and $\mathbb{E}^A$ for **C1** to **C5**. Recall from §4.2 that our dataset is divided into (3 users) × (4 locations) = 12 groups (*i.e.,* sub-datasets), and each group contains traces of 10 motion patterns. Also, each group's traces are randomly split into training (70%) and testing (30%). Since a model is created for a given (user u, location l) pair, we measure the impact on a per-group basis, and then average the impact across all groups. For **C1**, for a given group (u,l), $\mathbb{T}_B$ consists of the training data of (u,l); $\mathbb{T}_A$ and $\mathbb{E}^A$ contain the training and testing data of $\cup_{l'\neq l}(u,l')$, respectively. For **C2**, it is similar to **C1** except that $\mathbb{T}_A$ and $\mathbb{E}^A$ belonging to $\cup_{u'\neq u}(u',l)$. For **C3**, the three sets all belong to the same (u,l) pair but they contain different motion patterns: $\mathbb{T}_A$ contains all 10 motion patterns; we remove one motion pattern e from $\mathbb{T}_B$, and only keep e in $\mathbb{E}^A$. We repeat the above measurement 10 times, each time using one of the 10 motion patterns as e. For **C4** and **C5**, the three sets all belong to the same (u,l) pair, but we physically introduce the environmental changes and then recollect data for $\mathbb{T}_A$ and $\mathbb{E}^A$, as elaborated next.

We inject two static environmental changes (**C4**) and study them separately: (1) move four chairs in the room to fixed places (Figure 6 Left), and (2) put four large packages on designated spots. Then we ask the same users to exercise the same motion patterns (as those in $\mathbb{T}_B$) as if there were no environmental change (we ensure that the changes do not block any motion pattern). Injecting a dynamic change (**C5**) in a reproducible manner requires a more sophisticated setup. We use a robotic arm to move a box covered by aluminum foil back and forth between two designated spots to periodically introduce NLoS (Figure 6 Right). The box size and aluminum foil thickness are determined through a separate controlled experiment (details in Appendix §B) to mimic real humans in terms of NLoS-incurred throughput degradation. In this way, we programmatically emulate a passerby intermittently causing NLoS (demo video [2]). We study **C4** and **C5** only at *University Office* due to setup complexity.

5.2 Measurement Results and Insights

Figure 5 plots the impact of **C1** to **C5** on the mmWave throughput prediction accuracy. The two Y axes show both the absolute MAE growth ($\text{MAE}_B^A - \text{MAE}_A^A$) and the relative growth (normalized by the average 802.11ad throughput in our dataset). We highlight our findings next.

A new location incurs the highest impact. C1 degrades the model's accuracy by 19% (93 Mbps) on average. It reveals that the physical property of mmWave and its throughput distribution in a location is the fundamental knowledge learned by our model. Since the indoor location, characterized by its room layout, surface materials, furniture arrangement, *etc.* plays a dominant role in determining the mmWave propagation, changing the location will reshape the throughput distribution at different 3D coordinates.

A new user incurs a moderate impact. C2 increases the average MAE by 8% (38 Mbps). The impact is lower than that of **C1**, but still non-negligible. This suggests that our model also learns how a user's motion affects the throughput received by the headset. Changing the user alters the relative

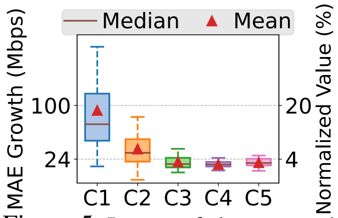


Figure 5: Impact of changes on the model prediction accuracy (*Seq2seq w/ Pose*, $pw = 1$ sec).

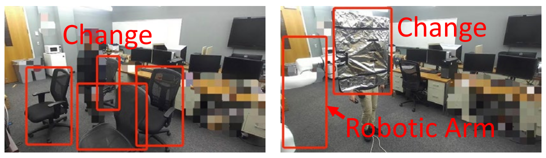


Figure 6: Static (left) and dynamic (right) environmental changes, from AP's view.

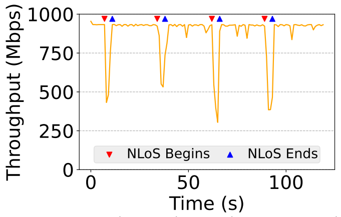


Figure 7: mmWave throughput drop caused by dynamic changes (**C5**) in our demo [2].

positions among the key points used by our model; it also changes the trajectory that the headset can reach even with the same motion pattern. Both degrade the model's accuracy. Nevertheless, the impact is lower than **C1** because the throughput distribution, mostly shaped by the surrounding environment, largely remains consistent.

New motion patterns and static/dynamic environmental changes may incur a small impact. Upon **C3**, **C4**, and **C5**, our model's prediction accuracy only drops marginally, by 4%, 3%, 4% (20, 17, 19 Mbps) on average, respectively. The primary reason is that the changes are spatially small (**C3** and **C4**) or temporarily short (**C5**). For **C3**, a new motion pattern oftentimes has positions overlapped with old ones, making the previously learned knowledge relevant. For our **C4** instances, the moved chairs and packages usually do not incur additional NLoS. Our **C5** instance does incur NLoS, but the overall impact is small due to its short duration (Figure 7). Note that, however, an environmental change may cause a big mmWave performance impact (*e.g.,* there are apparent blockages near the mmWave AP). Users of Habitus (and any mmWave system in general) should avoid such scenarios.

5.3 Methods of Handling Changes

The results in §5.2 suggest that besides taking the time-consuming approach of retraining a model from scratch, Habitus may adopt different strategies to tackle changes. We next introduce three orthogonal mechanisms. The first two (offline/online transfer learning) adopt the concept of *homogeneous transfer learning* [51,59,93] that transfers the knowledge learned from a past experience to a new setting. The two properties below make transfer learning a desirable solution. (1) Before and after the change, the feature space (*i.e.,* mmWave signal strength and throughput, 6-DoF motion, full-body pose) remains the same but their distributions (or domains) may differ [93]; (2) before and after a change, there is invariant knowledge (*e.g.,* the physical property of mmWave and the throughput distribution in certain positions) that can be reused. The third mechanism promptly handles **C5** by fusing real-time computer vision into Habitus.

Offline Transfer Learning. For **C1** and **C2**, given their infrequency and non-negligible impact on the model's performance, collecting data under the new setting to update old the model *before using it* helps mitigate large prediction accuracy drops. Specifically, when switching to a new location or a new user, Habitus asks the (new) user to exercise motion patterns (*e.g.,* those in Table 1) in the (new) location while measuring the mmWave bandwidth and collecting input features. A typical data collection only needs 1 to 2 minutes (see §8.5). Habitus then uses the collected training data to update the old model before starting streaming for the new user or location.

Online Transfer Learning. To handle **C3** to **C5** (and also **C1**, **C2**), since they occur much more frequently with usually a much smaller accuracy impact, Habitus can collect data under the new setting and update the model *on-the-fly*. Unlike offline transfer learning, online transfer learning does not incur additional data collection overhead and is transparent to users. Specifically, during a streaming session, Habitus updates the model in consecutive epochs. Epoch i produces a new model M_i and a set of training data samples D_i (input features and the measured ground truth mmWave throughput). Habitus also maintains a global training dataset D_G. At the beginning of Epoch i, Habitus (1) sets the current model for throughput prediction to M_{i-1}; (2) appends D_{i-1} to D_G, and start using D_G to update M_{i-1}; (3) start collecting D_i that will be used to update M_i in Epoch $i+1$. To bootstrap the above process, Epoch 0 only collects D_0 for a fixed period of 10 secs. To avoid D_G becoming too large, Habitus limits D_G to contain only data collected in the recent 5 minutes. We tune the batch size (64) to balance each epoch's convergence speed and the total model copy overhead after epochs. We also apply a small learning rate (0.001) given that we are fine-tuning the model rather than training it from scratch.

Vision-based Dynamic Change Handling. We find that even online transfer learning is too slow to react to **C5**. We thus devise a heuristic-based design to improve the responsiveness. The idea is to leverage the stereo camera, which already belongs to Habitus's infrastructure, to visually capture dynamic changes and penalize the predicted mmWave throughput accordingly. Specifically, we focus on the most common dynamic change: a person temporarily blocks the LoS between the viewer and mmWave AP. During a streaming session, the edge performs continuous human detection [25, 32, 78]. If a passerby is detected (we know the viewer's position so the viewer will not be confused with the passerby), Habitus uses the detected 3D bounding box, the known position of the mmWave AP, and 6-DoF motion reported by the headset, to determine if the passerby is causing NLoS or may cause NLoS in the near future by examining the distance from the bounding box center to the LoS between the viewer and AP. If so, Habitus adds an empirical penalty to the mmWave throughput B_{ad} predicted by the model: $B'_{ad} = -\frac{s_{max}-s}{s_{max}-s_{min}} \times B_{ad}$ where s is the observed signal strength; s_{min} and s_{max} denote the typical indoor signal strength range (empirically set to -70 and -30 dBm, respectively). We evaluate the above three methods in §8.6.

6 System Design of Habitus

We now detail the system design of Habitus that leverages the functionalities introduced in §4 and §5 as building blocks.

As shown in Figure 2, except the client-side feature collector, all the other components of Habitus reside on the edge. This helps minimize the energy consumption and heat dissipation on client devices.[2] The choice of the edge is flexible. It can be either a user's own desktop PC, or an edge node co-located with a mmWave 5G base station (*e.g.,* AWS wavelength [1]). The edge also acts as a proxy by forwarding the client's requests to the server and the server's streamed content to the client. Habitus supports both client request and server push. Our prototype uses the former.

6.1 Application Interface

Habitus jointly utilizes mmWave and omnidirectional radios (802.11ad and 802.11ac in our prototype) to deliver immersive content. It exposes simple interfaces to applications.

• Through a callback, Habitus keeps informing the application of the two radio links' *aggregated* bandwidth. The application should ensure that its actual streaming bitrate does not overshoot the aggregated bandwidth. The bandwidth update is at a fast pace (*e.g.,* 30 FPS) to match the viewer's fast motion.

• The application streams immersive contents on a per *data block* basis, which can be flexibly defined by the application based on its semantics. Each block can be independently decoded. We use examples in 6.4 to show that our block-based paradigm is aligned with the design of many existing immersive apps (*e.g.,* 360° videos, volumetric videos, and VR). When the client requests for (or the server pushes) a block, it uses Habitus API to attach two parameters: the block's *priority* and *playback deadline*. Habitus forwards the blocks based on their playback deadline in a FIFO manner, and distributes high-priority and low-priority blocks over the omnidirectional and mmWave radios, respectively. The rationale is that the omnidirectional radio is more reliable, so high-priority blocks get a higher chance of being delivered (and hence decoded and rendered) before its deadline than low-priority blocks.

6.2 Utilizing mmWave Throughput Prediction

Habitus exercises multipath content delivery in two steps. It first estimates the aggregated network bandwidth by treating the multipath ac/ad connections as one logical connection. Second, it splits the block stream over ac and ad paths. We now detail the first step, and describe the second step in §6.3.

[2]For example, in our experiment, running our *Seq2Seq* model (§4.3) on the ROG phone II [16] for only two minutes will trigger an overheating issue.

Recall that the predicted throughput of 802.11ad, denoted as B_{ad}, can be obtained from our mmWave throughput prediction model (§4, §5). The 802.11ac throughput, B_{ac}, is much more stable and largely not affected by the environment. We therefore simply use the harmonic mean of a past window of 5 secs to predict B_{ac}. Then Habitus predicts the aggregated capacity as $B_{ac} + c \times B_{ad}$. We use c, which we call the *trend-aware coefficient*, to further enhance the 802.11ad throughput prediction by considering the trend of a finite horizon in the future as produced by our model (§4.3). The idea is to analyze the monotonic trend of the predicted throughput sequence. If the trend is increasing (decreasing), we can use the 802.11ad link aggressively (conservatively).

To derive c, Habitus first uses Cox-Stuart Test [39, 70], a lightweight, non-parametric approach, to determine the trend (increase, decrease, or neither). Its details are in Appendix C.

$$c = \begin{cases} 1 + \text{avg}\left(\sum_{i=1}^{\frac{n}{2}} \mathbb{I}\left(z_i < z_{i+\frac{n}{2}}\right) \times \left|\frac{z_{i+\frac{n}{2}} - z_i}{z_i}\right|\right), & \text{if increase trend} \\ 1 - \text{avg}\left(\sum_{i=1}^{\frac{n}{2}} \mathbb{I}\left(z_i > z_{i+\frac{n}{2}}\right) \times \left|\frac{z_{i+\frac{n}{2}} - z_i}{z_i}\right|\right), & \text{if decrease trend} \\ 1, \text{ otherwise} \end{cases}$$

Next, Habitus uses the above formula to compute c by averaging the future changes in the predicted throughput sequence $\{z_i\}$. $\mathbb{I}(x) = 1$ iff x is true (otherwise 0), and n is the length of the predicted throughput sequence. The formula splits the sequence into two sub-sequences in the middle, and computes the normalized increase (decrease) of the second subsequence compared to the first one, on a per-element basis. The normalized changes are then averaged. We empirically set the lower and upper bound of c to 0.5 and 1.25, respectively.

6.3 Multipath Scheduling

Upon receiving the block stream from the server, Habitus splits it over 802.11ac and ad paths. Regarding the splitting mechanism, a straightforward solution is MPTCP [41] or its variants for wireless networks [48, 71, 82]. We reject this design due to three reasons. First, MPTCP exposes to the upper layer a single logical connection whose byte stream is delivered in-order; Habitus instead decouples the two paths that independently deliver blocks. Second, adapting MPTCP's scheduler to Habitus requires kernel modifications. Third, MPTCP is known to cause issues on ad/ac (*e.g.,* throughput drops over ad due to periodical network scans on ac [71]).

To avoid the above issues, Habitus establishes two single-path connections[3] and performs scheduling in the user space. The scheduling logic is straightforward: transmitting high-priority and low-priority blocks over 802.11ac and ad, respectively, with the reason explained in §6.1. Specifically, the server sends to Habitus's edge node the metadata (headers)

[3]Our prototype uses TCP. A better design would be using QUIC [54] to avoid head-of-line blocking across blocks within a path under packet losses.

of multiple blocks with the same playback deadline in a single bundle, followed by parallel streams of individual blocks' content. As the blocks' content arrives, the edge distributes them over the two paths according to their priority fields in the metadata and each path's estimated bandwidth. A block only usually uses one path, but a small number of blocks may be split over both paths if one path's bandwidth budget is insufficient. Once a block arrives at the client, it will be immediately passed to the application for decoding and rendering. The server-side transmission, edge-side forwarding, and client-side reception are pipelined.

6.4 Example Use Cases of Immersive Apps

Habitus can be easily integrated with a wide range of immersive applications and content formats as exemplified below.

360° Videos. State-of-the-art 360° video systems [46, 80, 90] spatially segment each panoramic video chunk into tiles. Tiles are selectively transmitted based on the viewport. Each tile naturally maps to a block in Habitus, and its priority can be set to the probability that it will appear in the viewport. Many existing systems already have this metric calculated [35, 66].

Volumetric Videos. The above viewport adaptation technique and henceforth the block/priority assignment also applies to volumetric videos, where a 2D tile becomes a 3D cube consisting of 3D points. Alternatively, since each volumetric frame consists of unstructured points, it can be arbitrarily split into multiple layers each constituting a block in Habitus. A "base layer" with a low-density point cloud can be assigned a high priority; one or more "enhancement layers" each encompassing additional details can be assigned lower priorities.[4]

Generic VR. Networked VR systems either stream raw 3D models [55, 57, 88, 89] or rendered 2D scenes [36, 58]. Depending on the content format, a block can be either a 3D model (or part of it) or a rendered 2D patch. The are several studies/systems on determining the priority of VR content, such as those based on foreground/background [53, 57, 85], the viewing distance [44, 61], and user gaze behaviors [36].

7 Implementation

Our implementation consists of three parts: (1) the main Habitus middleware in 3.5K LoC; (2) a 802.11 throughput measurement module plugged into Habitus; (3) two sample applications using the Habitus API (5.2K LoC, 4.4K LoC).

The Main Habitus System is implemented in C++ and Python. We use ROG Phone II [16] and plug it into a low-end VR headset [7] (costs $26) as the client-side device and the same server used in §4.2 as the edge node. On the client side, we use Linux **`iw`** [10] to monitor 802.11 signal strength; we use ARCore [3] for 6-DoF motion tracking (based on IMU and camera data [4]). On the edge side, we use PyTorch-1.10.0 [15] for training and transferring our models. For inference, we save the models in TorchScript [18] for C++ execution. We implement the body pose estimator over zed-openpose [26], using a pre-trained model [14, 34] to detect 2D poses. We pipeline the body pose estimation stages (capture, 2D detection, 2D-to-3D mapping). We use the object detection module in ZED SDK 3.8.2 [25] to detect passersby.

[4]A similar concept called Scalable Video Encoding (SVC [73]) can be applied to 2D content, albeit at a higher encoding overhead.

802.11 Throughput Measurement Module. Compared to traditional 2D video traffic, immersive content traffic is highly bursty [36, 47, 55]. This poses several challenges for 802.11 (in particular, mmWave) throughput measurement. We thus implement an 802.11 throughput measurement module using Libpcap-1.10.1 [8]. We detail its design in Appendix D.1.

Two Volumetric Streaming Applications using Habitus. To demonstrate how Habitus can benefit real immersive applications, we build two volumetric (point cloud) streaming systems with different logic and complexity using the Habitus API. The first app (**App1**) employs layered encoding of point clouds (§6.4) so each data block corresponds to a (frame, layer) pair. The second app (**App2**) performs viewport adaptation (§6.4) by spatially segmenting each frame (*i.e.,* point cloud) into cubical cells, so each data block constitutes a (frame, cell) pair. We build the first app from scratch in 5.2K LoC, and the second app replicating ViVo [47], a state-of-the-art, visibility-aware volumetric streaming system. For ViVo, we only change 47 LoC for Habitus integration. Both apps are equipped with the same bitrate adaptation algorithm whose details can be found in Appendix D.2.

8 Evaluation

8.1 Experimental Setup

Dataset, Devices, and Models. We use the dataset collected in §4.2 for our controlled experiments. The devices are the same as those used in §7 and §4.2. In §8.2, we use {*GBDT, BP8, RNN8, RNN20, Seq2Seq*} {*w/, w/o*} *Pose* models, and three prediction windows (*pw*): {0.5, 1, 2} secs. Experiments in other sections use *Seq2Seq* {*w/, w/o*} *Pose* with *pw*=1 sec.

Controlled Experiments. To ensure the reproducibility, during our controlled experiments, we replay the headset's 6-DoF motion traces and the signal strength traces on the smartphone, which is connected to the edge via real 802.11ac/ad links. On the edge side, we replay the RGB-D videos for online full-body pose estimation. We emulate ac/ad throughput traces by Linux **`tc`** [9]. We fix the smartphone static in LoS to the 802.11ad AP to keep a good mmWave signal for throughput emulation. We do not add additional RTT since the client already connects to the edge via real wireless links.

Volumetric Videos. We use three point-cloud-based volumetric videos (denoted as *V1*, *V2*, and *V3*, respectively) throughout our evaluation. {*V1*, *V2*, *V3*} has {2612, 2700, 3000} frames ({~87, 90, 100} secs), respectively. Each frame of them is split into 64 data blocks (§6.1) and details can be found in §7 and Appendix D.2. All the videos are at 30 FPS, encoded into ten quality levels. The highest bitrates are {570, 687, 738} Mbps for {*V1*, *V2*, *V3*}, respectively. Unless otherwise mentioned, the results reported in the remainder of

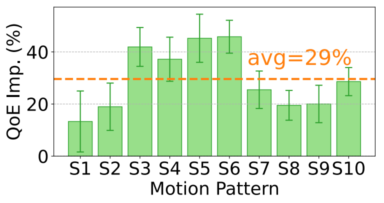


Figure 8: Per motion pattern QoE improvement of *Seq2Seq w/ Pose* over *Seq2Seq w/o Pose*.

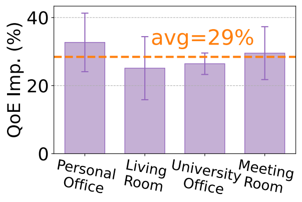


Figure 9: Per location QoE improvement of *Seq2Seq w/ Pose* over *Seq2Seq w/o Pose*.

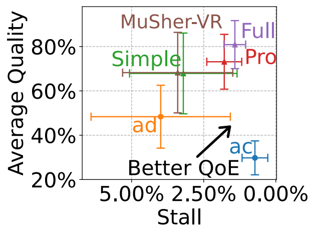


Figure 10: Quality vs. Stall of Habitus variants.

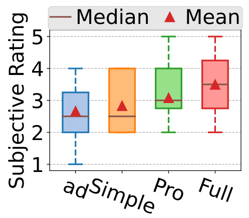


Figure 11: Subjective ratings of 12 users.

this section are generated using all three videos.

Roadmap and Metrics. §8.2 evaluates 802.11ad throughput prediction error reduction brought by full-body pose. §8.3 focuses on the QoE improvement brought by full-body pose. We assess the QoE using the QoE model for point cloud from [88]. It is a linear combination of frame quality, inter-frame & intra-frame quality switch, and stall. §8.4 and §8.7 evaluate the end-to-end performance (quality and stall) of Habitus. §8.5 conducts a user study to examine real users' QoE. §8.6 evaluates how our design handles unseen changes. §8.8 provides additional micro benchmarks. Except for §8.7, we use **App1** in §7 (details in Appendix D.2) for evaluation.

8.2 802.11ad Throughput Prediction Error

Recall that in §4.3, we perform 10-fold cross validation for {*GBDT, BP8, RNN8, RNN20, Seq2Seq*} {*w/, w/o*} *Pose* models on each {Location, User}'s dataset, with three *pws* {0.5, 1, 2} secs. As Figure 4 shows, leveraging full-body pose effectively reduces mmWave throughput prediction error for all these models, ranging from 5% (*GBDT*) to 29% (*RNN20*) in MAE and 5% (*GBDT*) to 25% (*RNN20*) in RMSE, respectively.

8.3 QoE over 802.11ad Network

We evaluate how full-body pose guided mmWave throughput prediction improves **App1**'s QoE through controlled experiments. First, for each {Location, User}, we train a *Seq2Seq w/ Pose* and a *Seq2Seq w/o Pose* model, respectively. We then run **App1** over a single-path 802.11ad network. For each data trace, we run the experiment twice, using *Seq2Seq* {*w/, w/o*} *Pose* model, respectively. We log the quality for each data block (§8.1) and the stall for each frame to assess the QoE. Figure 8 shows the QoE improvement by leveraging full-body pose for each motion pattern across all data traces. We have two findings. First, leveraging full-body pose effectively improves the QoE by 29% on average for all our motion patterns. Second, the QoE improvement varies across different motion patterns, from 13.30% (S1) to 45.82% (S6). The full-body pose does not help much for S1, S2, S8, and S9. This is due to two reasons. First, in S1 and S2, the user does not make translational movement; this reduces the effectiveness of the pose. Second, in S8 and S9, the LoS between the smartphone and the 802.11ad AP is well maintained; this makes the throughput prediction easier compared to other motion patterns. Figure 9 presents the QoE improvement for each location in Figure 3. The QoE improvement remains similar between simple (*Personal Office* and *Living Room*) and complex locations (*University Office* and *Meeting Room*).

Variant	Predict ad?	*trend-aware*?	Scheduler
ac	No	N/A	single-path ac
ad	No	N/A	single-path ad
Simple	No	N/A	ours in §6.3
Pro	*Seq2Seq w/o Pose*	Yes	ours in §6.3
Full	*Seq2Seq w/ Pose*	Yes	ours in §6.3

Table 2: Habitus variants.

8.4 End-to-end Performance of Habitus

We evaluate the end-to-end performance, including the content quality and stall, of diverse Habitus variants using all {Location, User}'s data and **App1** in §7.

Habitus Variants. Table 2 summarizes 5 Habitus variants. We consider two *single-path* variants, *ac* and *ad*, that only schedule data to ac and ad, respectively, without ad throughput prediction. We also consider three *multipath* variants, all using the multipath scheduler from §6.3: the *Simple* variant does not utilize 802.11ad throughput prediction; the *Pro* and *Full* variant apply the *Seq2Seq w/o Pose* and *Seq2Seq w/ Pose* model, respectively, for ad throughput prediction. Both *Pro* and *Full* enable the *trend-aware* feature (§6.2). As shown in Figure 10, compared to *ac* and *ad*, *Simple* boosts the quality (normalized by the highest quality level) by 127.88% and 40.36%, respectively. *Simple* incurs a much higher stall compared to *ac* because the ad network is highly fluctuating and *Simple* blindly uses it without predicting its future condition. Compared to *Simple*, *Pro* boosts the quality by 7.75% and reduces the stall by 44.25%, thanks to the ad throughput prediction and the *trend-aware* multipath scheduler. Compared to *Pro*, *Full* enhances the ad throughput prediction accuracy by using full-body poses, leading to a further stall reduction of 20.55% and video quality improvement of 10.58%.

Habitus vs. Existing Approaches. We compare Habitus with MuSher [71], a recently proposed MPTCP scheduler for ac/ad networks. Musher periodically probes the ratio between the current ad and ac throughput, and splits the traffic accordingly. In each probe, it tries to increase and decrease the radio, and greedily selects the direction to move based on the aggregated ac/ad throughput measurement. It also has a SCAN component to mitigate the negative impact of network scanning and a BLOCKAGE component to accelerate TCP congestion window recovery after an ad blockage event.

We implement MuSher's scheduling algorithm in the application layer. We plug it into Habitus and denote it as *MuSher-VR*. We do not implement the SCAN component because we use establish separate TCP connections over ac/ad links so network scans on one interface do not affect the other one. We repeat the same experiment on *MuSher-VR*.As shown in Figure 10, compared to *MuSher-VR*, Habitus (*Full*) significantly reduces the stall by 58.24% and boosts the quality by 18.52%. Habitus outperforms *MuSher-VR* due to two reasons. First, *MuSher-VR* incurs stalls when it aggressively probes the scheduling ratio by scheduling more data to one path than its actual capacity. Habitus instead takes a prediction-based approach to avoid the stall caused by aggressive probing. Second, Habitus prioritizes using the ac path and opportunistically uses the ad path if possible. In contrast, *MuSher-VR* lacks such prioritization. It schedules the data to the ac/ad paths based on a calculated ratio that ideally should converge to the ratio between ac/ad throughput. However, under the constant movement of the viewer, the actual instantaneous ratio may significantly deviate from the calculated ratio, leading to stalls or under-utilizing the ad path.

8.5 User Study

We conduct an IRB-approved user study at *University Office* (Figure 3) to assess real users' QoE when using **App1** (§7). We recruit 12 users with various demographics.[5] We let each user watch a video randomly selected from our test videos and then subjectively rate the watching experience through 5 choices {1=very bad, 2=bad, 3=fair, 4=good, 5=very good}. Each user performs the above assessment four times. Each time, we randomly plug a Habitus variant into **App1**. We consider four variants: *ad*, *Simple*, *Pro*, and *Full* as listed in Table 2. Before each user's trial begins, we collect 2 minutes' worth of data from the user to transfer a pre-trained model to the user. We let the user wear a low-end VR headset [7] with a ROG Phone II plugged into it. The user can freely make 6-DoF motions in the room during the study. As shown in Figure 11, compared to {*ad*, *Simple*, *Pro*}, *Full* improves the average subjective rating by {0.83, 0.67, 0.42} (in the scale of 1 to 5), respectively. Note that the best scheme (*Full*) has an average rating of 3.50 (between fair and good), likely because of the hardware limitation of the VR headset (costs $26) compared to a full-fledged VR headset.

8.6 Handling Unseen Changes

We evaluate the three techniques introduced in §5.3 for handling unseen changes. We reuse the datasets {$\mathbb{T}_B$, $\mathbb{T}_A$, $\mathbb{E}^A$} and models {$\mathbb{M}_B$, $\mathbb{M}_A$} introduced in §5.1. The experiments use the *Seq2Seq w/ Pose* model on an NVIDIA 1660Ti GPU.

Offline Transfer Learning. For **C1** and **C2**, we compare the training time between (1) transferring from $\mathbb{M}_B$ to $\widetilde{\mathbb{M}}_{B\to A}$ and (2) training a new model $\widetilde{\mathbb{M}}_A$ from scratch after the

[5] *Gender*: Male: 7, Female: 5; *Height*: <1.65m: 3, 1.65-1.75m: 5, >1.75m: 4. The subjects' ages vary between 20 and 30. 8 out of them do not have prior experience on watching volumetric videos.

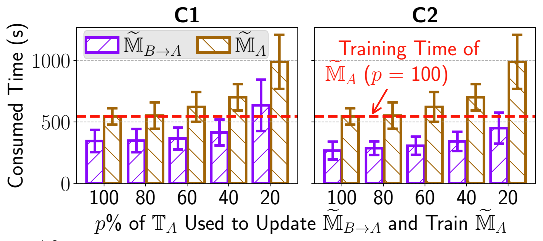


Figure 12: Time consumption for (1) transferring from $\mathbb{M}_B$ to $\widetilde{\mathbb{M}}_{B\to A}$ and (2) training a new model $\widetilde{\mathbb{M}}_A$ from scratch after the change, using p% of $\mathbb{T}_A$. (Left: **C1**; Right: **C2**).

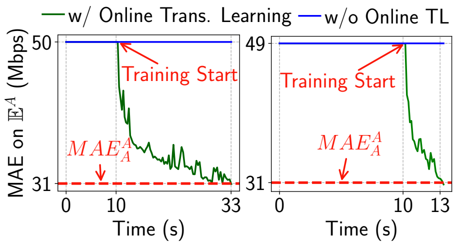


Figure 13: Online training cases (Left: **C3**; Right: **C4**).

change. $\widetilde{\mathbb{M}}_{B\to A}$ and $\widetilde{\mathbb{M}}_A$ denote the transferred model and the built-from-scratch model, respectively. For both $\widetilde{\mathbb{M}}_{B\to A}$ and $\widetilde{\mathbb{M}}_A$, we use $p\% \in \{100, 80, 60, 40, 20\}\%$ of the samples in $\mathbb{T}_A$ to transfer (train) them. Their training stops when the prediction accuracy evaluated on $\mathbb{E}^A$ reaches MAE$_A^A$ (*i.e.,* $\mathbb{M}_A$'s prediction accuracy on $\mathbb{E}^A$). We find that the training always converges even when p is as low as 20%. We show the measured training time in Figure 12, where the dashed red line marks the training time of $\widetilde{\mathbb{M}}_A$ with p = 100%. As shown, to achieve the same evaluation accuracy, $\widetilde{\mathbb{M}}_{B\to A}$ significantly reduces the training time by 36% to 41% (48% to 55%) for **C1** (**C2**) across all five p values, compared to $\widetilde{\mathbb{M}}_A$. In particular, training $\widetilde{\mathbb{M}}_{B\to A}$ using only 40% (20%) of the samples in $\mathbb{T}_A$ is still faster than training $\widetilde{\mathbb{M}}_A$ using all the samples in $\mathbb{T}_A$ for **C1** (**C2**). The reason, as explained in §5.3, is that $\widetilde{\mathbb{M}}_{B\to A}$ effectively reuses the invariant knowledge (*e.g.,* the physical property of mmWave and the throughput distribution in certain positions) that is already present in $\mathbb{M}_B$.

Online Transfer Learning. For **C3** and **C4**, we measure the time consumption when $\widetilde{\mathbb{M}}_{B\to A}$ first converges to the target prediction accuracy MAE$_A^A$ on $\mathbb{E}^A$. To accurately emulate the online setting in a reproducible manner, when training $\widetilde{\mathbb{M}}_{B\to A}$, we feed $\mathbb{T}_A$'s data at the same pace as the real-world training data collection rate. The results indicate that it takes on average 32 (15) secs for the training (*i.e.,* online transfer learning) to converge on **C3** (**C4**), with a standard deviation of 11 (12) secs. The convergence time includes the initial 10-sec bootstrapping (§5.3). Figure 13 shows case studies for **C3** and **C4**. Note that without online transfer learning, the prediction error on $\mathbb{E}^A$ will never decrease.

Dynamic Change Handling. For **C5**, we evaluate the end-to-end performance of the Habitus-supported volumetric streaming app (**App1**) *with* and *without* vision-based dynamic change handling (§5.3). The controlled experiment is conducted over a single-path 802.11ad network at *University*

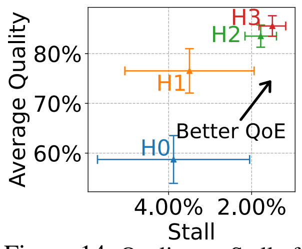


Figure 14: Quality vs. Stall of *H0* to *H3*.

H0	Vanilla ViVo [47] w/ single-path 802.11ad
H1	*H0* and multipath 802.11ac and 802.11ad
H2	*H1* and ad throughput prediction *w/o Pose*
H3	*H2* and ad throughput prediction *w/ Pose*

Table 3: Apply our solution to ViVo [47] (cumulative).

Office with the robotic arm. We pre-train the vision-based object detection model in a separate experiment so the model can reliably detect the aluminum-foil-covered box manipulated by the robotic arm (§5.1). We use $\mathbb{M}_B$ as the throughput prediction model. The results indicate that vision-based dynamic change handling reduces the stall by 7% with a video quality reduction of only 2.2%.

8.7 Applying Habitus to Existing Systems

We integrate Habitus into **App2** (ViVo [47], an existing volumetric streaming system) by changing only 47 LoC (details in Appendix D.2). ViVo adopts visibility-aware streaming by only fetching content that will appear in the future viewport. As listed in Table 3, *H0* is our comparison baseline: the vanilla ViVo over single-path ad. *H1* to *H3* involve key components of Habitus. Figure 14 shows the quality and stall of *H0* to *H3* across our dataset. As shown, by cumulatively enabling Habitus's components from *H1* to *H3*, both the average video quality and stall improve accordingly. Compared to *H0*, *H3* reduces the stall by 61% and improves the average quality by 46%. In addition, compared to not using pose (*H2*), full-fledged Habitus (*H3*) reduces the stall by 15.75% while slightly boosting the quality by 2.44%. The absolute stall rate of *H3* is 1.67%, meaning that the user encounters less than 0.9 secs of stall per minute on average.

8.8 Micro Benchmarks and Resource Usage

We run experiments to show: (1) The GPU memory usage (~4.7G out of 11G on 2080Ti) of Habitus is acceptable. The average processing time of pose estimation and throughput prediction is 27ms and 3.5ms on 2080Ti, respectively. (2) Compared to single-path ac, the additional energy usage and heat increase of **App1** using Habitus are only 1% and 1.3°C respectively. The details can be found in Appendix E.

9 Related Work

Immersive Content Delivery. We elaborate on some immersive content delivery systems mentioned in §6.4 (more can be found in [76]). Flare [66] and ViVo [47] apply viewport adaptation to optimize mobile 360° and volumetric videos streaming, respectively. M5 [89] investigates volumetric video streaming using adaptive mmWave beamforming. InstantReality [36] introduces a perceptual-aware approach to enhance VR media streaming. As a middleware framework, Habitus can be integrated into most of the above systems to enhance the application QoE (we have conducted a case study for ViVo in §8.7). Also note that Habitus is orthogonal to some VR systems (*e.g.,* MoVR [27]) that enhances the PHY layer (§1).

mmWave Throughput Prediction. Recent measurement studies have explored the feasibility of predicting mmWave throughput for various radios, such as commercial mmWave 5G [63], 802.11ad [28], and 802.11ay [83]. Lumos5G [63] establishes a composable machine learning framework to predict mmWave 5G throughput. Wu *et al.* uses Markov chain to predict the link quality of 802.11ay, based on the headset's motion data [83]. Aggarwal *et al.* conducts a measurement study on using a smartphone's motion sensor data to predict 802.11ad throughput [28]. They only consider 2-DoF (with a radio mounted on a guided rail) and LoS scenarios. None of the above studies employs the full-body pose, which we found to be an important feature for improving the prediction accuracy. Also, none of them conducts in-depth investigations on how to handle unseen changes as we do.

Multipath TCP Support for 802.11ac/ad. Despite a plethora of works on WiFi/cellular multipath [48,82,92], there are only a few studies on dual-band 802.11ac/ad multipath networking. MUST [74] predicts the best 60GHz beam and PHY rate setting, and switches between ac/ad links accordingly. We discussed MuSher [71], an MPTCP scheduler for 802.11ac/ad and compared it with Habitus in §8.4. Compared to the above works, Habitus is an application-layer solution designed specifically for immersive content delivery.

Improving mmWave Network Performance at PHY layer. There is rich literature on improving mmWave performance at the PHY layer, such as efficient beam selection [79], LiDAR-assisted beam management [81], and beam relay through smart metasurface [38], to name a few. Unlike the above, Habitus aims at optimizing the upper-layer network protocol stack for immersive content delivery without requiring modifying PHY-layer protocols.

10 Limitations and Concluding Remarks

Limitations. First, our prototype and experiments only use 802.11ac+ad. We expect Habitus's high-level design to also work with other radio technologies such as 4G + mmWave 5G, but field tests are needed to verify this claim. Second, Habitus's reaction to unseen changes could be further improved. We plan to employ more advanced techniques such as parameter sharing [93] to speed up transfer learning. Third, we focus on the single-user use case. Extending Habitus to multiple viewers will involve additional challenges such as dealing with the interplay among the viewers.

Despite the limitations, we have demonstrated through a working system and rich real-world data that, full-body-pose guided throughput prediction and joint use of omnidirectional+mmWave radio can significantly improve the QoE (up to 72%) for immersive applications. Furthermore, by fusing transfer learning and vision-based object recognition, Habitus can smoothly adapt to unseen changes.

Acknowledgments

We thank the anonymous reviewers and our shepherd Ravi Netravali for their insightful comments. This work was supported in part by NSF CNS-2212296 and CNS-2235049.

References

[1] AWS Wavelength. https://aws.amazon.com/wavelength/.

[2] Demo Video for Dynamic Change Injection. https://docs.google.com/presentation/d/e/2PACX-1vTokmujVFURIX6YBgyYz9aAjHUV9Ajr1a6dKqYHLmRbrcnI92flVA0O4TLOD338YXXUwyZOjIsSc0Hh/pub?start=false&loop=false&delayms=3000&slide=id.p.

[3] Google ARCore. https://developers.google.com/ar.

[4] Google ARCore Motion Tracking. https://developers.google.com/ar/develop/fundamentals.

[5] Habitus Code. https://github.com/zhan6841/Habitus-open-sourced-code.git.

[6] Habitus Data. https://drive.google.com/drive/folders/1B7rlIQG6ycg2OFML3CHxoPNge56xPhq4.

[7] KCXGHYI VR Headset. https://bit.ly/41UMh19.

[8] Libpcap. https://www.tcpdump.org.

[9] Linux TC Man Page. https://linux.die.net/man/8/tc.

[10] Linux Wireless. https://wireless.wiki.kernel.org/en/users/documentation/iw.

[11] Magic Leap. https://www.magicleap.com/device.

[12] Metaverse. https://about.facebook.com/metaverse/.

[13] Netgear Nighthawk X10. https://www.netgear.com/support/product/r9000.aspx.

[14] OpenPose Codebase. https://github.com/CMU-Perceptual-Computing-Lab/openpose.

[15] Pytorch. https://pytorch.org.

[16] ROG Phone II. https://rog.asus.com/phones/rog-phone-ii-model.

[17] Rokoko Smartsuit Pro II. https://www.rokoko.com/products/smartsuit-pro.

[18] TorchScript. https://pytorch.org/docs/stable/jit.html.

[19] Tp-link Archer A7. https://www.tp-link.com/us/home-networking/wifi-router/archer-a7.

[20] VIVE Cosmos Series. https://www.vive.com/us/product/#cosmos%20series.

[21] VIVE Pro Series. https://www.vive.com/us/product/#pro%20series.

[22] VIVE Tracker 3.0. https://www.vive.com/us/accessory/tracker3.

[23] VIVE Wireless Adapter. https://www.vive.com/us/accessory/wireless-adapter.

[24] ZED 2i Camera. https://www.stereolabs.com/zed-2i.

[25] ZED 3D Object Detection. https://www.stereolabs.com/docs/object-detection/.

[26] zed-openpose Codebase. https://github.com/stereolabs/zed-openpose.

[27] O. Abari, D. Bharadia, A. Duffield, and D. Katabi. Enabling {High-Quality} untethered virtual reality. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 531–544, 2017.

[28] S. Aggarwal, Z. Kong, M. Ghoshal, Y. C. Hu, and D. Koutsonikolas. Throughput prediction on 60 ghz mobile devices for high-bandwidth, latency-sensitive applications. In *International Conference on Passive and Active Network Measurement*, pages 513–528. Springer, 2021.

[29] B. Artacho and A. Savakis. Unipose: Unified human pose estimation in single images and videos. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7035–7044, 2020.

[30] S. Bak, E. Corvee, F. Bremond, and M. Thonnat. Person re-identification using spatial covariance regions of human body parts. In *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 435–440. IEEE, 2010.

[31] V. Bazarevsky and I. Grishchenko. On-device, real-time body pose tracking with mediapipe blazepose. *Google AI Blog*, 2020.

[32] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[33] J. Butepage, M. J. Black, D. Kragic, and H. Kjellstrom. Deep representation learning for human motion prediction and classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6158–6166, 2017.

[34] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[35] J. Chen, X. Qin, G. Zhu, B. Ji, and B. Li. Motion-prediction-based wireless scheduling for multi-user panoramic video streaming. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2021.

[36] S. Chen, B. Duinkharjav, X. Sun, L.-Y. Wei, S. Petrangeli, J. Echevarria, C. Silva, and Q. Sun. Instant reality: Gaze-contingent perceptual optimization for 3d virtual reality streaming. *IEEE Transactions on Visualization and Computer Graphics*, 28(5):2157–2167, 2022.

[37] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, et al. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778. IEEE, 2018.

[38] K. W. Cho, M. H. Mazaheri, J. Gummeson, O. Abari, and K. Jamieson. mmwall: A reconfigurable metamaterial surface for mmwave networks. In *Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications*, pages 119–125, 2021.

[39] D. R. Cox and A. Stuart. Some quick sign tests for trend in location and dispersion. *Biometrika*, 42(1/2):80–95, 1955.

[40] Q. Cui, H. Sun, Y. Kong, and X. Sun. Deep human dynamics prior. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 4371–4379, 2021.

[41] A. Ford, C. Raiciu, M. J. Handley, O. Bonaventure, and C. Paasch. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 8684, Mar. 2020.

[42] X. Gao, L. Dai, S. Han, I. Chih-Lin, and R. W. Heath. Energy-efficient hybrid analog and digital precoding for mmwave mimo systems with large antenna arrays. *IEEE Journal on Selected Areas in Communications*, 34(4):998–1009, 2016.

[43] Y. Ghasempour, C. R. Da Silva, C. Cordeiro, and E. W. Knightly. Ieee 802.11 ay: Next-generation 60 ghz communication for 100 gb/s wi-fi. *IEEE Communications Magazine*, 55(12):186–192, 2017.

[44] E. Gobbetti and F. Marton. Far voxels: a multiresolution framework for interactive rendering of huge complex 3d models on commodity graphics platforms. In *ACM SIGGRAPH 2005 Papers*, pages 878–885. 2005.

[45] Y. Guan, X. Hou, N. Wu, B. Han, and T. Han. Deepmix: mobility-aware, lightweight, and hybrid 3d object detection for headsets. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, pages 28–41, 2022.

[46] Y. Guan, C. Zheng, X. Zhang, Z. Guo, and J. Jiang. Pano: Optimizing 360 video streaming with a better understanding of quality perception. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 394–407. 2019.

[47] B. Han, Y. Liu, and F. Qian. Vivo: Visibility-aware mobile volumetric video streaming. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–13, 2020.

[48] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan. Mp-dash: Adaptive video streaming over preference-aware multipath. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pages 129–143, 2016.

[49] A. Hassan, A. Narayanan, A. Zhang, W. Ye, R. Zhu, S. Jin, J. Carpenter, Z. M. Mao, F. Qian, and Z.-L. Zhang. Vivisecting mobility management in 5g cellular networks. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 86–100, 2022.

[50] J. Jiang, V. Sekar, and H. Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 97–108, 2012.

[51] W. M. Kouw and M. Loog. An introduction to domain adaptation and transfer learning. *arXiv preprint arXiv:1812.11806*, 2018.

[52] T. Kucherenko, J. Beskow, and H. Kjellström. A neural network approach to missing marker reconstruction in human motion capture. *arXiv preprint arXiv:1803.02665*, 2018.

[53] Z. Lai, Y. C. Hu, Y. Cui, L. Sun, N. Dai, and H.-S. Lee. Furion: Engineering high-quality immersive virtual reality on today's mobile devices. *IEEE Transactions on Mobile Computing*, 19(7):1586–1602, 2019.

[54] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, et al. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the conference of*

the ACM special interest group on data communication, pages 183–196, 2017.

[55] K. Lee, J. Yi, Y. Lee, S. Choi, and Y. M. Kim. Groot: a real-time streaming system of high-fidelity volumetric videos. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2020.

[56] L.-H. Lee, T. Braud, P. Zhou, L. Wang, D. Xu, Z. Lin, A. Kumar, C. Bermejo, and P. Hui. All one needs to know about metaverse: A complete survey on technological singularity, virtual ecosystem, and research agenda. *arXiv preprint arXiv:2110.05352*, 2021.

[57] X. Liu, C. Vlachou, F. Qian, C. Wang, and K.-H. Kim. Firefly: Untethered multi-user vr for commodity mobile devices. In *Proceedings of the 2020 USENIX Conference on Usenix Annual Technical Conference*, pages 943–657, 2020.

[58] Y. Liu, B. Han, F. Qian, A. Narayanan, and Z.-L. Zhang. Vues: Practical mobile volumetric video streaming through multiview transcoding. *ACM MobiCom 2022*, 2022.

[59] Y. Ma, H. Tian, X. Liao, J. Zhang, W. Wang, K. Chen, and X. Jin. Multi-objective congestion control. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 218–235, 2022.

[60] J. Martinez, M. J. Black, and J. Romero. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2891–2900, 2017.

[61] S. R. Musse, C. Babski, T. Capin, and D. Thalmann. Crowd modelling in collaborative virtual environments. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 115–123, 1998.

[62] S. Mystakidis. Metaverse. *Encyclopedia*, 2(1):486–497, 2022.

[63] A. Narayanan, E. Ramadan, R. Mehta, X. Hu, Q. Liu, R. A. Fezeu, U. K. Dayalan, S. Verma, P. Ji, T. Li, et al. Lumos5g: Mapping and predicting commercial mmwave 5g throughput. In *Proceedings of the ACM Internet Measurement Conference*, pages 176–193, 2020.

[64] J. Navratil and R. L. Cottrell. Abwe: A practical approach to available bandwidth estimation. In *Proceedings of the 4th Passive and Active Measurement Workshop PAM 2003*. Citeseer, 2003.

[65] T. Nitsche, C. Cordeiro, A. B. Flores, E. W. Knightly, E. Perahia, and J. C. Widmer. Ieee 802.11 ad: directional 60 ghz communication for multi-gigabit-per-second wi-fi. *IEEE Communications Magazine*, 52(12):132–141, 2014.

[66] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan. Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 99–114, 2018.

[67] V. Raghavan, A. Partyka, L. Akhoondzadeh-Asl, M. A. Tassoudji, O. H. Koymen, and J. Sanelli. Millimeter wave channel measurements and implications for phy layer design. *IEEE Transactions on Antennas and Propagation*, 65(12):6521–6533, 2017.

[68] D. Rempe, T. Birdal, A. Hertzmann, J. Yang, S. Sridhar, and L. J. Guibas. Humor: 3d human motion model for robust pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11488–11499, 2021.

[69] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrell. pathchirp: Efficient available bandwidth estimation for network paths. In *Passive and active measurement workshop*, 2003.

[70] A. Rutkowska. Properties of the cox–stuart test for trend in application to hydrological series: the simulation study. *Communications in Statistics-Simulation and Computation*, 44(3):565–579, 2015.

[71] S. K. Saha, S. Aggarwal, R. Pathak, D. Koutsonikolas, and J. Widmer. Musher: An agile multipath-tcp scheduler for dual-band 802.11 ad/ac wireless lans. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2019.

[72] K. Sato, T. Manabe, T. Ihara, H. Saito, S. Ito, T. Tanaka, K. Sugai, N. Ohmi, Y. Murakami, M. Shibayama, et al. Measurements of reflection and transmission characteristics of interior structures of office building in the 60-ghz band. *IEEE transactions on antennas and propagation*, 45(12):1783–1792, 1997.

[73] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the h. 264/avc standard. *IEEE Transactions on circuits and systems for video technology*, 17(9):1103–1120, 2007.

[74] S. Sur, I. Pefkianakis, X. Zhang, and K.-H. Kim. Wifi-assisted 60 ghz wireless networks. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pages 28–41, 2017.

[75] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[76] J. van der Hooft, H. Amirpour, M. T. Vega, Y. Sanchez, R. Schatz, T. Schierl, and C. Timmerer. A tutorial on immersive video delivery: From omnidirectional video to holography. *IEEE Communications Surveys & Tutorials*, 2023.

[77] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko. Sequence to sequence-video to text. In *Proceedings of the IEEE international conference on computer vision*, pages 4534–4542, 2015.

[78] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Scaled-YOLOv4: Scaling cross stage partial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13029–13038, June 2021.

[79] S. Wang, J. Huang, and X. Zhang. Demystifying millimeter-wave v2x: Towards robust and efficient directional connectivity under high mobility. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020.

[80] S. Wang, S. Yang, H. Li, X. Zhang, C. Zhou, C. Xu, F. Qian, N. Wang, and Z. Xu. Salientvr: saliency-driven mobile 360-degree video streaming with gaze information. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 542–555, 2022.

[81] T. Woodford, X. Zhang, E. Chai, K. Sundaresan, and A. Khojastepour. Spacebeam: Lidar-driven one-shot mmwave beam management. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, pages 389–401, 2021.

[82] J. Wu, C. Yuen, B. Cheng, M. Wang, and J. Chen. Streaming high-quality mobile video with multipath tcp in heterogeneous wireless networks. *IEEE Transactions on Mobile Computing*, 15(9):2345–2361, 2015.

[83] Z. Wu, C.-Y. Huang, and P. Ramanathan. Measuring millimeter wave based link bandwidth fluctuations during indoor immersive experience. *IEEE Networking Letters*, 2022.

[84] G. Xia, H. Sun, B. Chen, Q. Liu, L. Feng, G. Zhang, and R. Hang. Nonlinear low-rank matrix completion for human motion recovery. *IEEE Transactions on Image Processing*, 27(6):3011–3024, 2018.

[85] C. Xie, X. Li, Y. Hu, H. Peng, M. Taylor, and S. L. Song. Q-vr: system-level design for future mobile collaborative virtual reality. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 587–599, 2021.

[86] T. Xu, B. Han, and F. Qian. Analyzing viewport prediction under different vr interactions. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, pages 165–171, 2019.

[87] Z. Yuan, H. Venkataraman, and G.-M. Muntean. ibe: A novel bandwidth estimation algorithm for multimedia services over ieee 802.11 wireless networks. In *IFIP/IEEE International Conference on Management of Multimedia Networks and Services*, pages 69–80. Springer, 2009.

[88] A. Zhang, C. Wang, B. Han, and F. Qian. {YuZu}:{Neural-Enhanced} volumetric video streaming. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 137–154, 2022.

[89] D. Zhang, P. Zhou, B. Han, and P. Pathak. M5: Facilitating multi-user volumetric content delivery with multi-lobe multicast over mmwave. 2022.

[90] W. Zhang, F. Qian, B. Han, and P. Hui. Deepvista: 16k panoramic cinema on your mobile device. In *Proceedings of the Web Conference 2021*, pages 2232–2244, 2021.

[91] P. Zhao, C. X. Lu, J. Wang, C. Chen, W. Wang, N. Trigoni, and A. Markham. mid: Tracking and identifying people with millimeter wave radar. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 33–40. IEEE, 2019.

[92] X. Zhu, J. Sun, X. Zhang, Y. E. Guo, F. Qian, and Z. M. Mao. Mpbond: efficient network-level collaboration among personal mobile devices. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, pages 364–376, 2020.

[93] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

Appendices

A Additional Details of mmWave Throughput Prediction Study

A.1 The OpenPose Format

Typically, a full-body pose can be represented by a set of key points where each key point corresponds to a joint/part of the human body. We customize the OpenPose [34] BODY_25 format – a popular format used in the computer vision community – to represent the full-body pose, as shown in Figure 15.

Location	Moving Area (m^2)	Room Space (m^3)
Personal Office	2.5×2.5	3.3×2.5×2.5
Living Room	3.0×2.7	6.6×5.4×2.5
University Office	4.0×2.0	9.0×7.0×3.0
Meeting Room	4.0×3.0	6.5×6.5×2.8

Table 4: The moving area and room space of the four locations.

We discard some key points (*i.e.,* eyes, ears, toes, and heels) that have little contribution to the full-body pose.

A.2 Missing Key Point Estimation

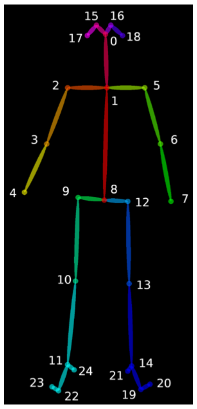
To estimate a missing key point's 3D coordinate on the fly, we consider two baseline approaches: simply reusing its most recently captured 3D coordinate (Method 1), and linearly extrapolating its coordinate using its historical trajectory (Method 2 [40,52,84]). To assess them, we select a subset of our dataset with no missing key point (as the ground truth, referred to as D_g). From D_g, we create a dataset D_a where key points are removed for k consecutive frames where k is exercised from 1 to 60. We apply the above two approaches to D_a, and find that when the missing duration is short (long), Method 1 (Method 2) gives a lower average estimation error (RMSE). This finding leads to our solution where we switch between the two baselines based on the missing duration. The switching threshold is empirically set to 7 or 14 frames for 30 and 60 FPS respectively, based on the data.

To evaluate our solution, we construct another dataset D_b from D_g. In D_b, key points are removed in such a way that their missing time follows the same distribution as that in our entire dataset. Compared to using the two baselines alone, our solution reduces the average RMSE by 21% and 15%, respectively.

Recall from §4.1 that a key point contains a confidence value w. We gradually decay w as a key point remains absent, because as the missing time t increases, its 3D coordinate estimation becomes less reliable. We let $w(t) = w_0 \times max(0, 1 - \frac{t}{T})$ where w_0 is the most recently captured confidence value of this key point and T is a threshold controlling the decay speed. We empirically set T to 1 sec, *i.e.,* the 90-th percentile missing time for a key point in our dataset. The confidence value will be used in §4.3 as an input to the prediction model.

A.3 Details of Data collection Locations

As shown in Figure 3, the four data collection locations we select (*Personal Office*, *Living Room*, *University Office*, *Meeting Room*) have diverse environments in terms of the layout, floor materials, furniture types, and spatial openness, *etc.* The data collection area of *Personal Office* covers almost the entire room. While *Living Room* also has a similarly simple setup, its data collection area only covers one-third of the room and appears more open, as shown in the floor plan. *University Office* is a large room with a complex layout. *Meeting Room* has a long table in the center of the data collection area. The relative positions between the camera and the WiFi AP also differ across the four locations. Table 4 summarizes the moving area and room space of the four locations. The moving area is the area for data collection. The room space refers to the total space of the entire room.



Used	Discarded
0: Nose	15: Right Eye
1: Neck	16: Left Eye
2: Right Shoulder	17: Right Ear
3: Right Elbow	18: Left Ear
4: Right Wrist	19: Left Big Toe
5: Left Shoulder	20: Left Small Toe
6: Left Elbow	21: Left Heel
7: Left Wrist	22: Right Big Toe
8: Mid Hip	23: Right Small Toe
9: Right Hip	24: Right Heel
10: Right Knee	
11: Right Ankle	
12: Left Hip	
13: Left Knee	
14: Left Ankle	

Figure 15: OpenPose BODY_25 Format.

B Details of Injecting Dynamic Change

We provide details on how to use a robotic arm to mimic real humans in terms of NLoS-incurred throughput degradation (§5.1). The idea is to find a material/object that can incur a similar throughput drop to that caused by a real human and is also lightweight enough for the robotic arm to carry.

To achieve the above, we perform the following experiment consisting of three steps. (1) We install the smartphone on a tripod and fix it in LoS to the 802.11ad access point. We measure the mmWave throughput when there is no blockage between the smartphone and AP. (2) We ask a real human (height: 1.75 m) to stand between the smartphone and the AP to introduce NLoS and measure the mmWave throughput, which now drops. (3) We ask our volunteer to walk away, and use the robotic arm to hold an object at the same position where the real human stands. We make sure the object blocks the LoS between the AP and the smartphone. We then measure the mmWave throughput and compare it with the measured throughput in Step (2). We want their difference to be small. We try three objects: a paper box, a box covered by an outwear, and a box covered by aluminum foil. We find that a box covered by aluminum foil has the most similar impact on mmWave throughput as a real human (with an average throughput difference of 5%, or 28 Mbps). We therefore use it in our dynamic change experiments in §5.1.

C Trend Test For Throughput Sequence

The Cox-Stuart test starts with two statistical hypotheses: (1) H_0: No monotonic trend exists in the series, and (2) H_A: The series is characterized by a monotonic trend, which is further considered as three cases, *i.e.,* (a) an increase or decrease trend exists, (b) an increase trend exists, and (c) a decrease trend exists. Mathematically, in the testing procedure, a throughput sequence $Z = \{z_1, z_2, ..., z_n\}$ (we suppose n is an even number for simplicity) is divided into two parts $\{z_1, z_2, ..., z_{\frac{n}{2}}\}$ and $\{z_{\frac{n}{2}+1}, z_{\frac{n}{2}+2}, ..., z_n\}$. The test statistic $T(+)$ and $T(-)$ is then calculated as $T(+) = \sum_{i=1}^{\frac{n}{2}} \mathbb{I}(z_i < z_{i+\frac{n}{2}})$ and $T(-) = \sum_{i=1}^{\frac{n}{2}} \mathbb{I}(z_i > z_{i+\frac{n}{2}})$, respectively, where $\mathbb{I} \in \{0,1\}$ is an indicator function. If the null hypothesis H_0 is true, the statistic $T(+)$ and $T(-)$ should obey the binomial distribution with parameters $\frac{n}{2}$ and $\frac{1}{2}$, *i.e.,* $T(+), T(-) \sim B(\frac{n}{2}, \frac{1}{2})$. Otherwise if $T(+) > T(-)$ (or $T(-) > T(+)$) and the p-value is less than a threshold (*e.g.,* 0.05 in our case), the hypothesis H_A case (b) (or H_A case (c)) is true.

D Additional Implementation Details

D.1 802.11ad Throughput Measurement

Compared to traditional 2D video traffic, immersive content traffic is highly bursty. Take volumetric content as an example. First, different from the traditional 2D videos that are encoded at a group of pictures (GOP) level, volumetric videos are typically encoded on a per-frame basis due to the difficulty of inter-frame encoding. Second, volumetric content players often apply visibility-aware techniques [36, 47, 55, 66, 88] per frame to only download the content inside the viewer's predicted viewport. To maintain accurate viewport prediction results, the client player has to maintain a shallow buffer (*e.g.,* 5 frames in ViVo [47]). Both factors above lead to an extremely frequent request/reply pattern, which renders traditional throughput measurement methods used by 2D video players (simply calculating the ratio between the video chunk size and the chunk download time) very inaccurate. Over mmWave that offers Gbps throughput, the inaccuracy is further deteriorated.

To address the above challenge, we adopt a cross-layer design to measure the throughput by passively examining incoming packets containing immersive content on the client side. Our approach works for both single-path and multipath cases. Specifically, at the application layer, the edge explicitly informs the client how much data will be transmitted over each path before sending data blocks belonging to each frame back-to-back. At the transport layer, the client tracks the arrival time and TCP sequence numbers of the incoming packets. The TCP sequence numbers indicate how much data has been received. Utilizing these information, the client-side throughput measurement module is able to group the *back-to-back* packets in each "burst" as a packet train [64, 69, 83, 87] and use their sizes and timing for throughput measurement. Our approach disregards the ordering and duplicate of packets, and is therefore robust to packet out-of-order and retransmission.

D.2 Development of Two Sample Volumetric Streaming Applications

To demonstrate how Habitus can benefit real immersive applications, we implement two sample volumetric content delivery applications with different logic and complexity.

App 1: Simple Volumetric Streaming. We build a simple volumetric streaming system using the Habitus API from scratch in 5,208 LoC. It delivers the volumetric content stored on a Linux server to an Android client over the Internet. The client player uses a shallow buffer of 5 frames (consistent with App 2) for streaming. The content format uses the layered encoding scheme described in §6.4: each volumetric frame (point cloud) is split into 64 layers each consisting of non-overlapped points through uniform sampling. Each (frame, layer) pair corresponds to a data block in Habitus's term. The priority of each block is inverse proportional to the number of points in the block. The intuition is to prioritize streaming blocks with sparse points so that the viewer can see the partial content as early as possible.

App 2: Visibility-aware Volumetric Streaming. We also replicate ViVo [47], a state-of-the-art networked volumetric video streaming system. ViVo performs visibility-aware streaming where it only fetches content falling into the viewer's predicted viewport. In ViVo, each volumetric frame (point cloud) is spatially segmented into 64 cubical cells. Each (frame, cell) thus constitutes to a data block in Habitus. The priority of a block is calculated at runtime, *i.e.,* inverse proportional to the Euclidean distance from the center of its cubical cell to the center of the predicted viewport. To integrate Habitus into ViVo, we only change 47 LoC that is mainly for library initialization and blocks transmission/reception.

For both applications, each data block is encoded into 10 quality levels with different point density levels. Both applications use the same throughput-based adaptive bitrate (ABR) algorithm [50] to determine the quality level of each data block. The bitrate selection logic works as follows. Initially, all the to-be-fetched blocks are set to the highest quality level. The ABR algorithm then greedily picks the block with the lowest priority and reduces its quality level by 1. The above process is repeated until the total calculated bandwidth usage does not exceed the aggregated network capacity reported by Habitus, or all the blocks reach the lowest quality level.

E Micro Benchmarks and Resource Usage

Resource Usage and Processing Time For a Habitus-enhanced volumetric content delivery system, the average CPU utilization is 36% on the client side (*i.e.,* ROG Phone II) and 169% (*i.e.,* equivalent to 1.69 cores being fully utilized) on the edge side. The peak GPU memory usage on the edge side is 4721MiB (out of 11GB on 2080Ti) in total, including 2101MiB for video capturing and pose estimation, 1017MiB for 802.11ad throughput prediction, and 1603MiB for object detection. The average processing time on an NVIDIA 2080Ti GPU is 27ms and 3.5ms for pose estimation and throughput prediction using a *Seq2Seq with Pose* model, respectively. The processing time meets the system's requirements.

Energy and Heat To profile the energy consumption and heat increase of the client device, we run our control experiment using *V2* and {*Personal Office*, User 1}'s data traces repeatedly on a ROG Phone II for 30 minutes. We use **App1** (§7) and three Habitus variants {*ac*, *Full*, *MuSher-VR*} in §8.4. We start each experiment on a fully-charged phone. After 30-minute running, the battery level drops from 100% to 93% for *ac*, 92% for *Full*, and 92% for *MuSher-VR*, while the device temperature rises from $30.0°C$ to $36.2°C$ for *ac*, from $30.5°C$ to $38.0°C$ for *Full*, and $30.2°C$ to $38.0°C$ for *MuSher-VR*. Compared to *ac*, the additional energy consumption and heat increase of *Full* is 1% and $1.3°C$, respectively. Overall, we believe the resource usage of Habitus is acceptable.