Defence of Master's Thesis

User Position Prediction in 6-DoF Mixed Reality Applications using Recurrent Neural Networks

Oleksandra Baga

Freie Universität Berlin
Matrikelnummer 5480722
Master Computer Science
E-Mail: oleksandra.baga@gmail.com

I am

Oleksandra Baga

I was born in **Odessa**, Ukraine 💙💛

I live in Berlin since **2014**

B.Eng

I earned B.Eng at Beuth Hochschule für Technik in 2021.

I studied Embedded Systems

HHI

I work as a **student assistant at Fraunhofer Institute**

Master thesis is created in cooperation with the Fraunhofer Institute.

Introduction

Mixed reality breaks down the border between the virtual and real world and **tricks human's senses.**

Picture source: https://www.nojitter.com/unified-communications-collaboration/horizons-holograms-and-headsets-whats-next-vrar

Introduction

Virtual objects with feeling of the size and density can be placed on the real table in the user's room, picked up with a hand and moved to another place.

Picture source: https://arstechnica.com/gadgets/2021/10/vr-ar-wearables-and-smart-home-tech-are-now-mainstream-research-says

3-DoF vs 6-DoF

Term **degrees of freedom** describes how users interact and move.

Within 3-DoF space user has only three possibilities: **look left, right, up down and pivot left and right.**

User can not move throughout the virtual space → only **rotation** can be tracked.

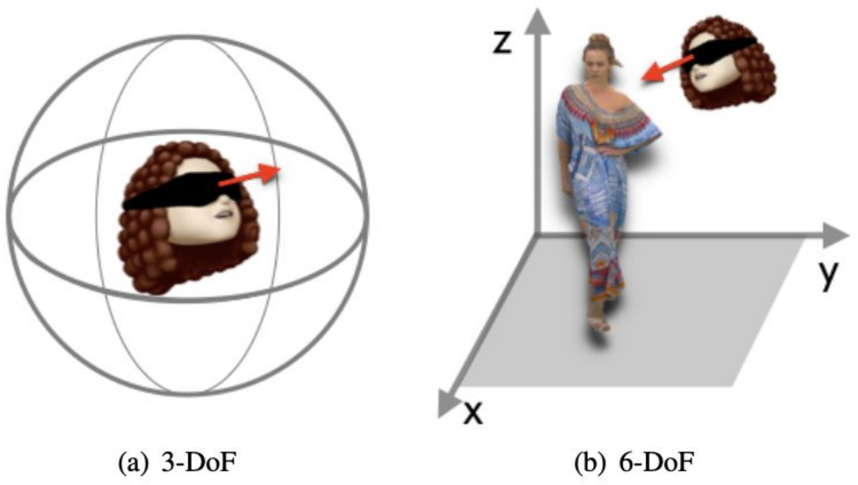


Fig.3. Users viewing point are inward in 3-DoF and outward in 6-DoF

Picture source: Silvia Rossi, Irene Viola, Laura Toni, and Pablo Cesar. From 3-DoF to 6-DoF: New Metrics to Analyse Users Behaviour in Immersive Applications

3-DoF vs 6-DoF

The 6-DoF means **tracking both position and rotation** and reflects the human's movement in a real life.

In 6-DoF VR user moves, walks, jumps!

Previous algorithms are done only for rotational data and thus **new approaches for user movement prediction must be created!**

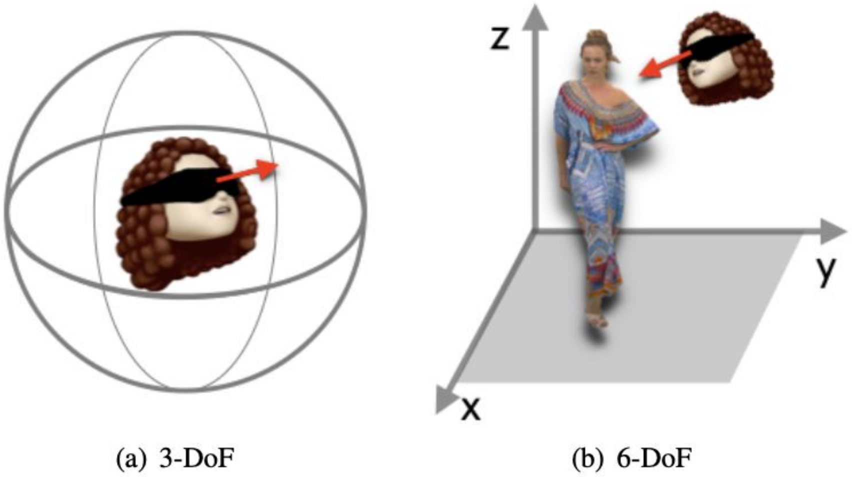


Fig.3. Users viewing point are inward in 3-DoF and outward in 6-DoF

Picture source: Silvia Rossi, Irene Viola, Laura Toni, and Pablo Cesar. From 3-DoF to 6-DoF: New Metrics to Analyse Users Behaviour in Immersive Applications

Problem statement

In the real world there is no time delay between action taken and reaction observed.

In VR Application there is always a delay!

A delay between the physical movement and the display output is defined as **motion-to-photon (M2P) latency.**

Display lag can produce spatial disorientation and dizziness → **Motion sickness**.

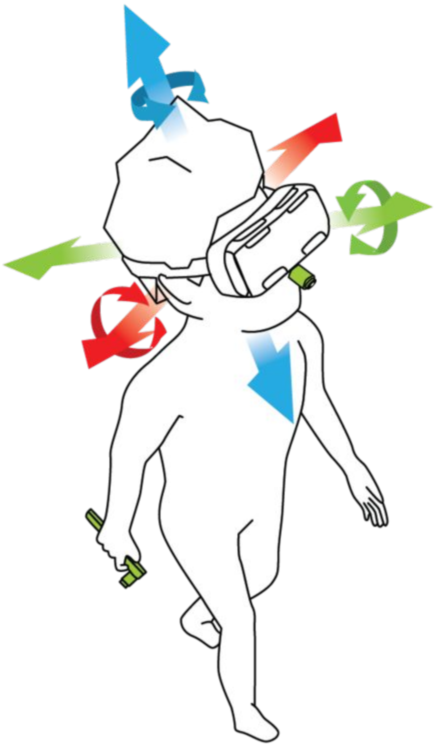


Fig.4. Pose tracking in virtual reality

Picture source: https://en.wikipedia.org/wiki/Pose_tracking

Motivation

Recently the new technique of the **rendering on a cloud server** was presented by the researches.

Thus it makes possible to decrease the computational load from client by offloading the task to a powerful server.

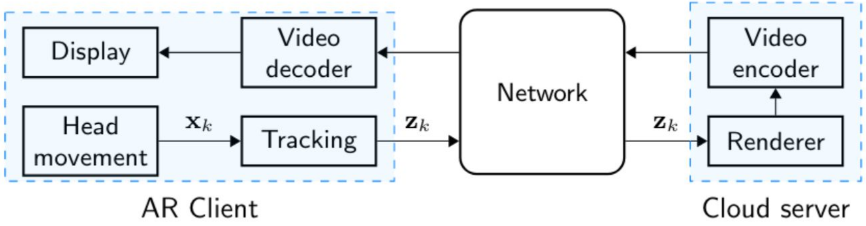


Fig.5. High level operation of a cloud-based volumetric streaming system.

Picture source: Serhan Guel, Sebastian Bosse, Dimitri Podborski, Thomas Schierl, and Cornelius Hellge. Kalman Filter-based Head Motion Prediction for Cloud-based Mixed Reality.

Motivation

But using a cloud **increases network latency** and **processing delays** due to uploading a data to a server, rendering and sending data back to a device.

The increased M2P latency can be **compensated by applying a prediction algorithm!**

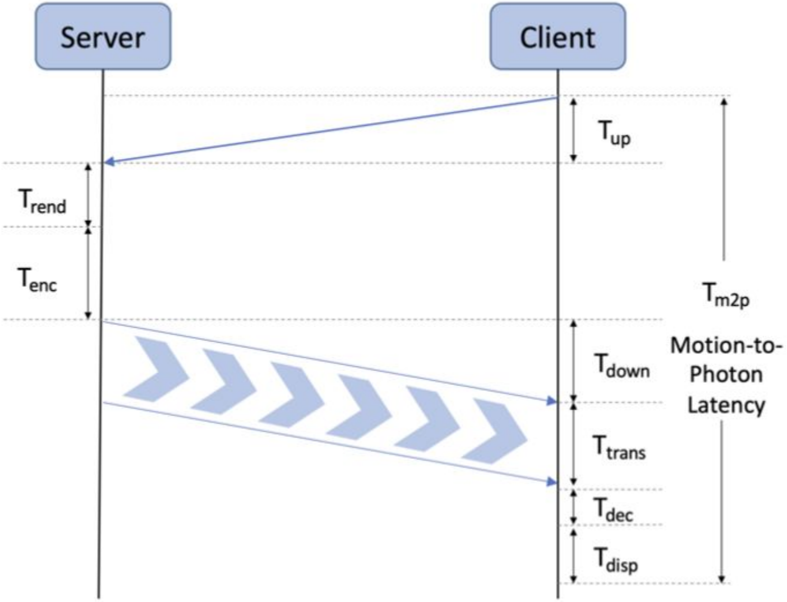


Fig.6. Components of the motion-to-photon latency for a remote rendering system.

Picture source: Serhan Guel, Sebastian Bosse, Dimitri Podborski, Thomas Schierl, and Cornelius Hellge. Kalman Filter-based Head Motion Prediction for Cloud-based Mixed Reality.

Introduction and fundamentals 01

02 Dataset and data exploration

Obtaining, analyze and preprocessing of 6-DoF dataset

Models 03

04 Evaluation

Experiments 05

06 Analyze

Obtaining 6-DoF Dataset

The user position and orientation were obtained with Unity application developed for HoloLens 2.

A volumetric animated object placed 3 metres ahead of the user in the MR environment.

Picture source: Oleksandra Baga

Obtaining 6-DoF Dataset

During data recording, users freely walked wearing HMD in laboratory space.

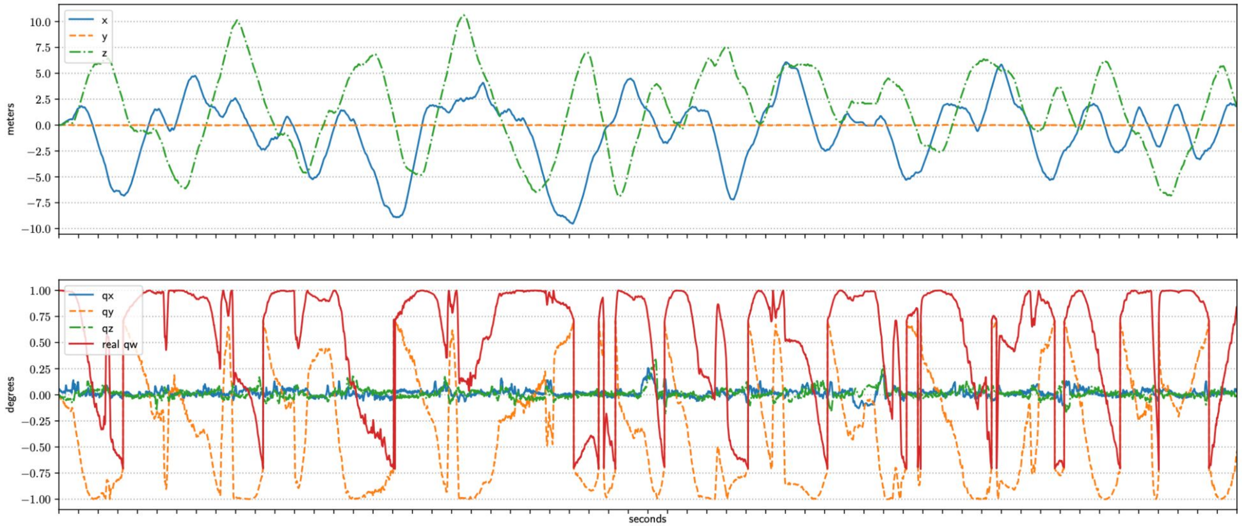
The 6-DoF dataset has 10 features used in training process:

- position (x, y, z)
- orientation (qx, qy, qz, qw)
- velocity (x, y, z).

<u>No personal data was recorded during these sessions and all traces are obtained anonymously.</u>

Picture source: Oleksandra Baga

Interpolated dataset

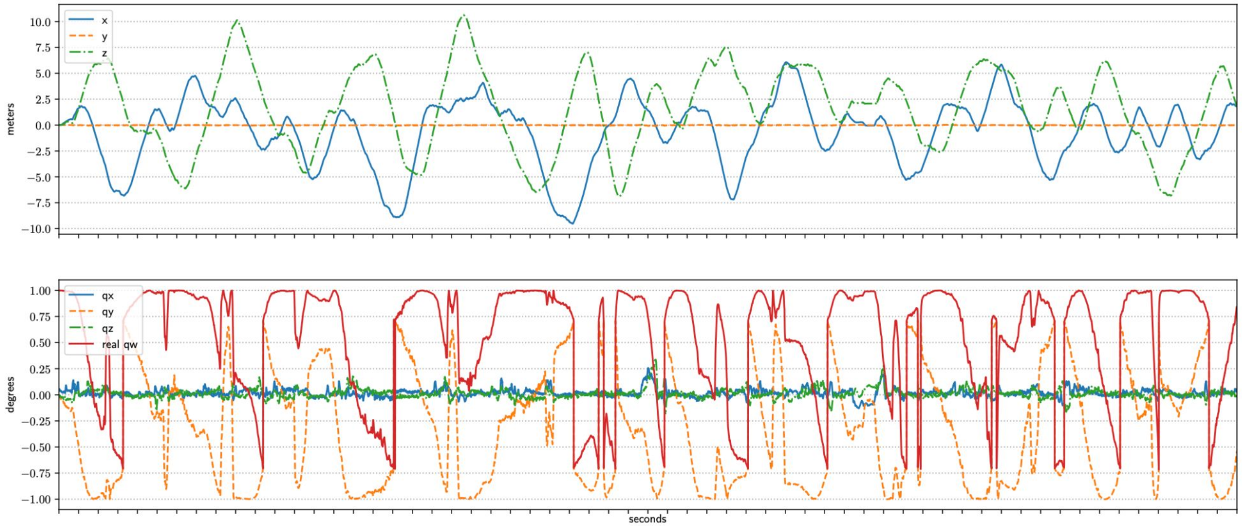


All traces were recorded over 10 minutes long on average 12 minutes.

A user rarely moves along the y-axis.

The y-axis shows the vertical movement that the users could make if they sit down or stand up what requires more effort.

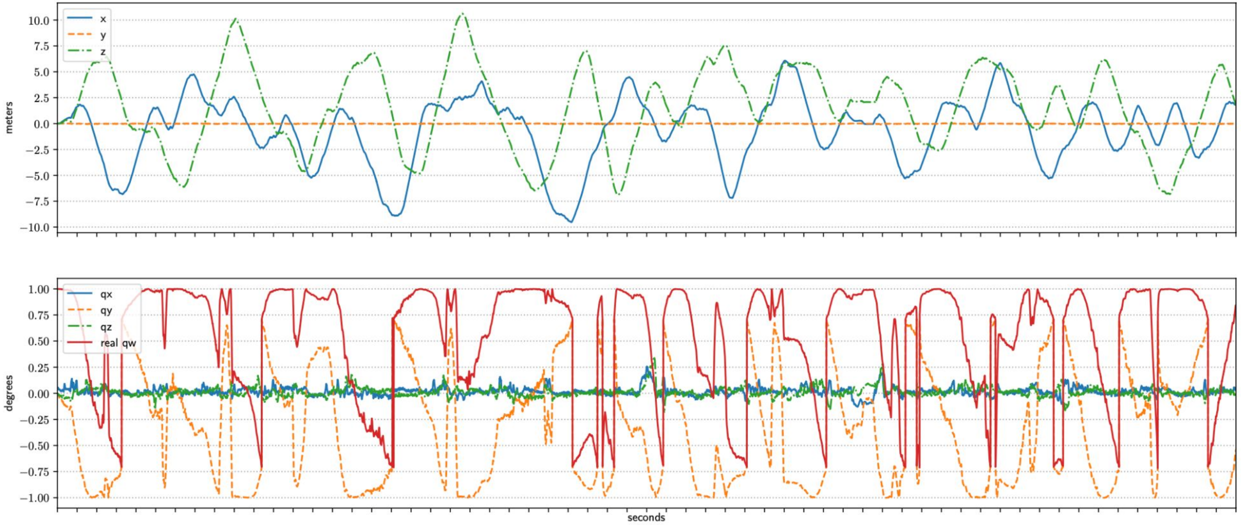
Interpolated dataset



Due to signal processing and propagation delays, distance in time between two consecutive samples was either increased or decreased.

Interpolated dataset

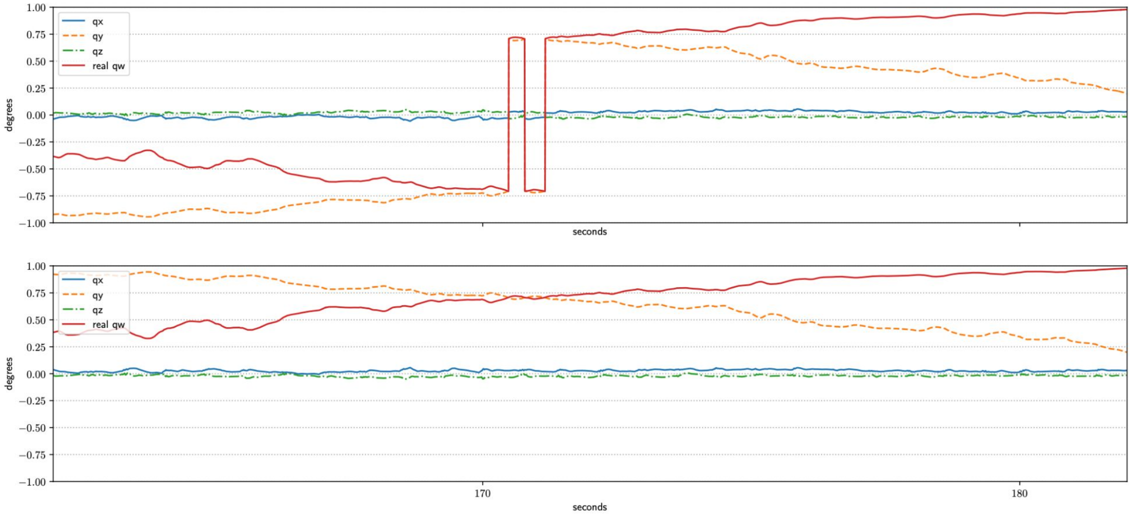
Data preprocessing



The position and velocity data were upsampled using linear interpolation. SLURP used for quaternions.

The components qy and qw of quaternion have sharp change of sign making it harder for a model to learn.

Flipped negative quaternions



Flipping the sign will not affect the rotation, but it will ensure that there are no large jumps in 4D vector space between the two neighboring quaternions with similar rotation.

Introduction and fundamentals 01

02 Dataset and data exploration

Models 03

Analyzed RNN variants

04 Evaluation

Experiments 05

06 Analyze

Model Inputs

The obtained 6-DoF dataset has 10 features used in training process:

- ➔ position (x, y, z)
- ➔ orientation (qx, qy, qz, qw)
- ➔ velocity (x, y, z).

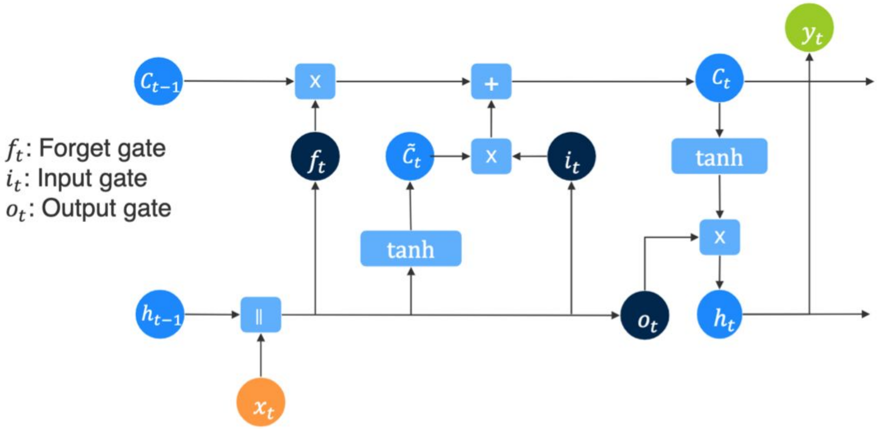
With *batch_first = true* the input and output tensors were provided as **(batch, seq, feature).**

seq = 20 rows = 100 ms

timestamp	x	y	z	qx	qy	qz	qw	velocity_x	velocity_y	velocity_z
0.0	0.004954389	0.003402365	0.01010712	0.0522510460919448	-0.0092471243083722	-0.0147093988998279	0.998482825319659	0.3015088	0.2081023	0.586858
5000000.0	0.0048331026666666	0.0033084996666666	0.0105062066666666	0.0528291300322694	-0.0094015253918042	-0.0147654075580485	0.9984501375031132	0.1943642	0.1336575666666666	0.41589388
10000000.0	0.0047118163333333	0.0032146343333333	0.0109052933333333	0.053407194836499	-0.0095559230697573	-0.0148214108678517	0.9984170880217684	0.0872196	0.0592128333333333	0.24492976
15000000.0	0.00459053	0.003120769	0.01130438	0.0539852402952434	-0.0097103172863047	-0.0148774088089516	0.998383676887596	-0.019925	-0.0152319	0.07396564
20000000.0	0.0044855761666666	0.0030990528333333	0.0115486099999999	0.0545632313576858	-0.0098646897875236	-0.0149333515881849	0.9983499069412224	-0.0215151583333333	-0.0145975916666666	0.0737863916666666
25000000.0	0.0043806223333333	0.0030773366666666	0.0117928399999999	0.0549670346500581	-0.0099280877120111	-0.014740475521052	0.9983299938184644	-0.0231053166666666	-0.0139632833333333	0.0736071433333333
30000000.0	0.0042756685	0.0030556205	0.01203707	0.0553708266921017	-0.0099914836044761	-0.0145475964369255	0.9983098763634082	-0.024695475	-0.013328975	0.0734278949999999
35000000.0	0.0041707146666666	0.0030339043333333	0.0122813	0.0557746074011709	-0.0100548774519432	-0.0143547143752826	0.9982895545801708	-0.0262856333333333	-0.0126946666666666	0.0732486466666666
40000000.0	0.0040657608333333	0.0030121881666666	0.01252553	0.0561783766946222	-0.0101182692414372	-0.0141618293756015	0.998269028472912	-0.0278757916666666	-0.0120603583333333	0.0730693983333333
45000000.0	0.003960807	0.002990472	0.01276976	0.0565821344898146	-0.0101816589599834	-0.0139689414773605	0.9982482980458324	-0.02946595	-0.01142605	0.07289015
50000000.0	0.00390328375	0.00300229975	0.0128401975	0.0568467445693149	-0.010241445519636	-0.0137800200359769	0.998235278615892	-0.023180431	-0.0082128855	0.0574620875

Model Architecture: LSTM

LSTM stands for Long Short-Term Memory and can predict future values based on previous sequential data and learn long-term dependencies.



The **input gate** decides what information will be stored in long term memory.

The **forget gate** decides which information from long term memory be kept or discarded.

The **output gate** produces new short term memory which will be passed on to the cell in the next time step.

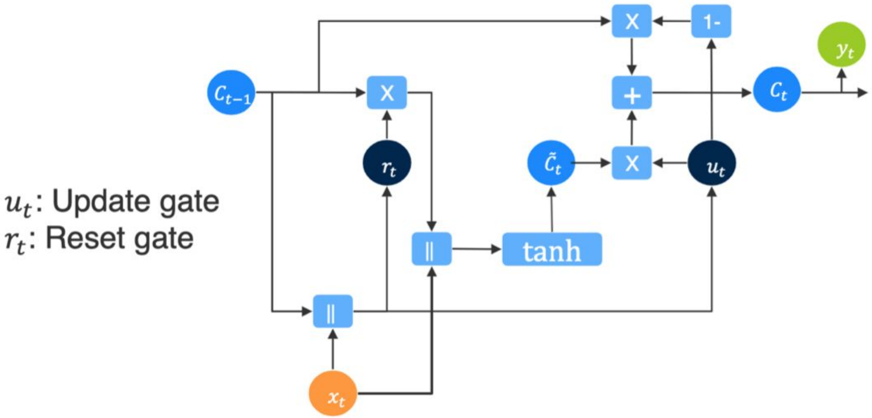
Picture source: Prof. Dr. Tim Landgraf, Lecture 13: Recurrent Neural Networks, WS 20/21: Machine Learning

Model Architecture: GRU

GRU stands for Gated recurrent unit and according to researchers is 29.29% faster than LSTM in training speed for processing the same dataset.

The **update gate** sets amount of previous information to pass along the next state.

The **reset gate** decides whether the previous cell state is important or not.



With powerful update gate the model can decide to copy all the information from the past and eliminate the risk of vanishing gradient.

Picture source: hhttps://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn

How the training and evaluation was done?

All evaluated models are implemented in Python with PyTorch.

Using **loss function**, PyTorch will adjust model weight parameters during training loop in such way that value of loss function **tends to be 0.00.**

With the **train() mode** the network's weights will be updated in order to reduce the training loss

The **eval() mode** turns off the calculation of the gradients.

During training on every epoch the training loss and validation loss both must decrease and stabilise at a specific point ⟶ **Early stopping algorithm.**

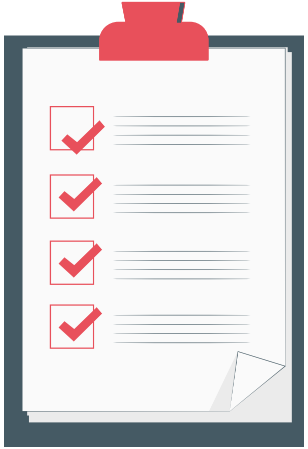
Model parameters

Train/val loop

Model is **trained on 60%, validated on 20% of dataset.**

The rest **20% used for test**.

No shuffle is applied during dataset slicing.



Datasets

Original interpolated dataset

Flipped negative quaternions

Position and rotation only

Normalized position [0..1]

Best: Flipped negative quaternions

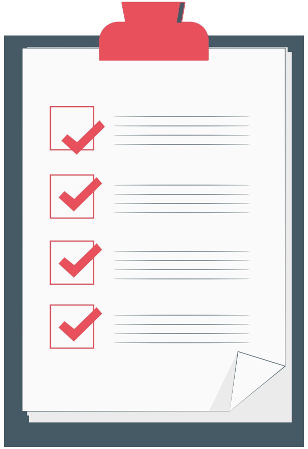
Model parameters

Learning rate

Too large learning rate **overshoots** the local minimum in a cost function.

Too small learning rate does not let to learn the dependencies in the dataset.

Finally the adaptively reducing LR is applied.



Weight decay

WD is a regularisation technique used to avoid overfitting: the weights are multiplied by a factor less than 1 → **prevents the weights from growing too large.**

Learning rate and weight decay of Adam Optimizer have highest influence!
After grid of parameter were tried, LR is set to 1e-4 and weight decay to 1e-12

How the hyperparameters were found?

The hyperparameters search is done using GPU cluster of Fraunhofer HHI.

Every training during 500 epochs took:

- **2 hours with up to 95% of CPU usage**
- **15-20 minutes on GPU cluster**

The grid search is used to search exhaustively through a manually specified subset of the hyperparameter.

For every evaluated model on average over hundred jobs were launched for the initial hyperparameters search and few dozens for parameter tuning.

Introduction and fundamentals 01

02 Dataset and data exploration

Models 03

04 **Evaluation**

Evaluation metrics

Experiments 05

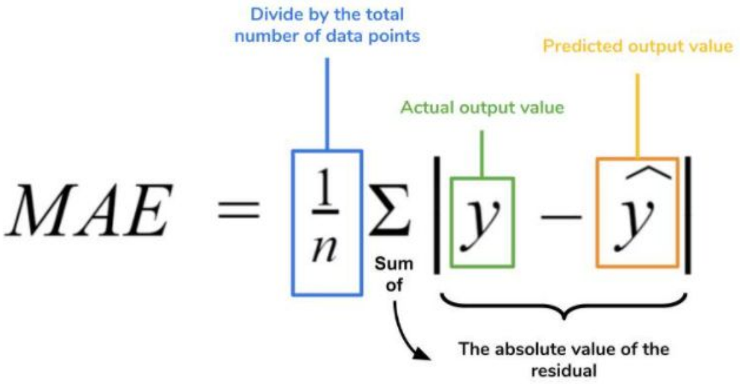
06 Analyze

Goal of evaluation

This master thesis aims to evaluate whether RNN modification as LSTM, GRU and bidirectional variant are **able to reduce the positional and rotation error for given look ahead time of 100 ms.**

Evaluation metrics

Mean Absolute Error (MAE) measures the average magnitude of the errors without considering their direction.



Picture source:https://medium.com/@polanitzer/the-minimum-mean-absolute-error-mae-challenge-928dc081f031

Evaluation metrics

Root mean squared error (RMSE)

squares errors before averages them.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Euclidean distances is used for position.

Angular distance between two quaternions is used for rotation.

Picture source: https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e

Baseline model

Simple model that acts as a reference.

By comparing the metrics it can be understood **how reasonable it is to implement** and use the chosen approach.

Baseline model

Evaluation metrics have value **in units of the original dataset.**

If a model predicts the prices of apartments in Berlin, then deviation from real price of 1000€ is a very good result

Model that predicts the price of average lunch in Berlin's restaurant with error of 1000€ works terrible.

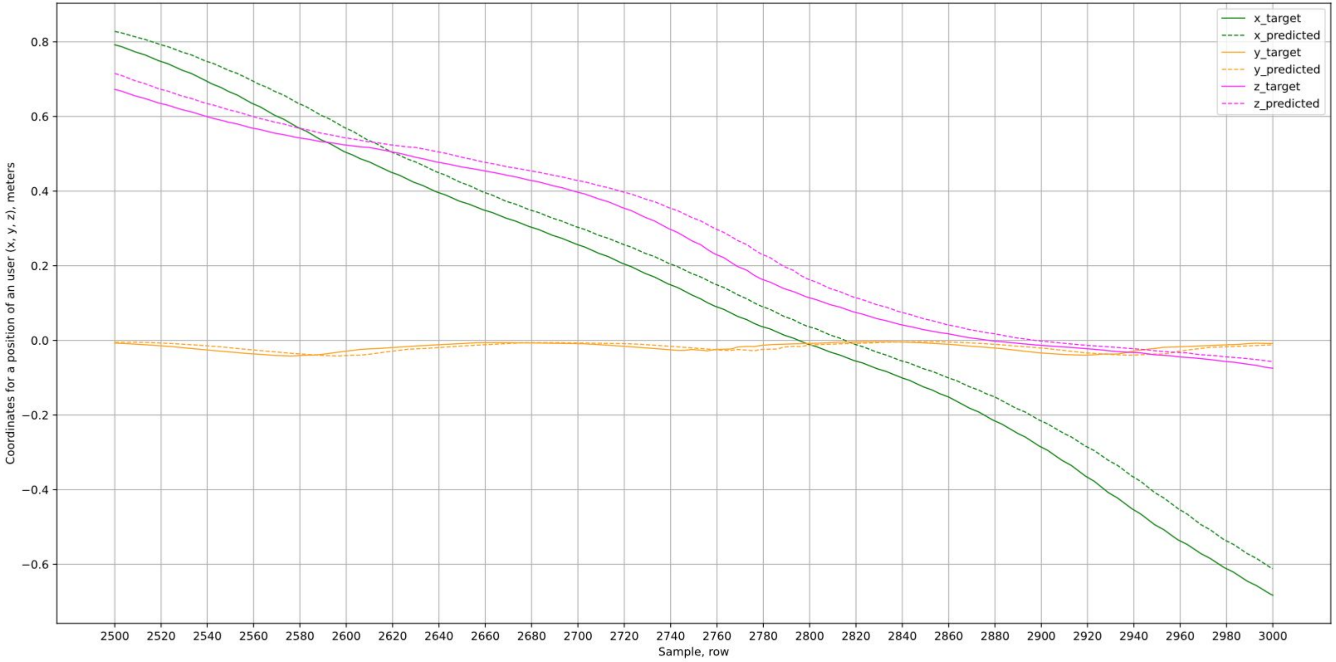
MAE: position: 0.067m
rotation: 14.61°

RMSE: position: 0.068m
rotation: 21.24°

Baseline: x, y z axes

MAE: position: 0.067m
rotation: 14.61°

RMSE: position: 0.068m
rotation: 21.24°



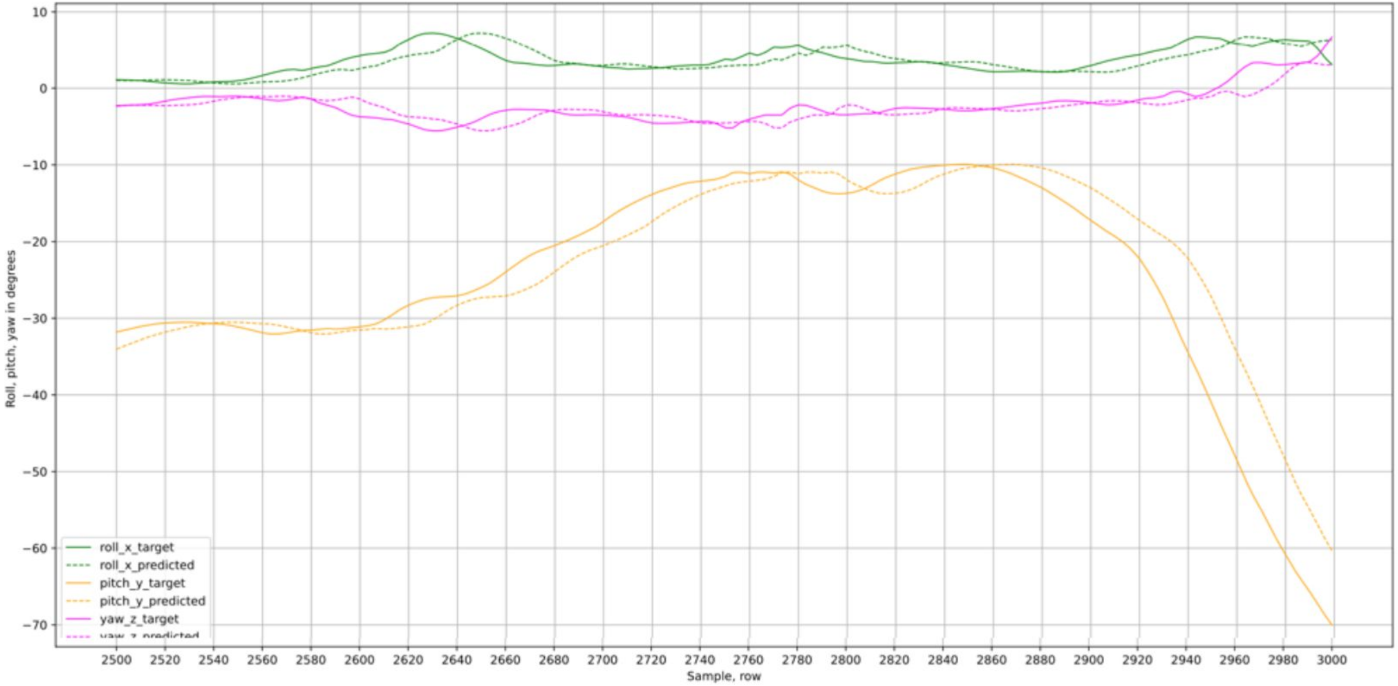
A baseline in forecast performance provides a point of comparison.

The plot of a baseline model predictions shows that the model is n-step behind reality.

Baseline: roll, pitch, yaw

MAE: position: 0.067m
rotation: 14.61°

RMSE: position: 0.068m
rotation: 21.24°



A baseline in forecast performance provides a point of comparison.

The plot of a baseline model predictions shows that the model is n-step behind reality.

Introduction and fundamentals 01

02 Dataset and data exploration

Models 03

04 Evaluation

Experiments 05

Results of experiments of different proprocessed RNN variants

06 Analyze

Overview of experiments

Criterion: MSELoss

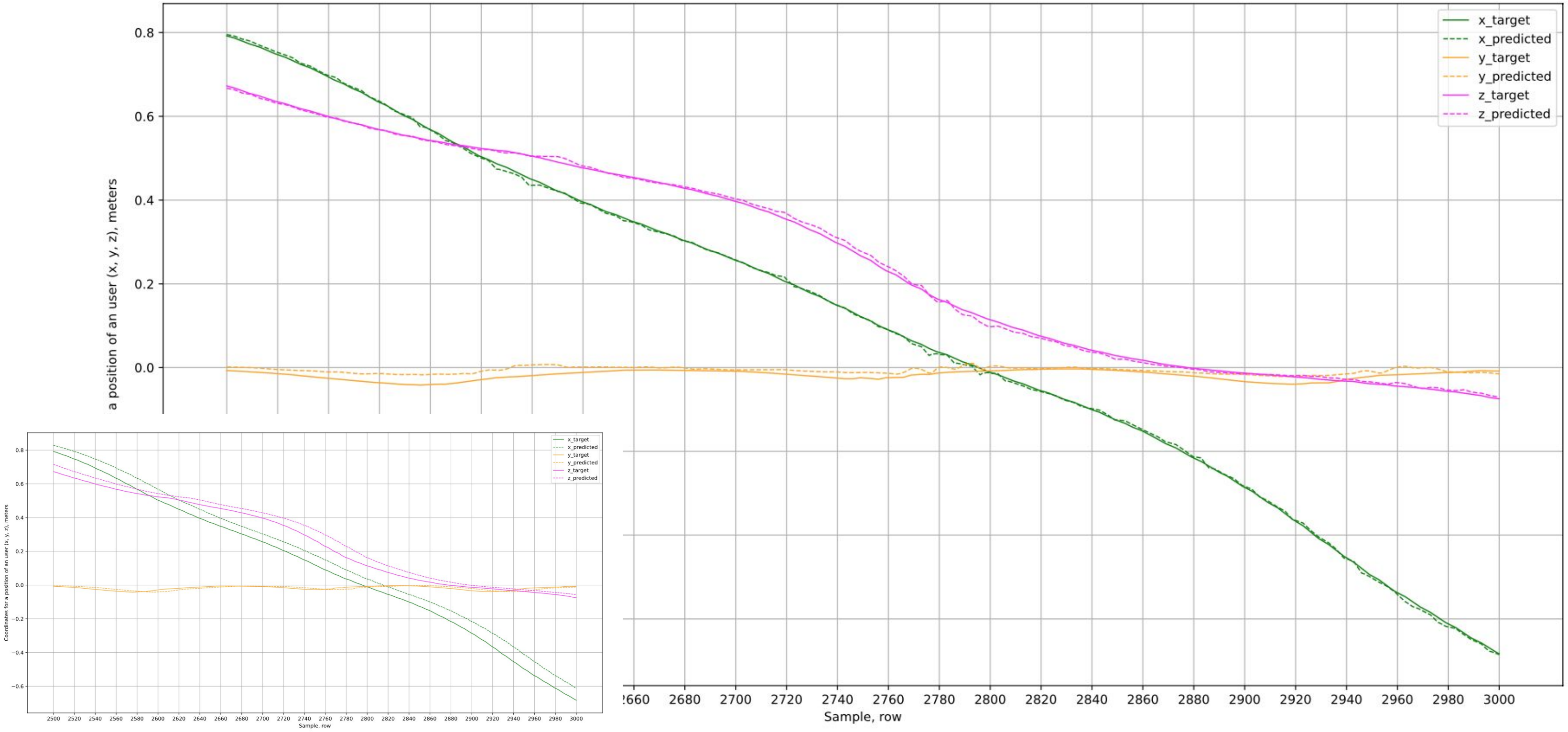
Optimizer: Adam

	MAEpos	RMSEpos	MAErot	RMSErot
Baseline	0.067	0.068	14.61	21.24
LSTM1 interpolated original	0.019	0.028	16.92	23.28
LSTM1 flipped negative quaternions	0.013	0.015	13.49	18.47
LSTM1 position only	0.078	0.091	-	-
LSTM1 rotation only	-	-	11.81	16.64
LSTM1 normalised [0..1]	0.056	0.197	9.87	12.72
LSTM2 Relu flipped	0.055	0.185	22.86	30.88
LSTM3 Mish flipped	0.012	0.014	13.18	17.28
LSTM4 layered flipped	0.055	0.185	42.0	48.0
GRU1 flipped	**0.009**	**0.011**	**8.68**	**12.29**
Bi-GRU flipped	0.030	0.041	30,19	33.78

LSTM3 with Mish():
x, y, z-axis

MAE: position: 0.012m↓
rotation: 13.18°↓

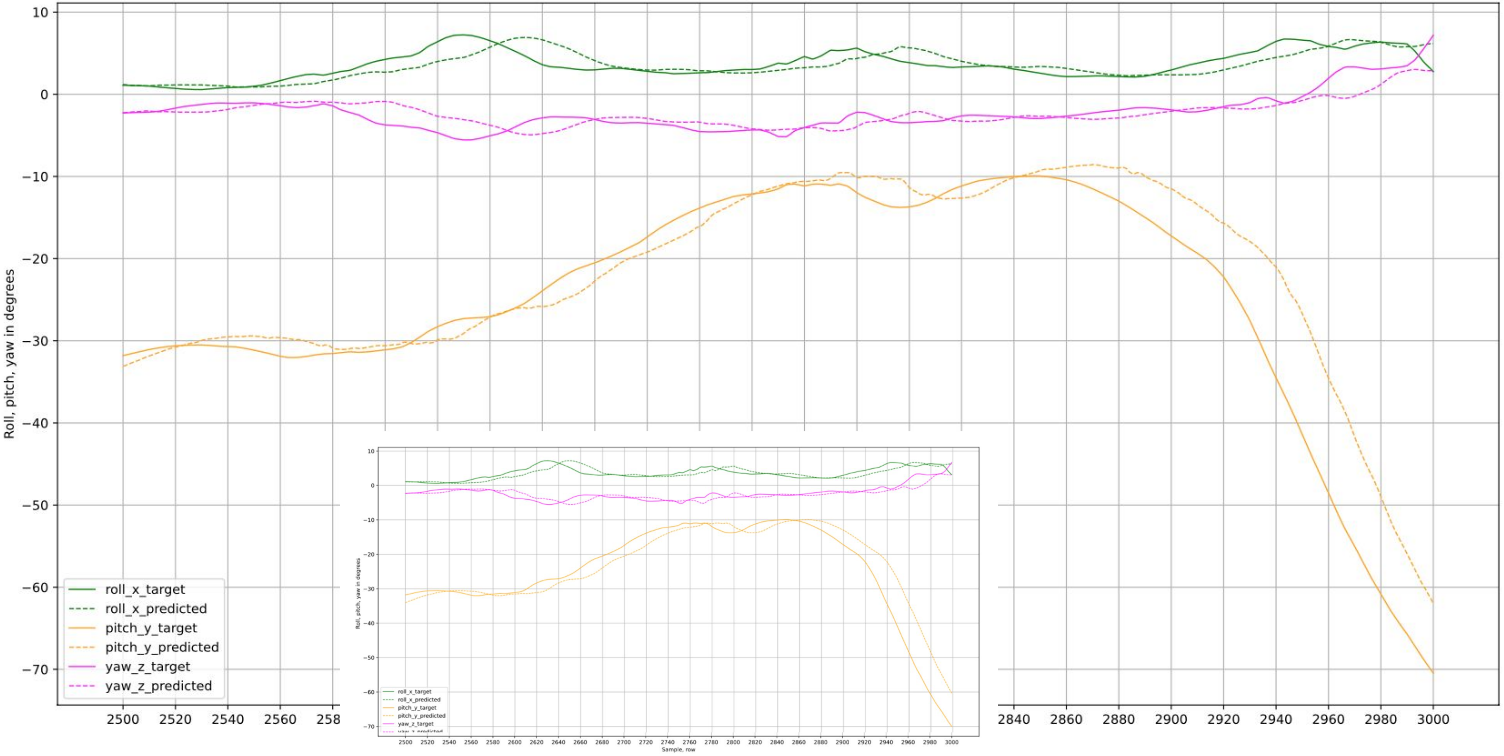
RMSE: position: 0.014m↓
rotation: 17,28°↓



LSTM3 with Mish(): **roll, pitch, yaw**

MAE: position: 0.012m↓
rotation: 13.18°↓

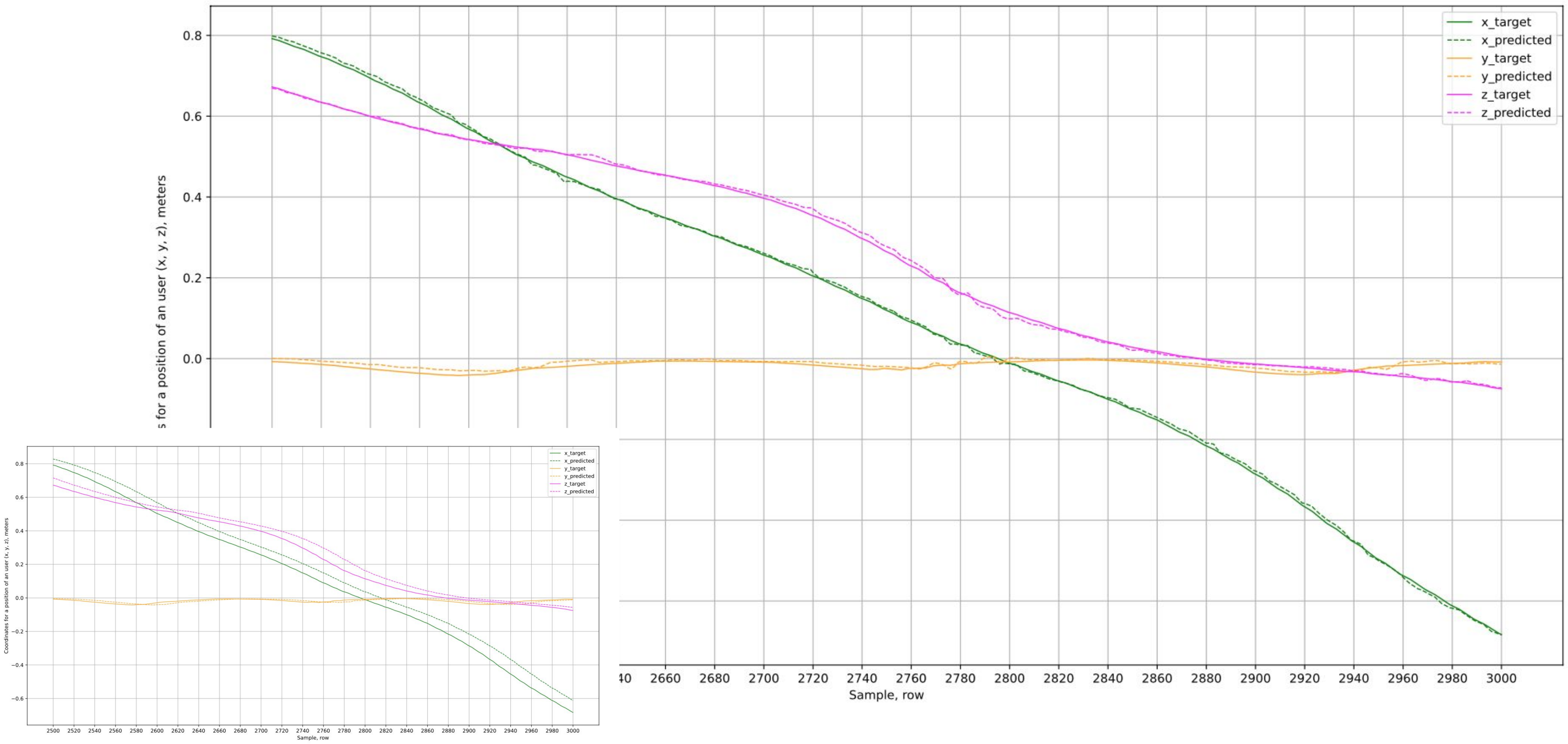
RMSE: position: 0.014m↓
rotation: 17,28°↓



GRU1: x, y, z-axis

MAE: position: 0.009m↓
rotation: 8.68°↓

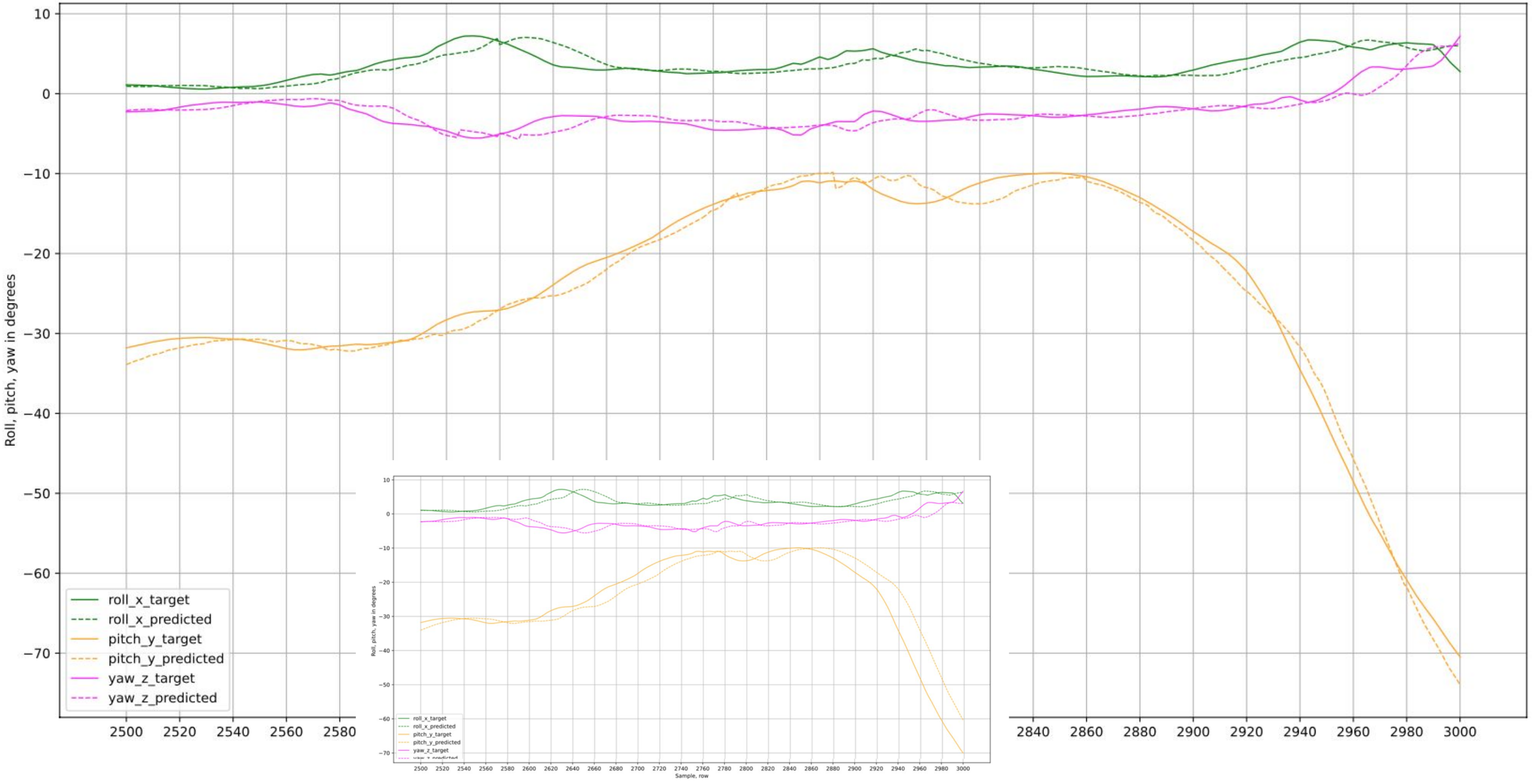
RMSE: position: 0.011m↓
rotation: 12,29°↓



GRU1: roll, pitch, yaw

MAE: position: 0.009m↓
rotation: 8.68°↓

RMSE: position: 0.011m↓
rotation: 12,29°↓



Introduction and fundamentals 01

02 Dataset and data exploration

Models 03

04 Evaluation

Experiments 05

06 **Analyze**

Discussion of the obtained results and steps for finishing the research

Conclusion & Analysis 1/4

The experiments done on a real head motion dataset collected from Microsoft HoloLens.

Proposed models can predict significant better than a Baseline model.

It is reasonable to build the best model in the prediction engine of the cloud-based streaming service.

Conclusion & Analysis 2/4

LSTM **improves** position error by **80%** and rotation error by **7,5%** compared to a Baseline.

Using Mish activation function **improves** position prediction by **82%** rotation prediction by **10%.**

Best performance is with **GRU-based model:**
Position error is **improved** by **85%**
and rotation error by **40%**!

Compared to LSTM prediction, GRU performs better by **24%** for position and by **36%** for rotation.

⟶same behaviour was found by other researchers*

* described in section "Related works" in my master thesis.

Conclusion & Analysis 3/4

Any variant of layered architecture needs significantly **more time to train** and can not catch the spatial dependencies despite complex architecture →
higher training and validation errors.

The bidirectional GRU **could not improve** metrics* of the unidirectional model.

* same behaviour was found by other researchers

Picture source: https://www.quora.com/What-are-some-good-memes-about-Machine-Learning

Conclusion & Analysis 4/4

For the prediction of rotational data the positional data can be eliminated from a dataset →user behaviour during dataset recording.

Users tended to turn their head always in direction to a VV because they were told to keep their sight mainly on this object.

Thank you for your attention

Oleksandra Baga
Master Computer Science (MSC)
Bachelor of Engineering (B.Eng.)
E-Mail: oleksandra.baga@gmail.com
https://github.com/oleksa-oleksa