Trace Match & Merge: Long-Term Field-Of-View Prediction for AR Applications

Adam Viola*, Sahil Sharma*,
Pankaj Bishnoi*
University of Massachusetts at Amherst, USA

Matheus Gadelha, Stefano Petrangeli,
Haoliang Wang, Viswanathan Swaminathan
Adobe Research, USA

***Abstract*—Photorealistic Augmented Reality (AR) experiences require high bandwidth. Streaming approaches have already been proposed to guarantee a low-latency and yet high quality experience for the end user despite this bandwidth requirement. To provide the best streaming performance, these approaches usually require accurate prediction of the Field-Of-View (FOV) of the AR device to prioritize the download of digital objects that are most likely to be viewed by the user. While current prediction approaches can be successfully applied for short prediction horizons (<2 seconds), it remains challenging to predict the user behavior for longer horizons. In this paper, we therefore present the *Trace Match & Merge* (TMM) algorithm for the FOV prediction in AR applications. TMM employs an improved nearest-neighbor-based approach that examines the traces of previous users of a particular AR scene, and merges the most similar segments to predict the future position and rotation of the AR device. Particularly, user traces are selected based on the overlap of the users' FOV, modeled as a pyramid. Extensive experimental results on an open-source AR exploration dataset, composed of 4 AR scenes explored by 50 users, confirm the benefits of the proposed solution. Particularly, TMM consistently outperforms a set of popular baselines in terms of predicted position, rotation and objects-in-view – especially for challenging, long-term prediction horizons (>2 seconds).**

***Index Terms*—Augmented Reality, Field-Of-View, Viewport, Prediction, Streaming**

I. Introduction

Augmented Reality (AR) is an interactive experience defined by the integration of digital objects into a real-world environment. Existing AR experiences are powered by a device with a display, such as a smartphone, tablet, or headset, that overlays digital objects over the physical environment within the device Field-Of-View (FOV). AR can enable novel interactive and immersive experiences in a variety of domains such as entertainment, manufacturing, education and others. Development toolkits such as ARKit by Apple and ARCore by Google have recently made AR accessible to anyone with a tablet or smartphone. However, the high bandwidth requirements of photorealistic AR creations can hinder the mainstream adoption of these high quality experiences. Many of the photorealistic models and materials that make up detailed AR objects have sizes ranging from tens to hundreds of megabytes. Since AR experiences cannot begin without their digital objects, the lengthy download phase before an AR experience starts may result in a poor user experience.

*These authors contributed equally to this work.

To mitigate this latency problem, streaming techniques – similar to those successfully applied in the video domain – have already been proposed in the AR domain [1], [2]. These techniques usually prioritize the download of digital objects near the AR device FOV, and select the optimal Level-Of-Detail (LOD) needed to guarantee a high quality AR experience. Although these systems can successfully reduce start-up latency, they do not take into consideration the future behavior of the user in the AR scene. With an accurate prediction of the AR device FOV several seconds in the future, an AR streaming solution could prioritize the download of digital objects at the optimal LOD based on the *expected* FOV of the user. As a result, this would guarantee an even better final user experience.

Wang et al. describe an approach for AR FOV prediction that predicts the nearest digital object in the FOV of the user [3]. Although results are promising, the identity of the single nearest digital object is not always enough, as many AR experiences feature multiple digital objects within the field-of-view at the same time. Beyond this approach, the problem of FOV prediction for AR applications has not been well investigated yet. Several FOV prediction methods have been proposed to support the delivery of 360-degree videos [4]–[6]. Similarly to AR, 360-degree video viewers are only able to view a particular region of the 360-degree video at any given time. 360-degree video streaming techniques take advantage of this fact by varying the streamed quality of each region of the video based on the viewer's current and predicted FOV. However, the AR FOV prediction problem features several unique challenges that remain unaddressed by existing FOV prediction methods tailored for 360-degree videos. First, AR enables users to move with six degrees of freedom (6DOF), in contrast to classical 360-degree videos that only support rotational movements (3DOF). Second, AR experiences are made of a sparse set of discrete objects that are blended with the physical world, while 360-degree videos are a "dense" medium, as the user is totally immersed in the virtual environment. Third, AR experiences are often highly interactive. Digital objects that react to particular actions of the user can complicate the FOV prediction task.

In light of the above challenges, we present in this paper the *Trace Match & Merge* (TMM) algorithm, a nearest-neighbour-based approach for predicting the FOV in AR applications. TMM uses a scene-specific index of trace segments and the

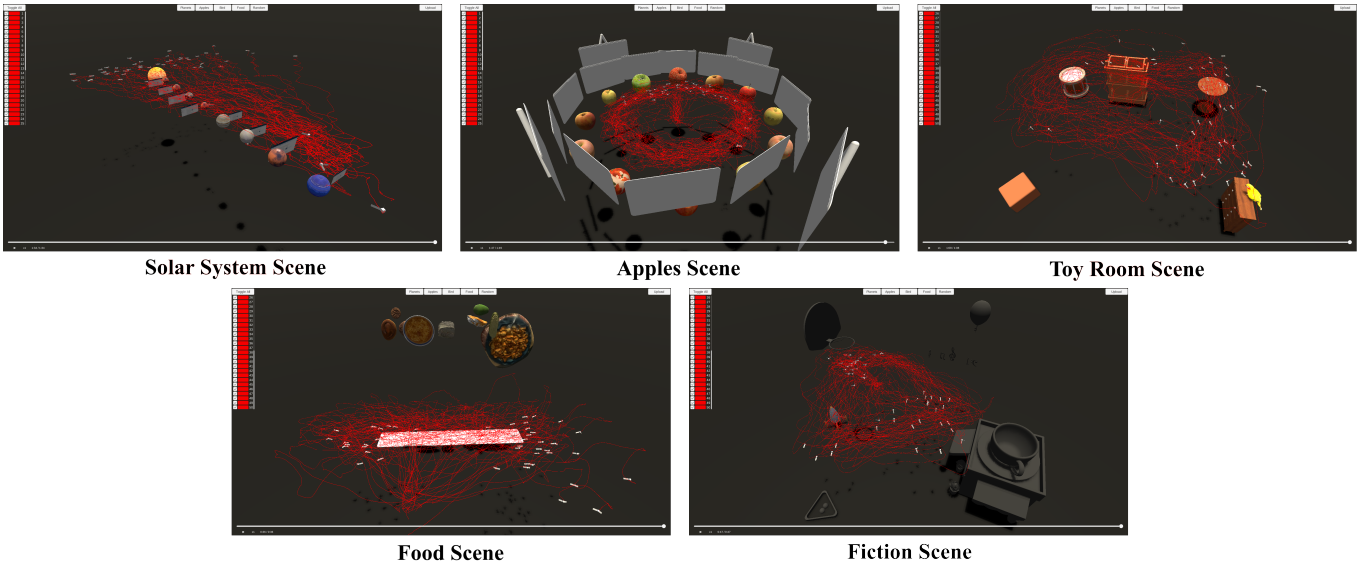

Fig. 1. The AR scenes composing the ACE dataset [3] and the user traces (in red). Each scene is explored by 50 different users.

recent user position history to locate similar position segments of previous users. This selection is performed by comparing the FOV of the current and past users – modelled as a pyramid, and finding past traces with the highest degree of similarity with that of the current user. The selected segments are then appropriately merged into a final trace that is used for prediction. Extensive experimental results on the ACE dataset [3], an open-source AR exploration dataset composed of 5 AR scenes explored by 50 users (Figure 1), confirm the benefits of our solution. Our TMM approach outperforms several baseline prediction algorithms across different AR scenes exhibiting different navigation patterns, especially for challenging, longer prediction horizons (> 2 seconds).

The remainder of this paper is organized as follows. Section II presents related work on streaming and prediction for immersive media. Sections III details the proposed TMM approach, while Section IV presents a comprehensive analysis of the performance of TMM when compared to several baselines on the ACE datasets. Finally, Section V concludes the paper and presents several directions for future work in this domain.

II. Related Work

A. AR Streaming

To improve user experience in bandwidth-intensive AR applications, Petrangeli et al. [1] propose an HTTP adaptive streaming (HAS) based framework for AR applications. It works by dynamically deciding which AR objects in the AR scene should be streamed and at what quality level. This choice is determined by computing an object utility value, which depends on the position of the object with respect to the viewer and the visual improvement a given level of detail provides with respect to the amount of bandwidth necessary to stream it. The proposed streaming framework improves the startup latency as compared to a benchmarking download-and-play strategy by 90%. Experiments performed on a simple AR scene comprised of 4 objects also show a reduction in the amount of delivered data by 79%, with minimal impact on the user experience – users are able to view objects at the optimal visual quality for almost 80% of the time. Noh et al. propose a cloud-assisted system for AR streaming in wireless environments [7]. An enhanced scene manifest is employed, which allows the client to select the best LOD level of the mesh to improve the visual quality contribution while taking into account the available bandwidth. Park et al. develop a 3D tiling system for the efficient streaming of AR volumentric media [8]. They develop a utility metric to dynamically select the best 3D tile, based on bandwidth conditions and the distance of the user from the digital object. van der Hooft et al. employ the latest point cloud compression standards to compress AR volumetric videos and a rate adaptation heuristic to select the best quality level for a multi-object AR scene [2].

Accurate, long-term FOV prediction can be leveraged to improve the performance of these AR streaming works, and provide an even better user experience.

B. AR FOV Prediction

Wang et al. present a novel user visual attention FOV prediction algorithm for AR applications [3]. Instead of predicting the 6DOF of the AR device, the authors use the users' historical movements and the positions of the digital objects to predict the nearest object within view. This approach outperforms baseline algorithms traditionally employed for 360-degree video FOV prediction (see Section II-C), including dead-reckoning-based [6] and linear regression-based [4] methods. This work also introduces the AR Content Exploration (ACE) Dataset, the first open-source dataset in this domain, which contains data from 50 users exploring 5 different AR scenes. We use the same dataset for our experiments.

C. 360-Degree Video FOV Prediction

360-degree video FOV prediction is an area of research closely related to FOV prediction for AR. Although, as highlighted before, there are significant differences between AR and 360-degree video, both technologies require streaming systems that can benefit from FOV prediction.

Motivated by the need for accurate prediction of the FOV in free viewpoint videos, Mavanklar et al. [6] present an online prediction method based on a Dead-Reckoning (DR) approach. The DR method makes FOV predictions purely based on the user's historical trajectory. The method assumes the user is going to maintain the current angular velocity, which is either read from the input device or computed from successive measurements of the viewing device. Petrangeli et al. [5] propose an algorithm to solve the problem of long-term FOV prediction of 360-degree videos, with prediction horizons varying from 1s to 10s. This is in contrast to most existing approaches, which usually focus on short-term (200 – 500ms) prediction horizons. The authors use a spectral clustering-based approach on a dataset of 16 public videos viewed by 61 users to cluster similar rotation trajectories, and use the generated clusters for prediction. This algorithm outperforms simpler linear regression and augmented linear regression approaches, due to the non-linear nature of the users' movements. In Fan et al. [9], the authors develop a fixation prediction network that uses sensor features (viewer orientation) and content features (image saliency and motion maps) to predict the viewer FOV for 360-degree videos. Image saliency maps are generated offline using pre-trained CNNs for image classification. Similarly, motion maps are generated offline by analyzing Lucas-Kanade optical flow over video frames. An online LSTM combines the offline features with view orientation data to predict the future FOV. This method, applied in a real 360-degree video delivery scenario, is able to produce comparable video quality to existing methods while providing a shorter initial buffering time and a reduction in bandwidth consumption up to 36%. Bao et al. [4] collect motion data from 153 subjects watching 360-degree videos and find strong short-term correlation across the movements of various users. Motivated by this finding, the authors predict the movements of the viewers through traditional linear regression and a neural network-based model that predicts the viewer's FOV, together with a prediction confidence score. The work also proposes several evaluation metrics, such as failure ratio and ratio of missing pixels, which can be used to compare different algorithms in this domain. Li et al. [10] propose two FOV prediction models: trajectory-based and heatmap-based models, by both considering the individual FOV of the user and the historical FOV patterns of other users as well. Several approaches are considered, as LSTM, MLP mixing and mixture of experts. Experimental results confirm that the models utilizing other users' information bring substantial performance gains over those utilizing the current user's data only. A Recurrent Neural Network (RNN) model is proposed by Liu et al. [11]. The FOV prediction model is used in combination with cloud rendering to offload the most computational expensive tasks from the client. The RNN model shows good prediction accuracy for short horizons. A similar approach is used by Hou et al. [12]. The authors use an LSTM model to predict the viewing probability of tiled 360-degree videos, by employing a large, proprietary dataset for training. Estimation of the saliency of 360-degree media can also be used to drive FOV prediction. Chao et al. introduce a novel saliency estimation model for omnidirectional images [13], which takes into account the distortion introduced by the equirectangular projection when performing this estimate. Romero Rondon et al. use an RNN architecture that fuses both absolute position and video saliency information to perform the prediction [14]. The so-called TRACK architecture merges embedding from position and saliency information using an LSTM architecture, which shows good prediction power up to 5 seconds. A clustering-based method is used by Nasrabadi et al. [15], which operates in the quaternion space for better accuracy. Similarly to this work, we also operate in the quaternion space when predicting the rotation component of the user's FOV. Depending on the video, this method shows good performance accuracy up to 5 seconds. The Sparkle algorithm is developed by Chen et al. with interpretability in mind [16]. A different model is trained for each user, which aims to capture how a particular user explores the VR video. Ban et al. [17] propose a KNN-based Viewport Prediction (KVP) approach that considers both user-specific and cross-user behavior to predict the future viewport. Given the user's recent fixation in Euler angles, KVP uses linear regression as its initial fixation prediction, which is later amended by a KNN-based method that finds the K nearest fixations using a sphere distance. A probability vote mechanism examines the K nearest fixations to determine the viewing probability of each video tile. In conjunction with a tile-based adaptive streaming system, the method achieves an average relative improvement in viewport prediction accuracy of 48.1% over a linear regression baseline. Park et al. exploit video semantic information and historical viewing patterns for prediction [18]. The authors propose SEAWARE, a semantic aware view prediction system, which solves this problem by performing video semantic analysis and recording it in a modified video manifest hosted on the server. SEAWARE performs better than existing 360-degree video streaming solutions in terms of accuracy and efficiency.

III. Trace Match & Merge Algorithm

To solve the problem of long-term FOV prediction in AR applications, we present in this section the *Trace Match and Merge (TMM)* algorithm, a nearest-neighbor-based approach for predicting the 6DOF of an AR device. At a high level, TMM makes a prediction of the user's future position and rotation by examining the recent position history of the current user – we refer to this position history as a *segment*. TMM then locates similar segments among the traces of past users who have explored the same scene, and merges these selected traces to create a new trace that is used to predict the behavior of the current user.

A. Input and Output Data

As input, TMM expects a list of user traces $x_1, x_2, ..., x_N$. Each user trace x_i is a pair of sequences of length l_i: a sequence of vector positions $p_i^{(1)}, p_i^{(2)}, .., p_i^{(l_i)}$ and a sequence of quaternion rotations $q_i^{(1)}, q_i^{(2)}, .., q_i^{(l_i))}$. These values correspond to the six degrees of freedom (position and rotation) of the AR device as it explores the AR scene over time.

Given a user trace:

$$x = ([p^{(1)}, p^{(2)}, .., p^{(l_i)}], [q^{(1)}, q^{(2)}, .., q^{(l_i)}])$$

The goal of the TMM algorithm is to predict the future H time-steps of the trace:

$$y = ([p^{(l_i+1)}, p^{(l_i+2)}, .., p^{(l_i+H)}], [q^{(l_i+1)}, q^{(l_i+2)}, .., q^{(l_i+H)}])$$

representing the future user position in the AR scene. It is worth noting that TMM builds a scene-specific model, since user behavior is expected to vary across AR scenes, as each scene can potentially have a different layout.

B. Trace Segments Creation

TMM exploits similarities in the exploration patters of past users of a particular AR scene to predict the position of the current user in future time steps. Initially, this similarity is computed on the position component only of the user trace. First, TMM divides each past user trace into so-called *segments* using three parameters: size z, stride s, and dilation d. The size z defines the number of consecutive time-steps in each segment (e.g., 2-seconds segements). The dilation d determines the number of time-steps skipped between each time-step in the segment. The stride s controls the number of time-steps in the trace between the start of each segment. As a result, the total time encompassed by a segment is equal to $d \cdot z$, and the number of segments generated by a single trace is approximately the length of the trace divided by s. This process produces n segments $S_1, S_2, ...S_n$. Each segment S_i contains $(z \times 3)$ values, where each row of S_i is a position vector.

Brute-force nearest neighbor search over all segments would require $O(zn)$ time. To accelerate the search, TMM stores the segments in a ball tree data structure, which enables nearest-neighbor search in $O(z \log n)$ time. However, ball trees (and other exact nearest neighbor data structures) require the segment distance metric to satisfy the triangle inequality. For distance metric d and arbitrary segments S_i, S_j, S_k, the triangle inequality is satisfied iff:

$$d(S_i, S_k) \leq d(S_i, S_j) + d(S_j, S_k)$$

To satisfy this requirement, TMM uses the squared Frobenius norm, which is equivalent to the sum of the pairwise squared Euclidean distance between each position vector of the segments:

$$d(S_i, S_j) = ||S_i - S_j||_F^2$$

The segment creation step of the TMM algorithm is visualized in step 1 of Figure 2.

At prediction time, TMM builds a segment of size z using dilation d from the user's recent position history and utilizes the ball tree to fetch the k nearest position segments.

C. Segment Relevance and Matching

Although the k segments provided by the nearest neighbor search exhibit similar patterns in terms of positional components compared to the position of the current user trace, their actual FOVs may still be dissimilar due to different orientations. To narrow down the segments and select only those exhibiting a similar FOV to that of the user, TMM examines the current FOV of the user and compares it to the FOV at the last time-step of the k segments (steps 2 and 3 in Figure 2). TMM explicitly compares FOVs using the volume of intersection between their pyramids of vision. Since the pyramid of vision has infinite depth, it is truncated at an arbitrary depth of 100 meters to enable volume computation. TMM computes the volume of intersection between two pyramids of vision using a Monte Carlo approximation. P_{tot} points are sampled uniformly within the pyramid of vision of the user; of these P_{tot} points, P_{int} points lie inside both pyramids of vision. The volume of intersection is given by:

$$V_{int} = \frac{P_{int}}{P_{tot}} V_{pyr}$$

where V_{pyr} refers to the volume of the user's truncated pyramid of vision, and V_{int} denotes the volume of the intersection. TMM ranks the k segments by their FOV intersection volume, and retains only the top-m segments with the greatest intersection. This process is highlighted in steps 3 and 4 of Figure 2.

It is worth noting that the calculation of the FOV intersection volume is computationally expensive and does not satisfy the triangle inequality. Therefore, it cannot be used as the distance metric in the nearest neighbor search; for each trace prediction, the volume is computed only k times.

D. Merging Segments

To produce a final prediction, TMM fetches the future position and rotation of each of the top-m segments selected at the previous step and merges them. As for traditional k-nearest neighbor approaches, this merging step helps reducing the variance of the prediction carried out by TMM.

The position and rotation at each time-step are determined independently using an average across the futures of the top-m segments. Since rotation is represented by a quaternion, a simple arithmetic mean would not be able to capture the intuitive average rotation, partially because a quaternion q and its negation $-q$ represent the same rotation. As a result, TMM uses Markley's quaternion average to merge the rotation values [19], which produces an optimal quaternion average using the eigenvalue/eigenvector decomposition of a matrix created from the quaternions. This average minimizes the

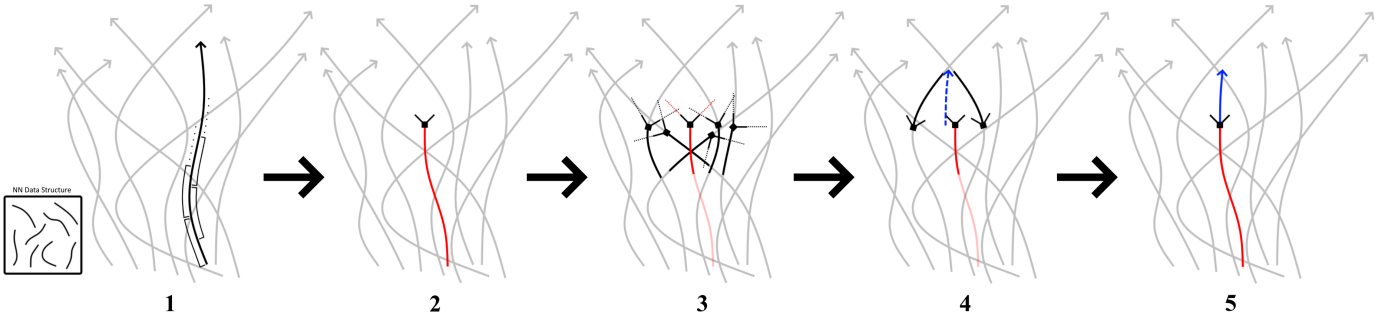

Fig. 2. *Trace Match & Merge* (TMM) uses a ball tree of trace segments (1) and the recent user position history (2) to locate similar position segments of previous users (3). The futures of the segments with the most similar field-of-view are merged (4) and translated (5) for continuity with the user's trace.

sum of the squared Frobenius norms of the attitude matrix differences:

$$\underset{q \in \mathbb{S}^3}{\operatorname{argmin}} \sum_{i=1}^{m} ||A(q) - A(q_i)||_F^2$$

where $A(q_i)$ denotes the attitude matrix of quaternion q_i and $\mathbb{S}^3$ denotes the unit 3-sphere. An example of the merging process is pictured in steps 4 and 5 of Figure 2.

The initial position of the merged prediction is equal to the average of the initial positions of the futures of the top-m segments, which is not necessarily close to the current position of the user. To improve the performance of TMM for short prediction horizons, the position values of the merged prediction are translated to enable continuity between the current user trace and its predicted trace. The difference in the prediction between steps 4 and 5 of Figure 2 provides an example of this translation. An analogous translation is made for rotation continuity between the current user trace and its predicted trace.

Formally, let $p^{(l)}$ and $q^{(l)}$ correspond to the current vector position and unit quaternion rotation of the user, respectively, and let $p_m^{(l+1)}, ..., p_m^{(l+h)}$ and $q_m^{(l+1)}, ..., q_m^{(l+h)}$ correspond to that of each time-step of the merged prediction. Let $o_p = p^{(l)} - p_m^{(l+1)}$ and $o_q = q_m^{(l)} q_m^{(l+1)*}$ be the vector position offset and unit quaternion rotation offset. The vector positions $p_f^{(l+1)}, ..., p_f^{(l+h)}$ and quaternion rotations $q_f^{(l+1)}, ..., q_f^{(l+h)}$ of the final prediction are given by:

$$p_f^{(l+i)} = p_m^{(l+i)} + o_p$$
$$q_f^{(l+i)} = o_q q_m^{(l+i)}$$

IV. Experimental Results

In this section, we present an analysis of the TMM prediction algorithm. We first briefly introduce the ACE Dataset [3] used for our experiments (Section IV-A), the baseline prediction algorithms employed for comparison (Section IV-B) together with the evaluation metrics (Section IV-C), and the main analysis of the results (Section IV-D).

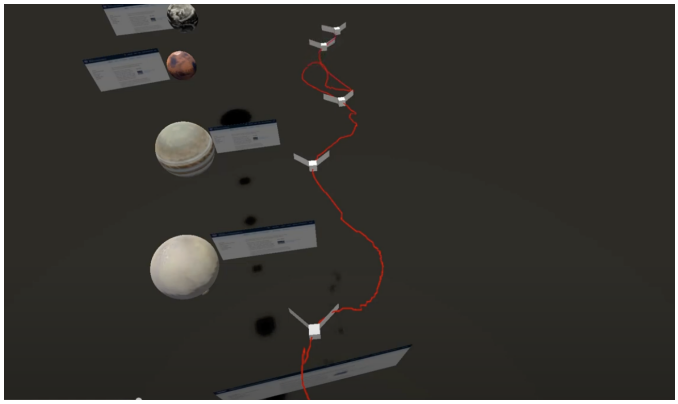

Fig. 3. The *No Prediction* (NP) approach simply assumes that the position at time t will be the same at time $t + H$, with H being the prediction horizon. The red line indicates the exploration pattern of a particular user in the *Solar System* scene, the gray box indicates the actual FOV.

A. Dataset

We utilize the ACE dataset [3], which contains the position and rotation of 50 users each exploring 5 different AR scenes. Figure 1 illustrates the 5 different AR scenes. Each scene has been created with different characteristics in mind, in order to validate whether different scene topologies would result in different exploration patterns from the users. The *Solar System* and *Apples* scenes present objects positioned in a very regular manner (a straight line and a circle, respectively). As it can be noted from the exploration patterns in Figure 1 (red lines), these scenes result in very consistent and similar explorations patterns across different users. On the other hand, the *Toy Room* and *Fiction* scenes present objects disposed with a less obvious exploration pattern in mind. Intuitively, the exploration patterns for these scenes will be harder to predict than then *Solar System* or *Apples* scenes. In this paper, we do not consider the *Food* scene, which contains user interaction events that are not recorded in the ACE dataset.

B. Baselines

We consider several baselines to compare with our proposed FOV algorithm, following the same approach used by Wang et al. [3]. The simplest baseline is *No Prediction* (NP), which

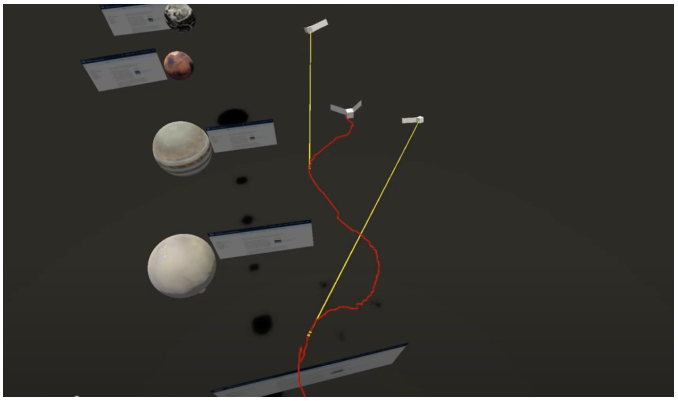

Fig. 4. The *Linear Regression* (LR) approach fit a linear regression model on the recent history of position and rotational values of the user trace. The yellow line indicates the prediction computed by this method over time.

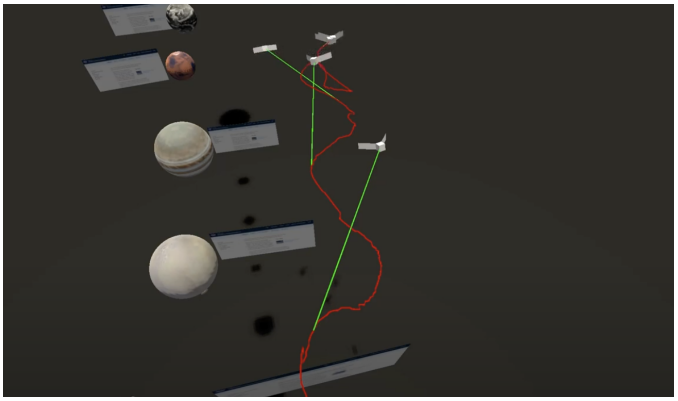

Fig. 5. In the *Dead Reckoning* (DR) baseline, we assume the user will maintain the current translational and rotational velocity between t and $t+H$, with H being the prediction horizon. The green line indicates the prediction obtained by using this method.

assumes that the user's field of view will not change between time t and time $t+H$, with H being the prediction horizon. The qualitative behavior of the NP approach can be seen in Figure 3, where the red line indicates the path taken by the user in the scene and the gray camera indicates the FOV of the user. The second baseline is a *Linear Regression* (LR) approach that applies linear regression on the recent trace history to predict the future position and rotation. Qualitatively, the resulting prediction is visualized in Figure 4. Intuitively, this method will perform the best for short prediction horizons and/or when the user movements do not present abrupt changes over time. We also consider a *Dead Reckoning* (DR) algorithm, which assumes the user will maintain the current translational and angular velocity. This velocity is estimated using a moving average over the last several time-steps. DR is a popular baseline for FOV prediction methods for 360-degree videos, and the prediction results are qualitatively similar to those of linear regression, as it can be seen in Figure 5.

While the previous approaches have been extensively used in the 360-degree FOV prediction literature, we also consider a more complex baseline based on a *Random Forest Regressor* (RFR), which uses a set of decision trees termed as estimators. In order to properly train the model, we follow the following strategy. First, we consider a fixed history of time-steps (referring to both position and rotation of the user trace) to be given as input to the regressor. To speed-up training, rather than considering all the contiguous time-steps in the history, we skip every 10^{th} time-step. Since data points in the ACE dataset are collected every 60^{th} of 1 second, this sampling operation helps training the regressor faster without losing important information on the user movements. To simplify training and obtain faster inference timing, given the history used as input at time t, the regressor only predicts position and rotation at time $t+H$, rather than the whole sequence of time-steps.

All experiments use optimal hyperparameters tuned for each scene, which were selected for each model using cross validation and Bayesian optimization.

C. Evaluation Metrics

We quantitatively evaluate each prediction model using two metrics: (1) position and rotation error and (2) *objects-in-view* error, which we detail next. In the result section, we evaluated both metrics at multiple prediction horizons, i.e., 0.5s, 1s, 5s, and 10s.

The position and rotation error metrics measure the difference in position/rotation between the ground truth trace (i.e., the actual position and rotation of the user) and the predicted traces (i.e., the predicted position and rotation of the user as computed by either TMM or one of the baseline methods). In the result section, we report the average errors computed over the interval $t; t+H$. The position error is computed using the Euclidean distance, and it is expressed in meters. The rotation error is computed as the angle between the ground truth and the predicted quaternions:

$$d_r(q_{gt}, q_{pr}) = \cos^{-1}(2(q_{gt}^T\, q_{pr})^2 - 1)$$

and it is expressed in radians.

While the absolute error gives a clear indication of the absolute performance of the prediction algorithm, we also use the objects-in-view error to get a more direct understanding of the impact of the algorithm on the FOV prediction task. The objects-in-view error compares the digital objects in the FOV as computed using the ground truth trace with respect to the objects in the FOV as computed using the predicted trace (obtained using a prediction algorithm). Based on these two sets of objects, we compute precision, recall and F1 score. In this context, a high precision entails that most of the predicted objects are indeed in the FOV, and is a measure of bandwidth efficiency (i.e., how much bandwidth would be wasted to potentially retrieve objects that are not in the FOV of the user). Recall is directly connected to the user experience: if the prediction cannot accurately determine all the objects in view, the user may have a poor AR viewing experience, since some objects would not be streamed. The F1 score ties both precision and recall together to provide a single value with

which to compare the performance of the different algorithms on the FOV prediction task.

D. Results and Analysis

To clearly showcase the benefits of the proposed TMM method, we first consider the performance on the traces from the *Fiction* scene which, as reported by Wang et al. [3], are the hardest ones to be predicted, given the position of the objects do not show a clear, intuitive exploration pattern. The results of the experiments for this scene are presented in Figure 6. The x-axis reports the prediction horizon, while the y-axis reports the position/rotation errors and the F1 score for the different methods.

We observe that all approaches, except for the Random Forest Regressor, result in very low position and rotation errors (left and center graphs in Figure 6) for shorter horizons smaller than 2 seconds. As expected, the corresponding F1 scores for these models are close to 1 (right graph in Figure 6). The simpler baselines (linear regression and dead reckoning) perform the best at this short prediction horizons, closely followed by our TMM approach. Nevertheless, as soon as the prediction horizon increases, these methods present a significant drop in performance, and provide significantly inferior performance compared to our TMM and the Random Forest Regressor, across all evaluation metrics. The NP, LR and DR baselines only use the recent position and rotation history of the particular user under consideration to make predictions, and therefore fail predicting the long-term behavior of the user. The results from TMM and RFR confirm that exploiting cross-user behavior is beneficial for predicting FOV at longer prediction horizons. Both approaches result in similar position error (for longer horizons), with TMM outperforming RFR in terms of rotation error. In turn, this translates into much higher F1 scores for TMM. As explained in Section IV-C, the F1 score provides a clear indication of the performance of a prediction algorithm: higher values indicate that TMM is able to retrieve most of and only the objects that are potentially viewed by the user.

In Figure 7, we present the F1 scores for all models across the *Apples* (left), *Toy Room* (center) and *Solar System* (right) scenes. The results for the *Apples* and *Toy Room* scenes follow similar trends as those for the *Fiction* scene: NP, LR and DR methods can provide good performance for short prediction horizons, but the models' performance rapidly deteriorates for prediction horizons larger than 2 seconds. On the other hand, both TMM and RFR result in high F1 score values for larger horizons, with TMM providing the best results overall. This indicates that TMM is able to capture the global exploration patterns of a particular AR scene, and re-use this knowledge to successfully predict the long-term behavior of a new user. As expected, the NP method provides the worst results for longer horizons. In the *Solar System* scene (right graph in Figure 7), we observe that most of the models are able to maintain a high F1 score, even for longer prediction horizons. This is mostly due to the simple linear layout of the objects in this scene (see Figure 1), which causes most users to move in an approximately straight line. In this situation, even simpler prediction algorithms are able to provide satisfactory results. Nevertheless, for longer horizons, TMM is always the best or near-best model in terms of F1 score.

These results highlight how the proposed TMM approach is effective in predicting the future user behavior and is able to provide the most consistent, high-performing results both across different scenes and more challenging, longer-term prediction horizons.

V. Conclusions

We presented in this paper the *Trace Match & Merge* algorithm, a novel approach for the FOV prediction in AR applications. Our approach employs a modified nearest neighbor approach that takes into consideration the unique characteristics of AR applications. First, TMM performs a search to find the past user traces closely resembling the trace of the user under consideration. Next, TMM merges those traces into one, which is used to predict the future FOV of the user. Extensive experimental results show that our approach provides better performance in terms of predicted position, rotation and objects-in-view compared to several baseline algorithms, especially for longer, more challenging prediction horizons. To conclude, we present several areas of improvements to be pursued as future work in this domain.

6DOF Position and Object Access Sequence. The TMM approach solely employs the 6DOF position of the user to perform a prediction. Even higher performance could be achieved, especially for longer prediction horizons, by considering the sequence of objects viewed by past users, in addition to their 6DOF positions.

Scene topography. User exploration patterns are highly impacted by the topology and the disposition of the AR objects. Approaches that take into account the topography of a scene, e.g., by employing path planning techniques with the objects as goals, can be used in this context. These techniques can also be useful when the amount of training data is not abundant, for example when a new scene is created.

Model Generalization. While TMM is used on a per-scene basis, it is worth investigating whether it is possible to use data across scenes to create a general model that can provide reasonable performance irrespective of the scene. This model can then be fine-tuned for each specific AR scene.

DNN-based Approaches. Albeit not presented in the main experimental section of this paper, we also experimented with an LSTM-based sequence-to-sequence model that uses the recent 6DOF history of a user to predict the future position. Preliminary results obtained using this approach are comparable to those obtained using linear regression- and dead-reckoning-based approaches. This is likely due to the limited amount of training data, which is not large enough for this specific task. As previously mentioned, a possible solution to this problem would be to aggregate the information from multiple scenes, and generate a cross-scene model that can work in combination with other scene-specific models.

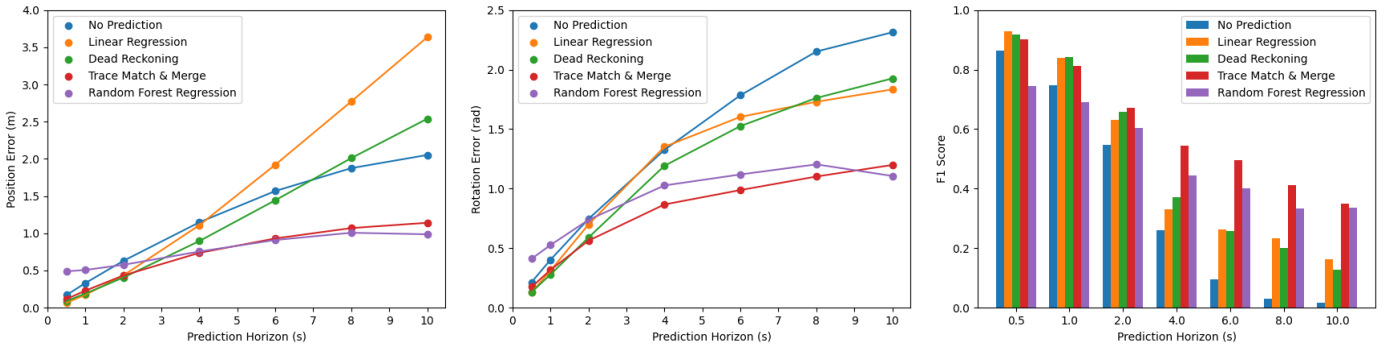

Fig. 6. Position error (left), Rotation error (center) and F1 score (right) for the *Fiction* scene of the ACE Dataset. Our TMM approach results in low position and rotation errors, especially for longer prediction horizons (>2 seconds), producing the highest F1 score across all methods.

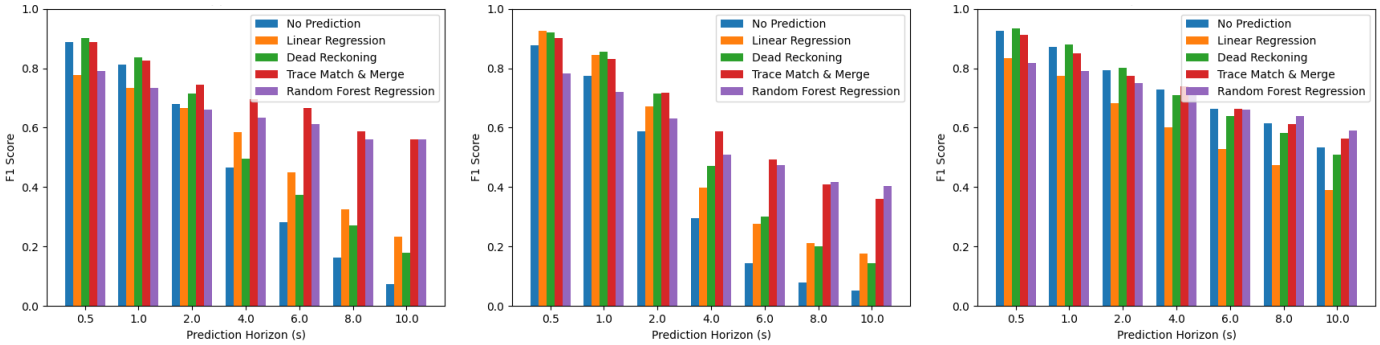

Fig. 7. F1 Scores for the *Apples* (left), *Toy Room* (center) and *Solar System* (right) scenes of the ACE dataset. Our TMM method provides the most consistent performance across all scenes and prediction horizons.

References

[1] S. Petrangeli, G. Simon, H. Wang, and V. Swaminathan, "Dynamic adaptive streaming for augmented reality applications," in *2019 IEEE International Symposium on Multimedia (ISM)*, 2019, pp. 56–567.

[2] J. van der Hooft, T. Wauters, F. De Turck, C. Timmerer, and H. Hellwagner, "Towards 6dof http adaptive streaming through point cloud compression," in *Proceedings of the 27th ACM International Conference on Multimedia*, ser. MM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2405–2413. [Online]. Available: https://doi.org/10.1145/3343031.3350917

[3] N. Wang, H. Wang, S. Petrangeli, V. Swaminathan, F. Li, and S. Chen, "Towards field-of-view prediction for augmented reality applications on mobile devices," ser. MMVE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 13–18. [Online]. Available: https://doi.org/10.1145/3386293.3397114

[4] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, and X. Liu, "Shooting a moving target: Motion-prediction-based transmission for 360-degree videos," in *2016 IEEE International Conference on Big Data (Big Data)*, 2016, pp. 1161–1170.

[5] S. Petrangeli, G. Simon, and V. Swaminathan, "Trajectory-based viewport prediction for 360-degree virtual reality videos," in *2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, 2018, pp. 157–160.

[6] A. Mavlankar and B. Girod, *Video Streaming with Interactive Pan/Tilt/Zoom*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 431–455. [Online]. Available: https://doi.org/10.1007/978-3-642-12802-8_19

[7] H. Noh and H. Song, "Cloud-assisted augmented reality streaming service system: Architecture design and implementation," in *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, 2020, pp. 363–366.

[8] J. Park, P. A. Chou, and J.-N. Hwang, "Volumetric media streaming for augmented reality," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[9] C.-L. Fan, J. Lee, W.-C. Lo, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Fixation prediction for 360 video streaming in head-mounted virtual reality," in *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2017, pp. 67–72.

[10] C. Li, W. Zhang, Y. Liu, and Y. Wang, "Very long term field of view prediction for 360-degree video streaming," in *Proceedings - 2nd International Conference on Multimedia Information Processing and Retrieval, MIPR 2019*. Institute of Electrical and Electronics Engineers Inc., Apr., pp. 297–302.

[11] X. Liu and Y. Deng, "Learning-based prediction, rendering and association optimization for mec-enabled wireless virtual reality (vr) network," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2021.

[12] X. Hou, S. Dey, J. Zhang, and M. Budagavi, "Predictive adaptive streaming to enable mobile 360-degree and vr experiences," *IEEE Transactions on Multimedia*, vol. 23, pp. 716–731, 2021.

[13] F.-Y. Chao, L. Zhang, W. Hamidouche, and O. Déforges, "A multi-fov viewport-based visual saliency model using adaptive weighting losses for 360-degree images," *IEEE Transactions on Multimedia*, vol. 23, pp. 1811–1826, 2021.

[14] M. F. R. Rondón, L. Sassatelli, R. A. Pardo, and F. Precioso, "Track: a multi-modal deep architecture for head motion prediction in 360-degree videos," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 2586–2590.

[15] A. T. Nasrabadi, A. Samiei, and R. Prakash, *Viewport Prediction for 360° Videos: A Clustering Approach*. New York, NY, USA: Association for Computing Machinery, 2020, p. 34–39. [Online]. Available: https://doi.org/10.1145/3386290.3396934

[16] J. Chen, X. Luo, M. Hu, D. Wu, and Y. Zhou, "Sparkle: User-aware viewport prediction in 360-degree video streaming," *IEEE Transactions on Multimedia*, pp. 1–1, 2020.

[17] Y. Ban, L. Xie, Z. Xu, X. Zhang, Z. Guo, and Y. Wang, "Cub360: Exploiting cross-users behaviors for viewport prediction in 360 video adaptive streaming," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, 2018, pp. 1–6.

[18] J. Park, M. Wu, K.-Y. Lee, B. Chen, K. Nahrstedt, M. Zink, and

R. Sitaraman, "Seaware: Semantic aware view prediction system for 360-degree video streaming," in *2020 IEEE International Symposium on Multimedia (ISM)*, 2020, pp. 57–64.

[19] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, "Averaging quaternions," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1193–1197, 2007.