# Reproducibility Companion Paper: Kalman Filter-based Head Motion Prediction for Cloud-based Mixed Reality

Serhan Gül
Fraunhofer HHI
Berlin, Germany
serhan.guel@hhi.fraunhofer.de

Sebastian Bosse
Fraunhofer HHI
Berlin, Germany
sebastian.bosse@hhi.fraunhofer.de

Dimitri Podborski*
Fraunhofer HHI
Berlin, Germany
podborskiy@gmx.de

Thomas Schierl
Fraunhofer HHI
Berlin, Germany
thomas.schierl@hhi.fraunhofer.de

Cornelius Hellge
Fraunhofer HHI
Berlin, Germany
cornelius.hellge@hhi.fraunhofer.de

Marc A. Kastner
National Institute of Informatics
Tokyo, Japan
mkastner@nii.ac.jp

Jan Zahálka
Czech Technical University in Prague
Prague, Czech Republic
jan.zahalka@cvut.cz

## ABSTRACT

In our MM'20 paper [4], we presented a Kalman filter-based approach for prediction of head motion in 6DoF. The proposed approach was employed in our cloud-based volumetric video streaming system to reduce the interaction latency experienced by the user. In this companion paper, we present the dataset collected for our experiments and our simulation framework that reproduces the obtained experimental results. Our implementation is freely available on Github to facilitate further research.

## CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Human-centered computing** → **Mixed / augmented reality**; • **Computer systems organization** → *Cloud computing*;

## KEYWORDS

volumetric video, mixed reality, augmented reality, remote rendering, head motion prediction, Kalman filter, reproducible research

**ACM Reference Format:**
Serhan Gül, Sebastian Bosse, Dimitri Podborski, Thomas Schierl, Cornelius Hellge, Marc A. Kastner, and Jan Zahálka. 2021. Reproducibility Companion Paper: Kalman Filter-based Head Motion Prediction for Cloud-based Mixed Reality. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21), October 20–24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3474085.3477936

## 1 INTRODUCTION

In media production, volumetric video has been gaining importance for photorealistic representation of 3D objects, especially in the context of rapidly developing Virtual Reality (VR) and Mixed Reality (MR) markets [8]. Volumetric video allows users to explore 3D content from any viewpoint and enables novel MR applications in which virtual objects can be integrated into real environments and animated depending on user interaction [3]. However, rendering complex volumetric videos is computationally challenging for today's mobile devices since high-quality volumetric videos may contain thousands of polygons (mesh) or millions of points (point clouds) posing a significant challenge in terms of decoding and rendering complexity on end devices [7, 9].

A promising technique to reduce the computational burden on mobile devices is to offload the rendering process to a cloud/edge server (more generally known as remote rendering [10]) and send a pre-rendered image to the client device. However, remote rendering systems (largely due to the additional network delay) suffer from an increased interaction latency that may cause registration errors in Mixed Reality (MR) environments. One way of reducing the effective interaction latency is to predict the viewer's future head pose at the remote server based on the most recent head pose sent from the client, render the corresponding view from the volumetric video in advance and send that view to the user. Thus, an accurate prediction algorithm can provide a smooth, low-latency user experience, especially in combination with client-side correction techniques such as asynchronous timewarping [12] and depth-based image warping [6] that perform corrections on the transmitted view based on the latest head pose of the user and thus mitigate potential prediction errors.

In our work presented at MM'20, we proposed a Kalman filter-based predictor for head motion prediction in 6DoF space. We compared the performance of the proposed approach to a benchmark autoregression model (AutoReg) [5] and a baseline (no prediction) using head motion traces recorded with Microsoft HoloLens. More details on our approach and experimental results can be found in our paper [4]. In this companion paper, we present the dataset we collected for our experiments and the simulation framework that reproduce our experimental results.

*This work was carried out as the author was affiliated with Fraunhofer HHI.

*MM '21, October 20–24, 2021, Virtual Event, China*

## 2 PROJECT DESCRIPTION

We developed our simulation framework using Python. The repository is available at:

`https://github.com/fraunhoferhhi/pred6dof`

including the head motion dataset used in the evaluation.

### 2.1 Folder structure

The high-level file hierarchy of the repository is given as follows:

```
pred6dof
├── autoreg_models
├── data
│   ├── raw
│   └── interpolated
├── pred6dof
│   ├── application.py
│   ├── runner.py
│   ├── evaluator.py
│   └── reporter.py
├── results
│   ├── figures
│   └── tabular
├── compute_all.sh
├── config.toml
├── README.md
├── environment.yml
└── requirements.txt
```

`autoreg_models`: Models for the benchmark AutoReg predictor, trained separately for different positional (x,y,z) and quaternion coordinates (qw,qx,qy,qz).
`data/raw`: Contains 14 user traces collected using Microsoft HoloLens. See format in Sec. 2.2.
`data/interpolated`: Contains the traces interpolated with a given sampling time (default: 5ms).
`application.py`: Command line interface for the application.
`runners.py`: Runs the predictors AutoReg, Kalman, baseline (no-prediction) over all traces.
`evaluator.py`: Compute evaluation metrics such as mean absolute error (MAE) and root-mean-square error (RMSE).
`reporter.py`: Computes and plots the trace statistics; per-trace and average results
`results/figures`: Plots for trace statistics, average performance of the predictors over all traces, and box plots for individual traces.
`results/tabular`: CSV files containing the per-trace and average prediction results for each trace.
`compute_all.sh`: Generates all results, plots everything and creates a PDF file with the reproduced results.
`config.toml`: Configuration parameters for the predictors.
`README.md`: Documentation for the repository
`environment.yml`: Package specification for recreating the environment using Conda.
`requirements.txt`: Package specification for recreating the environment using Pip.

### 2.2 Dataset

We created a Microsoft HoloLens application that overlays a virtual object on real world (see Fig.4, [4]) and collected 14 head movement traces while the user freely moved around and inspected the object for a duration of 60 s. We recorded the position (x, y, z) and orientation data (as quaternions) together with the corresponding timestamps. Recorded traces were saved as CSV files into the folder `data/raw`. Each line contains a sample represented in the following format: `<timestamp,x,y,z,qx,qy,qz,qw>`.

Since the raw sensor data from the HoloLens was unevenly sampled (i.e. different temporal distances between consecutive samples), we interpolated the data to obtain temporally equidistant samples at sampling rate of 200 Hz. This was achieved by upsampling the position data using linear interpolation and quaternions using SLERP [11]. The resulting traces were saved into the folder `data/interpolated`. For easy visualization of the trace statistics, we also included the Euler angles representation of the orientation components. Thus, the interpolated traces have the following format: `<timestamp,x,y,z,qx,qy,qz,qw,roll,pitch,yaw>`.

## 3 INSTALLATION AND RUNNING

### 3.1 Dependencies

You can install the dependencies either using Conda [2] or pip [1]. For both tools, package specification files (`environment.yml` and `requirements.txt`, respectively) are provided at the top-level directory. Conda provides an isolated environment and allows installing all dependencies automatically without interfering with your system. The following commands generate a conda environment called `pred6dof` containing all the dependencies and activate the environment:

```
$ conda env create -f environment.yml
$ conda activate pred6dof
```

Similarly, if you are using pip, you can create a *virtualenv* and install the dependencies there:

```
$ python3 -m venv pred6dof
$ source pred6dof/bin/activate
$ pip install -r requirements.txt
```

### 3.2 Running the code

The application should be run as a Python module (named `pred6dof`) from the top-level directory. For convenience, we provide a shell script `compute-all.sh` that runs the implemented predictors on all user traces and reproduces the plots presented in the paper. First make sure that the script is executable on your machine by running:

```
$ chmod +x compute-all.sh
```

and then run the script as follows:

```
$ ./compute-all.sh
```

Specifically, the script performs the following operations:

- Resample the collected raw traces evenly with a sampling time of 5ms and saves them to ./data/interpolated.
  `$ python -m pred6dof prepare`
- Run the predictors Kalman, AutoReg, baseline/no-prediction for the look-ahead times [20,40,60,80,100] ms.

```
$ python -m pred6dof run -a kalman -w 20 40 60 80 100
$ python -m pred6dof run -a autoreg -w 20 40 60 80 100
$ python -m pred6dof run -a baseline -w 20 40 60 80 100
```

- Compute mean results over all traces and generate the plots. Additionally, a PDF file is generated using the reproduced plots and saved to `acmmm20.pdf` at the top-level directory.

```
$ python -m pred6dof report
```

Different options for each sub-command (`prepare`/`run`/`report`), e.g. settings dataset or figures paths, look-ahead time, can be viewed by running the help command:

```
$ python -m pred6dof {prepare/run/report} -h
```

For example. you can run the Kalman predictor for a look-ahead time of 20 ms by executing:

```
$ python -m pred6dof run -a kalman -w 20
```

It takes around 35 minutes on a powerful laptop (MacBook Pro 2019, 2,4 GHz 8-Core Intel Core i9) to iterate over all evaluated traces and look-ahead times for the tested predictors where AutoReg takes the highest processing time.

Our implementation was tested on macOS 10.15 and Ubuntu 20.04.

## 4 REPRODUCIBILITY EFFORTS

The source code is well designed with a high standard of readability and comments. Further, the companion paper and readme files are sufficiently explained, making this framework easy to use. The results of the original paper are easily reproducible, directly generating not only all original figures, but the full original paper PDF.

As the sources were of already high quality in the initially submitted version, the reproducibility review has only consisted of some very minor fixes and ease-of-use improvements, such as a missing link in the companion paper and a missing shebang in one script. The authors have been very responsive to our feedback.

All of the above aspects lead us to believe this is a software worthy of the reproducibility badge.

## REFERENCES


[1] Ian Bicking. 2011. Pip: Package installer for Python. https://pypi.org/project/pip/. (2011). Accessed: 2021-01-26.

[2] Inc. Continuum Analytics. 2017. Conda package manager. https://conda.io. (2017). Accessed: 2021-01-26.

[3] Peter Eisert and Anna Hilsmann. 2020. Hybrid Human Modeling: Making Volumetric Video Animatable. In *Real VR - Immersive Digital Reality: How to Import the Real World into Head-Mounted Immersive Displays*, Marcus Magnor and Alexander Sorkine-Hornung (Eds.). Springer International Publishing, Cham, 167–187. https://doi.org/10.1007/978-3-030-41816-8_7

[4] Serhan Gül, Sebastian Bosse, Dimitri Podborski, Thomas Schierl, and Cornelius Hellge. 2020. Kalman Filter-Based Head Motion Prediction for Cloud-Based Mixed Reality. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20)*. Association for Computing Machinery, New York, NY, USA, 3632âĂŞ3641. https://doi.org/10.1145/3394171.3413699

[5] Serhan Gül, Dimitri Podborski, Thomas Buchholz, Thomas Schierl, and Cornelius Hellge. 2020. Low-Latency Cloud-Based Volumetric Video Streaming Using Head Motion Prediction. In *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '20)*. Association for Computing Machinery, New York, NY, USA, 27–33. https://doi.org/10.1145/3386290.3396933

[6] William R Mark, Leonard McMillan, and Gary Bishop. 1997. Post-rendering 3D warping. In *Proceedings of the 1997 symposium on Interactive 3D graphics*. 7–ff.

[7] O Schreer, I Feldmann, P Kauff, P Eisert, D Tatzelt, C Hellge, K Müller, T Ebner, and S Bliedung. 2019. Lessons learnt during one year of commercial volumetric video production. In *2019 IBC conference*. IBC.

[8] Oliver Schreer, Ingo Feldmann, Sylvain Renault, Marcus Zepp, Markus Worchel, Peter Eisert, and Peter Kauff. 2019. Capture and 3D video processing of volumetric video. In *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 4310–4314. https://doi.org/10.1109/icip.2019.8803576

[9] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A. Chou, Robert A. Cohen, Maja Krivokuća, Sebastien Lasserre, Zhu Li, Joan Llach, Khaled Mammou, Rufael Mekuria, Ohji Nakagami, Ernestasia Siahaan, Ali Tabatabai, Alexis M. Tourapis, and Vladyslav Zakharchenko. 2019. Emerging MPEG Standards for Point Cloud Compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (2019), 133–148. https://doi.org/10.1109/JETCAS.2018.2885981

[10] Shu Shi and Cheng-Hsin Hsu. 2015. A survey of interactive remote rendering systems. *Comput. Surveys* 47, 4 (May 2015), 1–29. https://doi.org/10.1145/2719921

[11] Ken Shoemake. 1985. Animating rotation with quaternion curves. In *ACM SIGGRAPH computer graphics*, Vol. 19. ACM, 245–254. https://doi.org/10.1145/325165.325242

[12] JMP Van Waveren. 2016. The asynchronous time warp for virtual reality on consumer hardware. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*. ACM, 37–46. https://doi.org/10.1145/2993369.2993375