View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by INRIA a CCSD electronic archive server

A Unified Evaluation Framework for Head Motion Prediction Methods in 360° Videos

Miguel Romero Rondon, Lucile Sassatelli, Ramon Aparicio-Pardo, Frédéric Precioso

▶ To cite this version:

Miguel Romero Rondon, Lucile Sassatelli, Ramon Aparicio-Pardo, Frédéric Precioso. A Unified Evaluation Framework for Head Motion Prediction Methods in 360° Videos. MMSys '20 - 11th ACM Multimedia Systems Conference, ACM, Jun 2020, Istanbul, Turkey. pp.279-284, 10.1145/3339825.3394934 . hal-02615979

HAL Id: hal-02615979

https://hal.archives-ouvertes.fr/hal-02615979

Submitted on 23 Jul 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A unified evaluation framework for head motion prediction methods in 360° videos

Miguel Fabián Romero Rondón
miguel-fabian.romero-rondon@etu.univ-cotedazur.fr
Université Côte d'Azur, CNRS, Inria, I3S, France

Lucile Sassatelli
lucile.sassatelli@univ-cotedazur.fr
Université Côte d'Azur, CNRS, I3S, France

Ramón Aparicio-Pardo
ramon.aparicio-pardo@univ-cotedazur.fr
Université Côte d'Azur, CNRS, I3S, France

Frédéric Precioso
frederic.precioso@univ-cotedazur.fr
Université Côte d'Azur, CNRS, Inria, I3S, France

ABSTRACT

The streaming transmissions of 360° videos is a major challenge for the development of Virtual Reality, and require a reliable head motion predictor to identify which region of the sphere to send in high quality and save data rate. Different head motion predictors have been proposed recently. Some of these works have similar evaluation metrics or even share the same dataset, however, none of them compare with each other. In this article we introduce an open software that enables to evaluate heterogeneous head motion prediction methods on various common grounds. The goal is to ease the development of new head/eye motion prediction methods. We first propose an algorithm to create a uniform data structure from each of the datasets. We also provide the description of the algorithms used to compute the saliency maps either estimated from the raw video content or from the users' statistics. We exemplify how to run existing approaches on customizable settings, and finally present the targeted usage of our open framework: how to train and evaluate a new prediction method, and compare it with existing approaches and baselines in common settings. The entire material (code, datasets, neural network weights and documentation) is publicly available.

CCS CONCEPTS

• **Computing methodologies → Model verification and validation**; **Virtual reality**.

KEYWORDS

360° videos, head motion prediction framework, saliency, dataset analysis

ACM Reference Format:
Miguel Fabián Romero Rondón, Lucile Sassatelli, Ramón Aparicio-Pardo, and Frédéric Precioso. 2020. A unified evaluation framework for head motion prediction methods in 360° videos. In *11th ACM Multimedia Systems Conference (MMSys'20), June 8–11, 2020, Istanbul, Turkey*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3339825.3394934

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys'20, June 8–11, 2020, Istanbul, Turkey
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-6845-2/20/06…$15.00
https://doi.org/10.1145/3339825.3394934

1 INTRODUCTION

360° videos are an important part of the Virtual Reality (VR) ecosystem, providing the users the ability to freely explore an omnidirectional scene and a feeling of immersion when watched in a VR headset. Given the closer proximity of the screen to the eye and the width of the content, the required data rate is two orders of magnitude that of a regular video [9]. To decrease the amount of data to stream, a solution is to send in high resolution only the portion of the sphere the user has access to at each point in time, named the Field of View (FoV). These approaches however require to know the user's head position in advance, that is at the time of sending the content from the server.

Owing to this acute need for head motion prediction in 360° video streaming, a number of recent approaches have proposed deep neural networks meant to exploit the knowledge of the past positions and of the content to periodically predict the next positions over a given horizon (e.g., [4, 7, 16, 17]). Some of these works have similar evaluation metrics or even use the same dataset, none of them however compares with their counterparts aiming the exact same prediction problem.

Our goal is to address the strong need for a comparison of existing approaches on common ground. For this reason, the main contribution of this work is a framework that allows researchers to study the performance of their new head motion prediction methods when compared with existing approaches on the same evaluation settings (dataset, prediction horizon, and test metrics). This software framework therefore contributes to progress towards efficient 360° systems. The entire material (code, datasets, neural network weights and documentation) is available at [12].

The paper is organized as follows. Section 2 introduces the definition of the problem of head motion prediction and each of the methods considered for reproduction and comparison. Section 3 describes the datasets used in these approaches and suggests an algorithm to create a uniform data structure from these heterogeneous dataset formats. Section 4 details the algorithms used to compute the saliency maps estimated from the raw video content or from the users' statistics. Section 5 details how to run existing approaches on customizable settings, and introduce two reference baselines. Section 6 presents how to prepare a testbed to assess comprehensively a new prediction method (on various datasets against several competitors). Finally, Section 7 concludes the paper.

2 REVIEW OF EXISTING HEAD MOTION PREDICTION METHODS

We first rigorously formulate the prediction problem which consists, at each video playback time t, in predicting the future user's head positions between t and $t + H$, as represented in Fig. 1, with the only knowledge of this user's past positions and the (entire) video content. We then provide a description and classification of each of the existing methods we compare with.

2.1 Problem Formulation

Let us first define some notation. Let $\mathbf{P}_t = [\theta_t, \varphi_t]$ denote the vector coordinates of the FoV at time t. Let $\mathbf{V}_t$ denote the considered visual information at time t: depending on the models' assumptions, it can either be the raw frame with each RGB channel, or a 2D saliency map resulting from a pre-computed saliency extractor (embedding the motion information). Let T be the video duration. We now refer to Fig. 1. Let H be the *prediction horizon*. We define the terms *prediction step* and video *time-stamp* as predicting for all *prediction steps* $s \in [0, H]$ from video *time-stamp* t. For every *time-stamp* $t \in [T_{start}, T]$, we run predictions $\hat{\mathbf{P}}_{t+s}$, for all *prediction steps* $s \in [0, H]$.

We formulate the problem of trajectory prediction as finding the best model $\mathbf{F}^*_H$ verifying:

$$\mathbf{F}^*_H = \arg\min \mathbb{E}_t \Big[D\Big(\big[\mathbf{P}_{t+1}, \ldots, \mathbf{P}_{t+H}\big], \mathbf{F}_H\big([\mathbf{P}_t, \mathbf{P}_{t-1}, \ldots, \mathbf{P}_0, \mathbf{V}_{t+H}, \mathbf{V}_{t+H-1}, \ldots, \mathbf{V}_0] \big) \Big) \Big]$$

where $D(\cdot)$ is the chosen distance between the ground truth series of the future positions and the series of predicted positions. For each s, we average the errors $D(\hat{\mathbf{P}}_{t+s}, \mathbf{P}_{t+s})$ over all $t \in [T_{start}, T]$.

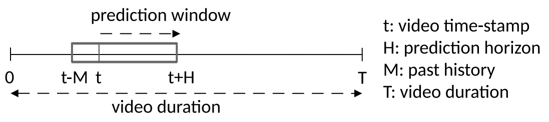

Figure 1: Head motion prediction: For each time-stamp t, the next positions until $t + H$ are predicted.

2.2 Methods for Head Motion Prediction

Various approaches to predict user motion in 360° video environments have been published in the last couple of years. Here we consider that the users' statistics for the specific video are *not* known at test time, hence we do not consider methods relying on these per-video statistics, such as [10, 13]. Each considered method from the literature is named according to the name of the conference or journal it was published in, appended with the year of publication.

PAMI18: Xu et al. in [16] design a Deep Reinforcement Learning model to predict head motion. Their deep neural network only receives the viewer's FoV and has to decide to which direction and with which magnitude the viewer's head will move. The prediction horizon is only one frame, around 30ms. By only injecting the FoV, the authors make the choice not to consider the positional information explicitly as input.

IC3D17: The strategy presented by Aladagli et al. in [1] extracts the saliency from the current frame with an off-the-shelf method, identifies the most salient point, and predicts the next FoV to be centered on this most salient point. It then builds recursively. We therefore consider this method to be a sub-case of PAMI18.

ICME18: Ban et al. in [2] use a linear regressor first learned to get a prediction of the displacement, which it then adjusts by computing the centroid of the k nearest neighbors corresponding to other users' positions at the next time-step, and hence assume more information than our case of study.

CVPR18: In [17], Xu et al. predict the gaze positions over the next second in 360° videos based on the gaze coordinates in the past second, and saliency and motion information on the entire equirectangular projection and the FoV of the current frame and next frame.

MM18: Nguyen et al. in [7] first construct a saliency model based on a deep convolutional neural network and named PanoSalNet. The so-extracted saliency map is then fed, along with the position encoded as a mask, into a doubly-stacked LSTM, to finally predict the tiles that pertain to the FoV.

NOSSDAV17: Fan et al. in [4] propose to concatenate the positional information, saliency and motion maps then fed into LSTM, they propose two neural networks to predict the head orientations or the likelihood that tiles pertain to future FoV in the future H time-steps.

ChinaCom18: Li et al. in [6] present a similar approach as NOSSDAV17, adding a correction module to compensate for the fact that tiles predicted to be in the FoV may not correspond to the actual FoV shape. We consider this model to be a sub-case of NOSSDAV17.

TRACK: In [11], we present a neural network that processes independently the time-series of positions and the visual features with dedicated recurrent units (LSTMs), before fusing the two embeddings with a third recurrent unit used to predict the sequence of future H time-steps.

Selected methods analyzed. From Sec. 3 we present the analysis of the following methods: PAMI18, NOSSDAV17, MM18 and CVPR18, the remaining methods are either considered sub-cases of the selected methods or assume there is more information available such as the viewers' position statistics.

3 UNIFORM DATA FORMATS

One of the challenges when evaluating a head motion prediction method across multiple datasets is to adapt it to the specific attributes of each dataset. Consider the case of a model trained with a specific sampling rate that is evaluated on a dataset with a different sampling rate, or where the size or the format of the visual input is different. It is important to have a convention on the structure of the datasets, as it becomes easier to read, sort, understand and compare homogeneous data. In this section, we first describe how to use our framework to post-process the datasets and get a uniform structure shared among all datasets considered in this work. We then provide a way to analyze the datasets.

3.1 Make the Dataset Structure Uniform

The datasets used to evaluate the methods discussed in Sec. 2.2 contain visual and head motion data for 360° videos, stored in different formats. In Table 1 we describe each of the datasets we analyze in our repository and we show how each dataset has a

Reference	Head Pos. Log Format	Saliency Maps	Raw Videos	Storage of Head Pos. Traces	Code and Neural Network
NOSSDAV17 [4]	Yaw, Pitch and Roll in range [-180, 180].	A MP4 file per video of size 1920×3840.	No.	A CSV file per trace.	No.
PAMI18 [16]	Longitude and latitude in range [-180, 180] and [-180, 180] respectively.	Not provided.	MP4 format.	MATLAB file with an entry per video, each with a matrix with a column per user and alternating longitude and latitudes in the rows.	Found in [15].
CVPR18 [17]	Longitude and latitude in range [-0, 1], origin in bottom-left corner.	Not provided.	MP4 format.	A folder per user with a text file per trace.	No.
MM18 [7]	3D position in the unit sphere, (x, y, z) in range [-0, 1].	A Python array per video of size 9×16.	No.	Python dictionary with an entry per video, each with a list with an entry per user.	Found in [8].
MMSys18 [3]	longitude and latitude in range [-0, 1], origin in top-left corner.	A binary file per video of size 1024×2048.	MP4 format.	A CSV file with the traces of all users per video.	N/A.

Table 1: Features of the file structure and format of the datasets used in each referenced method.

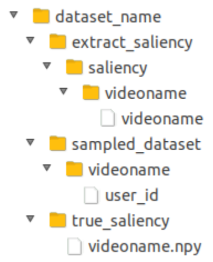
```
▼ dataset_name
  ▼ extract_saliency
    ▼ saliency
      ▼ videoname
          videoname
  ▼ sampled_dataset
    ▼ videoname
        user_id
  ▼ true_saliency
      videoname.npy
```

Figure 2: Uniform dataset file structure

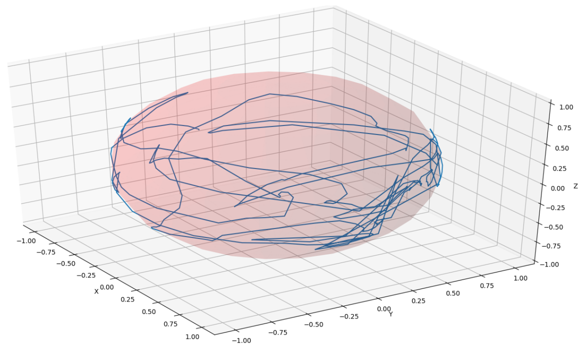

Figure 3: Exploration of user "45", in video "drive" from NOSSDAV17, represented in the unit sphere.

particular format and schema. Some of them store the head position data in language-specific formats (e.g. PAMI18 stores the data in a Matlab-file and MM18 stores the data in a Python dictionary), others store the head position data in text files (e.g. CVPR18 groups the files in a folder per user, MMSys18 uses a CSV file per video, while NOSSDAV17 uses a CSV file per user and video). Some datasets contain the saliency maps and the raw videos (MMSys18), others store only the saliency maps (NOSSDAV17, MM18) while others contain only the raw videos in mp4 file (PAMI18, CVPR18).

We propose an algorithm that allows to read each of the datasets, with methods to cleanse the data and produce traces in a uniform format, common for all the datasets. The uniform dataset structure is shown in Fig. 2. The following command is used to run the analysis on each dataset:

```
python {Fan_NOSSDAV_17, Nguyen_MM_18,
       Xu_PAMI_18}/Read_Dataset.py −analyze_data
```

Thanks to the analysis on the original datasets, we found that: (i) there are a few missing entries in the PAMI18 dataset, (ii) when re-implementing the tile mapping algorithm from NOSSDAV17, we found that there is a discrepancy in the tiles generated and the tile numbers provided in the dataset, and (iii) when observing the time-stamps in MM18's dataset, most of the traces are splitted and concatenated, and there are intersections between the time-stamps. By creating this dataset structure, we not only provide ways to read, parse and analyze the different datasets, we also allow to sample the datasets with a common sampling rate (by default 0.2 seconds). To subsample the datasets, we first transform the head position format from the original dataset to the quaternion representation. Then, we perform the spherical linear interpolation of the rotations (represented as quaternions) with a constant angular velocity, the rotations are interpolated at the rate of 0.2 seconds. Finally we transform the sampled quaternions into 3D coordinates. We provide a method on each dataset to visualize the exploration of the user in the unit sphere. For example, to obtain a plot similar to that of Fig. 3, the following command can be used:

```
python Fan_NOSSDAV_17/Read_Dataset.py −plot_3d_traces
```

In our repository [12] we provide a folder "sampled_dataset" for each of the original datasets (NOSSDAV17, CVPR18, PAMI18, MM18 and MMSys18), with a sub-folder per video. Inside, a text file per user stores the head motion trace indicating the time-stamp, followed by the 3D coordinates of the unit vector (x, y, z).

For example, to create the sampled dataset from the original dataset of PAMI18, the command to use is:

```
python Xu_PAMI_18/Read_Dataset.py −creat_samp_dat
```

We also provide functions to plot and verify that the sampling is correctly done. For example, the following command is used to compare the sampled trace against the original trace to get plots similar to Fig. 4:

```
python Xu_PAMI_18/Read_Dataset.py −compare_traces
```

3.2 Analysis of Head Motion in each Dataset

We provide the code to compute the Cumulative Distribution Function (CDF) of the maximum angular distance from the head position

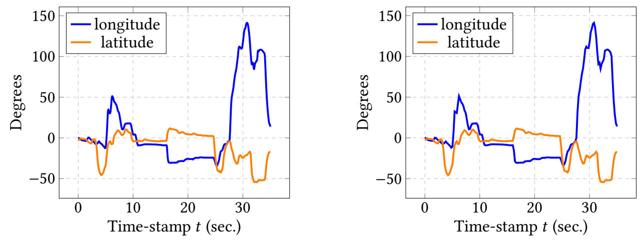

Figure 4: Comparison between the original trace (left) and the sampled trace (right) for user "m1_21" video "Terminator" in PAMI18 dataset.

at the start of the prediction window t for different prediction window lengths $H \in 0.2, 0.5, 1, 2, 5, 15$ seconds, as shown in Fig. 5. The following command can be used to get these plots for each dataset:

```
python DatasetAnalysis/DatasetAnalysis_{CVPR18, MM18,
        MMSys18, NOSSDAV17, PAMI18}.py
```

We observe that in the MMSys18 dataset [3], 50% of the users have shifted their FoV by more than its width (100°) after 5 sec., while in the datasets of CVPR18, NOSSDAV17, MM18 and PAMI18, the percentage is 30%, 20%, 20% and 15%.

4 SALIENCY EXTRACTION

The saliency map is a heatmap (2D distribution) that identifies what are the points in the 360° scene that attract the attention of the viewers the most. Besides the time series of past positions, the saliency map is one of the considered input modalities of the existing head motion prediction methods. Each of these methods extract the visual features in a different way. If we want to fairly compare different methods for head motion prediction, we would need to use the same post-processing to obtain the salient visual features. In our framework, we propose an algorithm to create saliency maps estimated from the video content using the same saliency detection model for all the datasets considered for reproduction and comparison. In a second case, if we want our evaluation to be independent from the imperfection of any saliency prediction model, we use a method based on users' statistics, namely *ground-truth saliency map*. It is the heatmap of the viewing patterns, obtained at each point in time from the users' traces. In this section, we describe how to compute each of these saliency maps using our code.

4.1 Ground-Truth Saliency Map

To compute the ground-truth saliency maps, we consider the point at the center of the viewport $P^t_{u,v}$ for user $u \in U$ and video $v \in V$ at the t^{th} time-stamp ($t = 0, \ldots, T$), where T is the length of the sampled traces. For each head position $P^t_{u,v}$, we draw a frame in equirectangular projection, and compute the orthodromic distance $D(\cdot)$ from $P^t_{u,v}$ to each point $Q_{x,y}$ in the equirectangular frame with longitude x and latitude y. Then, we use a modification of the radial basis function (RBF) kernel shown in Eq. 1 to convolve the points in the equirectangular projection.

$$GT\_Sal^t_{u,v,x,y} = \exp\left(-\frac{D(P^t_{u,v}, Q_{x,y})^2}{2\sigma^2}\right), \quad (1)$$

where $D(P^t_{u,v}, Q_{x,y})$ is the orthodromic distance, that is computed using Eq. 2.

$$D(P, Q) = \arccos(\vec{P} \bullet \vec{Q}), \quad (2)$$

where $\bullet$ is the dot product operation, and $\vec{P}$ are the coordinates in the unit sphere of point P. For a point $P = (x, y)$, where x is the longitude and y is the latitude, the coordinates in the unit sphere are $\vec{P} = (\cos x \cos y, \sin x \cos y, \sin y)$.

We compute saliency maps $GT\_Sal^t_{u,v}$ per user $u \in U$, video $v \in V$ and time-stamp t by convolving each head position $P^t_{u,v}$ with the modified RBF function in Eq 1, a value of $\sigma = 6°$ is chosen. The saliency map per video frame can be calculated as follows $GT\_Sal^t_v = \frac{1}{U}\sum_{u \in U} GT\_Sal^t_{u,v}$, where U is the total number of users watching this video-frame. An example of the ground-truth saliency map is shown in Fig. 6. The file Read_Dataset.py under the folder of each dataset contains all the methods to create the ground-truth saliency maps, for example, the command to compute and store the ground-truth saliency maps for David_MMSys_18 dataset is:

```
python David_MMSys_18/Read_Dataset.py −creat_true_sal
```

4.2 Content-Based Saliency Maps

To extract saliency maps from the content, we provide the workflow that uses PanoSalNet [7, 8], also considered in MM18. The neural network of PanoSalNet is composed by nine convolution layers, the first three layers are initialized with the parameters of VGG16 [14], the following layers are first trained on the images of the SALICON dataset [5], and finally the entire model is re-trained on 400 pairs of video frames and saliency maps in equirectangular projection. To create the content-based saliency maps we first need to transform the videos into scaled images. We provide a executable file to create images from each video with a rate of 5 samples per second (the same sampling rate used to create our "sampled_dataset" from Sec. 3). The file for each dataset is:

```
{Xu_CVPR_18, Xu_PAMI_18, David_MMSys_18}/dataset/
        creation_of_scaled_images.sh
```

The file panosalnet.py under the folder Extract_Saliency contains the methods to create the content-based saliency maps. As an example, we provide the command to create the saliency map for each frame in each video in CVPR18's dataset:

```
python Extract_Saliency/panosalnet.py −gpu_id 0
        −dataset_name CVPR_18
```

However, for the datasets that do not provide the 360° videos, but directly the saliency maps (NOSSDAV17 and MM18), we can create the content-based saliency maps using their provided information. For example, this command can be used for MM18:

```
python Nguyen_MM_18/Read_Dataset.py −creat_cb_sal
```

An example of content-based saliency map is shown in Fig. 6.

5 EVALUATION OF ORIGINAL METHODS

We provide the algorithms to run the experiments of PAMI18, CVPR18, MM18, ChinaCom18 and NOSSDAV17 with their original settings, evaluation metrics and datasets. To have a common

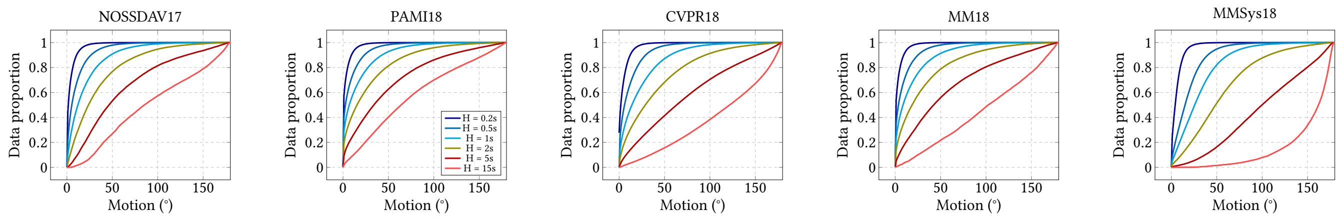

Figure 5: From left to right: Motion distribution of the datasets used in NOSSDAV17, PAMI18, CVPR18, MM18, and the MMSys18-dataset from [3]. The x-axis corresponds to the motion from position t to position t+H in degrees.

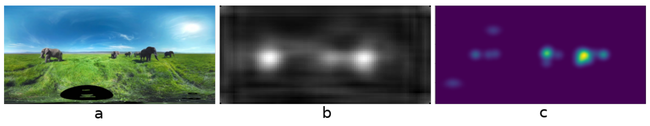

Figure 6: Saliency maps computed for frame "98" in video "160" from CVPR18 dataset. a) Original frame. b) Content-based saliency. c) Ground-truth saliency.

reference point on each evaluation experiment, we also present how to run two different baselines:

- *no-motion baseline*: Simple baseline that assumes that the head of the user stays still during the whole prediction horizon, no learning involved.
- *position-only baseline:* Sequence-to-sequence LSTM-based architecture that exploits the time series of past positions only (disregarding the video content).

The file Baselines.py under the folder of each dataset contains all the methods to run the experimental setup of each of the works for a given method. Depending on the type of result either a plot or a table is shown at the end of the execution, for the case of MM18, the figure shown in Fig. 7(left) is obtained by executing this file. The following command can be used to get the results for the no-motion baseline in the experimental setup of MM18:

```
python Nguyen_MM_18/Baselines.py –gpu_id "" –model_name
    no_motion
```

6 TYPICAL USAGE OF THE HEAD MOTION PREDICTION FRAMEWORK

Importantly, the software framework we propose enables to prepare a testbed and compare different methods on the same head motion prediction experiment. It therefore eases the assessment of new prediction techniques, and compare them with other existing prediction methods and baselines. Our framework allows to train and evaluate a given prediction model on a chosen dataset, specifying the prediction horizon and history window, among other parameters described below. We first detail the formatting of the commands and the available options, before developing some examples.

6.1 Training and Evaluation

To train or evaluate a given neural network in a specific dataset and configure some basic settings, the following command can be used:

```
python training_procedure.py –{evaluate, train} –gpu_id
    GPU_ID –dataset_name DATASET_NAME –model_name
    MODEL_NAME –init_window T_START –m_window
    M_WINDOW –h_window H_WINDOW [–end_window
    T_END] –exp_folder FOLDER_NAME [–provided_videos]
    –use_true_saliency –metric {orthodromic, mse}
```

Here is the detail of each option:

- **-evaluate/-train:** This option allows to decide if we want to train or evaluate the neural network defined in MODEL_NAME.
- **-gpu_id:** Specify the GPU_ID to load the neural network, if the parameter is left empty, the neural network will be loaded on CPU.
- **-dataset_name:** Select the dataset to use with the parameter DATASET_NAME, the options are: Xu_PAMI_18, Xu_CVPR_18, Fan_NOSSDAV_17, Nguyen_MM_18, Li_ChinaCom_18 and David_MMSys_18.
- **-model_name:** Select the model to train or evaluate, the options are: no_motion, pos_only, CVPR18, MM18, TRACK, among others.
- **-init_window:** In the experiment, the prediction will not be assessed over the first T_START time-stamps of the videos.
- **-m_window:** The neural network takes into account the last M_WINDOW time-stamps from time t, also named history window in Fig. 1.
- **-h_window:** The prediction horizon, we try to predict over the following H_WINDOW time-stamps from time t.
- **-end_window:** The prediction is not assessed over the last T_END time-stamps of the videos, by default T_END is equal to H_WINDOW.
- **-exp_folder:** The folder to read the traces. The default value is "sampled_dataset", the folder created when uniformly sampling all datasets in Sec. 3.
- **-provided_videos:** Flag to use in case the partition into train and test videos are provided in the original dataset.
- **-use_true_saliency:** Flag that tells whether to use true saliency, if not set, then content-based saliency is used.
- **-metric:** Metric used for the evaluation, by default *orthodromic distance* (orthodromic), but *mean squared error* (mse) can be used too. More metrics can be easily added by filling up the Python dictionary "all_metrics" in the script "Utils.py".

6.2 Examples of Usage

We now present a few examples on how to use our framework to get the results for the methods of CVPR18, MM18 and TRACK in an experimental setup where the prediction horizon is $H = 5$ seconds, the evaluation metric is the orthodromic distance, using the dataset of MMSys18, and using the ground-truth saliency for CVPR18 and MM18 and the content-based saliency for TRACK. The plot obtained by running the commands presented below is shown in Fig. 7(right).

CVPR18

Since the code of CVPR18 is not publicly available, the neural network of CVPR18 used here is a replica from the description in [17]. In our replica of CVPR18, we decided to prune the Saliency Encoder Module and replace it directly with the ground-truth to be independent from the imperfection of the saliency encoder module fed with the visual content. The following command is used to get the results for the replica of the neural network of CVPR18 on our experimental setup, trained and tested with ground-truth saliency maps:

```
python training_procedure.py –evaluate –gpu_id 0
     –dataset_name David_MMSys_18 –model_name CVPR18
     –init_window 30 –m_window 5 –h_window 25
     –exp_folder original_dataset_xyz –provided_videos
     –use_true_saliency
```

MM18

For the case of MM18, the model and weights are publicly available in [8]. Since the model of MM18 predicts the head orientation at time $t + H$, we had to retrain the model for each prediction step in the prediction horizon, i.e., for each $s \in \{0.2s, 0.4s, \cdots, H = 5s\}$. To get the results for the neural network of MM18 on our experimental setup, using ground-truth saliency maps, use the following command:

```
python training_procedure.py –evaluate –gpu_id 0
     –dataset_name David_MMSys_18 –model_name MM18
     –init_window 30 –m_window 15 –h_window 25
     –exp_folder original_dataset_xyz –provided_videos
     –use_true_saliency
```

TRACK

We provide as example the command to evaluate the neural network of TRACK:

```
python training_procedure.py –evaluate –gpu_id 0
     –dataset_name David_MMSys_18 –model_name TRACK
     –init_window 30 –m_window 5 –h_window 25
     –exp_folder original_dataset_xyz –provided_videos
```

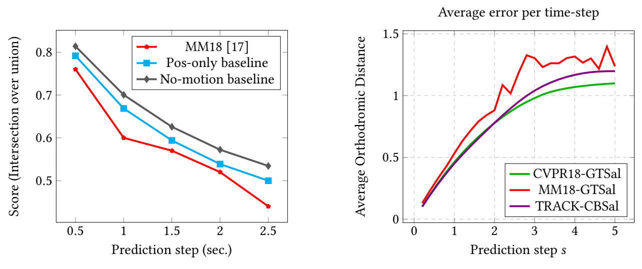

Figure 7: (Left) Performance of the no-motion baseline and position-only baseline compared with the results of MM18 in [7]. (Right) Average error of the models of MM18, CVPR18 using Ground-truth saliency and the model TRACK using Content-based saliency, tested on MMSys18's dataset [3].

7 CONCLUSIONS

In this paper we presented a framework to evaluate and compare different methods to predict head position in 360° videos. In this framework, we propose an algorithm to create a uniform data structure from each of the heterogeneous datasets evaluated in this work. We described the algorithms used to compute the saliency maps either estimated from the raw video content or from the users' statistics, considering a kernel fitted for the equirectangular projection used to encode 360° videos. To compare each of the head motion prediction settings to a common reference, we detailed the commands to estimate the performance of different approaches in each original evaluation context (prediction horizon, metrics and datasets). Finally, we presented how to use our framework to prepare a testbed to assess comprehensively a new prediction method (on various datasets against several competitors). This software framework therefore contributes to progress towards efficient 360° streaming systems. The entire material (codes, datasets, neural network weights and documentation) is available at [12].

ACKNOWLEDGMENTS

This work has been supported by the French government through the EUR DS4H Investments in the Future project ANR-17-EURE-0004.

REFERENCES

[1] A. D. Aladagli, E. Ekmekcioglu, D. Jarnikov, and A. Kondoz. 2017. Predicting head trajectories in 360° virtual reality videos. In *IEEE Int. Conf. on 3D Immersion.*
[2] Y. Ban, L. Xie, Z. Xu, X. Zhang, Z. Guo, and Y. Wang. 2018. Cub360: Exploiting cross-users behaviors for viewport prediction in 360 video adaptive streaming. In *IEEE Int. Conf. on Multimedia and Expo (ICME).*
[3] E. J. David, J. Gutiérrez, A. Coutrot, M. P. Da Silva, and P. Le Callet. 2018. A dataset of head and eye movements for 360-Degree videos. In *ACM MMSys.* 432–437.
[4] C. Fan, J. Lee, W. Lo, C. Huang, K. Chen, and C. Hsu. 2017. Fixation prediction for 360 video streaming in head-mounted virtual reality. In *ACM NOSSDAV.* 67–72.
[5] X. Huang, C. Shen, X. Boix, and Q. Zhao. 2015. Salicon: Reducing the semantic gap in saliency prediction by adapting deep neural networks. In *IEEE ICCV.* 262–270.
[6] Y. Li, Y. Xu, S. Xie, L. Ma, and J. Sun. 2018. Two-Layer FoV Prediction Model for Viewport Dependent Streaming of 360-Degree Videos. In *Int. Conf. on Comm. and Netw. in China (ChinaCom).* Chengdu, China, 501–509.
[7] A. Nguyen, Z. Yan, and K. Nahrstedt. 2018. Your Attention is Unique: Detecting 360-Degree Video Saliency in Head-Mounted Display for Head Movement Prediction. In *ACM Int. Conf. on Multimedia.* 1190–1198.
[8] A. Nguyen, Z. Yan, and K. Nahrstedt. 2019. PanoSalNet - Source code for Your Attention is Unique: Detecting 360-Degree Video Saliency in Head-Mounted Display for Head Movement Prediction. https://github.com/phananh1010/PanoSalNet.
[9] J. Park, P. A Chou, and J. Hwang. 2018. Rate-Utility Optimized Streaming of Volumetric Media for Augmented Reality. *arXiv preprint arXiv:1804.09864* (2018).
[10] S. Petrangeli, G. Simon, and V. Swaminathan. 2018. Trajectory-Based Viewport Prediction for 360-Degree Virtual Reality Videos. In *2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR).* 157–160.
[11] M. Romero, L. Sassatelli, R. Aparicio, and F. Precioso. 2020. Revisiting Deep Architectures for Head Motion Prediction in 360° Videos. *arXiv preprint arXiv:1911.11702* (2020).
[12] M. Romero, L. Sassatelli, R. Aparicio, and F. Precioso. 2020. Source code. https://gitlab.com/miguelfromeror/head-motion-prediction/tree/master.
[13] S. Rossi, F. De Simone, P. Frossard, and L. Toni. 2019. Spherical Clustering of Users Navigating 360-Degree Content. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 4020–4024.
[14] K. Simonyan and A. Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[15] M. Xu, Y. Song, J. Wang, M. Qiao, L. Huo, and Z. Wang. 2018. Dataset for Predicting Head Movement in Panoramic Video: A Deep Reinforcement Learning Approach (PAMI 2018). https://github.com/YuhangSong/DHP.
[16] M. Xu, Y. Song, J. Wang, M. Qiao, L. Huo, and Z. Wang. 2018. Predicting Head Movement in Panoramic Video: A Deep Reinforcement Learning Approach. *IEEE Trans. on PAMI* (2018).
[17] Y. Xu, Y. Dong, J. Wu, Z. Sun, Z. Shi, J. Yu, and S. Gao. 2018. Gaze Prediction in Dynamic 360° Immersive Videos. In *IEEE CVPR.* 5333–5342.