3D Memory for Augmented Reality

DISSERTATION

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

DOCTOR OF PHILOSOPHY (Computer Science)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Shaoyu Chen

May 2024

3D Memory for Augmented Reality

DISSERTATION

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

DOCTOR OF PHILOSOPHY (Computer Science)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Shaoyu Chen

May 2024

Approved:

_____

Department Chair Signature

_____

Date

University ID: N18526811

Net ID: sc6439

Approved by the Guidance Committee:

Major: Computer Science

Cláudio T. Silva
Institute Professor of Computer Science and Engineering and Data Science
New York University

Date

Fabio Miranda
Assistant Professor of Computer Science
University of Illinois Chicago

Date

Ken Perlin
Professor of Computer Science
New York University

Date

Qi Sun
Assistant Professor of Computer Science and Engineering
New York University

Date

Vita

Shaoyu Chen was born in Guangzhou, China in July, 1995. He earned his Bachelor of Engineering in Computer Science from the Hong Kong University of Science and Technology in 2017. While completing his undergraduate studies, he served as a student researcher at HKUST VisLab directed by prof. Huamin Qu, where he developed a variety of visualization tools. He started his Ph.D. in September 2017, working in various areas, including virtual reality, augmented reality, data visualization and computer vision. During his Ph.D., he was an intern at Adobe Research and Meta.

Acknowledgements

I would like to thank my advisor, Cláudio T. Silva, for the unwavering support, invaluable guidance, and fruitful exchange of ideas throughout my Ph.D. studies. Thank you for granting me the opportunity to work independently on challenging projects and consistently pushing my abilities to a higher level. I would also like to thank members of my Ph.D. committee, Fabio Miranda, Huy Vo, Ken Perlin, and Qi Sun for their expertise, insightful ideas, and constructive feedback.

I would like to thank my mother and father for their unconditional love and support.

I would like to thank all the members of the VIDA Lab for providing an excellent research environment and assistance with user studies. I feel incredibly fortunate to have been part of the VIDA Lab, where discussions with colleagues were consistently inspiring, and the atmosphere was fun.

I would like to especially thank Gromit Chan and Guande Wu for their insightful conversations. I would like to thank Bea Steers, Bowen Yu, Budmonde Duinkharjav, Chen Zhao, Connor Defanti, Corinne Brenner, Harish Doraiswamy, Iran Roman, Jianzhe Lin, Jing Qian, Joao Rulff, Jose Echevarria, Jun Yuan, Li-Yi Wei, Marcos Lage, Mengwei Ren, Nivan Ferreira, Shijie Li, Sonia Castelo, Stefano Petrangeli, Xin Sun and Yurii Piadyk. It was my pleasure to work with each of you.

I would like to thank funding agencies that supported the work presented in this thesis: the National Science Foundation (NSF awards CNS-1229185, CCF-1533564, CNS-1544753, CNS-1730396, CNS-1828576, CNS-1626098), DARPA PTG program, CNPq, and FAPERJ. Any opinions, findings, and conclusions or recommendations expressed in this thesis are those of the author and do not necessarily reflect the views of the funding agencies.

Shaoyu Chen
May 2024

To my family and friends.

ABSTRACT

3D Memory for Augmented Reality

by

Shaoyu Chen

Advisor: Prof. Cláudio T. Silva, Ph.D.

Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy (Computer Science)

May 2024

Augmented Reality (AR) headsets, with their capability to overlay digital information onto the physical world and equipped with diverse sensors for perceiving physical surroundings, emerge as powerful tools. This functionality offers a promising avenue for the development of AI assistants capable of guiding users in daily tasks, such as cooking. Understanding the physical environments where humans perform tasks establishes a universal foundation applicable to diverse scenarios. In AR, automated perception and memorization of physical surroundings empower intelligent assistants, facilitating tasks like querying the locations of objects, as known as episodic memory.

Recent advancements in computer vision have enabled object detectors to identify various classes, with open vocabulary object detectors now trained using large-scale image datasets and text embeddings. As a result, they can detect novel classes not present in the training set, allowing the creation of AR assistants capable

of generalizing across tasks without requiring re-training. However, these open vocabulary object detectors are constrained in providing information solely about the current RGB frame. To provide better task guidance, a 3D memory algorithm that mimics human episodic memory is needed to provide comprehensive data about observed objects, including object ID, 3D position, object class, and the time of last observation. Existing techniques, neglecting the unique challenges of limited camera field-of-view (FoV), frequent view changes and task object manipulation by AR users, encounter various issues when employed as 3D memory in AR scenarios. In this thesis, we present solution to empowers intelligent AR assistants with 3D memory.

First, we proposed a 3D memory framework grounded in tracking-by-detection paradigm for complex and dynamic AR scenarios. To take advantages of the power of both 2D open vocabulary object detectors and 3D depth information from depth sensors equipped by AR headsets, we presented a 2D-3D hybrid framework mimicking 3D Episodic Memory for AR. Additionally, we presented a tracking-by-detection method designed for complex and dynamic AR scenarios, considering the unique challenges inherent in such environments. In the absence of a suitable existing dataset, we created an annotated dataset for evaluating the performance of 3D memory. Experiments using this dataset demonstrated significant advantages of our proposed method over existing solutions.

Second, we created visual analytics tools to enhance comprehension of 3D memory outputs and facilitate the debugging of AI assistants' machine learning models. We built visual analytics tools on top of ARGUS, which is a state-of-the-art visual analytics system to support the development of intelligent AR assistants. We presented case studies to demonstrate the effectiveness of these visual analytics tools. One interesting direction is to visualize the 3D information within AR/VR headsets. However, given the large volume of the recorded data, it is impractical to store the data on the headset and streaming the data from server to the headset is required. We proposed a method to real-time stream the large volume of 3D data while providing a better user experience in AR/VR.

We envision the proposed open-source framework to form the foundation of AR task assistants with robust 3D episodic memory.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Augmented Reality has attracted growing attention for providing a novel mode user interaction. The rise in AR technology has led to the development of advanced AR headsets, exemplified by Microsoft HoloLens 2 and Apple Vision Pro. These headsets possess significant capabilities, not only overlaying digital information onto the physical world but also being equipped with diverse sensors for perceiving the physical surroundings. This functionality opens avenues for the creation of AI assistants capable of guiding users in daily tasks, such as cooking. Understanding the physical environments where humans perform tasks establishes a universal foundation applicable to diverse scenarios. In AR, automated perception and memorization of physical surroundings empower intelligent assistants, facilitating tasks like querying the locations of objects, as known as episodic memory. Episodic memory can help the AI assistants provide better task guidance like directing users to find objects outside field-of-view (FoV), or identifying the active tasks based on active objects in memory.

Recent advancements in computer vision have facilitated the identification of various object classes by object detectors. Open vocabulary object detectors, trained with large-scale image datasets and text embeddings, allow the detection of any class, even if the class is novel and not present in the training set. This not only enables the AI assistant to understand the physical world, but also facilitates the construction of AR assistants capable of generalizing across tasks without the

need for re-training. However, these open vocabulary object detectors are limited to providing information solely about the current RGB frame, lacking the ability to utilize temporal information from the video stream or depth information from the depth sensor. For example, if a static object becomes out-of-view due to headset movement, it won't be in the object detection result. Therefore, to provide better task guidance, a 3D memory algorithm mimicking human episodic memory is required to provide comprehensive data about observed objects, including object ID, 3D position, object class, and the time of last observation. This can enable the AI assistant not only to guide users to find objects needed for current task in 3D space, but also to provide information for other modules of the AI assistant, allowing inference of the current step of a task or correction of user errors. However, existing methods, neglecting the unique challenges of limited camera field-of-view, frequent view changes and task object manipulation by AR users, encounter various issues when employed as 3D memory in AR task guidance scenarios.

Other than 3D memory, AI assistants for task guidance may consist of other machine learning modules, including but not limited to object detection, action recognition, and reasoning for task steps and errors. Debugging and improving the AI assistant in such multi-module scenarios pose significant challenges for developers. While visual analytic tools can aid in comprehending and debugging machine learning modules, existing methods fall short in supporting the visual analytics of 3D memory.

Challenges

1. **Episodic Memory for AR.** Existing methods encounter three main roadblocks in AR scenarios: (1) fast camera movement due to user head motion, (2) objects disappearing and reappearing due to limited FoV, (3) diverse object classes. For example, 3D object-level SLAM estimates 3D positions, but has assumptions unsuitable for egocentric AR scenarios, such as slow/smooth camera motion and high frame rate of depth frames. Additionally, existing 3D multi-object tracking is limited to objects only within the "current" FoV, and the missing global memory and context prohibit queries about objects outside the current view, which is often prevalent and crucial. Unlike the common settings, scenes in egocentric settings typically feature diverse objects

in varying scales, which often induce classification errors, particularly for methods trained on non-egocentric dataset [1]. Furthermore, diverse objects can cause end-to-end methods with a limited vocabulary to fail due to novel classes.

2. **Datasets.** To fine-tune and evaluate the 3D memory algorithm, an ideal evaluation dataset shall contain (1) egocentric RGBD video, (2) dynamic scene in real-world task scenarios, (3) multi-object tracking annotation. To our best knowledge, existing datasets fail to simultaneously meet all three requirements, necessitating the development of a method requiring minimal data for fine-tuning and generalizing well across different tasks. Creating and annotating a suitable dataset is imperative for the evaluation of the proposed method.

3. **Visual Analytic for 3D Memory.** Visualization tools for 3D memory are crucial since they can help developers debug and improve the algorithm, especially for the case of AI assistants where there are multiple machine learning modules working together. Despite existing works focusing on object detection, creating visual analytic tools for 3D memory remains an open problem. Additionally, visualizing 3D information in AR/VR is an interesting direction. However, a 6 min HoloLens 2 recording can result in 3GB data [9]. The significant data volume poses challenges for interactive visualization due to network bandwidth limitations.

1.2 Contribution

In this dissertation, we describe two contributions that address the challenges described above. The main contributions can be summarized as follows.

1. **3D Memory Algorithms for AR (Chapter 3).** To address challenges 1 and 2, we proposed a 3D memory framework based on tracking-by-detection paradigm and developed a real-time spatial perception system for complex and dynamic AR scenarios. To take advantages of the power of both 2D open vocabulary object detectors and 3D depth information from depth sensors

equipped by AR headsets, we introduced a 2D-3D hybrid framework to mimic 3D Episodic Memory for AR. In addition, we presented a tracking-by-detection method designed for complex and dynamic AR scenarios considering unique challenges like limited FoV. Due to the absence of an existing suitable dataset, we created an annotated dataset for evaluating the performance of 3D dynamic episodic memories. Experiments using this dataset demonstrate significant advantages, including improved tracking accuracy and robustness to low frame rates, of the proposed method over existing solutions.

2. **Visualization for 3D Memory (Chapter 4).** To address challenge 3, we created analytic tools on top of ARGUS, a state-of-the-art visual analytics system supporting the development of intelligent AR assistants, and demonstrated their usefulness through case studies. Additionally, to tackle the real-time streaming of large volumes of 3D information, we propose a method for real-time streaming while providing a superior user experience in AR/VR. The effectiveness of the proposed method is validated through subjective studies and objective analysis under varying network conditions.

1.3 Organization

The remaining of this thesis is organized as follows. First, Chapter 2 discusses the related work on 3D memory, including existing tracking methods and datasets. Next, Chapter 3 introduces our proposed 3D memory for AR. Chapter 4 presents the visualization tools developed for the 3D memory, including ARGUS extensions for 3D memory and InstantReality for real-time 3D data streaming. Finally, Chapter 5 concludes the dissertation, highlighting potential future works.

Chapter 2

Related Work

3D memory is similar to multi-object tracking (MOT), since both of them focus on tracking the trajectory of objects in the scene. Multi-object tracking is one of the most active topics in computer vision and the core foundation for spatial computing systems such as VR/AR. Unlike the common settings, scenes in egocentric settings typically feature diverse objects in varying scales, which often induce classification errors, particularly for methods trained on non-egocentric dataset [1]. This makes robust tracking in AR scenarios remarkably challenging. Typically, the AR system only has limited knowledge about the physical surroundings at the beginning due to the limited FoV of the headset, but we will get more knowledge as the user moves around and rotates the head. In addition, objects may be transformed or consumed during the tasks. For example, in cooking tasks, ingredients are transformed and consumed during the cooking process. For the reasons above, to track and memorize objects in the scene, the tracking system must be able to distinguish between a new unseen object (to be added) and an object seen before (to be discarded). Some tracking methods such as XMem [13] only focus on tracking the object given manual initializations without automatically adding new objects and removing outdated objects using object detection results. Therefore, we mainly discuss related methods capable of object life cycle management in these three categories: multi-object dynamic SLAM, 3D Multi-object tracker, and 2D Multi-object tracker.

2.1 Multi-object Dynamic SLAM

Given RGBD input, dynamic SLAM can not only map the environment and provide the camera pose, but also reconstruct and track moving objects. The common idea is to separate the background and objects, reconstruct each object separately and then use the reconstructed representation to match and track objects in the following frames. Co-Fusion [64] and RigidFusion [82] rely on motion segmentation to segment the objects from background, while MaskFusion [62], MID-Fusion [84], EM-Fusion [70] and TSDF++ [28] rely on object masks provided by Mask-RCNN [29]. However, there are two major issues when using them in AR task guidance scenarios. *First*, the feature-based camera pose estimation and the tracking can fail under motion blur and large inter-frame movement [42, 47, 49]. Therefore, these methods rely on assumptions such as slow/smooth camera motion and high frame rate of camera frames, which are unsuitable for egocentric AR scenarios. *Second*, existing approaches typically assume the rigidity of 3D objects. However, deformable and consumable objects are common in real-world tasks, raising fundamental challenges in dynamic localization. In addition, transparent objects are a common part of everyday tasks, but the time-of-flight sensors in AR headsets have trouble in capturing the shape of transparent objects [91]. This prevents the reconstruction of object shape, and hence tracking by geometric shape matching will fail. These two issues make the multi-object dynamic SLAM not suitable for AR task guidance.

2.2 2D Multi-object Tracker

With the advent of high-performing object detector, many 2D multi-object trackers apply the tracking-by-detection paradigm, so that they can benefit from the enhanced object detection as object detectors improve. In the tracking-by-detection paradigm, for each frame, an off-the-shelf object detector is run first to find all the objects. Then the tracking problem becomes matching the detection to the tracklets and add/delete tracklets as necessary after the matching. SORT [7] demonstrated that 2D tracking-by-detection tracker can achieve state-of-the-art performance and run real-time by using a rudimentary combination of techniques

like IoU for similarity, Kalman Filter for motion model and Hungarian algorithm for matching detections to tracklets. Some methods improve the matching by using similarity between appearance features learned by neural networks. DeepSORT [81] proposed adding the cosine distance between the appearance feature to calculate the similarity. MOTDT [10] proposed using the appearance feature similarity to match the detection and tracklet first, and then using IoU to match the unmatched tracklet. However, these methods need an extra feature extractor which can be computationally expensive. CenterTrack [88] used object centers instead of bounding boxes to perform the tracking and modified the CenterNet [89] to let it not only output the object center, but also a 2D offset of the center from the adjacent frame. Since it requires CenterNet to output 2D offset of the center from the adjacent frame, it cannot take any off-the-shelf object detector as input and benefit from the improvements in the object detectors. ByteTrack [86] proposed a two-stage matching strategy. In the first stage, the high confidence detections are matched to tracklets and the unmatched high confidence detections are initialized to new tracklets after the second stage matching, while in the second stage, low confidence detections are matched to remaining tracklets and no new tracklets will be created for these detections. However, these 2D trackers cannot utilize the information from the depth sensor of AR headset, and do not take the fast camera movement of egocentric view into account. As shown in our evaluations, they will suffer from low frame rate and have poor performances in AR scenarios. In addition, there are methods to improve object detection result for frame-based detector on video stream like the method proposed by Sabater et al. [65] and Seq-Bbox-Matching [5]. They create tubelet, which is similar to tracklet, links detections across frames and aggregates the information to improve the detection results. They not only suffer from the drawbacks of 2D multi-object tracker such as low bounding box IoU under low frame rate and fast headset movement, but also issue like objects move in and out field-of-view breaks tubelet since tubelet only links detections in adjacent frames. To make things worse, these methods focus on improving the object detection results, and the identity from the tubelet is just a byproduct and they are not optimized for keeping track of object identity.

2.3 3D Multi-object Tracker

Many 3D multi-object trackers also adopt the tracking-by-detection paradigm to benefit from the strong power of 3D object detectors. Similar to SORT [7], AB3DMOT [80] created a baseline for 3D MOT and demonstrated that 3D tracking-by-detection tracker can achieve state-of-the-art performance and run real-time using 3D IoU, 3D Kalman Filter and Hungarian algorithm. Different improvements have been proposed on top of AB3DMOT. For example, when calculating similarity, IoU is replaced with L2 distance in CenterPoint [85], and Mahalanobis distance by Chiu et al. [14] SimpleTrack [50] analyzed and improved the four main modules in the 3D MOT pipeline: pre-processing, motion model, association and life cycle management. However, these methods require a 3D detector which outputs 3D bounding box of objects. Currently, there is no 3D detector that can detect tens of thousand classes like Detic [87]. Therefore, these methods are designed and evaluated using driving datasets like KITTI [24], Waymo Open [71] and nuScenes [8], and their generalizability is limited.

2.4 Datasets

Although there are 3D object tracking datasets like KITTI [24], Waymo Open [71] and nuScenes [8], they are for outdoor autonomous driving scenarios, which are very different from indoor AR task guidance scenarios. These datasets usually focus on limited number of object classes, mainly cars and pedestrians, while in AR task guidance, the object classes can vary a lot given the variety of tasks.

To evaluate the 3D memory algorithm, the dataset should satisfy three requirements: (1) egocentric RGBD video, (2) dynamic scene in real-world task scenarios, (3) multi-object tracking annotation. Lopes et al. [44] surveyed existing RGBD datasets. However, none of the existing datasets can satisfy all the three requirements. Datasets like GMU kitchen dataset [25] are recorded with static scene instead of dynamic scene. Real-world egocentric tracking datasets like EgoTracks [73] from Ego4D [26] and TREK-150 [21] from Epic-Kitchen [18] lack depth sensing, and focus on single-object tracking. Datasets like Epic-Kitchen VISOR [19] and MECCANO [56] are recorded with dynamic scene, but only provide active object

(object being manipulated by the user) annotations without object ID instead of annotating all objects with object ID. HOI4D [43] focuses on short video sequences (20s) depicting interactions between a single hand and a single object, deviating from real-world task scenarios. For the reasons above, we collected some recordings of user performing task wearing HoloLens 2 and create a dataset by annotating these recordings for evaluation.

2.5 Visual Analytics for 3D Object Tracking

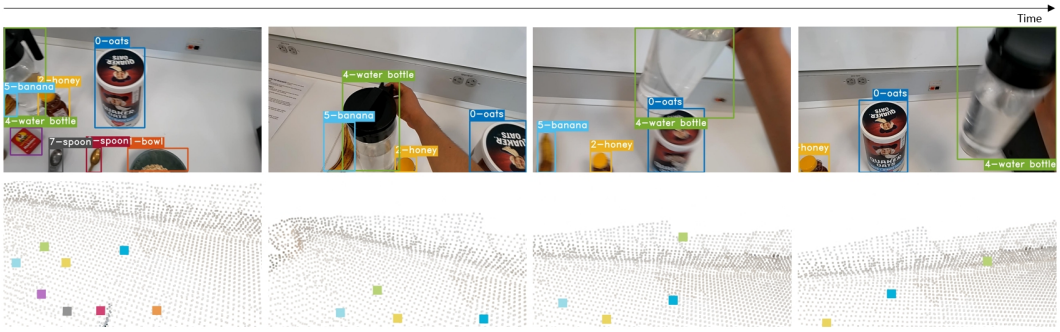
Visual analytics systems play an important role in understanding, debugging and improving machine learning models, especially neural networks due to their black-box-like nature. Hohman et al. [32] surveyed and summarized visual analytics systems for machine learning models. However, none of the existing works is developed for 3D object tracking. There are very few existing works that supports spatial-temporal visual analytics for machine learning models that take both RGB and depth data as input. Hou et al. [33] proposed a visual evaluation method for autonomous driving algorithms. The method has a 3D spatial view to help users understand the environment and other traffic participants. Wang et al. [76] developed a system which focuses on addressing the interpretability issue of 3D object detection in autonomous driving. The system helps users interpret the model failures and improve the models by providing information about when, where, and how the detection model fails. Castelo et al. [9] proposed ARGUS, which is a state-of-the-art visual analytics system to support the development of intelligent AR assistants. However, it only supports analyzing object detection results by object classes, and fails to consider the cases where there are multiple instances from the same class in the scene.

Chapter 3

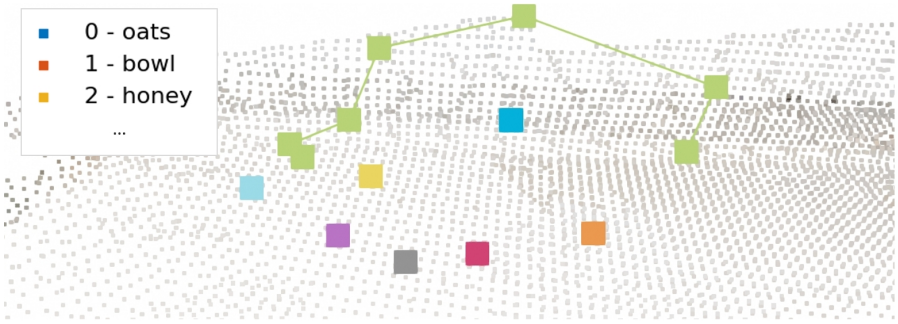
3D Memory Algorithms for AR

AR headsets not only superimpose digital information, but also provide sensing capabilities to understand both humans and the physical surroundings from a first-person perspective. The dynamic understanding of 3D environments also unlocks assistive "episodic memory" based on users' past actions [26, 74], e.g., responding to AR users' inquiry of "where was the mug that I just used?". However, unlike traditional machine vision tasks, AR assistants in real-world scenarios face unique challenges, such as highly dynamic movements of the head-mounted cameras, repetitive objects that may have indistinguishable appearances, or being frequently manipulated by users.

Recent advancements in computer vision allow the AR systems to see and detect objects of different classes. Object detectors like YOLO [57] and Faster-RCNN [59] can both localize and classify the objects in the image space. However, they are trained to predict a fixed set of predefined object categories. This limits the generality of these object detectors since they need to be re-trained with additional annotated data to identify new object categories. Although there are object detectors like YOLO9000 [58] which use a mix of image detection and image classification datasets for training to utilize the class-rich advantage of image classification datasets and can detect thousands of object categories, it is still possible that they will encounter new visual concept which they cannot handle. On the other hand, open-vocabulary object detectors, which use language embeddings of class names to classify objects, have the ability to detect objects outside of the training vocabulary. CLIP [55] provides an embedding allowing efficient learning



(a) Frames of user manipulating object. Top row shows the 2D bounding boxes of the objects. Bottom row shows the estimated 3D positions.



(b) Outputs from the Dynamic Episodic Memory.

Figure 3.1: The proposed Dynamic Episodic Memory can track the manipulated object and remember 3D position of objects even if objects are outside field-of-view.

of visual concepts from natural language supervision. Detic [87] achieves state-of-the-art performance on open-vocabulary and long-tail detection benchmarks by decoupling the localization and classification sub-problems and using CLIP to classify the objects. However, existing object detection approaches face critical roadblocks in the unique AR scenarios to *assist users' spatial awareness*. First, they commonly focus on identifying objects at present. However, in order to assist AR users in a real-world task sequence, the system must "memorize" the entire 3D environment and its changes over time, even if the objects are moved by users or beyond the current field-of-view. Second, machine-learning-based detections may face inevitable spatial (e.g., incorrect classes) and temporal (e.g., missing an object in a frame) imperfections, compromising the reliability of individual frames.

Therefore, spatiotemporally integrated "memory" is essential for an AR assistant to accurately understand and inform users of the physical task objects.

In dynamic tasks, objects are commonly moved by users' hands. With this observation, we proposed a multi-object tracking framework based on tracking-by-detection paradigm. Our goal is to develop a real-time spatial perception system for complex and dynamic AR scenarios. The system provides comprehensive task object information including object ID, position, and identity, as shown in Figure 3.1. To generalize to different tasks, it leverages the "tracking by detection" paradigm and a large vocabulary detector [87].

To summarize, we make the following main contributions

- An end-to-end run-time framework that combines the power of 2D detector and 3D depth information to mimic 3D Episodic Memory, assisting AR users' spatial awareness;
- A tracking-by-detection method designed for complex and dynamic AR scenarios;
- An annotated dataset for evaluating the performance of 3D dynamic episodic memories.

3.1 Method

Similar to many 2D multi-object trackers, we adopt the tracking-by-detection paradigm which has proven successful for multi-object tracking. We use Detic [87] as the object detector, allowing our method to be easily generalized to different tasks, as Detic can detect any class given class names using CLIP [55]. The pipeline of our method can be divided into four parts: inputs, pre-processing, association and life cycle management as shown in Figure 3.2. While most existing 2D/3D multi-object tracking-by-detection trackers include a motion model, our method does not incorporate one, given that objects are static in most cases.

We first analyze why existing 2D multi-object trackers fails in AR scenarios. Then we describe the four parts of our method.

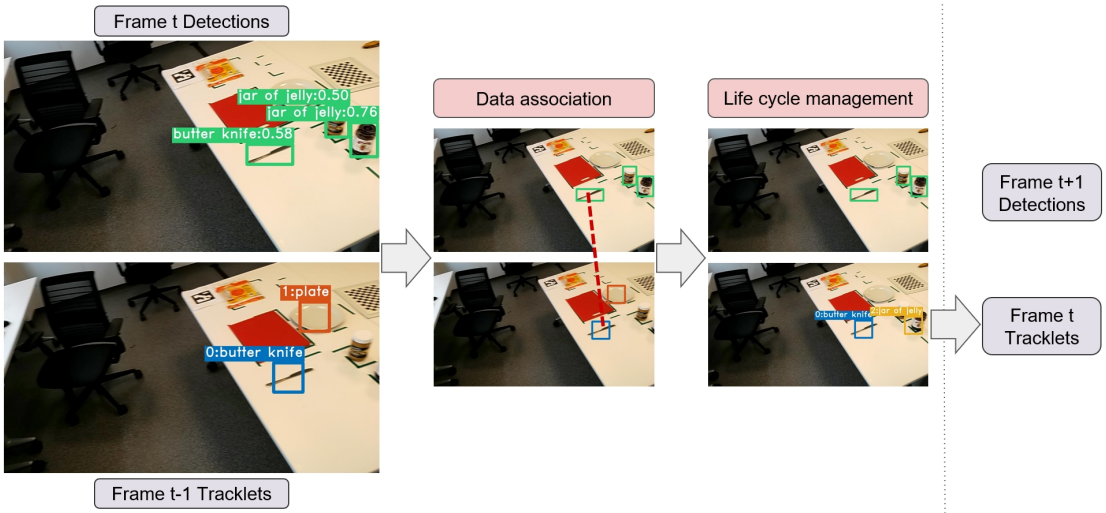
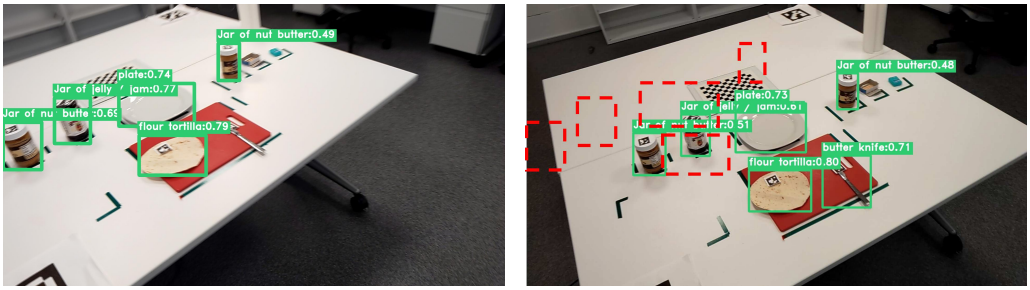


Figure 3.2: Pipeline of our algorithm. For simplicity, we only visualize the step between frame t-1 and frame t. In the data association, the only matched pair is the butter knife tracklet with ID 0 and butter knife detection. In the life cycle management step, new tracklets are created for the jar of jelly, while the tracklet of plate with ID 1 is deleted due to miss. Although the jar of butter is detected as jar of jelly, no tracklet is created since the confidence is lower than the threshold for new tracklet.

3.1.1 Analyze and Rethink 2D MOT

In AR scenarios, 2D MOT can fail at both the association and life cycle management stages. One issue during association is that 2D multi-object trackers use intersection-over-union (IoU) between 2D bounding boxes to calculate similarity Due to the low frame rate and abrupt headset movement, the 2D bounding boxes of a static object may not overlap at all, resulting in zero IoU and matching failures. However, in indoor AR scenarios, objects are static in most cases. Although the 2D bounding box position of a static object may change a lot due to headset movement, the 3D position reconstructed from depth camera of that object in world coordinate remains unchanged as shown in Figure 3.3. Therefore, 3D distance can be used to calculate the similarity instead of using 2D bounding box IoU. Because of the existence of false positives and false negatives in the object detection results, using only 3D distance may result in wrong associations. The class information from the object detection results can also be used to calculate the similarity to reduce errors.



(a) frame at time t (red)

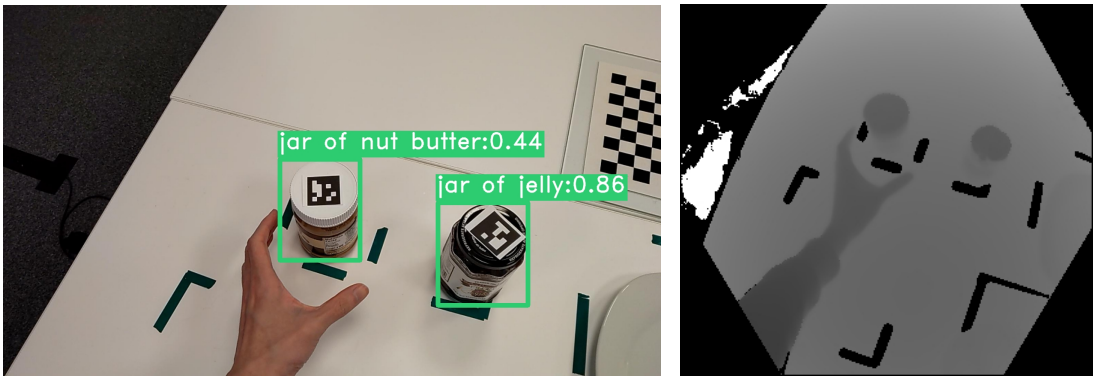
(b) frame at time t+1 (blue)

(c) 3D point cloud of the two frames

Figure 3.3: Comparison of 2D frames and 3D point clouds under movement. Although the 2D bounding box positions change a lot due to headset movement from (a) to (b) resulting in zero bounding box IoU for the static objects, the point cloud of these two frames overlap with each other, which means the 3D positions remain unchanged.

We use a similarity score calculated using the 3D distance and class information from object detector to replace the 2D bounding box IoU.

During life cycle management, most existing 2D trackers will delete an object if it is missed for a certain number of consecutive frames. This could be undesired for AR scenarios, because a static object can become outside FoV due to headset movement and get deleted. When it appears in FoV again, it will be added as a new object, resulting in an identity switch. Also, if the user cannot find an object outside FoV, we cannot guide the user to find it since it has been deleted. For the reasons mentioned above, we should only delete missing objects if they should



(a) RGB frame with object detection (b) depth frame

Figure 3.4: Inputs of our algorithm. Note that the RGB frame and depth frame are from separated cameras, resulting in different field-of-view.

appear inside FoV. AR headset can provide information like headset position, head orientation and camera parameters, so that given a 3D coordinate, we can tell whether it is inside FoV or not.

3.1.2 Input and Tracker State

As shown in Figure 3.4, for each incoming RGB frame, the input to our method contains:

- RGB frame with camera intrinsic/pose
- Object detection result from the RGB frame
- (Temporal nearest) depth frame with camera intrinsic/pose

Meanwhile, the information of each tracklet include:

- Tracklet ID
- 3D position
- Class labels of past N observations (a sliding window of max N frames)
- Timestamp of last seen
- Counts for life cycle management (seen count & consecutive miss count)

The 2D bounding box of matched detection is also saved to tracklet for visualization and evaluation purposes.

3.1.3 Pre-processing

3.1.3.1 Frames Alignment

AR headsets like HoloLens 2 have separated RGB camera and depth camera instead of a single RGBD camera. The RGB camera and depth camera can have different frame rates, field-of-view and resolutions. Therefore, it is necessary to align the frames from the RGB camera and depth camera. The alignment is performed by mapping the each point of the depth point cloud to a pixel of the RGB image.

The 3D directions of each pixel of the depth camera can be calculated from the camera intrinsic or provided by the manufacturer. By multiplying these 3D directions with the depth values of each pixel, we can reconstruct a 3D point cloud for the given depth image. We denote the position of pixel p in coordinate system s as $Pos_{p,s}$. Let (x_p, y_p, z_p) be the direction of pixel p in 2D depth image and d_p be its depth value. The 3D position of the pixel p in the depth camera coordinates can be calculated as $Pos_{p,depth} = (d_p x_p, d_p y_p, d_p z_p)$.

In addition to the local coordinate systems of the cameras on board, AR headsets also provide a world coordinate system and camera extrinsic parameters which transform the world coordinate system to camera coordinate system for each captured frame. Although the camera positions are relatively fixed to each other, we cannot transform from one camera coordinate system to another camera coordinate system directly since the frames could be from different timestamps. Therefore, the world coordinate system can link the camera coordinate systems of the different frames. By transforming the point cloud into the world coordinate system, we can further transform it to the coordinate system of other cameras to utilize information from other cameras. For example, we can further transform it to the RGB camera space, allowing us to generate a pseudo-RGBD image and take advantage of object detection results from the RGB camera.

Let $M_{depth2world}$ be the matrix transforming the coordinates from the depth camera space to the world space of a depth frame, and $M_{world2rgb}$ be the matrix transforming the coordinates from world space to RGB camera space of an RGB

frame. For a pixel p of the depth frame, we obtain its 3D coordinate in RGB camera space as $Pos_{p,rgb} = M_{world2rgb} \cdot M_{depth2world} \cdot Pos_{p,depth}$.

The camera intrinsic parameters of the RGB camera such as the focal length f_x, f_y and principal point p_x, p_y are also provided by the headsets. This allows us to reconstruct the camera intrinsic matrix $A = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}$ and convert the $Pos_{p,rgb}$ back to the 2D image space of the RGB camera. Using this transformation, the 3D points will be mapped to RGB pixels. Since the field of view of the RGB camera and depth camera are different and not completely overlapped, the converted pixel (x_p, y_p) may be outside the RGB image, thus omitted. This transformation can also help us decide if a 3D point should appear inside camera field-of-view during the life cycle management stage.

Given an RGB frame with object detection result, we first find the temporal nearest depth frame available, and then perform the alignment as described above. As a result, some RGB pixels will have the associated 3D point since the resolution of RGB camera is usually much higher than the depth camera.

3.1.3.2 3D Position from 2D Detection

Since 2D object detectors like Detic can only provide 2D bounding boxes instead of 3D positions, we then calculate the 3D positions of the detections. Given a 2D bounding box, we use all the 3D points inside the bounding box from depth frame to infer 3D position of the object. We use the average of the 3D points as the 3D position of the object.

3.1.3.3 Non-maximum Suppression

As discussed in [50], object detectors usually output a large number of bounding boxes to fulfill the recall requirements, leading trackers to select inaccurate detections for extending or forming tracklets. For example, as shown in Figure 3.5, Detic outputs 3 bounding boxes for the jelly, and 2 bounding boxes for the Nutella and nut butter. To address this issue, non-maximum suppression can be applied for detection results within the same class and across different classes. The IoU threshold for different class objects should be much higher than the threshold of

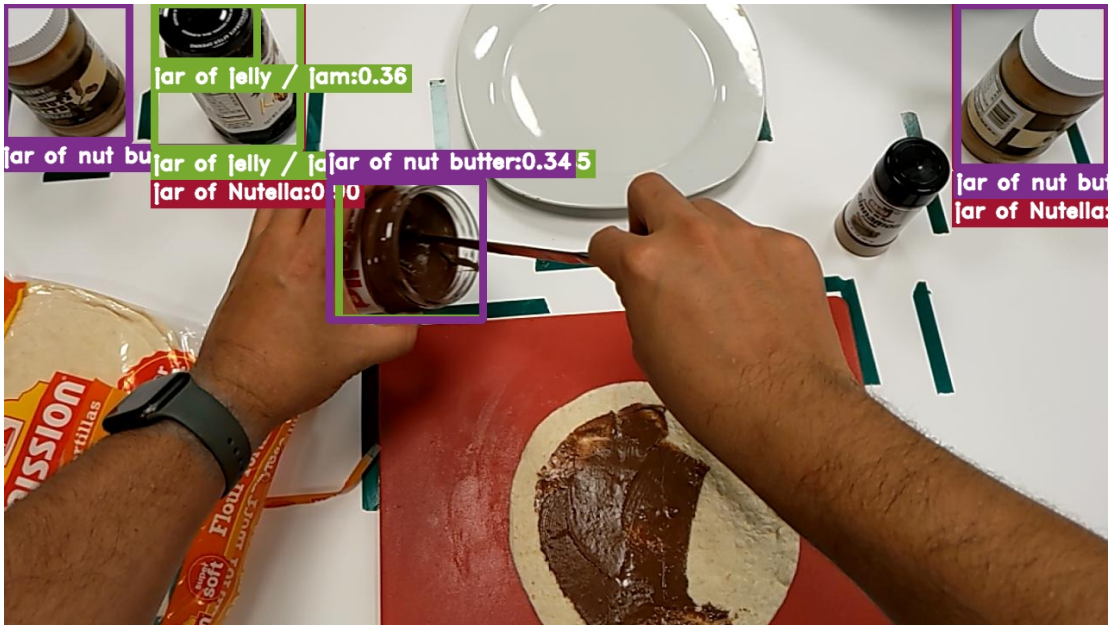


Figure 3.5: Raw outputs from Detic. For a single object, Detic can output 2-3 detection results, and this will lead the trackers to select the duplicate detections and create extra tracklets.

non-maximum suppression within the same class. This is because there are valid cases where the bounding box of an object is inside the bounding box of another object with a different class, such as a tortilla placed on a cutting board.

3.1.4 Association

In this stage, the tracker tries to match the tracklets with the detections in the incoming frames. When designing the data association, we mainly consider two factors: 3D position/distance and class label from the object detector. We don't take image features or object geometric shape into consideration because objects can be non-rigid, and their appearance and shape may change during the task process.

3.1.4.1 Similarity Metric

Based on the insight that objects are static unless moved by the hand in indoor AR scenarios, 3D distance is used to calculate the similarity instead of using

2D bounding box IoU. However, using only 3D distance may result in a wrong association. The class information from the object detection results is also used to calculate the similarity to reduce error.

Our similarity metric is based on 3D position/distance and class information from the object detector. Let d_{ij} be the 3D Euclidean distance between detection D^i and tracklet T^j, α is a parameter for the similarity distribution, the position similarity is calculated as:

$$S_{pos} = e^{-\alpha \cdot d_{ij}} \tag{3.1}$$

Let N be the size of the sliding window and C_{ij} be the count of class label of detection D^i in the the sliding window of tracklet T^j, and $Confidence_i$ be the confidence of detection D^i. The class similarity is calculated as:

$$S_{class} = Confidence_i \cdot \frac{C_{ij}}{N} \tag{3.2}$$

Given the parameter λ which balances the position similarity and class similarity, the final similarity score S is calculated as:

$$S = \lambda S_{pos} + S_{class} \tag{3.3}$$

3.1.4.2 Matching Strategies

After obtaining the similarity scores for all possible pairs of tracklets and detections, a matching strategy is needed. We use greedy algorithm to associate detections and tracklets. As discussed in [14, 50], distance-based metrics prefer greedy algorithms to Hungarian algorithm. Starting from the pair of detection D^i and tracklet T^j with the highest score, we iteratively associate the pairs until the score is smaller than a threshold $Thresh_{score}$. The match pairs of detection D^i and tracklet T^j, matched detections $D_{matched}$, unmatched detections $D_{unmatched}$, matched tracklets $T_{matched}$ and unmatched tracklets $T_{unmatched}$ will be saved and used to update tracker states later.

3.1.5 Life Cycle Management

Here the tracker deletes the outdated tracklets and create new tracklets when necessary. Since new objects may enter and existing objects may be transformed or consumed, life cycle management is required to add and delete tracklets. We use a count-based approach for the life cycle management. For matched tracklets $D_{matched}$, the tracklet will be updated by using the detection position as the new position, adding the class to sliding window, increasing the seen count by 1 and reset consecutive miss count to zero. For unmatched detections $D_{unmatched}$, a new tracklet will be created if the confidence of the detection $D^{i}_{unmatched}$ is larger than detection score threshold for creating new tracklet $Thresh_{conf}$. The new tracklet will be initialized with detection position as the position, the class label as the only element in the sliding window, seen count equals to 1 and consecutive miss count to 0. For unmatched tracklets $T_{unmatched}$, we first determine whether they should be inside the FoV by mapping the 3D position to the 2D camera plane as describe in Section 3.1.3.1 and filter out those outside FoV. For the unmatched tracklets inside the FoV, their seen counts will be decremented by 1 and their consecutive miss count will be added by 1. The unmatched tracklets can be deleted in two ways. First, they will be deleted if they miss for a number of consecutive frames so that consecutive miss count is larger than max number of consecutive misses allowed N_{max_miss}. Second, they will be deleted if the seen count reaches 0. This allow the tracker to delete both false positive and disappeared objects effectively. Disappeared objects usually have a large seen count, so it takes a large number of frames to decrease the seen count to 0 and deletion after missing for a number of consecutive frames can handle this case. If we only rely on number of consecutive missed frames, false positive can survive for that number of frame and this can be undesirable. Using the seen count can help handle this case.

When outputting tracklets, only matched tracklets $T_{matched}$ in this frame will be output. In order to be output, the matched tracklets need to have a seen count larger than min seen count required N_{min_seen}. For unmatched tracklets $T_{unmatched}$, they will become inactive and will not be output.

3.2 Evaluation

3.2.1 Metrics

We evaluate the tracking algorithms using several MOT metrics [6], including MOTA[1] (multi-object tracking accuracy), IDS (number of identity switches), TP (number of true positives), FP (number of false positives) and FN (number of false negatives). Since other algorithms output 2D bounding boxes and data annotations are also in 2D, our method will store the matched detection's 2D bounding box to the tracklet for evaluation.

3.2.2 Datasets

As discussed in Section 2.4, none of the existing labeled datasets can match all our targets for evaluation. Therefore, we created a dataset using recordings of users performing recipes and wearing HoloLens 2. This dataset contains 12 recordings which cover 4 recipes - pinwheels, dessert quesadilla, tea and oatmeal, containing including objects from 20 different classes. Apart from the headset movement during the task processes, it also contains many other challenges like duplicate objects, transparent objects and occlusions. In some cases, object movement process was outside field-of-view due to the limited camera FoV. It is split into a training set (8 recordings), and a test set (4 recordings). The dataset is annotated using Computer Vision Annotation Tool (CVAT), an industry-leading data annotation platform widely used by communities and companies. The dataset frame rate is around 2.5fps to simulate real-world, run-time AR scenarios where the object detection speed is limited by the available on-device computational power. Notably, various datasets with similar purposes also provide annotated videos at 1-2fps, such as TAO dataset [20] and nuScenes [8]. Corresponding tracking methods [88, 90] also operate at this rate. The intervals between frames may not be uniform given the variations of the network. There are 6045 frames and 35,490 object annotations in total. To facilitate the replication of our evaluations, we have also included the Detic detection results used in our analysis.

[1] $\text{MOTA} = 1 - \frac{\text{FP+FN+IDS}}{\text{GT\#}}$ is one of the most popular metrics used for ranking methods on MOT benchmarks

3.2.3 Implementation Details

We set the parameter for the similarity distribution α to 5, λ which balances the position similarity and class similarity to 2 and the min similarity score of a matched pair $Thresh_{score}$ to 1.4 through a pilot study using recordings with Aruco markers. For parameters that can be fine-tuned, we set the detection score threshold for new tracklet $Thresh_{conf}$ to 0.7, max number of consecutive misses allowed N_{max_miss} to 5 so that our algorithm can handle short-term occlusion under 2 seconds at 2.5 fps, windows size N to 15 and minimum number of seen count required N_{min_seen} to 2. These parameters are determined using the training set as shown in the ablation study later. For Detic, we use the Detic_C2_SwinB_896_4x_IN-21K+COCO model with their released weights. The proposed algorithm is implemented with Python. All the evaluations run on a PC with Intel i9-9880H 8-core CPU and 32GB of RAM.

3.2.4 3D Baseline

Since 2D trackers cannot utilize 3D depth and class label information, we create a simple baseline 3D tracker using greedy algorithm to validate the proposed similarity for data association. The baseline tracker also uses the tracking-by-detection pipeline. The only difference between our proposed method and this 3D baseline is the similarity score of data association. In data association, given Detic detections of the current frame, it greedily matches the tracklet to the closest detections of the same class. The life cycle management of this baseline is the same as our proposed method. It can be implemented by setting the sliding window size to 1 and min similarity score of a matched pair $Thresh_{score}$ to 0, and the similarity score S is calculated by

$$S = C_{ij} \cdot e^{-d_{ij}} \tag{3.4}$$

We also fine-tuned the confidence threshold parameter by selecting the value with highest MOTA on the training set and set it to 0.7.

	TP	FN	FP	IDS	MOTA
Ours	**7294**	**4517**	2047	**75**	**0.4379**
3D Baseline	7206	4605	2728	144	0.3669
ByteTrack	5570	6241	982	563	0.3408
SORT	5011	6800	**750**	470	0.3210

Table 3.1: Comparison with 2D trackers using Detic detections on the test set. The best results for each metric are shown in **bold**.

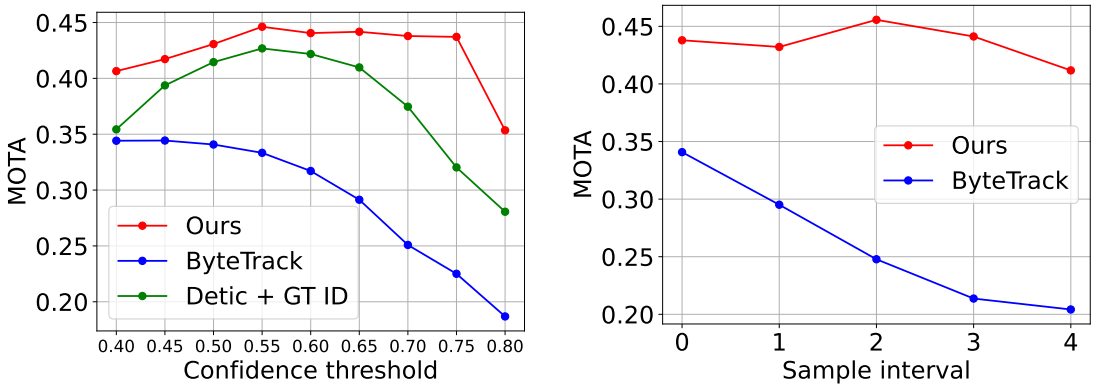
	TP	FN	FP	IDS	MOTA
Ours	11412	399	**0**	65	0.9607
3D Baseline	**11494**	**317**	**0**	**41**	**0.9697**
ByteTrack	9951	1860	122	896	0.7563
SORT	8999	2812	**0**	649	0.7070

Table 3.2: Comparison with 2D trackers using ground truths on the test set. The best results for each metric are shown in **bold**.

3.2.5 Comparisons with 2D trackers

We compare the proposed algorithm and the 3D baseline with 2D trackers, specifically SORT [7] and ByteTrack [86]. These 2D trackers were chosen as they don't require extra neural networks for tracking and can take bounding box (x1, y1, x2, y2, score) from any 2D detector as input. ByteTrack is a state of the art 2D tracker, and SORT is a classic 2D tracker and barebone of many other 2D trackers. Since SORT and ByteTrack do not incorporate class information, we run multiple instances of these trackers, one for each object class. Each instance receives only detections of the corresponding class, and all the output is labeled with that class. To ensure a fair comparison with our method and 3D baseline whose confidence thresholds were fine-tuned by selecting the value with highest MOTA on the training set, we fine-tuned the confidence threshold parameter for ByteTrack and SORT using the training set, setting it to 0.5.

Table 3.1 presents the results using Detic as detection inputs. For reference, the TP, FN and FP of the input Detic detection results at a 0.5 confidence threshold are 7228, 4583, and 2132, respectively. Our method significantly outperforms the 2D trackers, especially in terms of IDS. These results demonstrate that 2D trackers struggle with identity tracking. Although 2D trackers have fewer FPs compared to 3D trackers, this advantage comes at the expense of generating fewer bounding



(a) MOTA w.r.t conf threshold

(b) MOTA w.r.t frame rate

Figure 3.6: Comparison of performance under different confidence thresholds and framerates on the test set.

boxes, resulting in a higher number of FNs. Additionally, our method surpasses the 3D baseline, highlighting the effectiveness of the proposed similarity score.

Tracking-by-detection allows algorithms to benefit from enhanced object detectors. To evaluate different trackers given ground truth detections, we assessed them using ground truth annotations. Table 3.2 illustrates the results using ground truths as detection inputs. Despite using ground truths, the 2D trackers still perform poorly and exhibit a high number of IDS. This is due to their assumptions about camera stability and high frame rate, which do not hold in AR settings. Association using 2D bounding box IoU and motion model like Kalman filter does not work well under AR settings due to the low frame rate and abrupt movement of user hand/-head. In addition, the 2D trackers will delete an object under consecutive misses and this can be undesirable for objects outside FoV. The 3D Baseline surpasses our method since nearest neighbors matching within the same class overfits the ground truth inputs, where class labels are always correct, an unrealistic scenario for object detectors. Our method and the 3D baseline achieve near-perfect results with ground truth inputs, and we will discuss the limitations of our method that prevent us for achieving better results in Section 3.3.

Effect of input frame rate. Tracking performance can be influenced by the input frame rate. Therefore, we explore this effect by sampling one frame from every n frame of the dataset. Results in Figure 3.6b indicate that our method is

more robust to lower frame rates than ByteTrack. This resilience is attributed to the fact that 2D bounding boxes are not robust similarity indicators, being easily affected by low frame rates, abrupt camera movement, and even packet drops during transmission.

Runtime performance. We also compare the runtime performance of our algorithm with ByteTrack using recordings with an average of 8 visible objects per frame. During pre-processing, estimating 3D positions given 2D detections and RGBD image takes about 6.6ms/frame. For the data association and life cycle management, our algorithm takes 2.1ms/frame on average, while ByteTrack takes 4.0ms/frame. Therefore, the entire tracking pipeline can run at approximately 120fps, introducing minimal latency in interactive scenarios.

3.2.6 Comparison with Detic + GT ID

Let's consider the scenario where the Detic detections are first filtered by a threshold and then assigned with ground truth IDs (Detic + GT ID). This represents the upper bound performance of using the filtered Detic detections for tracking, since such an oracle tracker that can perfectly assign IDs does not exist in the real world. Figure 3.6a shows the MOTA under different filtering thresholds for Detic + GT ID, new tracklet thresholds for our method and detection score thresholds for ByteTrack. Our method outperforms the other two methods and is more robust to different confidence threshold values. Furthermore, since Detic + GT ID is using ground truth IDs, the IDS error for its result is 0. The MOTA is calculated by $1 - \frac{FP+FN+IDS}{\#\ of\ GT}$. In this case, Detic + GT ID will only be penalized by the FP+FN from Detic detections, while our method will be penalized by all FP, FN and IDS. Therefore, the higher MOTA of our method means that our method has fewer FP and FN than Detic + GT ID. As a result, if we apply our method as an object detector by only considering the object class and object position in the outputs, our method can achieve better results than the Detic detections. This suggests that our method can improve the detection quality by better utilizing the temporal information.

	TP	FN	FP	IDS	MOTA
Ours	14661	9015	4523	190	**0.4202**
w/o nms	14807	8869	5513	220	0.3833
w/o seen count	14737	8939	4697	198	0.4157
$Thresh_{conf}$=0.65	**15240**	**8436**	5103	225	0.4187
$Thresh_{conf}$=0.75	13352	10124	**4013**	**165**	0.3959
N_{max_miss}=3	14483	9193	4365	212	0.4184
N_{max_miss}=10	14930	8746	4944	174	0.4144
N_{min_seen}=1	14912	8764	4662	300	**0.4202**
N_{min_seen}=3	14493	9183	4416	168	0.4185

Table 3.3: Ablative analysis of different design choices of our proposed method on training set. The best results for each metric are shown in **bold**.

	TP	FN	FP	IDS	MOTA
N=1	14521	9155	4897	274	0.3949
N=5	14675	9001	4827	198	0.4076
N=10	14678	8998	4623	**190**	0.4167
N=15	14661	9015	4523	**190**	**0.4202**
N=20	**14686**	**8990**	4623	**190**	0.4170
N=50	14619	9057	**4551**	225	0.4157
N=∞	13818	9858	4674	439	0.3677

Table 3.4: Ablative analysis of window size of our proposed method on training set. The best results for each metric are shown in **bold**.

3.2.7 Ablation Study

In this section, we investigate the impact of different design choices using the training dataset, considering the imperfect nature of real-world object detectors and evaluating the design choices based on Detic detections as input.

Effect of non-maximum suppression. Non-maximum suppression, as shown in Table 3.3, reduces the number of false positives at the expense of decreasing true positives. Despite this trade-off, non-maximum suppression contributes to a higher MOTA. Detic's capability to produce multiple detection results for a single object, illustrated in Figure 3.5, can result in the creation of multiple tracklets for that single object, leading to additional false positives. In some instances, these extra detections generated by Detic might be associated with inactive tracklets, causing

false positives and identity switches.

Effect of seen count decrement. Existing trackers like SORT and AB3DMOT will start outputting matched tracklets once it satisfy the min hits requirement and only delete the tracklet by reaching the max consecutive misses allowed. However, this may lead to undesired outcomes, as false positive detections may have confidence score higher than the new tracklet threshold $Thresh_{conf}$. It takes time for the created tracklet to reach the max consecutive misses allowed N_{max_miss}, and during this time, false positive detections with low confidence can be associated and output. By decreasing the seen count during misses and apply output filtering and tracklet deletion based on the seen count, we achieve higher MOTA, reducing false positives and identity switches.

Effect of new tracklet threshold. As the new tracklet threshold $Thresh_{conf}$ increases, false negatives increase while false positives and identity switches decrease. A value of 0.7 yielded the highest MOTA when explored across the range of 0.5 to 0.8 on the training set.

Effect of max consecutive misses allowed. The choice of decreasing or increasing the max consecutive misses allowed (N_{max_miss}) has implications. Decreasing it may cause occluded tracklets to be deleted, resulting in identity switches when the objects become unoccluded and are detected again. In contrast, increasing its value may cause outdated tracklets may not be deleted in time, matching them to false positive detections and let them evade the the new tracklet threshold $Thresh_{conf}$, causing an increase in false positives. Setting it to 5 is deemed appropriate based on MOTA.

Effect of min seen required. When changing the min seen required for a tracklet to be output N_{min_seen} to 1, the newly created tracklets will be output immediately instead of needing an extra detection from the future frames to be confirmed. This will increase both true positives and false positives since true positives can be output starting from first frame and false positives that only last one frame can also be output. This change also significantly affects the number of identity switches. Since for newly created objects, their class label windows are not

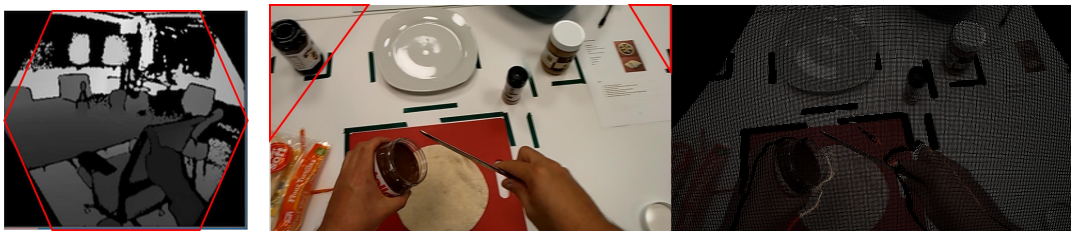
yet filled and usually results in low class score. As a result, when they are under fast movement, the detection cannot be matched to the existing tracklets so that new tracklets are created for each frame, resulting in identity switches. Alternatively, changing the N_{min_seen} to 3 will result in lower the MOTA. A compromise is found by setting N_{min_seen} to 2, aligning with ByteTrack's implementation behavior.

Effect of window size. We investigate the effect of the class label sliding window by varying its size. A small window limits the ability to utilize information from the past. In the most extreme case where window size is 1, only the class information from the last seen frame can be used. On the contrary, large window size allows for more extensive use of historical information. Since it takes time for the window to be filled, tracklets generated by false positive detections usually has few observations and few elements in the window. As a result, they will have lower class similarity score comparing with true positive tracklets, and the tracking will be more robust. However, if the window size is too large, the sliding window can hardly be filled and each observation will only increase the class similarity score by a very small amount. This can be undesirable since it makes the data association heavily relies on the position score. Table 3.4 demonstrates that both too small and too large values lead to inferior performance. Setting it to 15 is deemed appropriate based on MOTA.

3.3 Limitations

In case where the headset has separated RGB and depth cameras instead of a single RGBD camera like HoloLens 2, it is possible that some parts of the RGB image will lack depth information as shown in Figure 3.7 due to the different field of view of the RGB and depth cameras. As a result, 2D detections in those parts cannot convert to 3D, and the corresponding tracklets cannot be matched to the detections and will be deleted due to consecutive misses. One way to mitigate this is to find a better way to utilize depth information from the past instead of just using the temporal nearest depth frame.

In addition, as shown in Figure 3.8, some objects in the scene can be transparent, such as a glass cup. A 2D detector can see through the transparent objects while



(a) depth field-of-view (b) left: RGB frame, right: pseudo-RGBD frame after alignment

Figure 3.7: Aligned frame whose corners (highlighted in red) don't have depth information. The depth FoV of HoloLens 2 is in hexagon shape, which results in the top left jar of jelly detection doesn't have any depth information available and cannot be converted into 3D.

the depth camera may not. This will result in inaccurate 3D position estimation for the objects behind transparent objects.

In the evaluation of the 3D baseline, we fine-tuned only the parameters that significantly impact the MOTA, as identified in the ablation study, such as the confidence threshold. Parameters with negligible effects on the MOTA, like N_{max_miss} and N_{min_seen}, were maintained at the same values as those used in our method. This approach may have slightly disadvantaged the 3D baseline's MOTA performance.

Although our method can handle short-term occlusion up to max number of consecutive misses allowed N_{max_miss}, it doesn't attempt to solve tracking objects under long-term occlusion. The object will be deleted if there are misses for N_{max_miss} consecutive frames. Therefore, long-term occlusion can result in identity switches.

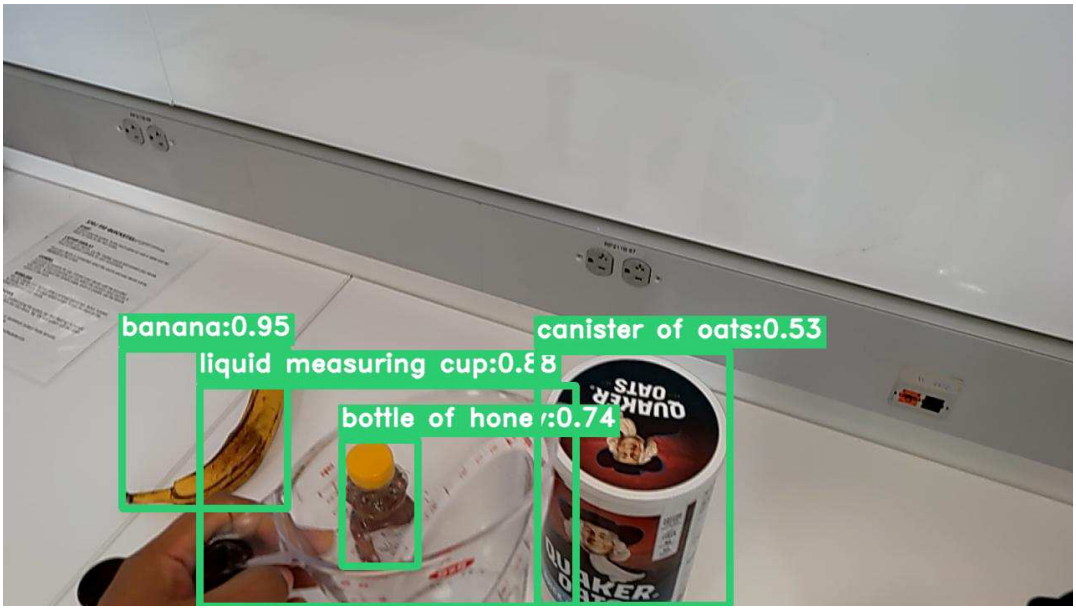


Figure 3.8: Example of transparent objects. 2D detector can detect the bottle of honey behind the liquid measuring cup while the depth camera cannot provide correct 3D position of the bottle of honey.

3.4 Discussion

We present an end-to-end run-time framework that combines the power of 2D detector and 3D depth information to mimic 3D Episodic Memory for AR, along with a tracking-by-detection method designed for complex and dynamic AR scenarios. Using Detic as the object detector, it can be generalized on different tasks. The average runtime per frame is about 8ms, which means it won't introduce latency for interactive scenarios and can run at 120 fps. The evaluation shows that it can work well under low frame rate and abrupt camera movement, where the existing state-of-the art 2D trackers can fail. In addition, it can benefit from the improvement of object detectors and output near-perfect results with perfect detections.

In future, object detectors such as Segment Anything [36] may output accurate mask instead of bounding box for detected objects, resulting in a more accurate estimation of the 3D position given the 2D detections. However, for headsets with separated RGB and depth cameras like HoloLens 2, since the RGB frame and depth frame are from different timestamps, it is still quite challenging to estimate

the position of moving objects. Another interesting direction of future work is to incorporate image features to solve the long-term occlusion issue and the moving object issues by re-identifying the long-term occluded objects and moving objects using image features.

Chapter 4

Visualization for 3D Memory

4.1 ARGUS for 3D Memory

Visualization tools for 3D memory play a crucial role in helping developers to debug and enhance the algorithm, especially for the case of AI assistants where multiple machine learning modules work together. Figure 4.1 illustrates the architecture of TIM, a Transparent, Interpretable, and Multimodal AR Personal Assistant designed to guide users through various tasks. To automatically guide users through task steps, machine learning modules for object detection, action detection, step reasoning, and 3D memory are employed. Given the complexity of the pipeline, visual analytics tools become essential for comprehending and debugging these modules.

ARGUS [9] is a state-of-the-art visual analytics system to support the development of intelligent AR assistants. It has the capability to visualize not only data

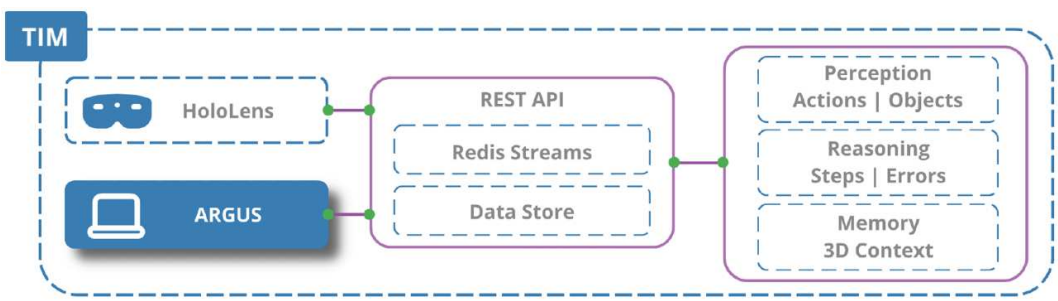


Figure 4.1: The architecture of TIM

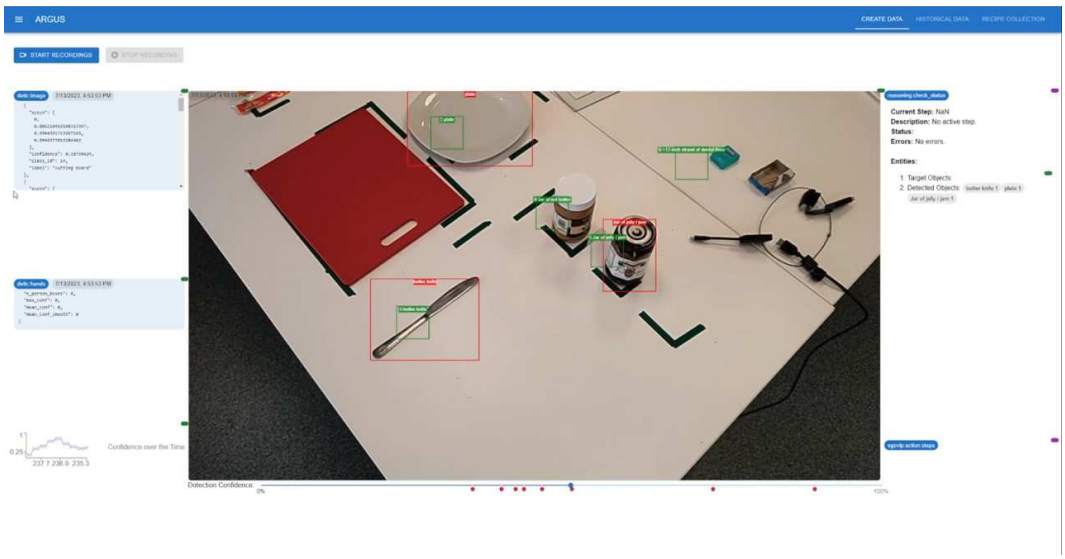


Figure 4.2: ARGUS Online Mode. The memory output is rendered in green.

streams from multimodal sensors on HoloLens 2 but also the machine learning model outputs of TIM. Therefore, we choose to build our visual analytics tools for 3D memory on top of ARGUS to leverage the advantages offered by this system. ARGUS has two modes, online mode and offline mode, both of which are modified for visualizing the 3D memory.

4.1.1 ARGUS Online Mode

ARGUS online mode (shown in Figure 4.2) enables developers to observe a live view from the main RGB camera of HoloLens 2, along with online visualizations of object, action, and step detection. Additionally, developers can use the buttons in the top left to start/stop recording the data for subsequent offline analysis We incorporate visualization for live 3D memory output, similar to the visualization of object detection, by drawing a green bounding box for each tracklet and providing the object ID and class label.

4.1.2 ARGUS Offline Mode

ARGUS Offline mode enables analysis of the recorded data generated by performer actions and the models of TIM. Given the spatiotemporal characteristics

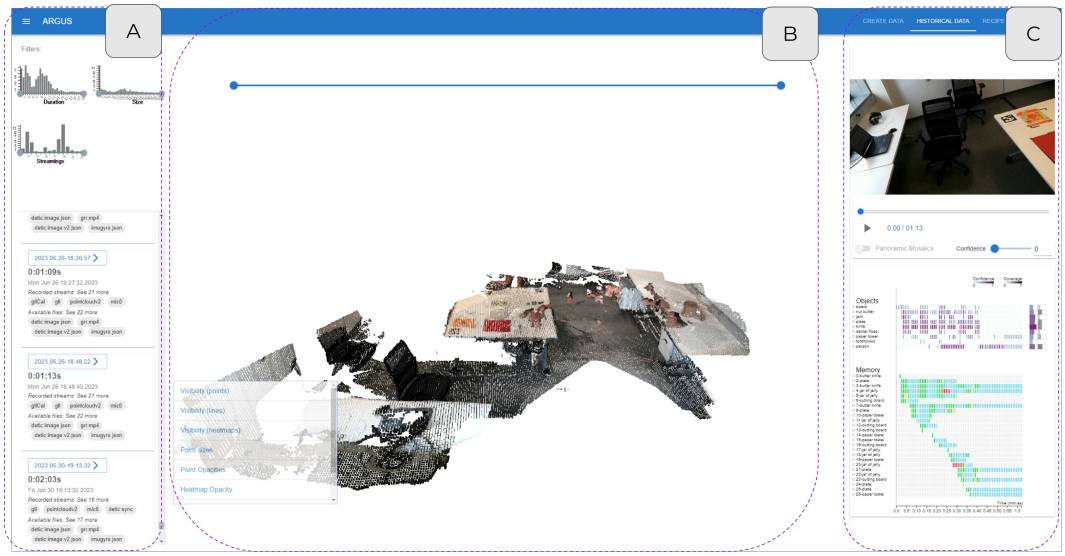


Figure 4.3: ARGUS Offline Mode. (A)data manager, (B)spatial view, (C)temporal view.

of the data, both spatial and temporal visualization widgets are provided to allow developers analyze the data from different perspectives. As shown in Figure 4.3, the offline mode consists of 3 parts: the data manager, spatial view and temporal view. Both spatial view and temporal view are modified to support visualize memory output

Temporal View. A summary of the model output is crucial for starting an analysis. The visual summary of the temporal view allows users to quickly grasp a global picture of the model outputs. As shown in Figure 4.4, the memory output is visualized by using each row represents the object status of an object across time. The color of the cell in the row represent the object status. Green indicates detection by the object detector. Yellow represents undetected status with the tracklet extended by the memory. Red denotes manipulation by the performer's hand Blue represents being outside the camera field-of-view. Some objects can occupy multiple rows if their class labels change during the session, as illustrated by objects with ID 3,4,6 in the Figure 4.4.

The video player is also modified to support displaying memory outputs by rendering the bounding boxes in green. This allows developers to analyze specific time frames. When a frame is replayed in the video player, the rows of objects

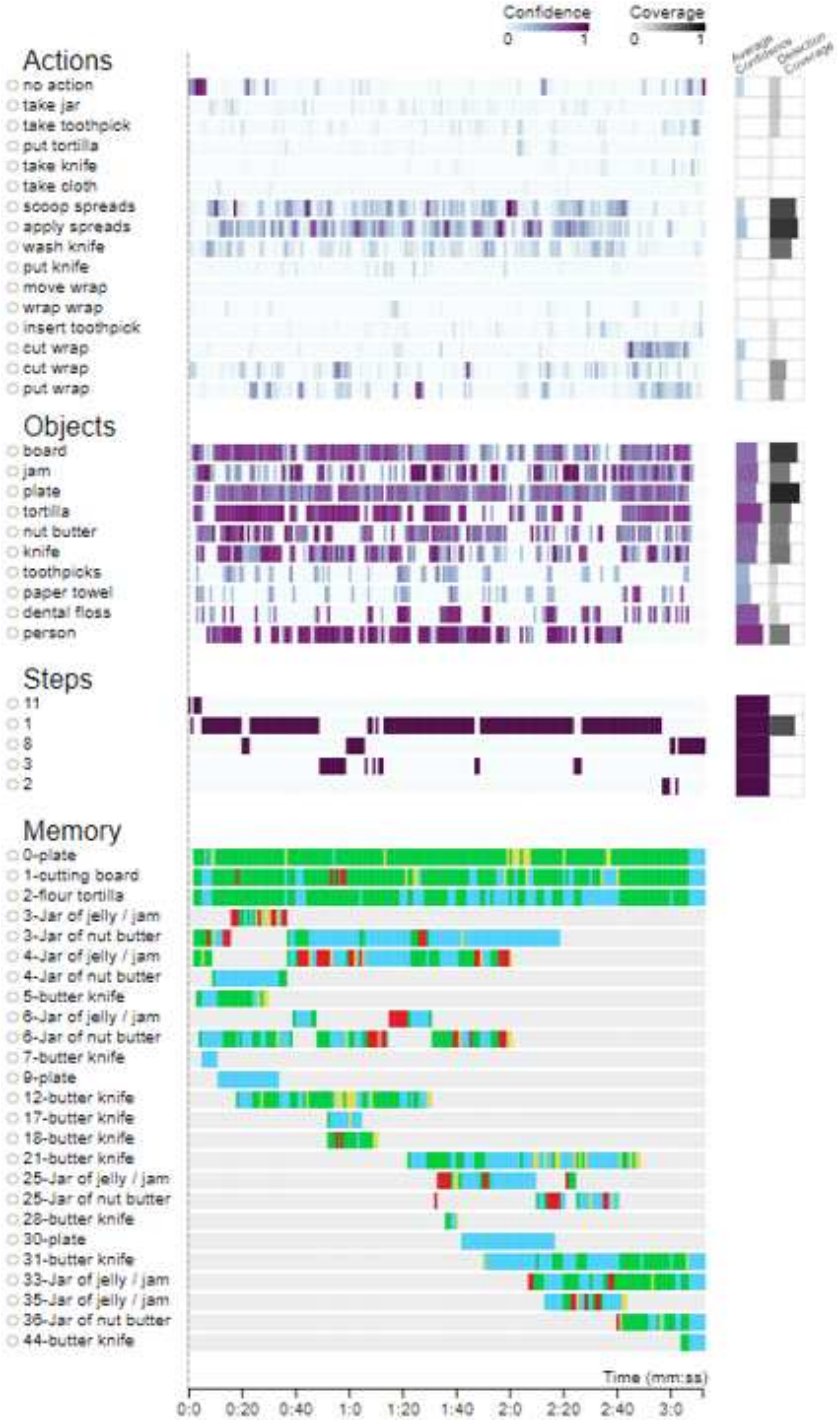


Figure 4.4: Temporal view of ARGUS offline mode. A visual summary of the memory output for one session is provided.

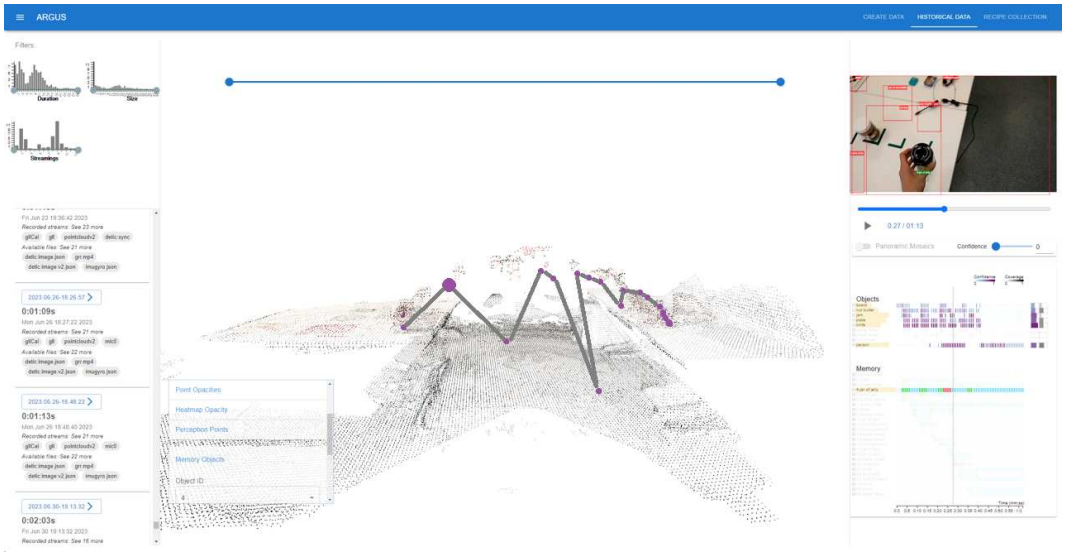


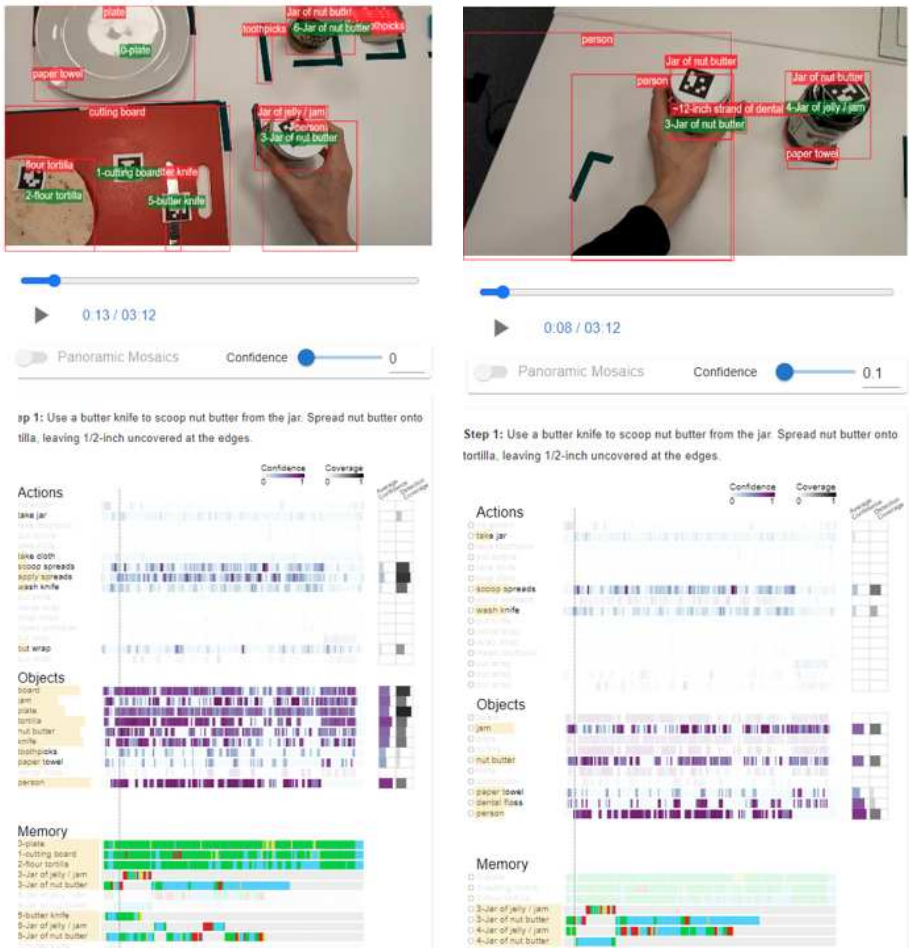
Figure 4.5: Spatial view of ARGUS offline mode. The trajectory of an object is visualized. When pointing towards a point of the trajectory, the temporal view and video player will show the show the corresponding frame.

in that frame will be highlighted, and the corresponding bounding boxes will be rendered in the video player.

Spatial View. The spatial view (Figure 4.5) allows the visualization of streamed spatial data, such as the point cloud from the depth frames. The memory output is visualized by showing the trajectory of an object. Additionally, when pointing toward the points on the trajectory, the temporal view and video player will show the corresponding frame of that point.

4.1.3 Case Studies

Debugging the modules. When a developer sees the temporal view of Figure 4.4, it can be noticed that there are many objects (object id 3,4,6,25) occupy more than one row, indicating a change in the class label stored in memory. This suggests that the object detector struggles to differentiate between these two classes and predicts a mix for the objects. The developer can investigate the issue further by using the video player to replay the frame before the label change, as shown in Figure 4.6a. For example, in this frame, the object with ID 3 is misdetected as jam



(a) debug object detection (b) debug action recognition

Figure 4.6: Using temporal view of memory and video player to debug the modules.

jelly, likely due to the viewing angle, resulting in the label change.

An issue of the action prediction can also be identified. The memory shows that the user was moving the jar of butter or jelly around most of the time. However, the actions output has very low coverage and confidence for the "take jar" action. By replaying the video (Figure 4.6b), it can be confirmed that "take jar" has lower confidence than the action "scoop spreads" and "wash knife" even though there is no knife detected.

Comparing different memory algorithms. The temporal view of memory can also be used to compare the performance of different memory algorithms. Here,

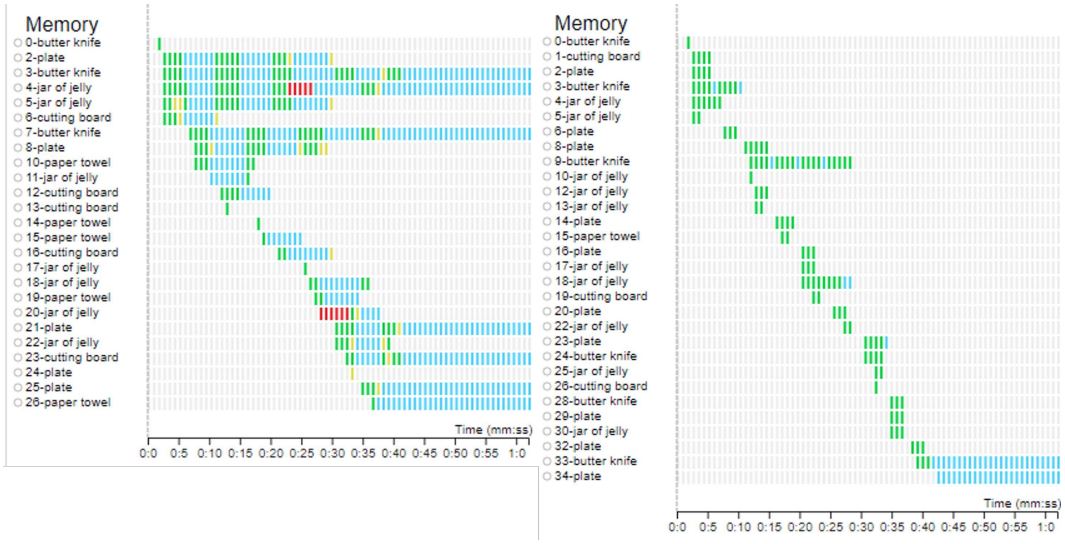


Figure 4.7: Temporal view of memory for same clip from 2 different memory algorithms.

we compare two 3D memory algorithms: one that can utilize temporal information (left) and another that can only utilize information from the last frame (right). As shown in Figure 4.7, it can be seen that the performance of the algorithms of the left is better since the lifespan of tracklets are longer, and it has fewer rows.

4.1.4 Discussion

We present visual analytic tools for 3D memory built on top of ARGUS. They empower the developer to understand and debug the 3D memory algorithm together with other machine learning modules of an AI assistant. Here, we focus on develop tools for visualizing a single recording session. One interesting future work could be visualizing the machine learning outputs across different recording sessions at the same time, allowing users to analyze and compare the trajectories of different performers doing the same task.

4.2 Instant Reality: Gaze-Contingent Perceptual Optimization for 3D Virtual Reality Streaming

4.2.1 Introductions

Thanks to the rapid growth of Internet technologies such as 5G, streaming services have unprecedentedly revolutionized how we access and consume high-quality multimedia, from listening to audios, watching videos, to scientific discovery (e.g., NASA's Eyes[1]). The edge-cloud streaming setting significantly reduces edge-side computation and redundant offline storage, allowing for portable and wearable consumer devices.

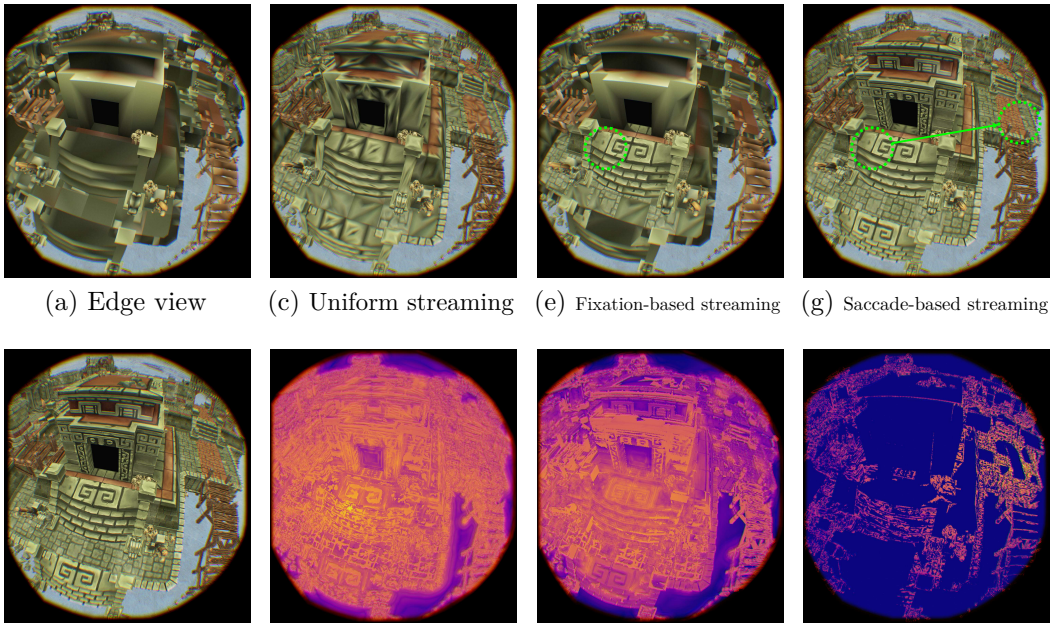
When it comes down to VR/AR, storing all the assets offline is impractical for dynamic and location-based scenarios. Thus, streaming assets from the cloud and perform the rendering on the edge-side become an effective solution. However, VR streaming poses unique challenges due to the required high resolution, high frame rate, low latency, and stereo rendering. Frame-based 2D streaming falls unrealistic given the exceptionally high demands of data volume and responsiveness. Still, current approaches fail in streaming large high-quality 3D assets, introducing unacceptably long loading times or undesirable visual and interactive artifacts (please refer to our video for examples). Comparisons of the recent 250% FLOPS gains in the last two consecutive GPU generations [2] versus a smaller growth of only 26% in global internet bandwidth during the same time [3] also suggest the main roadblock for immersive 3D experiences lies in the delayed access to 3D assets, instead of computational resource.

Perceptual mechanisms, such as foveation, have been harnessed in recent efforts to accelerate interactive rendering [38, 48]. However, the acceleration only occurs after all assets are stored at the user-end, thus improving the computation from, but not the transmission of, scene assets (especially complex geometries). Gaze-contingent effects have been used to optimize the limited network bandwidth for

[1] https://eyes.nasa.gov/

[2] https://www.digitaltrends.com/computing/nvidia-rtx-3090-vs-rtx-2080-ti-most-powerful-gaming-gpus-duke-it-out/

[3] https://blog.telegeography.com/466-tbps-the-global-internet-continues-to-expand



(a) Edge view (c) Uniform streaming (e) Fixation-based streaming (g) Saccade-based streaming

(b) Cloud data (d) Quality for (c) (f) Quality for (e) (h) Quality for (g)

Figure 4.8: Our gaze-contingent immersive 3D assets streaming interface. Starting from the partially-streamed 3D assets on the edge-side rendered at a given time (a), our method streams additional updates from the cloud to the edge for perceptually closer rendering to the full assets stored on the cloud (b). Standard uniform streaming (c) evenly updates all visible assets in the scene, causing suboptimal perceptual quality in (d), which visualizes both the temporal (popping with respect to (a)) and the spatial (quality with respect to (b)) perceptual errors. Brighter colors indicate worse artifacts. Our method, in contrast, optimizes the subset of the assets to be streamed to the edge for better spatio-temporal quality under the same network bandwidth. Our perceptual model considers both eccentricity-based acuity during fixation (e) and temporal masking during eye movement (g). If the user fixes the gaze (e), our model prioritizes regions near the gaze point (green circle) while reducing potential popping artifacts. If a saccade is detected (g), our model can safely ignore popping to stream more aggressive updates for further quality improvement (h).

2D frame-based streaming [35, 61] but not 3D assets. Existing solutions for 3D content streaming often have globally uniform granularity, such as game levels or camera distance. Finer-grained controls can be achieved based on visibility [31] or viewpoint [66] to optimize level-of-details (LoD). However, without comprehensive optimization for human perceived quality, the adaptive streaming may cause strong visual artifacts such as temporal popping or low quality with limited network bandwidth.

To this end, we propose a 3D LoD-based streaming framework that automatically schedules the data transmission priority towards optimal perceptual quality. Our method is based on modeling human spatio-temporal perception and adapts to varying network conditions. Our key idea is to estimate perceptual sensitivity based on spatio–temporal user gazes, including foveation, saccade, and popping, so that we can more aggressively optimize static image quality and dynamic frame change, such as image regions outside the visual fovea or quickly glanced over during saccade. The perceptual importance is then transferred to transmission priorities from the cloud to the edge. To further enable real-time responses without extra latency caused by the complex frequency domain computations, we also implemented a simple neural-network-based accelerator for fast importance computation. Our method is general enough to support various data formats, such as meshes, volumes, height fields, and point clouds. It targets high field-of-view and eye-tracked displays that require interactive streaming, such as cloud-based gaming. The framework is also compatible with transmitting geometric transformations in dynamic scenes, as our proof-of-concept shows in the supplementary video.

We conducted both subjective studies (via VR and traditional displays) and objective analysis based on the recorded data, which indicated higher quality and lower artifacts of our method than alternative solutions under identical bandwidth. We also tested our method under varying latencies by simulating real-world network transmission rates (from 3G to 5G).

In summary, the main contributions of this work are:

- a perceptually optimized high-quality and low-latency 3D immersive streaming framework, supporting various 3D computer graphics data formats;
- a gaze-contingent perceptual model based on foveation, saccade, and popping to depict spatio-temporal visual behaviors during procedural streaming,

including static quality acuity and dynamic change suppression;

- a neural-network-based accelerator for real-time computation of the complex perceptual mechanisms and large data volumes;
- a series of subjective studies and objective analysis validating our method under varying network conditions.

Details on the implementation can be found in the InstantReality Github repository[4].

4.2.2 Method

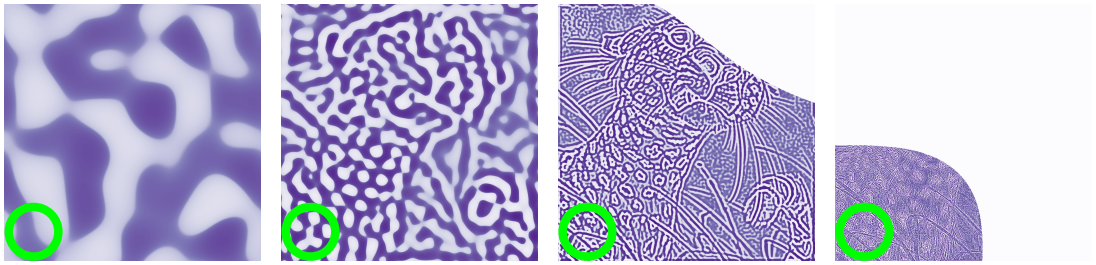
Given a limited network bandwidth and a set of scene data, we optimize and determine the streaming priority with regard to individual user's spatio-temporal perception. The priority is updated according to users' static and dynamic gaze behaviors. Our goal is to maximize perceived quality (compared with a fully local rendering) and ensure dynamic smoothness (minimize popping artifacts).

We first describe the factors in our spatio-temporal perception model (Section 4.2.2.1), followed by our method in 2D image space (Section 4.2.2.2) and 3D object space (Section 4.2.2.3). Equation (4.3) summarizes our key idea. Finally, to enable the cloud's real-time responses to users' dynamic head/gaze motions, we accelerate the system via a deep neural network (Section 4.2.2.4). At each moment, the cloud (for computation and storage) receives the gaze and existing content from the edge (for rendering and interaction). Then, guided by our gaze-contingent and perception/content-aware model, the cloud automatically determines the streaming priority of each asset component based on their added perceptual quality and data size (with regard to a given network bandwidth). Figure 4.13 shows an overview of our system.

4.2.2.1 Modeling Spatio-temporal Vision

Spatial visual acuity. The human visual acuity is foveated, with a deterioration along age [22] and visual field eccentricity [46, 51]. The importance of a pixel $E(\mathbf{x})$ is determined by its distance to the retinal eccentricity $r = \sqrt{x^2 + y^2}$. As the retinal

[4]https://github.com/chenshaoyu1995/InstantReality



(a) 4 cycle / im (b) 16 cycle / im (c) 64 cycle / im (d) 256 cycle / im

Figure 4.9: Decomposition visualization of bandpass filtered contrast multiplied by contrast sensitivity function of corresponding frequency band ($c(\mathbf{x}, \mathbf{f}_i, I)$ from Equation (4.2) and Equation (4.17) in Section 4.2.7.3). The green circle indicates gaze position. When $\mathbf{f}_i$ is low ((a) and (b)), the bandpass filtered contrast is computed over the entire visual field based on visual content. With high $\mathbf{f}_i$ ((c) and (d)), the periphery sensitivity was clamped by B_r, thus the empty content.

eccentricity r increases, the importance function $E(\mathbf{x})$ decreases. Please refer to Section 4.2.7.1 for the details of $E(\mathbf{x})$.

Static stimuli. As shown in Figure 4.9, we model the perception of a static image by separating it into a number of frequency bands. The perception for each frequency band $\mathbf{f}_i$ is the product of its bandpass-filtered contrast and its contrast sensitivity function. The foveated retinal band $B_r(\mathbf{g}, \mathbf{x}) = E(\mathbf{g} - \mathbf{x})$ will also affect the perception, results in clamped peripheral part for high frequency content. Please refer to Section 4.2.7.3 for details.

Dynamic stimuli. In runtime, the level of content LoD constantly changes and the quality is improved gradually, because more detailed data is streamed continually with the available network bandwidth. However, abrupt changes within a small time period may cause popping artifacts. Similar to [67], we define the perceived *change* as a temporally adapted Weber's contrast considering short-term memory.

Change blindness. The gaze fixations and the dynamic transitioning motions among them also significantly alter the perceptual quality. Besides motion parallax [68] and smooth pursuit movements to keep tracking on objects, people perform very

fast eye movements (a.k.a. saccades) to change among fixations. Due to the fast movement, the visual contrast sensitivity (c) is suppressed as studied in [37]. We ran a pilot study and validated the suppressed perception of the popping artifacts along with the sensitivity drop. Therefore, we adaptively model visual sensitivities w.r.t dynamic gaze behaviors including fixations and saccades, as detailed next in Section 4.2.2.2. Albert et al. [2] suggested for foveated rendering, an overall system latency of 50-70ms is acceptable when users are specifically tasked with finding artifacts, and the requirements shall be further relaxed for regular tasks. Although reading-introduced saccades can be as short as 20ms, both short- and long- peri-saccades are combined with post-saccade suppression that stretches the applicable duration to mitigate popping artifacts. Previous literature has shown that saccadic suppression may typically last for at least 100ms even with a 50ms saccade [34]. Thus, the allowable saccadic suppression durations contain modern mobile network latencies ($\sim$ 100ms for 3G) [69].

4.2.2.2 Model in Screen Space

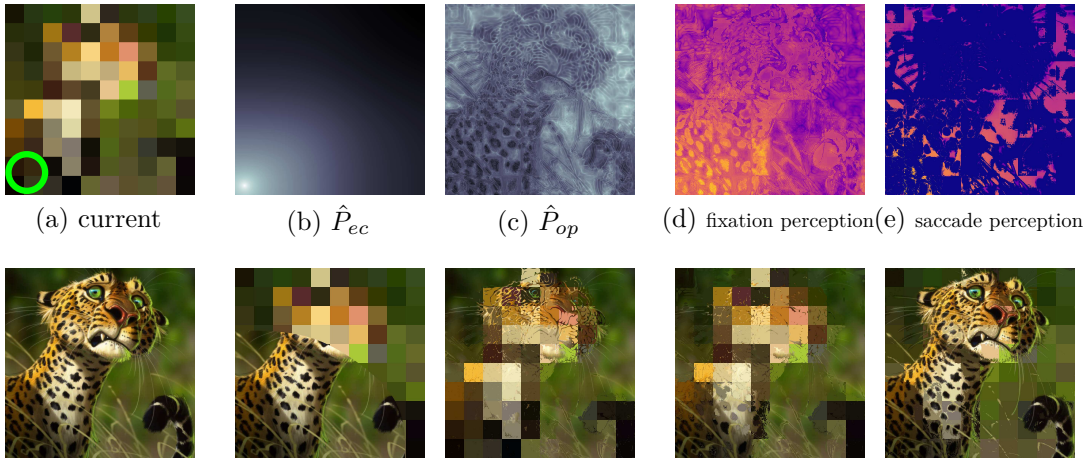
Spatial acuity and quality. Due to foveated vision, we prioritize quality and details in the fovea over the periphery. Thus, the importance of a given pixel $\mathbf{x}$ under gaze position $\mathbf{g}$ is computed as:

$$\hat{P}_{ec}(\mathbf{g}, \mathbf{x}) = E(\mathbf{g} - \mathbf{x}), \tag{4.1}$$

where E is the importance function defined in Equation (4.11) in Section 4.2.7.1.

Temporal consistency. A major problem from traditional LoD-based procedural rendering is the visual popping effect [45]. That is, when the LoD level of an area receives an update, the abrupt visual changes may easily be noticed and distractive to the experience. The human visual system perceives LoD-introduced popping artifacts in spatial frequency and retinal velocity [67].

By extending the perceived *change* as a temporally adapted Weber's contrast considering short-term memory to individual frequency band, we obtain the approximated popping (i.e., perceived temporal intensities) between two varied frames (I



(a) current (b) $\hat{P}_{ec}$ (c) $\hat{P}_{op}$ (d) fixation perception (e) saccade perception

(f) target (g) $\hat{P}_{ec}$-only update (h) $\hat{P}_{op}$-only update (i) fixation update (j) saccade update

Figure 4.10: Visualization of our perceptual model in 2D screen space. Here each pixel is treated as a unit with its own LOD. (a) is a semi-transmitted low LoD image on the edge and the green circle indicates the gaze. (f) is the full LoD target on the cloud. (b) and (c) visualize the importance of the eccentricity in Equation (4.1) and the temporal consistency in Equation (4.2). Our model exploits both eccentricity and temporal consistency in Equation (4.3). (d) and (e) visualize the corresponding perceptual quality measures considering both static visual quality and dynamic consistency. The second row shows the corresponding screen space update.

and I' in the screen space):

$$\hat{P}_{op}(\mathbf{g}, I, I', \mathbf{x}) = \sum_{i=0}^{b-1} s\left(\mathbf{f}_i, L\right) \times \frac{|c(\mathbf{x}, \mathbf{f}_i, I) - c(\mathbf{x}, \mathbf{f}_i, I')|}{|c(\mathbf{x}, \mathbf{f}_i, I)| + \omega}, \tag{4.2}$$

where $\mathbf{g}$ is the tracked gaze positions in I and I', respectively. ω is the controlling parameter that balances for low-intensity stimuli where Weber's law may fail. s is the contrast sensitivity function given the frequency band $\mathbf{f}_i$ and luminance L, as detailed in Equation (4.12) under Section 4.2.7.2. The transition from I to I' can be from the changes of scene or camera. The corresponding coefficients are omitted here for the brevity of presentation. Although the spatial contrast sensitivity function is not reflecting the sensitivity of the human visual system to temporal changes, we have modeled the sensitivity temporal changes by introducing Weber's law on two temporally adjacent frames I and I'. c is the bandpass filtered point contrast given the frequency band $\mathbf{f}_i$, as detailed in Section 4.2.7.3. It assumes

slow gaze/head motion with fast (90FPS) frame update. Thus, $\mathbf{g}$ in the two frames are approximately identical. The next paragraph discusses cases when the gaze moves fast (saccade).

Adapting to dynamic gaze behaviors. As studied by [37], our visual sensitivity gets significantly suppressed during saccades. Due to the change blindness, we only perceive weak popping artifacts in this period. Thus, when saccades are detected, we can instead stream the most noticeable popping elements to reduce the popping intensity after it lands. This motivates our gaze-behavior-adaptive per-pixel sensitivity by combining both spatial acuity and temporal consistency models:

$$\hat{P}(\mathbf{g}, I, I', \mathbf{x}) = \begin{cases} \hat{P}_{ec}(\mathbf{g}, \mathbf{x}) - \lambda \hat{P}_{op}(\mathbf{g}, I, I', \mathbf{x}) & \text{during fixation} \\ \int_{\mathbf{g}' \in I'} \hat{P}_{op}(\mathbf{g}', I, I', \mathbf{x}) \mathrm{d}\mathbf{g}' & \text{during saccade} \end{cases}, \tag{4.3}$$

where λ is the balance between maximizing foveated perceptual quality and minimizing popping artifacts during fixation. A saccade is considered to happen if the gaze speed is greater than 180 deg/sec. Because of the open challenge in real-time saccade landing prediction [3], we integrate over the entire visual field while computing the popping for saccade instead of assuming the landing position. This ensures global robustness to any user's attentional changes. Figure 4.10 visualizes individual importance and the resulting image-space update.

We can adapt Equation (4.3) for progressive LoD update from level $i+j$ to i as follows:

$$\hat{P}(\mathbf{g}, I_{i+j}, I_i, \mathbf{x}) = \sum_{l=i}^{i+j-1} \hat{P}(\mathbf{g}, I_{l+1}, I_l, \mathbf{x}), \tag{4.4}$$

where I_i represents the image at the i-th LoD.

4.2.2.3 Mapping from 2D Screen to 3D Streaming

In this section, we present how to extend the screen-space model from Section 4.2.2.2 to 3D assets for a real-world cloud-based 3D streaming system. The 3D assets can have various representations, including triangle meshes, volumes,

terrains, or large crowded objects, as shown in Figure 4.12.

Map to 3D assets. So far, our perceptual model depicting static quality and dynamic artifacts in Section 4.2.2.2 applies to individual pixels. The 3D assets, however, comprise non-uniformly distributed content such as depths and connectivities. We apply a deferred shading algorithm to convert various types of 3D primitives to 2D perception evaluations.

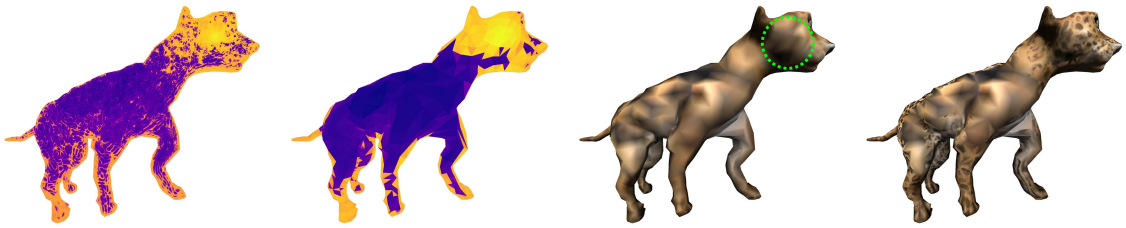
We divide the 3D content based on the coarsest level of LoD. We denote an individual computational unit as U_i, where i is its index among all units. It can be a coarsest triangle in a 3D mesh, a largest super-voxel in a volume, a texel in the coarsest mipmap level of a height/displacement texture, or a separate object in a swarm scene. We denote the LoD level of U_i at time frame t as $L_{U_i,t}$.

At time frame $t-1$ when the LoD levels of all units are determined and transmitted to edge already, we render a framebuffer at cloud side for the whole scene without anti-aliasing to retrieve the unit indices of every pixel, i.e., a mapping $M_{t-1} : \{\mathbf{x}\} \longmapsto \{U_i\}$ from the set of pixels $\{\mathbf{x}\}$ to the set of units $\{U_i\}$.

If the LoD level of U_i is updated to $L_{U_i,t}$ at time frame t, we approximate its sensitivity by accumulating all pixels $M_{t-1}(\mathbf{x}) = U_i$ of the unit at time frame $t-1$,

$$\begin{aligned} &\hat{P}_{U_i,t}\left(L_{U_i,t}, \mathbf{g}_{t-1}, M_{t-1}\right) \\ &\approx \sum_{\mathbf{x} \in M_{t-1}^{-1}(U_i)} \hat{P}\left(\mathbf{g}_{t-1}, I_{t-1}, \hat{I}_t \middle| L_{U_i,t}, \mathbf{x}\right), \end{aligned} \tag{4.5}$$

where $\mathbf{g}_{t-1}$, M_{t-1}, and I_{t-1} are the gaze position, unit mapping, and render image at time frame $t-1$. Figure 4.11 visualizes the mapping. The mapping M implicitly represents the camera of each time frame. It also varies according to the LoD of all units. The approximation in Equation (4.5) simplified the evaluation by assuming $M_{t-1} = M_t$. $\hat{I}_t \middle| L_{U_i,t}$ is the render image at time frame t if the LoD level of U_i is $L_{U_i,t}$. We denote it with a hat because it is an approximation by assuming the LoD level of other units is not changed between time frames of $t-1$ and t. $\hat{P}_{U_i,t}$ is also dependent on the LoD levels of other units $\left\{L_{U_j,t-1} \middle| j \neq i\right\}$, which we omit in Equation (4.5) for brevity. The evaluation of $\hat{P}_{U_i,t}$ in Equation (4.5) is computationally expensive, so we propose a neural evaluation, as will be detailed in Section 4.2.2.4.



(a) 2D heatmap (b) 3D heatmap (c) before (d) update

Figure 4.11: Visualization of mapping importance from 2D pixels to 3D triangles. (a) shows the gaze-aware importance $\hat{P}$ in 2D. (b) shows mapped importance in 3D. (c) and (d) show the object before and after streaming, guided by (b). The green circle indicates the user's gaze fixation position.

Cloud-based streaming. Finally, we evaluate the perceptual quality *per bit* by updating the LoD level of a unit,

$$W_{U_i,t}(L_{U_i,t}) = \frac{\hat{P}_{U_i,t}\left(L_{U_i,t}, \mathbf{g}_{t-1}, M_{t-1}\right)}{D_{U_i}(L_{U_i,t-1}, L_{U_i,t})}, \tag{4.6}$$

where D is the data volume difference by updating U_i's LoD level from $L_{U_i,t-1}$ at time $t-1$ to $L_{U_i,t}$ at time t.

The edge side is a device with limited network bandwidth and storage. Therefore, the 3D scene is fully stored on cloud side, including all LoD levels of all units. In runtime at time frame $t-1$, the edge side sends the gaze position $\mathbf{g}_{t-1}$. The cloud side knows the LoD levels of all units on the edge side as the edge side acknowledges the received network packages containing the 3D data. According to that, the cloud will compute the perceptual quality per bits for updating LoD levels of every unit accordingly, with the updates holding highest Ws to achieve the best improvement of perceptual quality, given the constraint on the available update package size S.

$$\underset{\{L_{U_i,t}\}}{\operatorname{argmax}} \sum_i W_{U_i,t}(L_{U_i,t}), s.t. \sum_i D_{U_i}(L_{U_i,t-1}, L_{U_i,t}) \leq S, \tag{4.7}$$

4.2.2.4 Neural Acceleration

In a practical implementation, the cloud needs to compute Equation (4.5) for each update. Although the cloud can afford more computation than the edge,

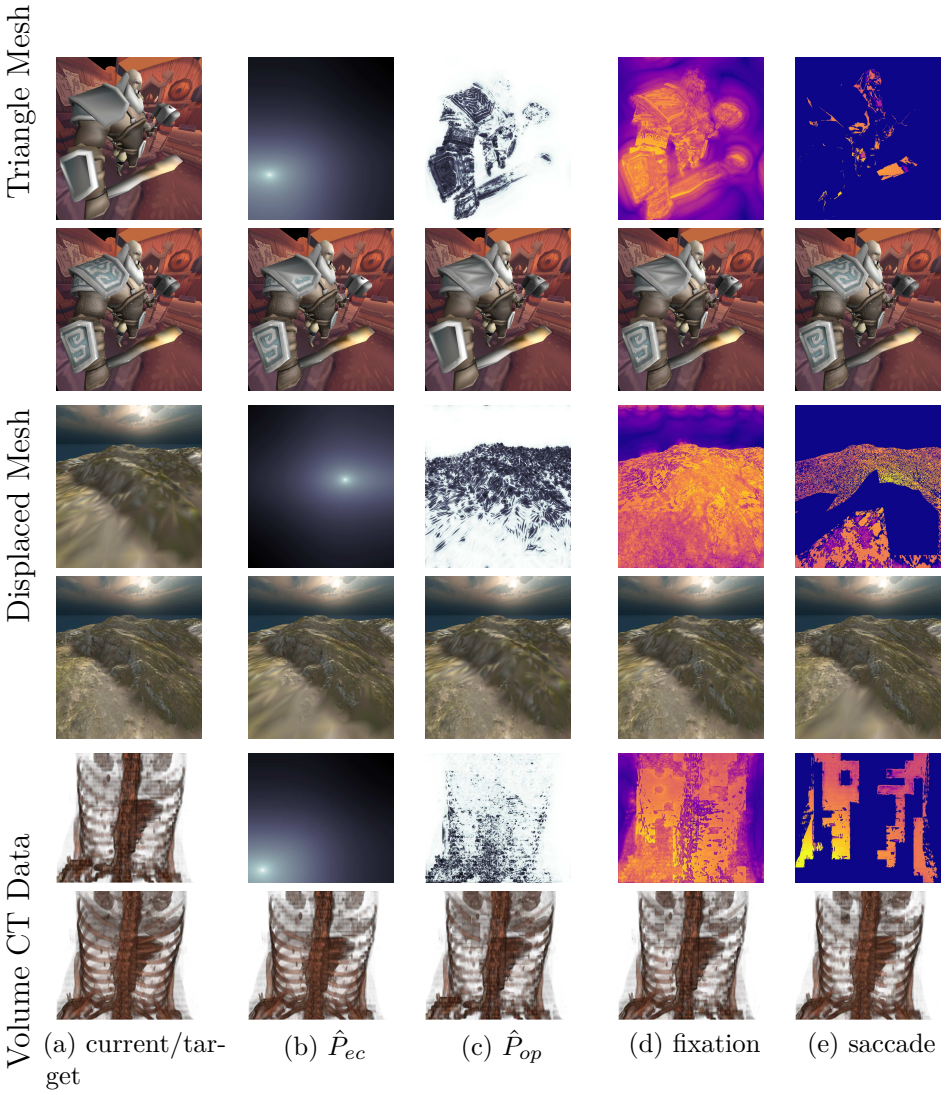


Figure 4.12: Results for various 3D data types. From top to bottom, the 3D assets are triangle mesh, displacement-based terrain, and CT scanned volume. In each group of images, (a) shows the semi-transmitted scenes at edge on top and full scenes at cloud on bottom. (b)/(c) is the importance and corresponding updated rendering by considering only eccentricity $\hat{P}_{ec}$/temporal consistency $\hat{P}_{op}$. Brighter means higher importance. (d)/(e) is the update if the gaze fixes/saccades considering both (b)$\hat{P}_{ec}$ and (c)$\hat{P}_{op}$. Similar to Figure 4.8, their first rows indicate estimated perceptual quality (brighter means stronger spatio-temporal artifacts or worse quality).

the heavy frequency domain decomposition for individual LoD and frames in Equation (4.16) in Section 4.2.7.3 may cause intolerable latencies on the end users. Thus, for each scene, we train a multilayer perceptron (MLP) neural network for the fast prediction of $\hat{P}_{U_i}$.

Specifically, with a camera view $\mathbf{c}_{t-1}$ and gaze position $\mathbf{g}_{t-1}$ as input, the network learns to compute

$$\mathcal{N}(\mathbf{c}_{t-1}, \mathbf{g}_{t-1}) = \left\{ \hat{P}_{U_i} \left(L_{U_i,t}, \mathbf{g}_{t-1}, M_{t-1} \right) \right\} \tag{4.8}$$

for all U_i and possible $L_{U_i,t}$. The camera view $\mathbf{c}_{t-1}$ contains 3 vectors, which are coordinates of camera position, camera direction and up direction. In the input, there is also a flag indicating whether a saccade is detected. The current LOD state is not in the input since the network always predicts for all LODs, which will be used by the cloud to make the streaming decision. Although we do not provide the mapping M_{t-1} explicitly to the network, the network also infers it from $\mathbf{c}_{t-1}$ once trained on a specific scene. The diversity of projected areas of various units U_i cause $\hat{P}_{U_i}$ to have no upper bound, making it difficult to optimize the network. To combat this, we normalize $\hat{P}_{U_i}$ by the pixel counts of individual $\hat{P}_{U_i}$ in the projected screen space. After normalization, the value is bounded by the maximum value of Equation (4.2). The cloud stores the network and retrieves the fast-inferred $\hat{P}_{U_i}$ for streaming decision (Equations (4.6) and (4.7)), so the cloud doesn't need to render the actual image for the decision.

We report specific implementation details such as network architecture, loss function, and dataset generation in Section 4.2.3 and the performance/precision analysis in Section 4.2.5.2. Please also refer to our code repository for the actual implementation of the network structure. We evaluate both performance gains and prediction precision of our neural network in Section 4.2.5.2. Change the super parameters like the ω will require retrain the neural network. To avoid retraining the network when changing the λ, one solution is to train 2 networks to predict the $\hat{P}_{ec}$ and $\hat{P}_{op}$ separately.

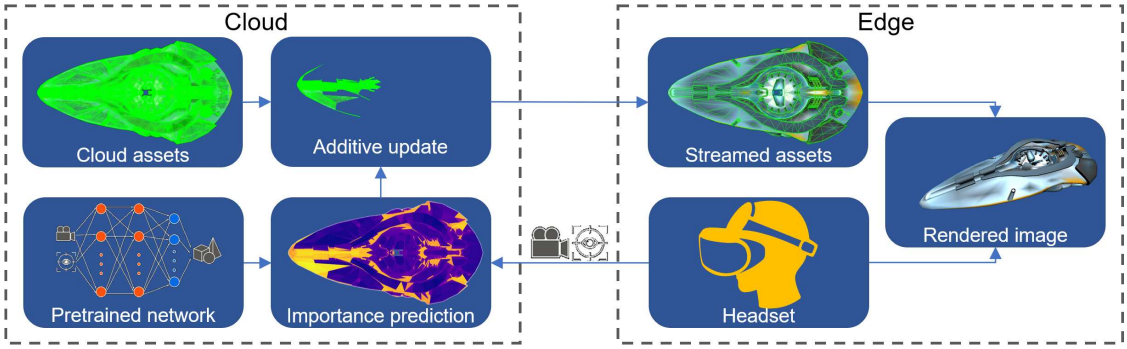


Figure 4.13: Overview of the system. The parameters of the camera and gaze are sent to cloud, and they are used as the input of the pretrained neural network to predict the perceptual importance of primitives. The image is rendered on edge with the streamed assets.

4.2.3 Implementation

3D Assets. Our current method assumes we can set the LoD of each U independently from one another. While this works for asset types with more independent units such as point clouds [63, 66], volumes, and crowd agents, it might introduce artifacts for others, such as cracks or T-junctions for triangle meshes. To ensure quality without introducing complex implementation, we currently use vertex colors instead of textures, and build the mesh hierarchies by sub-sampling existing vertices without changing their positions. The displaced mesh (terrain scene from Figure 4.12) is handled in a similar way, except that additionally a height map is also sampled for vertices. For volume data, the LOD is created by sub-sampling the voxels.

Solving Equation (4.7) is equivalent to the knapsack problem, which is NP-Complete. For this reason, we use a greedy approximation, which always selects the unit with the largest $W_{U_i,t}(L_{U_i,t})$ in Equation (4.6) if the bandwidth allows so.

System. We simulate the streaming network via ZeroMQ library (`https://zeromq.org/`). The rendering system was implemented with OpenGL. The system runs on a PC with Intel i9-9880H 8-core CPU and with 32GB of RAM, and an NVIDIA RTX 2080 graphics card. For the VR headset, we use an HTC Vive Pro Eye with an integrated eye tracker. It has a 110 degree FoV and 1440×1600 spatial resolution (14 pixels per degree, i.e., 7 cycles per degree for B_d). Our system runs

at 90-FPS with neural acceleration.

Neural acceleration. We prepare the dataset for the neural network in Section 4.2.2.4 by first sampling short sequences of camera and/or gaze movements inside a 3D scene at 90-FPS. The views are freely controlled by the HMD users. Then the offline computed (via Equation (4.5)) $\{(\mathbf{c}_{t-1}, \mathbf{g}_{t-1}), \hat{P}_{U_i}\left(L_{U_i,t}, \mathbf{g}_{t-1}, M_{t-1}\right)\}$ pairs are used to train the network. Since the adjacent frames are likely to have similar data, we sample 4 frames per second uniformly and compute the ground truth $\hat{P}_{U_i}$ for all U_i and LOD levels offline. There are 16 sequences, and each sequence lasts for about 30 seconds and contains ~ 120 samples. We use 15 sequences as the training set and the remaining one as the test set.

The neural network is fully-connected and consists of 3 inner-layers with $(100, 1000, 1000)$ neurons, each with $ReLU$ activations, and an output layer with a *Sigmoid* activation. The network was trained with *L1* loss for 100 epochs. In the training, we set the learning rate to 0.001 and batch size to 128. we used an SGD optimizer with momentum = 0.9. For the neural network in the user study (Section 4.2.4.1), since we have ~ 2000 samples and each sample contains 722 (triangles) $\times 4$ (LOD) ≈ 3000 scores. The sample size, higher than the parameter numbers, avoids biased overfitting. We refer the readers to our open source implementation for reproduction details.

Parameters. We determine the optimal spatio-temporal balancing λ via a pilot user study. Users reported 3.0 as the most plausible experience. In our experiment, the Weber's law adjustment for low intensities ω was set to 10. With low network bandwidths, the peripheral content may not receive the priority of transmission. Thus, we included a pedestal constant 2.0 to the $\hat{P}_{ec}$ for each pixel.

4.2.4 Evaluation: User Study

We compare different progressive streaming methods with different usages of perceptual mechanisms under the same network bandwidth - uniformly increasing the LoD of each U based on visibility (**UNI**), "foveated" streaming without considering dynamic consistency (**ECC**, a 3D version of [61]), and our method (**OURS**). We experimented with two display environments, active HMD-based 6 DoF (degrees

(a) VR setting (b) Video setting

Figure 4.14: The setting of our user study. Both (a) and (b) were captured from parts of our user studies.

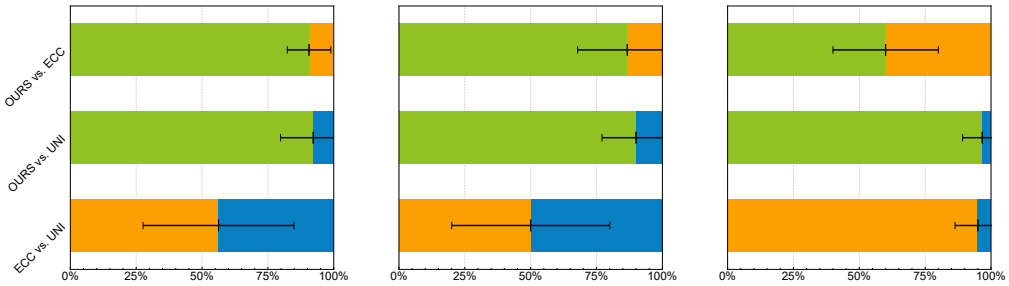
of freedom) navigation (Section 4.2.4.1) for temporal consistency evaluation only and passive screen-based observation (Section 4.2.4.2) for evaluation of both visual temporal consistency and static visual quality.

4.2.4.1 Eye-Tracked Study

We first conducted a user study with an eye-tracked VR HMD to assess our method's effectiveness on perceived temporal consistency.

Stimulus and setup. We used a terrain displacement scene from Figure 4.12 in this experiment. Based on users' real-time head and gaze motions, the stimuli is a sequence of simulated progressive streaming updates (from a coarse to adaptively finer level). During the study, users wore an HTC Vive Pro Eye Headset and remained seated. They were instructed to freely observe the virtual scene and keep their fingers on the keyboard to make selections after each trial, as shown in Figure 4.14a. Eight users (age $23-31$, 1 female) participated in the study. To accommodate the scene data size and study duration, we experimented with a simulated 5G network bandwidth locally.

Task. Due to the limited access to participants during COVID-19, we aim to maximize trial randomness and numbers to ensure objective measurements. We designed a two-alternative-forced-choice (2AFC) experiment. Each trial consists of a pair of conditions among the three (**UNI**/**ECC**/**OURS**). In each condition, the stimulus was initiated with the coarsest LoD. The participants were then



(a) VR eye-tracked temporal consistency (b) video-based temporal consistency (c) video-based quality

Figure 4.15: User study results. We compare pair-wise 2AFC user vote percentage in both HMD and traditional display studies from Section 4.2.4. The green/orange/blue bar indicates the percentage voting for **OURS**/**ECC**/**UNI** and the errorbars indicates the standard deviations.

instructed to freely observe the environment that was streamed and updated with the corresponding method. Each condition lasted for 10 seconds. At the end of each trial, the participants used the keyboard to select which one of the two conditions appeared more smoothly and comfortably updated with fewer artifacts over the entire duration.

Each experiment began with a warm-up session to familiarize the participants with the interface and task, followed by 24 counter-balanced and randomly ordered trials. Thus, each pair of conditions was evaluated 8 times per participant. To minimize fatigue, a 2-second break was enforced between trials.

Results. Figure 4.15a plots the results of the pair-wise percentages that participants indicated as the preferred method. Specifically, **OURS** was voted as significantly more preferred than both **ECC** ($90.6\% \pm 8.3\%$) and **UNI** ($92.2\% \pm 12.4\%$). With a significance level established at $p = 0.05$ and power $> .999$, binomial tests indicate the difference is significant for both comparisons ($p < .001$). **ECC** showed marginally higher voting rate than **UNI** ($56.3\% \pm 28.6\%$). We did not observe significance with a binomial test ($p = .38$).

Discussion. The results above showed our method's significant temporal quality preference over the alternative solutions: under the same network conditions, our method could stream 3D assets more smoothly by reducing popping artifacts.

However, this shall be achieved without compromising spatial-visual quality. Our initial study design considered several evaluation metrics, including visual fidelity to the reference, temporal smoothness, and perceptual comfort. We planned to evaluate them both individually and collectively: ask participants to focus on individual metric (such as visual fidelity or temporal smoothness only), and additional post-study comments (such as why they choose one condition over another).

However, during our pilot runs, we noticed several problems with this initial design. Some participants commented that it was difficult for them to focus on more than one aspect of the evaluation criteria, especially for those with less VR or gaming experiences. Meanwhile, visual quality depends on the temporal integration of multiple instead of individual frames. The integration also depends on the exact view path, which can be freely moved by the participants and thus may differ across pairs of conditions for a valid comparison. To address these issues, we decided to focus on only temporal smoothness, which tends to be less view-path dependent than spatial quality, for this free navigation HMD experiment, and leave per-frame visual quality evaluation by fixing the gazes, as described in the following experiment.

4.2.4.2 Screen-Based Study

Due to the aforementioned design challenges and extra difficulties of physically distributing the eye-tracked VR display, we further simulate the study procedure in Section 4.2.4.1 to a remote setting with recorded stimuli. This ensures a thorough evaluation of both temporal smoothness and spatial quality.

Specifically, we recorded videos by randomly pre-defined gaze paths. During the study, users were remotely monitored and instructed to keep one eye open, fix their head positions, and gaze on the green cross and follow the same procedure as Section 4.2.4.1.

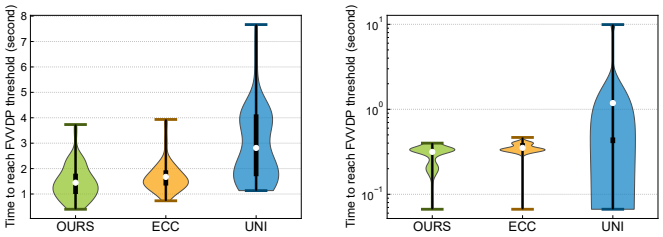
Stimulus and setup. To evaluate a different data type from the eye-tracked study, we chose the triangle mesh scene in Figure 4.12 for this study. The visual stimuli were rendered with 1920×1080 resolution and 60 degree of vertical field of view, so that the B_d is the same for every participant. In the monitored remote

study, participants first input their computer resolution and screen size. Our study protocol would automatically compute and inform participants of the correct eye-display distance. The user then fixed their head at the reported distance cross accordingly. Twelve users (age $24-41$, 5 females) participated in this study. All users had normal or corrected-to-normal vision.

Tasks. The experiment was conducted in two phases: video-based stimuli for temporal smoothness and static frame-based stimuli for visual quality. The task of the *first* phase was similar to Section 4.2.4.1. The users observed two sequentially played stimuli from the three conditions. After each trial, they selected the one with preferred temporal consistency (i.e., fewer artifacts). In the *second* phase, each trial consisted of two static images randomly sampled from the sequences in the first phase at the same timestamp. In each trial, a full-quality rendering (second row of Figure 4.12a) was first shown for 3 seconds as the quality reference. Then, 2 images of the 3 conditions were sequentially displayed for 3 seconds. After each trial, the participants selected the one that appeared to be closer to the reference in visual quality. Both tasks contain 15 counter-balanced and randomly ordered trials. Thus, each pair of conditions was evaluated 5 times per participant.

Task Design Rationale. Fixing gaze in 2AFC experiment has been practiced in assessing retinal acuity and perceived visual quality in VR [51, 72, 75]. In fact, the human visual perception is limited during natural, active viewing conditions [15]. Further, because of the suppressed vision during saccades, the visual quality is primarily determined by the frames at gaze fixations. For this reason, the current experiment where the gaze is fixed may generally represent free viewing conditions where fixations are connected by saccades.

Results. Figures 4.15b and 4.15c plot the results. For temporal consistency (phase-1), a consistent trend to Section 4.2.4.1 was observed: **OURS** had a significant higher ration of voting over the alternatives ($86.7\% \pm 18.8\%$ against **ECC** and $90.0\% \pm 12.9\%$ against **UNI**). With a significance level established at $p = 0.05$ and power $> .999$, binomial tests indicate the difference is significant for both comparisons ($p < .001$). **ECC**, however, did not show major preference gain over **UNI** ($50.0\% \pm 30\%, p = 1.0$).



(a) static scene (b) dynamic scene

Figure 4.16: The statistics of the time FovVideoVDP first reaches threshold per sequence among all users (Section 4.2.5.1). The sub-figure (a)/(b) shows the results with static/dynamic scenes. The violin plot shows mean values, 1st/3rd quartiles, and whiskers min/max. Lower values indicate better streaming efficiency in additively improving the quality.

On the extended visual quality (phase-2), marginally higher voting rate on **OURS** was observed over **ECC** ($60.0\% \pm 20\%, p = .16$). On the other hand, with a significance level established at $p = 0.05$ and power $> .999$, both **OURS** and **ECC** were voted to have significantly improved quality than **UNI** ($96.7\% \pm 7.4\%$ and $95.0\% \pm 8.7\%$ respectively, $p < .001$ for both).

Discussion. This study simulates and extends Section 4.2.4.1 to a remote screen-based setting. Beyond the agreeing observation of temporal consistency, the phase-2 study also indicated that **OURS** preserves the high visual fidelity as **ECC**. The experiments with both eye-tracked VR and traditional monitors show that our method significantly improves the perceived temporal consistency (i.e., minimizing popping artifacts) without compromising the visual quality.

4.2.5 Evaluation: Objective Analysis

4.2.5.1 Visual Quality

Section 4.2.4 shows our significant benefits of subjective quality judgement in terms of temporal consistency and artifact reduction. For further objective evaluation considering dynamic gaze adaptation stimuli (Equation (4.3)), we conducted a series of analytical experiments.

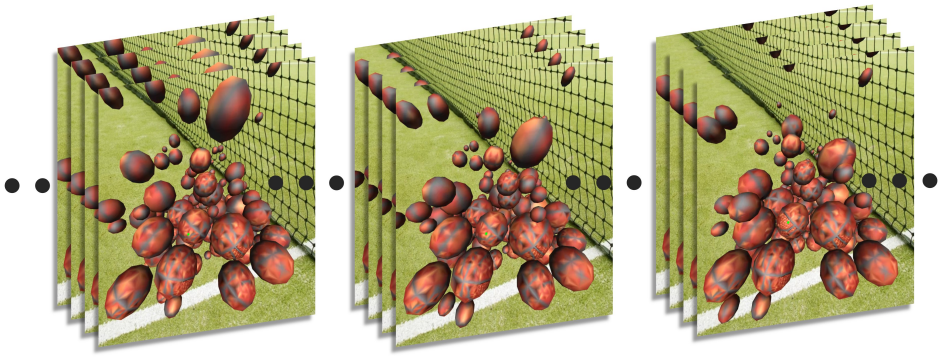


Figure 4.17: Example frames of the dynamic scene. The animation is composed of free-falling soft balls. Each image represents a frame along time. Please refer to the supplementary video for the full visualization.

Static scene. Recently, a thoroughly considered metric, FovVideoVDP [46], was proposed to measure video quality considering both spatio-temporal and foveated effects. We compute the FovVideoVDP with rendering on the fully transmitted asset as a reference. The analysis comprises full 10-second sequences from each user (one sample from each tri-condition group, 8×8 in total) of the eye-tracked study. Since edge-cloud streaming is a temporally procedural quality enhancement, we measure the timing when FovVideoVDP reaches a shared threshold. Intuitively, this measures the efficiency of the streaming achieving high quality. In the experiment, we chose the threshold as 6.5 to ensure all conditions, even the slowest, can reach it with the limited trial durations. As shown in Figure 4.16a, the average time of **OURS** (1.446 ± 0.669) and **ECC** (1.674 ± 0.589) are significantly shorter than **UNI** (2.813 ± 1.429). Pairwise t-tests show the difference between all pairs are significant ($p < .001$ for **OURS** vs. **UNI** and **ECC** vs. **UNI**, $p = .032$ for **OURS** vs. **ECC**).

Dynamic scene. We further preformed an experiment on a dynamic scene which consists of an animation of a pile of falling soft balls as shown in Figure 4.17. We refer the complementary video for full animation. The gaze is from the eye-tracked study with the same setting as the static scene study, except that the dynamic assets and threshold as 6.7. As shown in Figure 4.16b, the average time of **OURS** (0.316 ± 0.069) and **ECC** (0.352 ± 0.049) are significantly shorter than **UNI** (1.184 ± 2.548). Pairwise t-tests show the difference between all pairs are

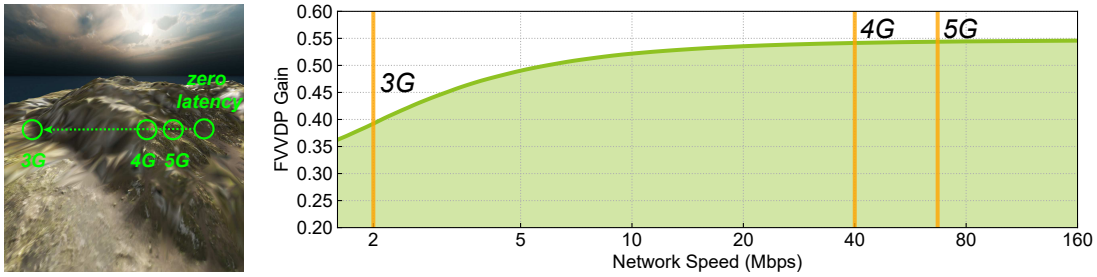
significant ($p = .008$ for **OURS** vs. **UNI**, $p = .011$ for **ECC** vs. **UNI**, and $p < .001$ for **OURS** vs. **ECC**).

The above analysis on the eye-tracked gaze data revealed a consistent observation to the subjective studies in Section 4.2.4. That is, **OURS** and **ECC** deliver high visual quality more efficiently than **UNI**. Meanwhile, comparing with **ECC**, **OURS** does not compromise the visual quality with the benefits of improved temporal consistency.

4.2.5.2 System Performance

Performance gain from neural acceleration. We conducted a performance analysis of the neural acceleration effectiveness. With all the scenes and gaze trajectories, it takes an average of 389.28 ± 14.44 ms without the neural acceleration. In comparison, the neural network accelerates the computation to 20.30 ± 0.89 ms.

Cloud/edge latency. Large-scale (e.g., 3D) data transmission inevitably introduces latency between the cloud and the edge due to the underlying network. That is, the gaze analyzed in the cloud may be outdated when the edge receives the additive asset. To validate the impact under real-world network conditions, we conduct a simulated experiment with varied transmission speeds ranging from 3G to 5G+. With a recorded gaze motion trajectory, we compute the FoVVideoVDP gain between **OURS** and **UNI**. By assuming 100KB packets, the approximated network speeds for 3G (∼2Mbps) / 4G (∼40Mbps) / 5G (∼67Mbps) are referred from [54, 60]. As seen in Figure 4.18, **OURS** shows significantly elevated spatio-temporal quality (i.e., higher FVVDP) compared with **UNI** under all network conditions. The elevation slightly decreases when network speed is slower than 10Mbps. The analysis demonstrates our approach's consistent outperform than **UNI** under both modern (4G and 5G) and legacy (3G - 4G) network conditions. Using the same data, we further perform a pressure experiment with artificially created latencies (from e.g., pings, database retrieval, etc.) with 3G/4G/5G network speeds. Figure 4.19 shows the result. Our method is only subtly affected by latencies across all conditions. Under 4G and 5G network, **OURS** can reach 0.5 in JODs compared with **UNI** when the latency is lower than 100ms. The 4G and 5G results are almost identical because their speeds are close.



(a) illustration (b) quality gain w.r.t. bandwidth

Figure 4.18: Extendability test with varied network bandwidths. (a) shows a gaze trajectory from right to left, starting at the position circled by *zero latency*. The 3G/4G/5G circles indicate the gaze positions for which the required data packets arrive. (b) shows the FoVVideoVDP gain of **OURS** from **UNI** along with varied simulated network conditions. The significant quality elevation in **OURS** is only subtly affected by gaze transmission latency when the speed is lower than 10Mbps.

Approximation errors. Unlike 2D frame streaming, our system faces challenges of the complex 3D structures including connectivity and sparsity. Although our method analytically transforms screen-space perceptual models to 3D (Section 4.2.2.3) with neural acceleration (Section 4.2.2.4), each of them may introduce numerical loss. In this section, we also analyzed the potential error. Specifically, we compute the MSE loss among the image-based per-pixel update, per-triangle update with/without neural acceleration. Figure 4.20 visualizes an example sequence. The experiments showed that the loss is relatively minor and worth the performance gain enabled by the neural network.

4.2.6 Discussion

We present a method for high-quality 3D immersive streaming with gaze-contingent perceptual optimization. Compared with 2D frame-based streaming systems, our 3D streaming method enables low-latency interaction, low cloud overload, and consistent viewing. Our evaluation demonstrates that our system delivers a statistically significant reduction of temporal artifacts without compromising the visual quality.

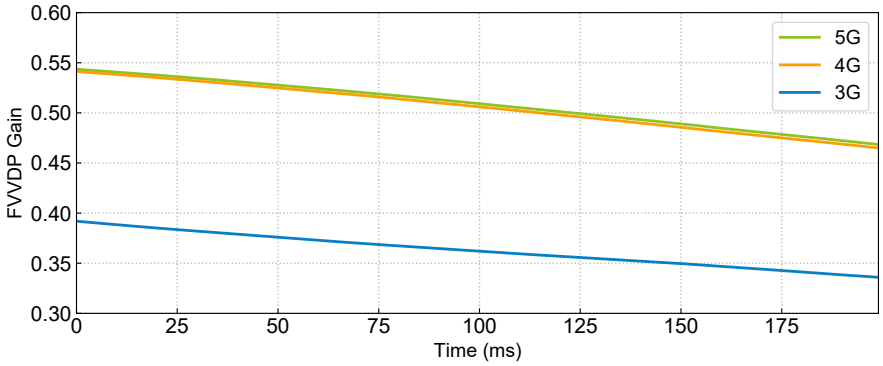


Figure 4.19: Pressure test with artificially introduced network latencies. X-/Y-axis indicates the introduced latency/corresponding average FVVDP values. Across all network conditions, the low FVVDP changes demonstrate our method's robust benefits under real-world scenarios.

Limitations and Future Work. The two main parts of our method, modeling human perception for visual importance in 2D, and mapping the 2D importance to 3D for rendering and streaming, are largely orthogonal, as various perceptual factors can be considered for the 2D importance, before mapping it to 3D (Section 4.2.2.3). This work focuses on combining foveation and saccade as the main perceptual mechanisms to optimize spatial quality and temporal smoothness (Equation (4.3)). Other possibilities include applying other perceptual phenomena to compute 2D importance, such as color, saliency/attention [30, 79], and masking [23, 27, 39], as well as applying our 2D importance estimation for streaming 2D and 360-degree videos [17]. The modeling, implementation, and evaluation of all these possibilities are promising future works, beyond the scope of a single paper.

In our system and analysis, we presumed an ideal and stable network environment without bandwidth sharing. Moreover, the data transmission was considered as individual packages of 100KB without compression and decomposition between the cloud and the edge [16, 40]. As shown in Section 4.2.5.2, all these factors may introduce additional latencies of the timely gaze tracking data sharing between the cloud and the edge, causing approximation errors. There are several super parameters in our method like the spatio-temporal balancing λ and the Weber's law adjustment for low intensities ω. One interesting future work is to adaptively optimize these super parameters for each scene.

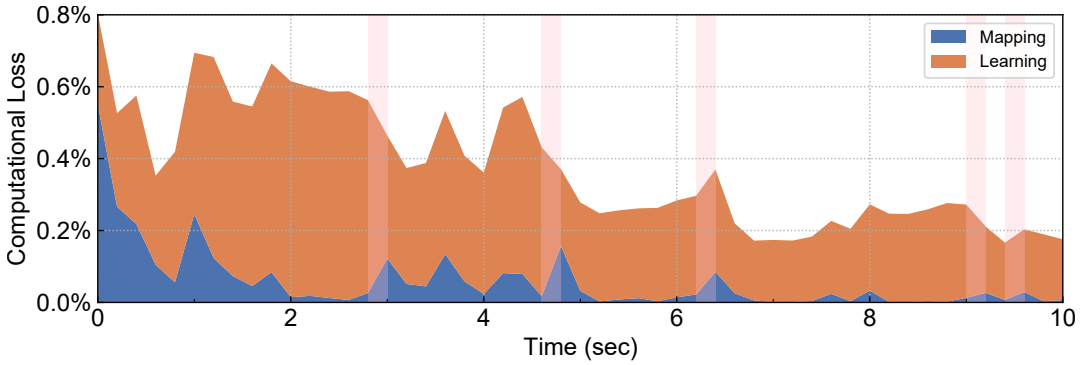


Figure 4.20: Approximation error visualization. Areas highlighted by pink indicates that there is a saccade. Due to the 2D to 3D mapping (Section 4.2.2.3, blue area) and neural acceleration (Section 4.2.2.4, orange area), the final importance prediction may contain approximation errors. However, their overall error is below 1%.

The neural network's prediction precision is shown to support local 6DoF motions, as shown in Section 4.2.5.2. However, its capability of supporting very large motions such as in a flight simulation may require larger training data volume. Similarly, they are currently trained with static scenes. Predicting high accuracy results would also increase data samples due to introducing time as an additional input dimension.

To address the quality drop due to the gaze information latency, we foresee gaze motion prediction with machine learning may shed light on further reducing the perceptual latency in low bandwidth and unstable networks. In our current method, the rendering updates immediately upon receiving the relevant assets. Our framework *mitigates* the perceived flickering. For instance, during saccadic movement, we reduce the future (after the gaze lands) perceived flickering by taking advantage of the change blindness. However, targeting at progressive streaming systems, the temporal artifacts are not to be completely eliminated. Instead, a locally adaptive rendering may increase temporal consistency at the cost of delayed full quality. Recent research on balancing quality and latency [41] could make the system adaptable to various users.

One important application scenario that may require 3D assets (instead of 2D frames) is AR. High resolution and high FoV AR displays are yet to be available. Thus, we conducted the experiments with VR. In the future, considering not only

virtual assets but also its interaction with physical surroundings may inspire more efficient transmission.

4.2.7 Appendix

4.2.7.1 Spatial Visual Acuity

The human vision is foveated. Watson [78] proposed a formula that approximates midget ganglion cells density as a function of retinal eccentricity $r = \sqrt{x^2 + y^2}$, where x and y are the eccentricity degree in horizontal and vertical direction respectively, for the meridian type m:

$$\begin{aligned}\rho(r,m) =& 2\rho_{cone}\left(1+\frac{r}{41.03}\right)^{-1}\times \\ & \left[a_m\left(1+\frac{r}{r_{2,m}}\right)^{-2}+(1-a_m)\exp\left(-\frac{r}{r_{e,m}}\right)\right],\end{aligned} \tag{4.9}$$

where $\rho_{cone} = 14804.6 deg^{-2}$ is the density of cone cell at fovea and $a_m, r_{2,m}, r_{e,m}$ are all fitting constants along the four meridians of the visual field. And the cell-wise spacing is calculated by [72]:

$$\sigma(x,y) = \frac{1}{r}\sqrt{\frac{2}{\sqrt{3}}\left(\frac{x^2}{\rho(r,1)}+\frac{y^2}{\rho(r,2)}\right)} \tag{4.10}$$

Then, given the eccentricity, the importance function E of a given eccentricity $\mathbf{x} = (x, y)$ is modeled as [72]:

$$E(\mathbf{x}) = 0.5\sigma(\mathbf{x})^{-1}. \tag{4.11}$$

4.2.7.2 Perceiving Static Stimuli

The visual sensitivity s of a certain frequency f and illumination L is determined by [4]:

$$s(f,L) = afe^{-bf}\sqrt{1+ce^{bf}}, \tag{4.12}$$

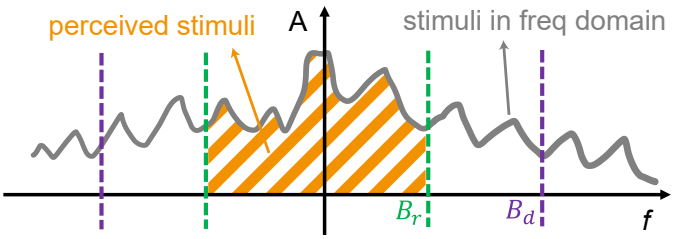


Figure 4.21: Perceived static visual stimuli. The perceived stimulus (crossed area) is an integration of frequency amplitudes. Its integral domain is bounded by the densities of display pixels (B_d) and retinal receptors (B_r).

where $a = \frac{540(1+0.7/L)^{-0.2}}{1+\frac{1}{1+f/3}}$, $b = 0.3(1+100/L)^{0.15}$ and $c = 0.06$.

Because illumination of commodity displays is largely insensitive to the displaying content, we assume a constant illumination L for all frequencies. Given an image I, its visual sensitivity is the integral of all frequencies weighted by their amplitude,

$$\int s\left(|\mathbf{f}|, L\right)|A\left(\mathbf{f}\right)|\mathrm{d}\mathbf{f} \tag{4.13}$$

where $\mathbf{f} = (f_x, f_y)$ is the two-dimensional frequency of I, and A is the amplitude of $\mathbf{f}$.

As shown in Figure 4.21, because of the finite display angular resolution and the finite retinal receptor densities, the non-zero frequency in Equation (4.13) lies in a finite domain as well,

$$\int_{|\mathbf{f}|<\min(B_d,\sup B_r)} s\left(|\mathbf{f}|, L\right)|A\left(\mathbf{f}\right)|\mathrm{d}\mathbf{f}, \tag{4.14}$$

where B_d is the display band from the pixel density and eye-panel distance (0.5 cycle per pixel (CPP) in our display); $\sup B_r$ is the supremum of the foveated retinal band.

The amplitude A in Equation (4.14) will become spatially-variant if combined with foveated vision (Equation (4.9)). That requires the evaluation of spatially-variant spectrum, such as windowed Fourier or wavelet, which is prohibitively expensive in a real-time system. Therefore, we evaluate local contrast c, which can

be more efficiently computed than A, of every pixel:

$$\Phi(\mathbf{g}, I) \triangleq \int_{\mathbf{x}\in I} \overbrace{\int_{|\mathbf{f}|<\bar{B}(\mathbf{g},\mathbf{x})} s\left(\mathbf{f}, L\right) c(\mathbf{x}, \mathbf{f}, I) \mathrm{d}\mathbf{f}}^{\hat{\Phi}(\mathbf{g},\mathbf{x},I)} \mathrm{d}\mathbf{x}, \tag{4.15}$$
$$\bar{B}(\mathbf{g}, \mathbf{x}) = \min\left(B_d, B_r(\mathbf{g}, \mathbf{x})\right)$$

where c is the local contrast of $\mathbf{x} = (x, y)$ of I under the frequency $\mathbf{f}$. $\mathbf{g} = (g_x, g_y)$ is the tracked gaze position on the screen space. $\hat{\Phi}$ is the corresponding sensitivity value for a spatial position $\mathbf{x}$, thus the frequency is integrated within a narrower foveated retinal band $B_r(\mathbf{g}, \mathbf{x}) = E(\mathbf{g} - \mathbf{x})$ according to the gaze position and spatial position as shown in Equation (4.11). By applying clamping with $B_r(\mathbf{g}, \mathbf{x})$, the eccentricity variances on the contrast sensitivity function are considered. Note that our model is for importance-based decision than pixel rendering so we assume the visibility of certain spatial frequencies is defined by receptor density. The peripheral vision can "reason" (other than directly perceive) higher-than-receptor-frequency information from the aliasing on the receptors [77]. Therefore, fully approximate the value by clamping the integration with the receptor frequency may cause loss of importance in the periphery. Thus, we introduced a constant ω in Equation (4.2) that compensates the peripheral region with low importance.

4.2.7.3 Bandpass filtering

To analytically compute the temporal consistency considering not only the content but the retinal receptors and display capability, we discretize the $\hat{\Phi}$ in Equation (4.15). Specifically, we perform a series of bandpass filtering of I to first obtain the gaze- and content-aware pixel-wise sensitivity:

$$\hat{\Phi}(\mathbf{g}, \mathbf{x}, I) \approx \sum_{i=0}^{b-1} s\left(\mathbf{f}_i, L\right) c(\mathbf{x}, \mathbf{f}_i, I), \tag{4.16}$$

where we divide the frequency domain of integral in Equation (4.15) into b bands, where $\mathbf{f}_i = 2^{i\frac{\bar{B}(\mathbf{g},\mathbf{x})}{b}}$ are the representative frequencies located at the mid-point of every band (following [53]). I_i is the $\mathbf{f}_i$-filtered version of I. Figure 4.9 visualizes $\hat{\Phi}$

across different bands $\mathbf{f}_i$. The contrast c at point $\mathbf{x}$ is defined as

$$c(\mathbf{x}, \mathbf{f}, I) = \frac{a_{\mathbf{f}}(\mathbf{x}, I)}{a_0(\mathbf{x}, I)}. \tag{4.17}$$

Here $a_{\mathbf{f}}$ is the approximated local frequency of the $\mathbf{f}$-filtered I, as detailed in [52], and a_0 is $a_{\mathbf{f}}$ with $\mathbf{f}_i = 0$.

Chapter 5

Conclusions and Future Work

This dissertation presented two main contributions to 3D memory in AR. The first contribution was a 3D memory framework, which addresses the unique challenges in AR scenarios such as rapid camera movement and limited FoV. It combines the power of 2D open vocabulary object detectors and 3D depth information from depth sensors equipped by AR headsets. In the absence of a suitable existing dataset, we created an annotated dataset, wherein users engaged in cooking tasks. The efficacy of the proposed framework was validated through the dataset, showcasing its superiority over existing methods. The second contribution was visualization tools for 3D memory, including extensions to ARGUS for debugging machine learning modules of AI assistants, and InstantReality[11] for real-time streaming large volume 3D data while providing a better user experience in AR/VR.

The contributions presented here enable AI assistants to integrate a 3D memory module with other essential modules, such as task reasoning and user interface. This integration has unveiled novel research challenges, providing interesting directions for further exploration in the realm of AI assistants in AR. In the following, we present some of these challenges and highlight interesting directions of future research.

Multimodal inputs. In this thesis, the proposed 3D memory framework utilizes the information from both RGB and depth sensors. However, modern AR headsets are often equipped with additional sensors beyond the RGB and depth sensors. For example, the Microsoft HoloLens 2 not only equips with an RGB and an

infrared depth camera, but also contains four peripheral grayscale cameras, a microphone array, an IMU sensor and eye sensors for calculating the user's gaze. An interesting direction is to explore how to leverage these multimodal inputs from the various sensors on board. The four peripheral grayscale cameras not only enlarge the FoV, but also enable stereo depth estimation to overcome the limitations of the time-of-flight infrared depth sensor. Eye gaze from users could provide information about active objects. Furthermore, different headsets can have distinct sensor configurations. For instance, the Meta Quest 3 has two RGB cameras, two grayscale cameras and a depth camera, which differs from the HoloLens 2. The Quest 3 also lacks eye tracking. Building a generic 3D memory framework that can handle multimodal inputs and adapt to different sensor configurations remains a challenge.

Task guidance with 3D memory. To provide task guidance to the user, the outputs of 3D memory must be presented through a user interface. Designing such a user interface to improve user task performance is still an open problem. AI assistants need to carefully choose the information shown to the user, as unnecessary information displayed in the headset could obstruct the user's view or distract them from performing the task, given the limited display space available. AI assistants also need to decide when and how to guide the user to find a required object outside the user's FoV to help them complete the current step. Adding spatial information in instructions are considered important by domain experts [83]. Interface design that better utilizes the available screen space to show both local and global 3D information simultaneously has been explored for virtual reality [12], and it can be an interesting direction for augmented reality. Moreover, the outputs of 3D memory can aid the reasoning module in inferring the current task/step and determining whether the user has made an error. With 3D memory, AI assistants can memorize seen objects that are now outside the FoV, allowing them to use global context when inferring the current task/step instead of relying solely on the current RGB frame. Exploring how to leverage information from 3D memory to enhance reasoning performance could be an interesting direction.

Human-in-the-loop AI. Unlike computers using a mouse and keyboard for input, AR headsets support natural ways of interaction such as hand gestures, eye gaze, and voice commands. This give us new opportunity to make the 3D memory human-in-the-loop. Some cases that are very challenging for computer vision, such as differentiating a jar of nut butter and a jar of Nutella, could be easily done by the human. 3D memory allows memorizing not only objects in the scene, but also user's feedback to the machine learning results. Exploring how to build an interactive AI system in AR that allows users to correct AI errors and assist AI in decision-making could be an interesting direction.

Bibliography

[1] P. Akiva, J. Huang, K. J. Liang, R. Kovvuri, X. Chen, M. Feiszli, K. Dana, and T. Hassner. Self-supervised object detection from egocentric videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5225–5237, 2023.

[2] R. Albert, A. Patney, D. Luebke, and J. Kim. Latency requirements for foveated rendering in virtual reality. *ACM Trans. Appl. Percept.*, 14(4), Sept. 2017.

[3] E. Arabadzhiyska, O. T. Tursun, K. Myszkowski, H.-P. Seidel, and P. Didyk. Saccade landing position prediction for gaze-contingent rendering. *ACM Trans. Graph.*, 36(4), July 2017.

[4] P. G. J. Barten. Evaluation of subjective image quality with the square-root integral method. *J. Opt. Soc. Am. A*, 7(10):2024–2031, Oct 1990.

[5] H. Belhassen., H. Zhang., V. Fresse., and E. Bourennane. Improving video object detection by seq-bbox matching. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2019) - Volume 5: VISAPP*, pages 226–233. INSTICC, SciTePress, 2019.

[6] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *J. Image Video Process.*, 2008, jan 2008.

[7] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016.

[8] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society.

[9] S. Castelo, J. Rulff, E. McGowan, B. Steers, G. Wu, S. Chen, I. Roman, R. Lopez, E. Brewer, C. Zhao, J. Qian, K. Cho, H. He, Q. Sun, H. Vo, J. Bello, M. Krone, and C. Silva. Argus: Visualization of ai-assisted task guidance in ar. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):1313–1323, 2024.

[10] L. Chen, H. Ai, Z. Zhuang, and C. Shang. Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, Los Alamitos, CA, USA, jul 2018. IEEE Computer Society.

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

[12] S. Chen, F. Miranda, N. Ferreira, M. Lage, H. Doraiswamy, C. Brenner, C. Defanti, M. Koutsoubis, L. Wilson, K. Perlin, and C. Silva. Urbanrama: Navigating cities in virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 28(12):4685–4699, 2022.

[13] H. K. Cheng and A. G. Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*, page 640–658, Berlin, Heidelberg, 2022. Springer-Verlag.

[14] H.-K. Chiu, J. Li, R. Ambruş, and J. Bohg. Probabilistic 3d multi-modal, multi-object tracking for autonomous driving. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14227–14233, 2021.

[15] M. A. Cohen, T. L. Botch, and C. E. Robertson. The limits of color awareness during active, real-world vision. *Proceedings of the National Academy of Sciences*, 117(24):13821–13827, 2020.

[16] D. Cohen-Or, Y. Mann, and S. Fleishman. Deep compression for streaming texture intensive animations. In *SIGGRAPH '99*, pages 261–267, 1999.

[17] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski. Viewport-adaptive navigable 360-degree video delivery. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–7, 2017.

[18] D. Damen, H. Doughty, G. M. Farinella, A. Furnari, J. Ma, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision (IJCV)*, 130:33–55, 2022.

[19] A. Darkhalil, D. Shan, B. Zhu, J. Ma, A. Kar, R. Higgins, S. Fidler, D. Fouhey, and D. Damen. Epic-kitchens visor benchmark: Video segmentations and object relations. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 13745–13758. Curran Associates, Inc., 2022.

[20] A. Dave, T. Khurana, P. Tokmakov, C. Schmid, and D. Ramanan. Tao: A large-scale benchmark for tracking any object. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V*, page 436–454, Berlin, Heidelberg, 2020. Springer-Verlag.

[21] M. Dunnhofer, A. Furnari, G. M. Farinella, and C. Micheloni. Visual object tracking in first person vision. *International Journal of Computer Vision*, 131(1):259–283, 1 2023.

[22] D. B. Elliott, K. C. Yang, and D. Whitaker. Visual acuity changes throughout adulthood in normal, healthy eyes: seeing beyond 6/6. *Optometry and vision science : official publication of the American Academy of Optometry*, 72 3:186–91, 1995.

[23] J. A. Ferwerda, P. Shirley, S. N. Pattanaik, and D. P. Greenberg. A model of visual masking for computer graphics. In *SIGGRAPH '97*, pages 143–152, 1997.

[24] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[25] G. Georgakis, M. Reza, A. Mousavian, P. Le, and J. Kosecka. Multiview rgb-d dataset for object instance detection. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 426–434, Los Alamitos, CA, USA, oct 2016. IEEE Computer Society.

[26] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, M. Martin, T. Nagarajan, I. Radosavovic, S. Ramakrishnan, F. Ryan, J. Sharma, M. Wray, M. Xu, E. Xu, C. Zhao, S. Bansal, D. Batra, V. Cartillier, S. Crane, T. Do, M. Doulaty, A. Erapalli, C. Feichtenhofer, A. Fragomeni, Q. Fu, A. Gebreselasie, C. Gonzalez, J. Hillis, X. Huang, Y. Huang, W. Jia, W. Khoo, J. Kolar, S. Kottur, A. Kumar, F. Landini, C. Li, Y. Li, Z. Li, K. Mangalam, R. Modhugu, J. Munro, T. Murrell, T. Nishiyasu, W. Price, P. Puentes, M. Ramazanova, L. Sari, K. Somasundaram, A. Southerland, Y. Sugano, R. Tao, M. Vo, Y. Wang, X. Wu, T. Yagi, Z. Zhao, Y. Zhu, P. Arbelaez, D. Crandall, D. Damen, G. Farinella, C. Fuegen, B. Ghanem, V. Ithapu, C. V. Jawahar, H. Joo, K. Kitani, H. Li, R. Newcombe, A. Oliva, H. Park, J. M. Rehg, Y. Sato, J. Shi, M. Shou, A. Torralba, L. Torresani, M. Yan, and J. Malik. Ego4d: Around the world in 3,000 hours of egocentric video. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18973–18990, Los Alamitos, CA, USA, jun 2022. IEEE Computer Society.

[27] W. Griffin and M. Olano. Evaluating texture compression masking effects using objective image quality assessment metrics. *IEEE Transactions on Visualization and Computer Graphics*, 21(8):970–979, 2015.

[28] M. Grinvald, F. Tombari, R. Siegwart, and J. Nieto. Tsdf++: A multi-object formulation for dynamic object tracking and reconstruction. In *2021*

IEEE International Conference on Robotics and Automation (ICRA), pages 14192–14198, 2021.

[29] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[30] S. Hillaire, A. Lecuyer, T. Regia-Corte, R. Cozot, J. Royan, and G. Breton. Design and application of real-time visual attention model for the exploration of 3d virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):356–368, 2012.

[31] J. Hladky, H.-P. Seidel, and M. Steinberger. The camera offset space: Real-time potentially visible set computations for streaming rendering. *ACM Trans. Graph.*, 38(6), Nov. 2019.

[32] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE Transactions on Visualization and Computer Graphics*, 25(8):2674–2693, 2019.

[33] Y. Hou, C. Wang, J. Wang, X. Xue, X. L. Zhang, J. Zhu, D. Wang, and S. Chen. Visual evaluation for autonomous driving. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):1030–1039, 2022.

[34] M. Ibbotson and B. Krekelberg. Visual perception and saccadic eye movements. *Current Opinion in Neurobiology*, 21(4):553–558, 2011. Sensory and motor systems.

[35] A. S. Kaplanyan, A. Sochenov, T. Leimkühler, M. Okunev, T. Goodall, and G. Rufo. Deepfovea: Neural reconstruction for foveated rendering and video compression using learned statistics of natural videos. *ACM Trans. Graph.*, 38(6), Nov. 2019.

[36] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[37] J. Knoell, P. Binda, M. C. Morrone, and F. Bremmer. Spatiotemporal profile of peri-saccadic contrast sensitivity. *Journal of vision*, 11(14):15–15, 2011.

[38] G. A. Koulieris, K. Akşit, M. Stengel, R. K. Mantiuk, K. Mania, and C. Richardt. Near-eye display and tracking technologies for virtual and augmented reality. *Computer Graphics Forum*, 38(2):493–519, 2019.

[39] G. Lavoué, M. Langer, A. Peytavie, and P. Poulin. A psychophysical evaluation of texture compression masking effects. *IEEE transactions on visualization and computer graphics*, 25(2):1336–1346, 2018.

[40] M. Levoy. Polygon-assisted jpeg and mpeg compression of synthetic images. In *SIGGRAPH '95*, pages 21–28, 1995.

[41] M. Li, Y.-X. Wang, and D. Ramanan. Towards streaming perception. In *European Conference on Computer Vision*, pages 473–488. Springer, 2020.

[42] P. Liu, X. Zuo, V. Larsson, and M. Pollefeys. Mba-vo: Motion blur aware visual odometry. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5530–5539, 2021.

[43] Y. Liu, Y. Liu, C. Jiang, K. Lyu, W. Wan, H. Shen, B. Liang, Z. Fu, H. Wang, and L. Yi. Hoi4d: A 4d egocentric dataset for category-level human-object interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21013–21022, June 2022.

[44] A. Lopes, R. Souza, and H. Pedrini. A survey on rgb-d datasets. *Computer Vision and Image Understanding*, 222:103489, 2022.

[45] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of Detail for 3D Graphics*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[46] R. K. Mantiuk, G. Denes, A. Chapiro, A. Kaplanyan, G. Rufo, R. Bachy, T. Lian, and A. Patney. Fovvideovdp: A visible difference predictor for wide field-of-view video. *ACM Trans. Graph.*, 40(4), July 2021.

[47] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[48] H. Murphy and A. Duchowski. Gaze-contingent level of detail rendering. *EuroGraphics*, 2001, 01 2001.

[49] R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 343–352, 2015.

[50] Z. Pang, Z. Li, and N. Wang. Simpletrack: Understanding and rethinking 3d multi-object tracking. In L. Karlinsky, T. Michaeli, and K. Nishino, editors, *Computer Vision – ECCV 2022 Workshops*, pages 680–696, Cham, 2023. Springer Nature Switzerland.

[51] A. Patney, M. Salvi, J. Kim, A. Kaplanyan, C. Wyman, N. Benty, D. Luebke, and A. Lefohn. Towards foveated rendering for gaze-tracked virtual reality. *ACM Trans. Graph.*, 35(6), Nov. 2016.

[52] E. Peli. Contrast in complex images. *J. Opt. Soc. Am. A*, 7(10):2032–2040, Oct 1990.

[53] C. Párraga, T. Troscianko, and D. Tolhurst. The effects of amplitude-spectrum statistics on foveal and peripheral discrimination of changes in natural images, and a multi-resolution model. *Vision Research*, 45(25):3145 – 3168, 2005.

[54] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan. Beyond throughput, the next generation: A 5g dataset with channel and context metrics. In *MMSys '20*, pages 303–308, 2020.

[55] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021.

[56] F. Ragusa, A. Furnari, and G. M. Farinella. Meccano: A multimodal egocentric dataset for humans behavior understanding in the industrial-like domain. *Comput. Vis. Image Underst.*, 235(C), oct 2023.

[57] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society.

[58] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2016.

[59] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[60] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen. Commute path bandwidth traces from 3g networks: Analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*, MMSys '13, pages 114–118, New York, NY, USA, 2013. Association for Computing Machinery.

[61] M. F. Romero-Rondón, L. Sassatelli, F. Precioso, and R. Aparicio-Pardo. Foveated streaming of virtual reality videos. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, pages 494–497, 2018.

[62] M. Runz, M. Buffier, and L. Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 10–20, 2018.

[63] S. Rusinkiewicz and M. Levoy. Qsplat: A multiresolution point rendering system for large meshes. In *SIGGRAPH '00*, pages 343–352, 2000.

[64] M. Rünz and L. Agapito. Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4471–4478, 2017.

[65] A. Sabater, L. Montesano, and A. C. Murillo. Robust and efficient post-processing for video object detection. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10536–10542, 2020.

[66] M. Schütz, G. Mandlburger, J. Otepka, and M. Wimmer. Progressive real-time rendering of one billion points without hierarchical acceleration structures. *Computer Graphics Forum*, 39(2):51–64, 2020.

[67] M. Schwarz and M. Stamminger. On predicting visual popping in dynamic scenes. In *APGV '09*, pages 93–100, 2009.

[68] A. Serrano, I. Kim, Z. Chen, S. DiVerdi, D. Gutierrez, A. Hertzmann, and B. Masia. Motion parallax for 360° rgbd video. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):1817–1827, May 2019.

[69] C. Serrano, B. Garriga, J. Velasco, J. Urbano, S. Tenorio, and M. Sierra. Latency in broad-band mobile networks. In *VTC Spring 2009 - IEEE 69th Vehicular Technology Conference*, pages 1–7, 2009.

[70] M. Strecke and J. Stueckler. Em-fusion: Dynamic object-level slam with probabilistic data association. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5864–5873, 2019.

[71] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2443–2451, 2020.

[72] Q. Sun, F.-C. Huang, J. Kim, L.-Y. Wei, D. Luebke, and A. Kaufman. Perceptually-guided foveation for light field displays. *ACM Trans. Graph.*, 36(6), Nov. 2017.

[73] H. Tang, K. Liang, M. Feiszli, and W. Wang. Egotracks: A long-term egocentric visual object tracking dataset. *arXiv:2301.03213*, 2023.

[74] V. Tschernezki, A. Darkhalil, Z. Zhu, D. Fouhey, I. Laina, D. Larlus, D. Damen, and A. Vedaldi. Epic fields: Marrying 3d geometry and video understanding. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 26485–26500. Curran Associates, Inc., 2023.

[75] O. T. Tursun, E. Arabadzhiyska-Koleva, M. Wernikowski, R. Mantiuk, H.-P. Seidel, K. Myszkowski, and P. Didyk. Luminance-contrast-aware foveated rendering. *ACM Trans. Graph.*, 38(4), July 2019.

[76] J. Wang, Y. Li, Z. Zhou, C. Wang, Y. Hou, L. Zhang, X. Xue, M. Kamp, X. Zhang, and S. Chen. When, where and how does it fail? a spatial-temporal visual analytics approach for interpretable object detection in autonomous driving. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–16, 2022.

[77] Y.-Z. Wang, A. Bradley, and L. N. Thibos. Aliased frequencies enable the discrimination of compound gratings in peripheral vision. *Vision Research*, 37(3):283–290, 1997.

[78] A. B. Watson. A formula for human retinal ganglion cell receptive field density as a function of visual field location. *Journal of Vision*, 14(7):15–15, 06 2014.

[79] A. B. Watson, A. J. Ahumada, and J. E. Farrell. Window of visibility: a psychophysical theory of fidelity in time-sampled visual motion displays. *J. Opt. Soc. Am. A*, 3(3):300–307, Mar 1986.

[80] X. Weng, J. Wang, D. Held, and K. Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10359–10366, 2020.

[81] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649, 2017.

[82] Y.-S. Wong, C. Li, M. Nießner, and N. J. Mitra. Rigidfusion: Rgb-d scene reconstruction with rigidly-moving objects. *Computer Graphics Forum*, 40(2):511–522, 2021.

[83] G. Wu, J. Qian, S. Castelo, S. Chen, J. Rulff, and C. Silva. Automated text simplification for the task guidance in augmented reality. In *2023 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 585–586, 2023.

[84] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger. Mid-fusion: Octree-based object-level multi-instance dynamic slam. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5231–5237, 2019.

[85] T. Yin, X. Zhou, and P. Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11784–11793, June 2021.

[86] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang. Bytetrack: Multi-object tracking by associating every detection box. In S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, editors, *Computer Vision – ECCV 2022*, pages 1–21, Cham, 2022. Springer Nature Switzerland.

[87] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra. Detecting twenty-thousand classes using image-level supervision. In S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, editors, *Computer Vision – ECCV 2022*, pages 350–368, Cham, 2022. Springer Nature Switzerland.

[88] X. Zhou, V. Koltun, and P. Krähenbühl. Tracking objects as points. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 474–490, Cham, 2020. Springer International Publishing.

[89] X. Zhou, D. Wang, and P. Krähenbühl. Objects as points. *arXiv:1904.07850*, 2019.

[90] X. Zhou, T. Yin, V. Koltun, and P. Krahenbuhl. Global tracking transformers. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8761–8770, Los Alamitos, CA, USA, jun 2022. IEEE Computer Society.

[91] J. Zhu, L. Wang, R. Yang, J. E. Davis, and Z. pan. Reliability fusion of time-of-flight depth and stereo geometry for high quality depth maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1400–1414, 2011.