FarfetchFusion: Towards Fully Mobile Live 3D Telepresence Platform

Kyungjin Lee
Seoul National University
Seoul, Republic of Korea
jin11542@snu.ac.kr

Juheon Yi
Seoul National University
Seoul, Republic of Korea
johnyi0606@snu.ac.kr

Youngki Lee
Seoul National University
Seoul, Republic of Korea
youngkilee@snu.ac.kr

Abstract

We present FarfetchFusion, a fully mobile live 3D telepresence system. Enabling mobile live telepresence is a challenging problem as it requires i) realistic reconstruction of the user and ii) high responsiveness for immersive experience. We first thoroughly analyze the live 3D telepresence pipeline and identify three critical challenges: i) 3D data streaming latency and compression complexity, ii) computational complexity of volumetric fusion-based 3D reconstruction, and iii) inconsistent reconstruction quality due to sparsity of mobile 3D sensors. To tackle the challenges, we propose a disentangled fusion approach, which separates invariant regions and dynamically changing regions with our low-complexity spatio-temporal alignment technique, topology anchoring. We then design and implement an end-to-end system, which achieves realistic reconstruction quality comparable to existing server-based solutions while meeting the real-time performance requirements (<100 ms end-to-end latency, 30 fps throughput, <16 ms motion-to-photon latency) solely relying on mobile computation capability.

CCS Concepts

• **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; • **Computer systems organization** → **Real-time system architecture**; • **Computing methodologies** → **Mixed / augmented reality**; **Volumetric models**.

Keywords

3D Telepresence, Mobile Systems, Volumetric Fusion

ACM MobiCom '23, October 2–6, 2023, Madrid, Spain

ACM ISBN 978-1-4503-9990-6/23/10…$15.00
https://doi.org/10.1145/3570361.3592525

ACM Reference Format:
Kyungjin Lee, Juheon Yi, and Youngki Lee. 2023. FarfetchFusion: Towards Fully Mobile Live 3D Telepresence Platform . In *The 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '23), October 2–6, 2023, Madrid, Spain.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3570361.3592525

1 Introduction

Live 3D telepresence provides unprecedented immersion and interactiveness, overcoming fundamental limitations of prior 2D video-based communication. It captures the user and surroundings in 3D media format and streams it to the remote user who enjoys the media with 6 Degrees-of-Freedom (6DoF). There have been early trials to build live 3D telepresence systems [28, 43], but use cases are limited to indoor studios instrumented with many high-cost cameras and server-grade computers. In this work, we aim to design a fully mobile live 3D telepresence system for everyday use. This capability will enable highly immersive co-activity experiences even for physically apart users (e.g., long-distance couples and families as if they are eating, chatting, or walking together).

In this light, we present FarfetchFusion, the first mobile 3D live telepresence system. It enables an end-to-end pipeline to capture, reconstruct, stream, and render 3D media in real time, solely based on mobile cameras and computing power. As the first step, we focus on the 3D face, which is not only the essential part of interpersonal communication conveying a multitude of emotions but also one of the most complex objects for 3D reconstruction. We believe that our system and the underlying design considerations will be cornerstones for future mobile 3D telepresence systems (e.g., supporting other body parts and objects as discussed in Section 10).

Designing our system involves the following challenges.

• **Realistic 3D Reconstruction with Mobile Cameras.** Enabling a high level of presence requires a realistic reconstruction of the user's face. However, the human face is a challenging 3D capture and reconstruction target. It consists of 42 individual muscles that jointly generate intricate, subtle facial expressions during conversations (especially around the mouth and the eye regions). Failing to capture these details can lead to the uncanny valley phenomena, somewhat

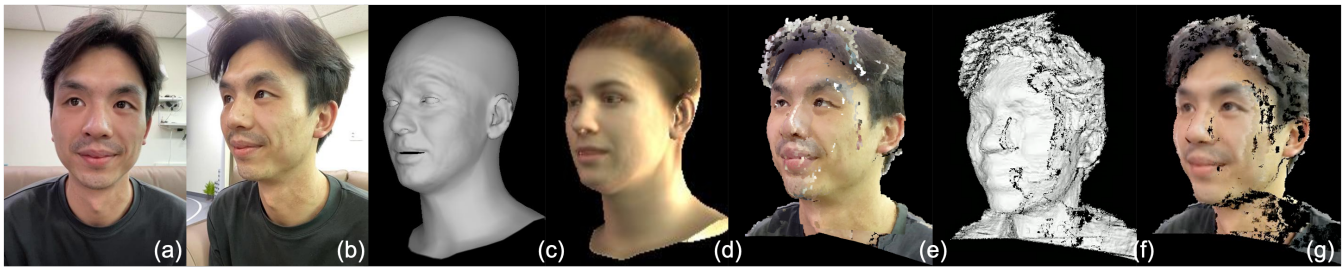

Figure 1: Example results of different 3D reconstruction methods. From left: (a)-(b) input RGB images, (c) geometry and (d) texture reconstruction result of 3DMM with deep learning-based method [13], (e) point cloud fusion result, (f) geometry and (g) texture reconstruction result of TSDF-based fusion method [24]

resembling reconstruction results rather generates negative emotional effects [8].

Mobile deployment complicates the problem in two folds. First, the number of cameras and resolution is significantly smaller than in studio settings (e.g., 3 vs. 7, 640×480 vs. 1280×1024 [28]). This results in sparse 3D data due to partial facial areas out of camera views or inconsistency between multiple depth camera data. The problem becomes more serious when the face moves quickly and continuously. Such sparse data leads to significant perceptual quality degradation (e.g., Figure 1(g)). Second, the 3D data from multiview depth streams need to be accurately aligned and fused to generate a smooth surface; even with state-of-the-art deep learning techniques [13, 25], misalignment occurs and generates reconstruction artifacts and quality fluctuation (Figure 18).

• **High Responsiveness in Mobile Setup.** FarfetchFusion have tight latency requirements. First, it should support less than 100 ms end-to-end latency [28] including face capturing, 3D reconstruction, streaming, and rendering, for high user responsiveness. Second, each component should achieve a high throughput (e.g., 30 fps) to capture and reconstruct subtle motions such as utterance-level mouth movements; This is challenging because 3D reconstruction alone takes 250 ms even on high-end mobile devices (e.g., iPhone 12 Pro). Finally, the rendering latency should be much smaller (e.g., <16 ms) to update the viewer's screen upon his viewpoint change; this is known as motion-to-photon latency, different from the end-to-end latency. However, even with highly optimized rendering techniques (i.e., raycasting [26] or meshing and rasterization [35]), the latency takes 100 ms∼180 ms on mobile GPUs. As such, enabling fully mobile 3D telepresence requires rigorous end-to-end design and optimizations.

Our key approach to address the challenges is *disentangled fusion* (Section 5). It allows our system to utilize the temporality of 3D face data even under dynamically changing facial poses and expressions. In particular, it separates the static part of facial information ('invariants') such as the head, forehead, and ears from the dynamic part ('variants') such as the mouth and eyes. Then, it combines the invariants first from multiple frames while additionally fusing only the 'variants' from the most recent frames. This provides two benefits: (i) high reconstruction quality by effectively utilizing spatio-temporal redundancy in multiview video streams, and (ii) high computation efficiency by minimizing the redundancy in processing the 'invariant' part.

The key to realizing our approach is aligning the multiview RGBD video streams in spatio-temporal axes. However, this is a non-trivial problem, as finding the correspondence between multiview RGBD streams of a dynamically moving face is either computationally heavy or erroneous. Prior registration techniques such as iterative closest point (ICP) algorithms find accurate alignments minimizing the distance between two 3D data pairs, but incur significant overhead [60]. Another approach is using facial landmark features for alignment [10]. However, the topological inconsistency between the landmarks with different facial expressions (e.g., different mouth shapes and moving eyebrows while talking) does not guarantee accurate alignments [52].

To tackle the challenge, we propose a topologically consistent anchoring mechanism, namely *topology anchoring* (Section 6). We are inspired by the 3D Morphable Model (3DMM)-based method that fits a generic face template to the target captured face by morphing the shape and texture, rotating the head direction, and estimating the camera parameters. We use the generic face template coordinates as the static anchor space where all input spatio-temporal RGB-D data should be aligned and integrated. Then, we generate pseudo-static anchors, which is anchored in the static anchor space but reflects the dynamic facial expressions of every input by morphing. This enables the alignment of spatio-temporal inputs with topologically consistency.

Based on the topology anchoring, we design the entire telepresence pipeline to realize our approach. Among the vast design space, we first build the base pipeline based on the volumetric fusion [24, 39, 40], a representative 3D reconstruction approach that generates realistic and smooth surfaces from multiview RGB-D inputs. Then, we optimize the

base pipeline to meet 30 fps throughput by enabling spatio-temporal fusion and rendering. In the optimized pipeline, the invariants are computed in the anchor space at an offline phase to remove computational redundancy for the voxels corresponding to invariant face areas. The pipeline updates only the voxels for variants at run time while maintaining smooth boundaries with the pre-computed voxels for invariants. It also handles noises in input depth streams and occlusions, well-utilizing previous reconstruction frames.

Our key contributions are summarized as follows.

- To our knowledge, FarfetchFusion is the first fully mobile live streaming system for 3D face-to-face telepresence.
- We design a disentangled fusion approach that jointly tackles the reconstruction quality and latency challenges.
- We implement a prototype of FarfetchFusion with commercial mobile devices and conduct extensive real-world evaluation with custom collected multiview RGB-D data in mobile settings both quantitatively and qualitatively.
- We achieve an end-to-end latency of ≈100 ms and meet the component-wise requirement of 30 fps throughput and <16 ms rendering latency.

2 Background on Volumetric Fusion

We design our system based on the concept of volumetric fusion, a representative 3D reconstruction approach that generates realistic and smooth surfaces from multiview RGB-D inputs (e.g., KinectFusion [40] and its variants [24, 39, 41]). We choose this approach to achieve high realism and practicality. Other 3D reconstruction methods such as modeling-based methods lack realism [11, 13, 32, 46, 50, 51, 57, 58] or require high quality pre-scanning and compute-intensive personalized model generation [5, 9]. Most recent approaches based on neural radiance fields (NeRF) are still limited to reconstructing static or pre-scanned objects and scenes [7, 15, 21, 37, 45, 53, 61].

2.1 Cameras and Data

Input. Volumetric fusion requires multiview RGB-Depth(D) data as input for 3D reconstruction. RGB-D data can be easily acquired from commodity sensors such as Microsoft Kinect, Intel RealSense, and LiDARs. Many state-of-the-art smartphones and tablets are equipped with high-resolution (e.g., 640x480) time-of-flight (ToF) frontal depth sensors used for accurate face identification. Users can customize a multiview capture setup utilizing personal smart devices in any desired location.

Output. The output of volumetric fusion can be rendered on head-mounted displays (e.g., Oculus Quest, HTC Vive). Users can move around freely, viewing the 3D reconstruction from any viewpoint. 3D data is rendered by generating two 2D images rendered for each left and right eye. Users can also view them on 2D displays with a touch or mouse interface for 6 degrees of freedom (DoF).

Data Representation. The underlying data representation is 3D voxels (i.e., equivalent to pixels in 2D images), each associated with a Truncated Signed Distance Function (TSDF) value and colors. A TSDF value (e.g., float or short) represents the distance of the voxel to the nearest surface truncated in the range of [-1,1] (Figure 3). TSDF values of voxels on the surface are zero, the outside positive, and the inside negative. The TSDF voxels are stored in a hash data structure due to the sparse nature of 3D data, rather than a 3D array to reduce the memory footprint (e.g., 300×300×300 voxels for 1*mm* resolution, 4 bytes for TSDF value, 3 bytes for RGB colors, 1 byte for the weight value results in 216 MB per frame) [24, 41]. The hash function maps (x,y,z) positions of voxels to different hash entries. Each entry contains a pointer to allocated voxels with valid TSDF values. This enables compute- and memory-efficient reconstruction.

2.2 Volumetric Fusion Overview

Volumetric fusion includes multiple stages (Figure 2).

• **Multiview Alignment.** The first stage is multiview RGB-D alignment, which consists of feature extraction and registration. We can use face-specific feature extraction methods (e.g., landmark detection) for both robustness and efficiency, especially with recent deep learning-based methods [13, 25, 54]. The 2D face landmarks are first extracted from input RGB images. The 2D face landmarks are then lifted to 3D landmarks using depth images. The 3D landmarks are used to find the similarity transform parameters (i.e, rotation, translation, scaling) between multiple views [52].

• **Fusion.** This stage comprises three steps to calculate the TSDF voxels with the aligned multiview RGB-D: (i) TSDF voxel reset, (ii) voxel allocation, and (iii) TSDF calculation.

In the first step, we reset the hash data structure managed for storing TSDF values (i.e., remove all the hash entries allocated for the previous frame). This avoids the afterimage effect, traces left from previous frames in the final reconstruction outcome. Second, the voxel allocation step creates hash entries only for the effective voxels (i.e., voxels on or near the face surface). For this, a ray is casted from each pixel of the RGB-D image using the camera's intrinsic parameters. Then, the hash entries are allocated for the voxels intersecting with the ray. This step allows to update useful voxels only, rather than the entire 3D grid, saving computational and memory resources by reducing the number of voxels to process. Finally, for TSDF calculation, each allocated voxel is filled with a TSDF value. In particular, the TSDF value is set as the difference between the z-axis position of the voxel (in the 3D space projected to the camera viewpoint using the inverse camera projection parameters) and the depth

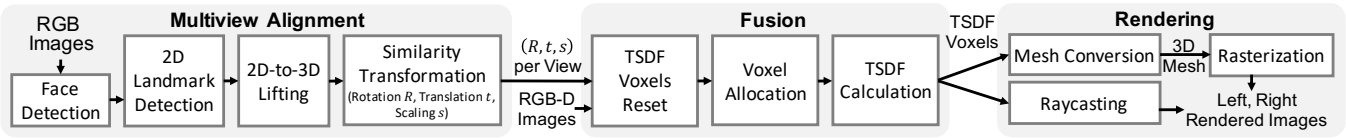

Figure 2: Multiple stages of volumetric fusion.

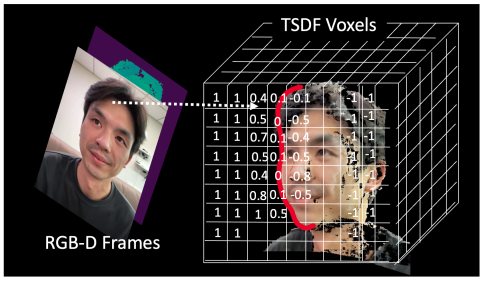

Figure 3: TSDF voxel generation from RGB-D frames.

value. Figure 3 shows an example of calculated TSDF values. When a voxel has inconsistent TSDF estimation from multiple views, the values are averaged or aggregated with different weights using the confidence scores [28].

• **Rendering.** The final step is rendering the reconstructed face, using the calculated TSDF values. There are two alternatives, rasterization and raycasting. Rasterization is a highly optimized rendering method supported by most modern graphics hardware on mobiles. The method, however, requires the conversion of TSDF voxels to mesh or point cloud. The conversion can be efficiently done with the Marching Cubes algorithm [35], but the performance depends on the number of voxels to process. Raycasting generates the rendered images without converting TSDF voxels to explicit 3D data representation. It first finds the voxels corresponding to the surface (i.e., TSDF value closest to zero compared to nearby voxels), by casting a ray from the user's viewpoint. Then, the color information of such found voxels is used to form the 2D pixels. Here, the performance is affected by the number of cast rays determined by the rendered image size and the number of allocated voxels.

3 Baseline Pipeline Design

Many design choices exist to build a telepresence pipeline with volumetric fusion. This involves i) dividing the volumetric fusion stages into the sender and receiver devices and ii) deciding on a 3D data format to stream. We need to narrow down the choices with careful consideration of network resources and mobile computation power. First, we focus on the data streaming challenge, which is critical to reducing the end-to-end latency. We introduce three candidates for distributing the volumetric fusion components and choose one baseline. Then, we illustrate the remaining challenges to achieve high reconstruction quality and efficiency.

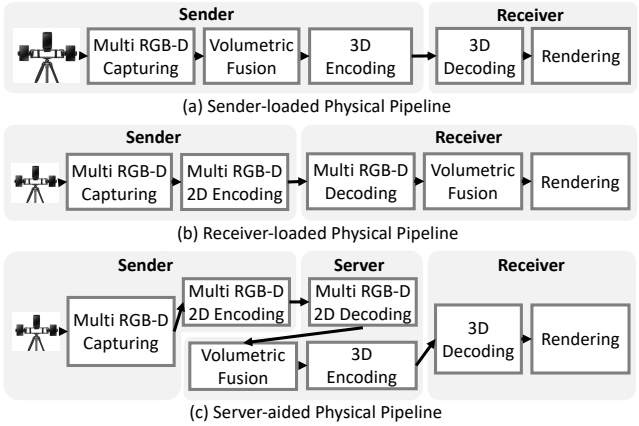

Figure 4: Variants of physical pipelines for volumetric fusion-based telepresence.

3.1 Distributed Pipeline Candidates

There are three possible participating devices: i) the sender mobile device capturing the multiview RGB-D, ii) the receiver mobile device for rendering, and iii) the (optional) remote server to aid computing.

Sender-loaded (Figure 4 (a)). In this pipeline, the compute-intensive multiview alignment and volumetric fusion components run on the sender device, and the reconstructed 3D data is streamed to the receiver. The reconstructed 3D data is encoded at the sender device, streamed, and decoded at the receiver device.

Receiver-loaded (Figure 4 (b)). The sender device sends the multiview RGB-D data, encoded with 2D video codecs, to the receiver device. The receiver device performs volumetric fusion and directly renders the 3D results. This pipeline does not require 3D streaming 3D, but the receiver device runs the computation-intensive volumetric fusion component.

Server-aided (Figure 4 (c)). We can offload the volumetric fusion component to a remote server to reduce the mobile's computational load. The sender encodes multiple RGB-D data and streams them to the server. The server performs volumetric fusion and sends the compressed 3D data to the receiver. The computation load on the sender and receiver devices are low. However, streaming 3D data is still required.

3.2 Baseline Selection

We first narrow down the design space to mitigate the networking challenge to stream 3D data (and further address

Table 1: Data rate and encoding/decoding latency for streaming.

RGB-D (Mobile)			3D Mesh (Desktop)		
Data (Mbps)	Enc (ms)	Dec (ms)	Data (Mbps)	Enc (ms)	Dec (ms)
7.2	3	5	288	656	248

Table 2: Processing latency (ms) for multiview alignment stage.

	Face Detection	Landmark Detection
CPU	6(±0.82)	232(±19.17)
GPU	9(±1.01)	140(±11.43)

Table 3: Processing latency (ms) for fusion stage.

	Scene Reset	Voxel Allocation	TSDF Calculation
CPU	6(±0.82)	62(±10.69)	1524(±264.90)
GPU	46(±4.45)	60(±8.34)	<1

computational challenges). We made the design decision under the assumption that the network resources are more constrained for 3D data streaming than the 3D computation (with fast-improving neural processors), but other designs remain to be studied in future systems.

We compare two streaming options: 3D data streaming (required for sender-loaded and server-aided pipelines) and multiview RGB-D streaming with 2D video codecs (required for receiver-loaded one). We use Draco [16], the state-of-the-art k-d tree-based point cloud data compression library, and H264 video codecs for multiview RGB-D. The depth data is quantized into 8 bits and stored in the Y-channel for YUV420 format-based compression, which is hardware accelerated on mobile devices. We test them on iPhone 12 Pro and a desktop computer equipped with Intel Xeon Gold 5218 CPU@2.30GHz and 1× NVIDIA RTX 3090 GPU.

Table 1 shows the data rate and encoding/decoding latency for the two streaming options. First, 3D data streaming incurs very high network bandwidth (≈300 Mbps) and compute latency even on desktop computers, mainly due to the difficulty of 3D data compression and lack of hardware support. Recently, numerous attempts improve the performance of 3D streaming (e.g., view-adaptive [20, 29, 34, 48], neural-enhanced [59]). However, it is far from reaching the performance of 2D video streaming. Cloud-offloaded rendering, which renders the 3D data to 2D at the server regarding the user's viewpoint and streams 2D videos, is another alternative [17–19]. This is extremely challenging because the round trip time has to be below 16ms to prevent motion sickness [12]. Multiview RGB-D streaming, however, relieves such issues as 2D video codecs are highly optimized. The data rate and compression latency are significantly smaller.

The results show that the receiver-loaded pipeline, which does not require 3D streaming, is preferable over sender-loaded and server-aided pipelines. Note that the three possible pipelines are conceptualized to show the streaming options, and many variants exist.

4 Challenges

The baseline pipeline has the merits of efficient 3D data streaming but still involves multiple computational challenges. In this section, we characterize its performance in terms of latency and reconstruction quality to discover the remaining challenges in the system design. We measure them all on iPhone 12 Pro.

Table 4: Processing latency (ms) for rendering stage.

	Raycasting (per view)			Mesh-based rasterization	
	Visibility Check	Depth Estimation	Cast Ray & Rendering	Mesh Generation	Rasterization
CPU	4(±0.41)	17(±1.87)	599(±63.17)	2254(±273)	–
GPU	3(±0.04)	11(±1.11)	31(±3.69)	183(±17.79)	<2

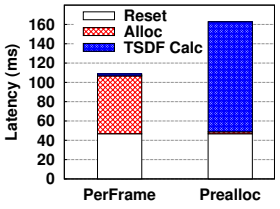

Figure 5: Comparison of TSDF fusion with voxel preallocation.

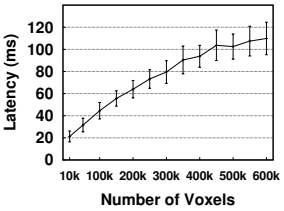

Figure 6: Mesh extraction latency for different number or voxels.

4.1 Latency

Multiview Alignment. Table 2 shows the latency breakdown. The face and landmark detection is the main bottleneck taking 149 ms to process three multiview input RGB images. We use state-of-the-art deep learning models for face detection [3] and face landmark detection [13]. The two remaining components, 2D-to-3D lifting and similarity transformation, incur negligible latency.

Fusion. As shown in Table 3, the fusion stage takes a total latency of ≈2 seconds on the CPU and ≈106 ms on the GPU. The performance breakdown shows that the main bottleneck is the voxel resetting and allocation stage, mainly because of the large number of voxels to process. One method to reduce the voxel allocation latency is to preallocate all the voxels in the bounding box that tightly fits the input data and reuse the allocation for subsequent frames. However, this results in the increase of TSDF calculation time as it significantly increases the number of voxels to evaluate, as shown in Figure 5. This implies that we should eliminate the voxel allocation and resetting latency by preallocation. However, the number of voxels to process in runtime should be minimized.

Rendering. Table 4 shows that both methodologies fail to achieve the tight latency requirements. First, the total latency for raycasting is 45 ms per each 640×480 view (i.e., 90 ms for the left and right eyes). Even when the visibility check and depth estimation results are shared between the two views, the total latency of 74 ms exceeds the <16 ms motion-to-photon latency requirement. The mesh-based rasterization, on the other hand, enables real-time rendering (i.e., < 2 ms latency) independent from the mesh generation throughput. However, mesh generation latency is proportional to the number of voxels, as shown in Figure 6. The latency easily exceeds 33 ms (i.e., the 30 fps camera source frame latency) even with 100k voxels and becomes as high as 180 ms for realistic rendering (e.g., 1M voxels).

4.2 Reconstruction Quality

There are two factors that affects the reconstruction quality: i) inaccurate multiview alignment and ii) temporal fluctuations from missing or noisy RGB and depth data.

Multiview Alignment. We can leverage robust facial features for alignment, but the accurate alignment is not straightforward due to the following challenges: i) lightweight landmark detection models for mobile device suffers from frequent errors that leads to inaccurate alignment results, and ii) more accurate model still suffers from finding the accurate alignment due to the detection inconsistencies across multiple views. Figure 18 shows the failure cases of lightweight landmark detection and multiview inconsistencies.

Temporal Fluctuations. Our experiments show that three cameras are sufficient to capture the majority of the face regions. However, the reconstruction quality suffers from temporal fluctuations. The main cause is the continuous movements of the face, which induces partial occlusions leading to missing depth data. Especially the region between the nose and the cheeks suffers from frequent occlusions (Figure 1(g)). Additionally, the depth cameras based on Time-of-Flight (ToF) active sensors suffer from unexpected noises. The fundamental reasons behind the noises are not in the scope of this work. However, we discover that regions with strong lighting generate noisy data, also shown in Figure 1(g) [4].

5 System Overview

5.1 Design Goals

High Immersiveness. Our primary goal is to enable a live 3D face telepresence system with a high level of immersion and presence comparable to that of state-of-the-art studio solutions. To do so, we need both realistic reconstruction and low end-to-end latency.

Fully Mobile System. We aim to design a plug-and-play system. We assume no external servers for computation or any offline user-specific training process with high-quality sensors (e.g., studio pre-scanning, DNN model training).

System Requirements. As the system consists of multiple components, we carefully set the three following latency requirements.

- < 100*ms* End-to-End Latency: This refers to the total latency between the source data (sender user's face) capture time and the render time of the reconstruction results at the receiver. This requirement is required so that the users do not recognize the interaction delay [2, 28].
- 30 fps Throughput: The 3D content at the receiver side should be updated at the same rate as the input rate at the sender's camera. Rapid facial expression changes require at least 30fps to be captured [47].
- < 16 *ms* Motion-to-Photon Latency: This is required to update the scenes in reaction to the receiver's 6 DoF viewpoint to minimize motion sickness [12]. Note that this is a different requirement from the above end-to-end latency.

5.2 System Architecture

Figure 7 shows the system architecture of FarfetchFusion. The sender devices (synchronized over wireless network [1]) capture multiview RGB-D data; we currently use three smartphones for our implementation, but other personal devices (e.g., laptops, tablets, home cameras) can be used in real deployment scenarios. Each sender device performs the front part of the multiview alignment, including face detection and topology anchoring-based landmark detection (Section 6). The results (3D landmarks, head rotation, camera parameters) are concatenated with the encoded RGB-D data (H264 video codec [49]) and individually streamed to the receiver.

The receiver has an offline and online phase to reconstruct and render the face. In the offline phase, the TSDF voxels are pre-allocated. Here, invariant facial areas are marked so that they do not need to be reset or updated across frames. Then, in the online phase, the TSDF values for variant facial areas are efficiently calculated and converted to a 3D mesh to ensure spatio-temporal consistency (Section 7.2). Lastly, the generated mesh is rotated by the head pose and rendered to the screen via rasterization.

In our architecture design, we make a change to the receiver-loaded pipeline to share the computational load between the receiver and sender devices while still using 2D encoding for RGB-D data. In particular, we make the sender device share the computation for the multiview alignment to lessen the load on the receiver, which contributes to achieving 30 fps throughput. Also, we adopt rasterization-based rendering (enabled with our disentangled fusion approach) to achieve both 30 fps throughput and <16 ms motion-to-photon latency. The raycasting-based method inevitably drops the output image resolution to meet the latency.

5.3 Our Approach: Disentangled Fusion

The core of our approach is to leverage the time-invariant face structure (the head, ear, and upper forehead) to update time-variant face regions (eyes and mouth) only and reuse the voxels with no value updates. This is essential in meeting our latency requirement by significantly reducing the number of voxels to process (down to 30% of the original). Our approach is analogous to data or computation caching methods in 2D videos (e.g., inference result caching in live video analytics [33, 55], object and background split rendering in AR/VR [27, 36]). However, new challenges arise in reusing the previous computation in the case of unorganized 3D data.

The main challenge lies in accurately determining which TSDF voxels to reuse from previous frames or to update newly. This requires precise alignments between consecutive 3D frames, but unlike 2D video frames, accurate and efficient 3D frame alignments are known to be a challenging problem. The problem is aggravated with feature-based (e.g., face landmarks) alignment since landmarks from dynamically changing facial expressions are topologically inconsistent [54]. Furthermore, even after alignment, voxel-wise computation is still necessary to identify variant TSDF voxels from invariants; even for invariants, reusing the previous results require per-voxel coordinate transformation. This incurs a significant overhead with an increasing number of voxels due to limited mobile GPU cache size [22] (Figure 5).

We devise a *topology anchoring* technique to tackle the challenges. The key idea is to generate a pseudo-static anchor, which have the same facial landmark features as the changing inputs for topologically consistent alignment, but maintains a consistent head positionThe consistency of the head position allows the invariant regions to occupy fixed-position voxels in the anchor space after alignment. Thus, separating invariant and variant voxels can be pre-determined offline, reducing the runtime overhead of allocation and resetting. Then, only the variant voxels require TSDF value updates and mesh generation reducing the latency. Additionally, the invariant voxels and some variant voxels that were not updated can be reused to handle missing or noisy depth data.

6 Topology Anchoring

Topology anchoring requires two features: i) the alignment between multiview, temporal RGB-D inputs should be based on topologically consistent features (i.e., pseudo-static anchors) for robustness, and ii) finding the alignment should run in real-time (e.g., 30 fps) on mobile devices.

6.1 Why 3DMM as Topology Anchor?

Pseudo-static anchors are necessary to find accurate spatio-temporal alignment, which is essential for our disentangled

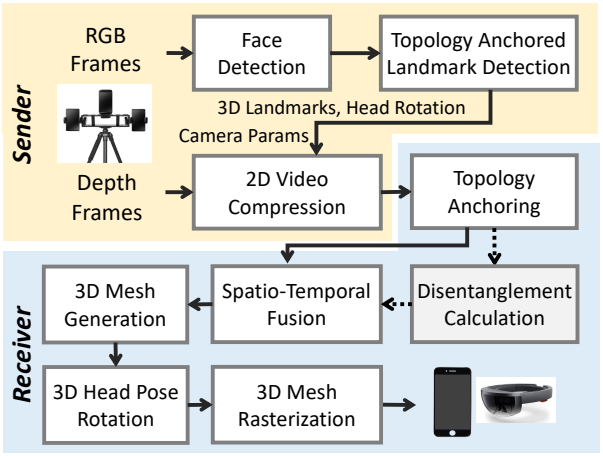

Figure 7: System architecture of FarfetchFusion**.**

fusion approach. Specifically, an intuitive way of temporal alignment is to use the 3D reconstructed face from the first and single-view RGB-D data as an anchor. The subsequent input frames from different views and times can align the detected 3D face landmarks to the anchor using least-square rigid alignment algorithms [52]. However, the anchor and the input frames have landmarks with different topologies (i.e., dynamically changing eye and mouth movements). It is theoretically proven that the least-square algorithm guarantees accurate alignment only if the two sets of 3D points are topologically identical [52]. Furthermore, consistency is not guaranteed for spatial frames even with the same facial expressions (Figure 8). The landmark detection based on 2D images generates reprojection errors between multiple views [54].

We realize pseudo-static anchoring with a different method inspired by 3D morphable models (3DMM). 3DMM is a generic template face model which can be modified by a set of parameters to fit user-specific shapes and expressions. Recently, deep learning-based approaches were proposed to estimate a set of information to morph the face model to the target user with high accuracy [11, 13, 32, 46, 50, 51, 57, 58][1]. The information includes i) geometry coefficients, which are used to represent the unique facial structure of each person and facial expressions, ii) the head rotation matrix, which represents the global rigid transformation of the template, and iii) camera parameters, which are used to project the reconstructed 3D face into the 2D input space. This means that we can create a topology anchor customized to the user's face shape and dynamic facial expression eliminating the transformation from head rotation and camera projection. Then, it is guaranteed that the features used for alignment between the input data and the anchor are topologically consistent.

[1]These papers refer to this task as 3D reconstruction. Note that the reconstruction here generates avatar-like results, far from being realistic. We target realistic reconstruction directly using RGB-D data. Refer to Section 11.

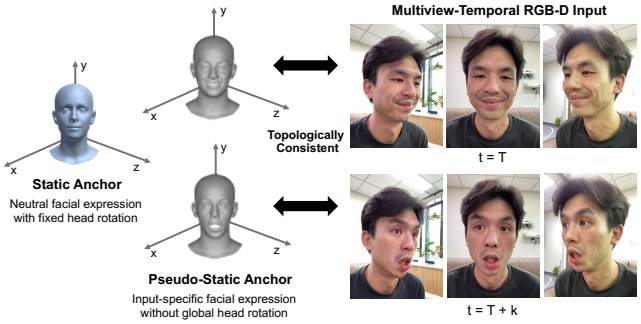

Figure 8: Generating pseudo-static anchors for accurate spatio-temporal topology anchoring.

3DMM-based topology anchoring brings other advantages. As we use facial landmarks for feature-based alignment, extracting accurate landmarks is essential. The 3DMM pipeline can be used as a robust 2D/3D landmark detector, outperforming the performance of 2D-based landmark detection methods. The aforementioned deep learning-based 3DMM methods leverage 2D and 3D landmark loss as key component to improve the morphing quality. The landmark loss refers to the difference between ground-truth 2D landmarks and the estimated landmarks from the reconstructed 3DMM model projected onto the 2D image. Thus, our 3DMM-based topology anchoring method does not incur additional overhead compared to using landmark-based alignments. Lastly, state-of-the-art 3DMM-based reconstruction methods require only a single-pass DNN execution (e.g., ResNet50), which can run in real-time (>30 fps) even on mobile devices. We refer to this model as 3DMM-DNN hereinafter.

6.2 3DMM-based Topology Anchoring

We detail how we generate the pseudo-static 3DMM anchors for multiview-temporal alignment. We first find the pseudo-static anchor's 3D landmarks by running 3DMM-DNN on the RGB frame. The 3D landmarks are projected to 2D and lifted with the depth data to extract the 3D landmarks of the RGB-D input. These two sets are used for topologically consistent alignment.

Applying 3DMM-DNN. The output is a set of parameters including geometry coefficients, head rotation matrix, and camera projection parameters. The geometry coefficients are first used to morph the generic 3DMM template to the target user represented as a mesh. We employ state-of-the-art solution named DECA [13]. Specifically, DECA is based on a non-linear 3D morphable face model, FLAME [30], with linear shape/expression blendshape coefficients and non-linear joint-wise transformation. It starts from a template model represented as a mesh with a set of vertices ($\overline{\mathbf{T}} \in \mathbb{R}^{3N}$, $N = 5023$ vertices (x, y, z)). The linear blendshapes are first applied to create an offset from the template towards the target face shape ($\mathbf{T_S} \in \mathbb{R}^{3N}$) as

$$\mathbf{T_S} = \overline{\mathbf{T}} + \sum_{n=1}^{|\overrightarrow{\beta}|} \beta_n \mathbf{S_n} \tag{1}$$

, where $\overrightarrow{\beta} = [\beta_1, \beta_2, ..., \beta_{|\overrightarrow{\beta}|}]^T$ are the shape/expression coefficients, and $\mathcal{S} = [\mathbf{S_1}, \mathbf{S_2}, ..., \mathbf{S}_{|\overrightarrow{\beta}|}]^{\mathbf{T}} \in \mathbb{R}^{3\mathbf{N}\times|\overrightarrow{\beta}|}$ are the orthonomal shape/expression basis vectors. The non-linear joint-wise transformation includes translation and rotation of four joints, neck, jaw, and two eyeballs. The translation of the joints is already modeled by the previous linear blendshapes. It is calculated by a predetermined joint regression matrix ($\mathcal{J} \in \mathbb{R}^{K\times N}$, $K = 4$), which estimates the joint positions ($\mathbf{J_p} \in \mathbb{R}^{K\times 3}$) from the morphed mesh vertices.

$$\mathbf{J_p} = \mathcal{J}\mathbf{T_S} \tag{2}$$

The joint rotations ($\mathbf{J_R} \in \mathbb{R}^{\mathbf{K}\times\mathbf{3}\times\mathbf{3}}$), represented as rotation matrices, are predicted as a parameter by 3DMM-DNN included in the geometry coefficients. As the rotation of the parent joints affects the child's joints, the joint transformations are represented as relative transformations starting from the global head rotation. Thus, the final transformations are represented as $\mathbf{J_T} \in \mathbb{R}^{(\mathbf{K+1})\times\mathbf{4}\times\mathbf{4}}$, where $K + 1$ includes the global head rotation and four joints, and the rotations and translations are represented in homogeneous coordinates. We can obtain the final vertices of the mesh as

$$\mathbf{T} = \mathcal{W}\mathbf{J_T}\mathbf{T_H} \tag{3}$$

, where the skinning blendweights $\mathcal{W} \in \mathbb{R}^{\mathbf{N}\times(\mathbf{K+1})}$ calculates how the joint transformations are applied to the vertices and $\mathbf{T_H} \in \mathbb{R}^{N\times 4}$ is the vertices after applying linear blendshapes ($\mathbf{T_S}$) represented in homogeneous coordinates. The weighted joint transforms ($\mathcal{W}\mathbf{J_T}$) are 4×4 transformation matrices for each vertex (4×1 vector). We can obtain the pseudo-static anchor, which reflects only the facial expressions but not the global rotation, by applying the inverse of the global head rotation matrix to the final vertices (T).

3DMM 2D/3D Landmark Extraction. The final vertices are used to calculate the 3D landmarks. Similar to the joint regression matrix, the landmark embedding information is a predetermined set of mesh triangle indices and their barycentric coordinates. The 2D landmarks can be extracted by projecting the 3D landmarks to the 2D image space using the camera projection parameters also predicted by the 3DMM-DNN.

$$\begin{aligned} \mathbf{L_{mesh3D}} &= \mathcal{F}(\mathbf{T})\mathbf{B_L} \\ \mathbf{L_{2D}} &= \mathcal{P}_{\mathbf{cam}}(\mathbf{L_{mesh3D}}) \end{aligned} \tag{4}$$

, where $\mathcal{F}$ extracts the three vertex coordinates of the indexed mesh triangles with landmarks ($\mathcal{F}(\mathbf{T}) \in \mathbb{R}^{\mathbf{A}\times\mathbf{3}\times\mathbf{3}}$), $\mathbf{B_L} \in \mathbb{R}^{\mathbf{A}\times\mathbf{3}\times\mathbf{1}}$ represents the barycentric coordinates, $\mathbf{L_{mesh3D}} \in \mathbb{R}^{\mathbf{A}\times\mathbf{3}}$ is the 3D mesh landmarks, $\mathbf{L_{mesh2D}} \in \mathbb{R}^{\mathbf{A}\times\mathbf{2}}$ is the 2D landmarks, and $\mathcal{P}_{\mathbf{cam}}$ is the orthographic camera projection function including scaling and translation.

Figure 9: Generated mask example for offline disentanglement calculation.

Anchoring. We can now align the input RGB-D data to the topology anchor using the detected landmarks. We first lift the 2D landmarks to 3D landmarks using the depth information. We refer to this 3D landmark as point cloud landmark ($\mathbf{L}_{\mathbf{pc3D}}$), while the landmark directly extracted from the 3DMM as mesh landmark ($\mathbf{L}_{\mathbf{mesh3D}}$). The alignment between the point cloud and mesh landmarks can be found with the least-square estimation algorithm [52], with a higher guarantee of topological consistency as they share the same 2D landmarks. Finally, the transformed RGB-D data can be calculated as

$$\mathbf{P} = \mathcal{P}_{\mathbf{head}}^{-1}(\mathcal{P}_{\mathbf{cam}}^{-1}(\mathcal{A}(\mathbf{P}_{\mathbf{RGB-D}}))) \tag{5}$$

, where $\mathcal{A}$ is the alignment matrix between $\mathbf{L}_{\mathbf{pc3D}}$ and $\mathbf{L}_{\mathbf{mesh3D}}$, $\mathcal{P}_{\mathbf{cam}}^{-1}$ is the inverse camera projection matrix, and $\mathcal{P}_{\mathbf{head}}^{-1}$ is the inverse global head rotation matrix. The anchored RGB-D data can now be utilized for volumetric fusion.

7 Pipeline Design and Implementation

7.1 Offline Disentanglement Calculation

We first conduct a short offline scanning phase to reduce runtime computation. We preallocate all the voxels that require TSDF calculation, both invariants and variants determined by landmark-based masking, to eliminate the per-frame voxel allocation latency (Table 3). This enables our system to calculate the TSDF values and generate the 3D mesh only for the invariant regions at runtime. Also, the results are cached and directly reused in subsequent frames. Eliminating computation of the invariant regions reduces the number of per-frame voxels, effectively reducing the latency.

Offline Scanning. During the initialization stage before the telepresence session starts (e.g., 1~5 seconds), we capture the user moving from a neutral facial expression to the one where one's mouth and eyes are open to the maximum extent.

Mask Generation We run topology anchoring for every input frame and generate a region mask. The purpose of the region mask is to segment the captured face into variant regions, invariant regions, and interpolation regions. The interpolation region is an overlapping region between the variant and invariant to smooth out the boundaries of the two regions. We reuse the 2D landmarks to easily calculate the region masks. The variant regions are determined by a filled polygon using the landmarks of the two eyebrows and the jaw lines as the contour. We enlarge the polygon by a percentage to generate the interpolation region (e.g., 120% enlargement used heuristically). The remaining regions are determined as the invariants. The region mask is represented as a single-channel image, the variant regions with a pixel value of 255, the overlapping regions with 127, and the invariants with 0 (Figure 9).

```
struct Voxel{
    short sdf;
    uint8_t clr[3];from the anchoring transformation matrix
    uchar salient;
    uchar padding[2]; // for 8 bytes aligned struct
};
```

Voxel Allocation and Invariant Region Caching. The generated masks are utilized to pre-allocate the voxels that require TSDF value calculation. We add an additional salient variable to the voxel struct to represent which regions the voxels are included in. During the allocation stage, we set the salient variable according to the region mask. If the current voxel is raycasted from the invariant regions, the voxel's salient value is set to 0, 1 for the interpolation region, and 2 for the variant region. Voxels included in the variant regions at least once during this stage are allocated as variants. The TSDF values of the voxels from invariant regions are precalculated and reused for subsequent frames.

7.2 Spatio-Temporal Fusion

Algorithm 1 shows TSDF calculation procedure for each preallocated voxel in the online phase. If the voxel saliency value is zero (i.e., included in the static region), the calculation is skipped and reuses the cached results from the offline phase. If the voxel is included in the variant region, the voxel SDF and the color value are updated using the weighted values of the multiview inputs. Lastly, for the voxels included in the interpolation regions, the voxel values that were calculated in the previous frames are interpolated with the current weighted multiview inputs. We implement this process in a single GPU kernel function for optimized performance.

Additionally, our pipeline can improve the reconstruction quality by minimizing temporal fluctuations. We first prevent false reconstruction from noisy depth data (e.g., compression artifacts or sensor defects) by adopting the confidence-based TSDF weighting technique for spatial fusion [28]. Each pixel i of the depth frame is associated with a confidence value ($conf_i$) calculated by comparing it with nearby pixels as

$$conf_i = min(0.001/\sigma_i, 1.0)$$
$$\sigma_i = \sqrt{\frac{1}{|N_i|}\sum_{k \in N_i} min((d_i - d_k)^2, \mu^2)} \tag{6}$$

Algorithm 1 Spatio-Temporal TSDF Calculation for Disentangled Fusion

```
salient ← voxel.salient
thresh ← CONFIDENCE_THRESH
size ← VOXEL_SIZE
if salient is 0 then return                    ▷ invariant region
else if salient is 1 then                      ▷ interpolation region
    sdf ← voxel.sdf
    clr ← voxel.clr
    sdf_cnt ← 1
else if salient is 2 then                      ▷ variant region
    sdf ← 0
    clr ← 0
    sdf_cnt ← 0
for each view input rgb, depth, confidence do
    image.x, image.y ← view.mat * voxel.pos.xy
    if confidence[x, y] ≥ thresh then
        sdf ← sdf + (depth[x, y] − voxel.pos.z) * size
        sdf_cnt ← sdf_cnt + 1
    if !isColor then
        clr ← rgb[x, y]
        isColor ← true
voxel.sdf ← sdf/cnt
voxel.clr ← clr
```

, where N_i are the 7×7 nearby pixels, d_i, d_k are the depth values, and μ is the TSDF voxel size ($\mu = 0.02$ in our case). In FarfetchFusion, we simply ignore depth values with confidence values less than a certain threshold as in [28]. The reconstruction errors due to missing depth data from occlusions are handled by temporal fusion. As topology anchoring allows accurate temporal alignment, the TSDF values from even the variant regions of previous frames can also be reused. Applying this technique without anchoring would leave a noticeable trace from previous frames. As such, disentangled fusion with topology anchoring reduces the latency significantly by only processing the variant regions while improving the consistency of the reconstruction quality.

7.3 Rendering with Cached Mesh

The last stage is rendering with disentangled invariant and variant regions. The mesh triangles corresponding to the invariant regions are generated in the initialization stage, and only the meshes of the variant and interpolation regions are updated in the subsequent frames. The mesh rendering pipeline on mobile GPUs takes mesh vertex positions and the indices of the vertex comprising the triangles as input. The mesh triangles of the invariant regions are first calculated in the offline phase and stored in memory. Then, the variant and interpolation mesh triangles calculated in runtime are concatenated to the invariants and committed to the rendering queue. As the mesh is still in the anchor space without head rotation, the rendering requires applying the global head rotation matrix before rasterization with the user's 6 DoF viewpoint. This can be done with no overhead, as it is equivalent to multiplying a transformation matrix by the user's view matrix.

8 Implementation

Sender. The sender is implemented on iOS with Objective-C++ and C/C++. The hardware setup for the sender device is shown in Figure 15. We leverage multiple smartphones equipped with active depth sensors (e.g., iPhone 12 and 11 pros) to capture multiview RGB-D frames (inter-camera distance of 15cm and angle of 30°). We use the Network Time Protocol (NTP) [1] for multiview camera time synchronization. Depth data is captured with the highest supported resolution (640×480) with corresponding RGB data. We train the facial landmark detection model for topology anchoring with PyTorch, convert it to a Tensorflow-Lite model, and integrate it into the mobile application using the iOS TensorFlow-Lite C API. For multiview RGB-D frame compression, we use the H264 codec with YUV420 chroma subsampling, which achieves the highest efficiency on commercial mobile phones [14]. For RGB frame compression, we apply the quantization parameter to 21. For depth frame compression, we first quantize the frames into 8 bits and store the data in the Y-channel (UV channels are set to gray).
Receiver. The receiver application is also implemented on iOS with Objective-C++ and C/C++. It runs on iPhone 13 Pro Max running on iOS 14.7.1 equipped with an A14 Bionic chipset (Hexa-core CPU (2×3.1 GHz GHz Firestorm + 4×1.8 GHz Icestorm) and a 4-core GPU with 6GB RAM). The multiview TSDF fusion pipeline is implemented on top of the InfiniTAM library [44]. The GPU functions are implemented with the iOS Metal library. The sender and receiver devices are both connected to Wi-Fi through a commodity 802.11ac AP at 5 GHz. The average downlink and uplink throughputs are 102 Mbps and 85 Mbps, respectively.

9 Evaluation

9.1 Experimental Setup

Dataset. As there are no benchmark datasets for mobile 3D live face telepresence, we collect our custom dataset using our implementation. We recruit a total of 10 people (5 females and 5 males, 6 Asians, 3 white, 1 Hispanic, 25-35 age group). Each person is captured for three sessions in total. The first session captures various facial expressions by following the expressions shown in the sample image. In the second and third sessions, the user is given a set of text paragraphs to

read out loud while freely moving the head and changing the facial expressions.

Receiver Baseline. (*multiviewFusion*): we build a multiview fusion pipeline based on a state-of-the-art volumetric-fusion system [24], which is known to achieve real-time performance on mobile devices. However, since this system was initially built for reconstructing static scenes through scanning doing spatial fusion over temporal frames. We directly extend this system to enable spatial fusion for multiview RGB-D inputs.

Sender Baselines. The main component of the sender is the temporal multiview fusion with DNN-based landmark detectors. Hence, we compare the alignment accuracy and latency with multiple state-of-the-art deep learning methods: i) *lightweightFusion* uses the MediaPipe face landmark detection model that achieves 100 fps on mobile devices and ii) *modelFusion* uses the 3DMM-based landmark detection model for higher accuracy.

9.2 Overall Performance

Figure 10 shows the end-to-end latency performance of FarfetchFusion. We achieve all three latency goals: 30 fps throughput, <16 ms motion-to-photon latency, and <100 ms of end-to-end latency. Specifically, the latency of the volumetric fusion pipeline at the receiver side improved by 6.45×. The TSDF fusion latency reduced from 60 ms to 8 ms and the mesh generation latency from 160 ms to 28 ms. We also improved the perceptual quality. Our user study result shows that the perceptual quality of FarfetchFusion was noticeably better than the *multiviewFusion* baseline, mainly due to the alignment robustness of our topology anchoring and smooth surface generation with spatio-temporal fusion.

9.3 3D Reconstruction Quality

9.3.1 Quantitative Analysis

We provide a quantitative analysis of the temporal alignment accuracy and the effectiveness of spatio-temporal fusion in filling the missing geometry. Temporal alignment accuracy is measured by RMSE (mm) between randomly selected two reconstruction results from different time frames. Better temporal alignment would lead to smaller RMSE values. Figure 12 shows that quantitatively there was minimal difference from the baselines (*modelFusion* and *lightweightFusion*). However, the qualitative result in the following section shows that misalignment can lead to perceptual quality degradation.

9.3.2 Qualtitative Analysis

Figure 16 shows the visual example of 3D reconstruction results of FarfetchFusion and baseline *multiviewFusion*. FarfetchFusion generates noticeably smoother results with less number of missing regions compared to *multiviewFusion* while reducing the computation complexity. The alignment results are also compared in Figure 18. FarfetchFusion provides consistent alignment results regardless of facial expression or head poses while the misalignment from modelFusion and lightweightFusion leaves artifacts in the boundaries

9.3.3 User Study

For qualitative analysis, we conduct a user study with our 3D reconstruction results. Through the user study, we aim to evaluate (i) the overall reconstruction quality of FarfetchFusion, and (ii) the spatio-temporal alignment accuracy.

User Study Procedure. We conducted a user study with 7 users (3 females and 4 males, all in the age group 25-35). We conducted two sessions per user. Within each session, the user views the reconstruction results of two different people from our dataset. Each session was conducted using two devices with different screen sizes: iPhone 13 Max and an iPad Pro that has a bigger screen. In both sessions, the user views the 3D reconstruction result in 6 DoF. Each user views four different rendering options in random order: FarfetchFusion, *multiviewFusion*, *lightweightFusion*, and *modelFusion* for detailed pairwise comparison. Afterward, the users fill out a questionnaire asking (i) a pairwise comparison of different rendering options, (ii) alignment accuracy comparison, (iii) and ranking of the overall quality.

Results. First, all users reported that the overall reconstruction results of FarfetchFusion have a much higher quality compared to *multiviewFusion*. No user noticed the inconsistency of FarfetchFusion even when the invariant and variant regions were processed separately. Second, in terms of alignment accuracy, the perceptual alignment accuracy of FarfetchFusion was the highest from all users compared to the two baselines. All participants gave the unified opinion that FarfetchFusion has the most smooth reconstruction with fewer holes. One participant also commented that the overall shape of FarfetchFusion stayed consistent which makes it look most natural. This was an unexpected benefit of our disentangled fusion approach (i.e., fixing the invariant region).

9.4 Performance of Disentangled Fusion

We conducted an ablation study to show the effectiveness of the individual techniques in our disentangled fusion approach. Figure 13 shows the latency breakdown of the 3D reconstruction stage. Spatio-temporal fusion, only updating the variant region, reduces the latency by 2.8×. Optimizing the GPU code by integrating the separate GPU kernels per view RGB-D frame into a single GPU kernel code improved the latency by two-folds. The elimination of the preallocation stage reduced the latency by 20 ms. Note that in the

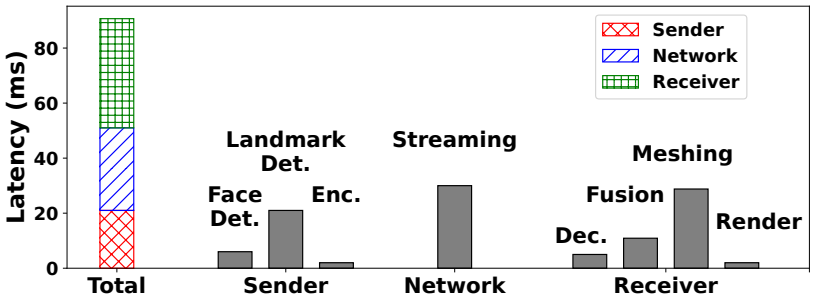

Figure 10: End-to-end latency of FarfetchFusion.

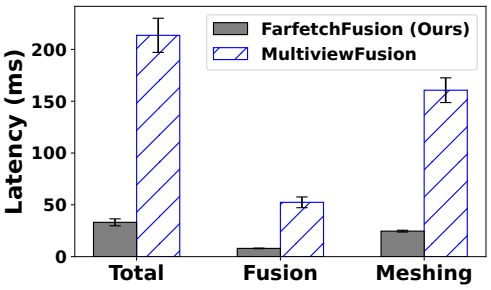

Figure 11: Latency result of FarfetchFusion **receiver.**

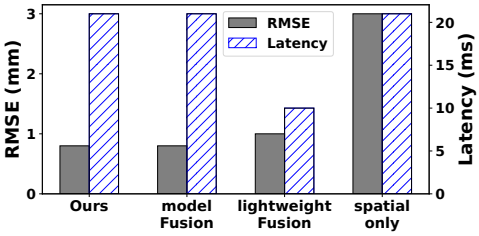

Figure 12: Accuracy and latency result of FarfetchFusion **sender.**

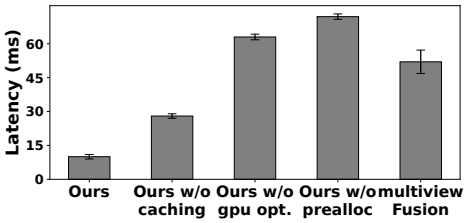

Figure 13: 3D reconstruction latency breakdown.

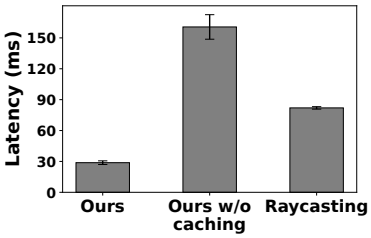

Figure 14: Rendering latency breakdown.

multiviewFusion baseline, the elimination of per-view allocation with preallocation rather increases the latency as the number of allocated voxels is inevitably larger (Figure 5).

Computation caching of the invariant regions was also effective in the rendering stage as shown in Figure 14. With caching, the latency could be reduced by 5.7×. Without applying our technique, the mesh generation stage incurs significant overhead in the pipeline taking 183 ms per frame, which is even heavier than using raycasting.

9.5 Performance on Other Mobile Devices

Figure 17 shows the latency comparison across various devices. The end-to-end latency of FarfetchFusion was achieved on iPhone 13 Pro Max, one of the latest mobile devices. The latency on older mobile devices was higher, however, the trend shows that future mobile devices would allow FarfetchFusion achieve less than 100 ms end-to-end latency.

10 Discussion and Future Works

Further Quality Improvement. FarfetchFusion achieves higher perceptual quality compared to the baselines, however, the reconstruction quality still needs improvement. FarfetchFusion fully rely on the quality of the multiview depth inputs. Thus, the reconstruction fails when the depth data from multiple views are all ill-captured (e.g., when user rotates the head to extreme angles, inner mouth or detailed eyeball movements, thin geometries like hair). Leveraging data-driven generative techniques [15, 61] to overcome these limitations remains as our future work. There are also possible additional optimizations to improve the texture quality. We only use the RGB frames captured with the same resolution as the depth frames (640 × 480). Higher resolution RGB data (e.g., 1920 × 1080) could be utilized. Furthermore, relighting the face texture regarding the receiver's lighting environment can improve the realism and sense of presence.

Extension to Other Body Parts. We plan to extend our topology anchoring to other body parts and objects as well. There has been an extensive amount of work in creating 3DMMs for different body parts (e.g., hand, body, hair, ears, foot). Other objects can also be represented as a global model (i.e., initial shape) and dynamic deformations. Leveraging these anchors, the key would be to accurately disentangle the rigid transformation that can anchor the current input to the anchor space regardless of movements or deformations.

Extension to AR/VR Devices. We plan to replace the receiver side device with AR/VR glasses for higher-level of immersion. Currently, we rely on 2D devices to create high resolution rendering results. Using AR glasses would be a practical direction to support bi-directional communication.

11 Related Work

3D Face Reconstruction. There is an extensive amount of research in 3D face reconstruction. The most representative ones are modeling-based methods. These methods leverage the semantic properties of the face to create a generic template and fit the template to the target with optimizations [30]. Recent advancements in computer vision enables

Figure 15: FarfetchFusion **sender prototype.**

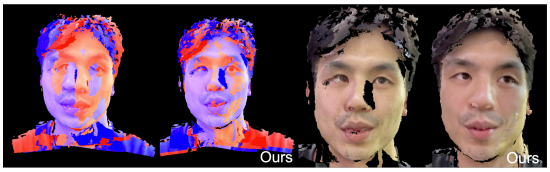

Figure 16: Left: Subsequent frames represented in blue and red before and after spatio-temporal fusion, Right: reconstruction result without and with spatio-temporal fusion (ours).

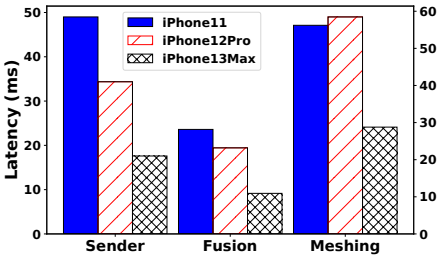

Figure 17: Latency on various mobile devices.

Figure 18: Qualitative comparison of alignment techniques. A: *lightweightFusion*, B: *modelFusion*, Ours: topology anchoring-based alignment of FarfetchFusion**. Best viewed in color.**

reconstruction of the shape and facial expressions [11, 32, 46, 50, 51, 57, 58]. Some uses a single RGB input, enabling numerous applications such as real-time emoji generation [23], video filters for social media, or real-time avatar generation for future metaverse [31]. However, the reconstruction still remains to be unrealistic. The reasons are two folds: (i) face models are represented with a set of finite parameters and (ii) the modeling is highly affected by the data distribution including ethnicity, gender, age, etc. Such limitations average out person-specific details, making the reconstruction results far from realistic and rather avatar-like (Figure 1(c) and (d)). Recent works attempt to overcome the problem by pre-scanning the user's face in high-performance capture studios [9] or using mobile sensors [6], but it is nontrivial to scale them due to costly and time-consuming studio usage or model training. FarfetchFusion adopt the volumetric fusion-based reconstruction method to guarantee realism and newly address computational challenges for mobile deployment.

3D Telepresence Systems. A few immersive 3D telepresence platforms were proposed recently. Google's Project Starline [28] is a 3D telepresence booth that enables face-to-face communications between physically distanced people. It requires four high-resolution RGB-D cameras and server computing resources with wired streaming (four NVIDIA sever-grade GPUs). Pixel codec avatar [9] from Meta Research is a VR-based 3D telepresence system. The runtime system is based on six infrared cameras equipped on the VR headset, which is wired to a single server-grade GPU. However, the realistic reconstruction requires intense offline user pre-scanning session and personalized model training.

Mobile 3D Reconstruction Systems. Existing mobile 3D reconstruction systems that leverages TSDF-based volumetric fusion focus on reconstructing static 3D scenes [38, 42, 56]. Therefore, the key focus of this stream of work is finding robust pose estimation of the scanning mobile camera or enabling 3D reconstruction with monocular video input.

Mobile 3D Streaming Systems. A line of work supports lossless depth compression or temporal compression that minimizes the distortion. However, the compression rate is yet lower than the high-performance video codecs. The main challenges of these approaches are the low compression rate for large-size 3D data and the computational complexity for encoding and decoding. Recent works aim to tackle these challenges by viewpoint-aware partial decoding [20] or enabling fast parallel decoding using mobile GPUs [29].

12 Conclusion

We presented FarfetchFusion, one of the first fully mobile end-to-end systems for live 3D telepresence based on an in-depth analysis of the design space. We proposed a novel disentangled fusion approach that jointly tackles the realistic reconstruction quality and low end-to-end latency, critical limitations of existing works. We realized our approach through topology anchoring, which finds accurate spatio-temporal alignment of dynamic data to maximize the computation reusability. Our results show that FarfetchFusion achieves <100 ms end-to-end latency with realistic reconstruction results.

Acknowledgments

We sincerely thank our anonymous shepherd and reviewers for their valuable comments. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MIST) (No. 2022R1A2C3008495). Youngki Lee is the corresponding author of this work.

References

[1] S. Ansari, N. Wadhwa, R. Garg, and J. Chen. Wireless software synchronization of multiple distributed cameras. In *2019 IEEE International Conference on Computational Photography (ICCP)*, 2019.

[2] M. Baldi and Y. Ofek. End-to-end delay analysis of videoconferencing over packet-switched networks. *IEEE/ACM Transactions on Networking*, 8:479 – 492, 09 2000.

[3] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann. Blazeface: Sub-millisecond neural face detection on mobile gpus. *CoRR*, abs/1907.05047, 2019.

[4] A. Breitbarth, T. Schardt, C. Kind, J. Brinkmann, P.-G. Dittrich, and G. Notni. Measurement accuracy and dependence on external influences of the iPhone X TrueDepth sensor. In *Photonics and Education in Measurement Science 2019*, volume 11144, page 1114407. International Society for Optics and Photonics, SPIE, 2019.

[5] C. Cao, T. Simon, J. K. Kim, G. Schwartz, M. Zollhoefer, S.-S. Saito, S. Lombardi, S.-E. Wei, D. Belko, S.-I. Yu, Y. Sheikh, and J. Saragih. Authentic volumetric avatars from a phone scan. *ACM Trans. Graph.*, 41(4), jul 2022.

[6] C. Cao, T. Simon, J. K. Kim, G. Schwartz, M. Zollhoefer, S.-S. Saito, S. Lombardi, S.-E. Wei, D. Belko, S.-I. Yu, Y. Sheikh, and J. Saragih. Authentic volumetric avatars from a phone scan. *ACM Trans. Graph.*, 41(4), jul 2022.

[7] Z. Chen, T. Funkhouser, P. Hedman, and A. Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *arXiv preprint arXiv:2208.00277*, 2022.

[8] S. Cho, S.-w. Kim, J. Lee, J. Ahn, and J. Han. Effects of volumetric capture avatars on social presence in immersive virtual environments. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 26–34, 2020.

[9] H. Chu, S. Ma, F. De la Torre, S. Fidler, and Y. Sheikh. Expressive telepresence via modular codec avatars. In *European Conference of Computer vision (ECCV)*, pages 330–345, Cham, 2020. Springer International Publishing.

[10] J. Deng, G. Trigeorgis, Y. Zhou, and S. Zafeiriou. Joint multi-view face alignment in the wild. *IEEE Transactions on Image Processing*, 28:3636–3648, 2017.

[11] Y. Deng, J. Yang, S. Xu, D. Chen, Y. Jia, and X. Tong. Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set. In *IEEE Computer Vision and Pattern Recognition Workshops*, 2019.

[12] S. R. Ellis, K. Mania, B. D. Adelstein, and M. I. Hill. Generalizeability of latency detection in a variety of virtual environments. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 48(23):2632–2636, 2004.

[13] Y. Feng, H. Feng, M. J. Black, and T. Bolkart. Learning an animatable detailed 3D face model from in-the-wild images. *ACM Transactions on Graphics (ToG), Proc. SIGGRAPH*, 40(4):88:1–88:13, Aug. 2021.

[14] FFmpeg. Ffmpeg. https://ffmpeg.org/.

[15] G. Gafni, J. Thies, M. Zollhöfer, and M. Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8649–8658, June 2021.

[16] Google. Google Draco. https://google.github.io/draco/.

[17] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai. An overview of ongoing point cloud compression standardization activities: video-based (v-pcc) and geometry-based (g-pcc). *APSIPA Transactions on Signal and Information Processing*, 9:e13, 2020.

[18] S. Gül, D. Podborski, J. Son, G. S. Bhullar, T. Buchholz, T. Schierl, and C. Hellge. Cloud rendering-based volumetric video streaming system for mixed reality services. In *Proceedings of the 11th ACM Multimedia Systems Conference*, MMSys '20, page 357–360, New York, NY, USA, 2020. Association for Computing Machinery.

[19] S. Gül, C. Hellge, and P. Eisert. Latency compensation through image warping for remote rendering-based volumetric video streaming. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 2026–2030, 2022.

[20] B. Han, Y. Liu, and F. Qian. Vivo: Visibility-aware mobile volumetric video streaming. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, MobiCom '20, New York, NY, USA, 2020. Association for Computing Machinery.

[21] Y. Hong, B. Peng, H. Xiao, L. Liu, and J. Zhang. Headnerf: A real-time nerf-based parametric head model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[22] L. N. Huynh, Y. Lee, and R. K. Balan. Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '17, page 82–95, New York, NY, USA, 2017. Association for Computing Machinery.

[23] A. Inc. Apple memoji. https://support.apple.com/en-us/HT208986.

[24] O. Kahler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S. Torr, and D. W. Murray. Very High Frame Rate Volumetric Integration of Depth Images on Mobile Device. *IEEE Transactions on Visualization and Computer Graphics*, 22(11), 2015.

[25] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann. Real-time facial surface geometry from monocular video on mobile gpus. In *CVPR Workshop on Computer Vision for Augmented and Virtual Reality 2019*, Long Beach, CA, 2019.

[26] O. Kähler, V. Adrian Prisacariu, C. Yuheng Ren, X. Sun, P. Torr, and D. Murray. Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Transactions on Visualization and Computer Graphics*, 21(11):1241–1250, 2015.

[27] Z. Lai, Y. C. Hu, Y. Cui, L. Sun, and N. Dai. Furion: Engineering high-quality immersive virtual reality on today's mobile devices. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, MobiCom '17, page 409–421, New York, NY, USA, 2017. Association for Computing Machinery.

[28] J. Lawrence, D. B. Goldman, S. Achar, G. M. Blascovich, J. G. Desloge, T. Fortes, E. M. Gomez, S. Häberling, H. Hoppe, A. Huibers, C. Knaus, B. Kuschak, R. Martin-Brualla, H. Nover, A. I. Russell, S. M. Seitz, and K. Tong. Project starline: A high-fidelity telepresence system. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 40(6), 2021.

[29] K. Lee, J. Yi, Y. Lee, S. Choi, and Y. M. Kim. Groot: A real-time streaming system of high-fidelity volumetric videos. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, MobiCom '20, New York, NY, USA, 2020. Association for Computing Machinery.

[30] T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6):194:1–194:17, 2017.

[31] J. Liang, Y. Liu, and F. Lu. Reconstructing 3d virtual face with eye gaze from a single image. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 370–378, 2022.

[32] J. Lin, Y. Yuan, T. Shao, and K. Zhou. Towards high-fidelity 3d face reconstruction from in-the-wild images using graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5891–5900, 2020.

[33] L. Liu, H. Li, and M. Gruteser. Edge assisted real-time object detection for mobile augmented reality. In *The 25th Annual International Conference on Mobile Computing and Networking*, MobiCom '19, New York, NY, USA, 2019. Association for Computing Machinery.

[34] Y. Liu, B. Han, F. Qian, A. Narayanan, and Z.-L. Zhang. Vues: Practical mobile volumetric video streaming through multiview transcoding. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, MobiCom '22, page 514–527, New York, NY, USA, 2022. Association for Computing Machinery.

[35] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 163–169, New York, NY, USA, 1987. Association for Computing Machinery.

[36] J. Meng, S. Paul, and Y. C. Hu. Coterie: Exploiting frame similarity to enable high-quality multiplayer vr on commodity mobile devices. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, page 923–937, New York, NY, USA, 2020. Association for Computing Machinery.

[37] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022.

[38] O. Muratov, Y. Slynko, V. Chernov, M. Lyubimtseva, A. Shamsuarov, and V. Bucha. 3dcapture: 3d reconstruction for a smartphone. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2016.

[39] 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.

[40] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011.

[41] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. Graph.*, 32(6), nov 2013.

[42] P. Ondrúška, P. Kohli, and S. Izadi. Mobilefusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones. *IEEE Transactions on Visualization and Computer Graphics*, 21(11):1251–1258, 2015.

[43] S. Orts-Escolano, C. Rhemann, S. Fanello, W. Chang, A. Kowdle, Y. Degtyarev, D. Kim, P. L. Davidson, S. Khamis, M. Dou, V. Tankovich, C. Loop, Q. Cai, P. A. Chou, S. Mennicken, J. Valentin, V. Pradeep, S. Wang, S. B. Kang, P. Kohli, Y. Lutchyn, C. Keskin, and S. Izadi. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, page 741–754, New York, NY, USA, 2016. Association for Computing Machinery.

[44] V. A. Prisacariu, O. Kähler, S. Golodetz, M. Sapienza, T. Cavallari, P. H. Torr, and D. W. Murray. InfiniTAM v3: A Framework for Large-Scale 3D Reconstruction with Loop Closure. *ArXiv e-prints*, Aug. 2017.

[45] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *arXiv preprint arXiv:2011.13961*, 2020.

[46] J. Shang, T. Shen, S. Li, L. Zhou, M. Zhen, T. Fang, and L. Quan. Self-supervised monocular 3d face reconstruction by occlusion-aware multi-view geometry consistency. In *European Conference of Computer vision (ECCV)*, 2020.

[47] C. Su and L. Huang. Spatio-temporal graphical-model-based multiple facial feature tracking. *EURASIP Journal on Advances in Signal Processing*, 2005(13):215497, Aug 2005.

[48] S. Subramanyam, I. Viola, A. Hanjalic, and P. Cesar. User centered adaptive streaming of dynamic point clouds with low complexity tiling. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 3669–3677, New York, NY, USA, 2020. Association for Computing Machinery.

[49] G. Sullivan and T. Wiegand. Video compression - from concepts to the h.264/avc standard. *Proceedings of the IEEE*, 93(1):18–31, 2005.

[50] A. Tewari, M. Zollhöfer, P. Garrido, F. Bernard, H. Kim, P. Pérez, and C. Theobalt. Self-supervised multi-level face model learning for monocular reconstruction at over 250 hz. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2549–2559, 2018.

[51] L. Tran, F. Liu, and X. Liu. Towards high-fidelity nonlinear 3d face morphable model. In *Proceeding of IEEE Computer Vision and Pattern Recognition*, Long Beach, CA, June 2019.

[52] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13:376–380, 1991.

[53] D. Wang, P. Chandran, G. Zoss, D. Bradley, and P. Gotardo. Morf: Morphable radiance fields for multiview neural head modeling. In *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22, New York, NY, USA, 2022. Association for Computing Machinery.

[54] Z. Xu, B. Li, M. Geng, Y. Yuan, and G. Yu. Anchorface: An anchor-based facial landmark detector across large poses. In *AAAI Conference on Artificial Intelligence*, 2020.

[55] K. Yang, J. Yi, K. Lee, and Y. Lee. Flexpatch: Fast and accurate object detection for on-device high-resolution live video analytics. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, pages 1898–1907, 2022.

[56] X. Yang, L. Zhou, H. Jiang, Z. Tang, Y. Wang, H. Bao, and G. Zhang. Mobile3drecon: Real-time monocular 3d reconstruction on a mobile phone. *IEEE Transactions on Visualization and Computer Graphics*, 26(12):3446–3456, 2020.

[57] T. Yenamandra, A. Tewari, F. Bernard, H.-P. Seidel, M. A. Elgharib, D. Cremers, and C. Theobalt. i3dmm: Deep implicit 3d morphable model of human heads. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12798–12808, 2021.

[58] X. Zeng, X. Peng, and Y. Qiao. Df2net: A dense-fine-finer network for detailed 3d face reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2315–2324, 2019.

[59] A. Zhang, C. Wang, B. Han, and F. Qian. YuZu: Neural-Enhanced volumetric video streaming. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 137–154, Renton, WA, Apr. 2022. USENIX Association.

[60] C. Zhang, S. Du, J. Liu, and J. Xue. Robust 3d point set registration using iterative closest point algorithm with bounded rotation angle. *Signal Processing*, 120:777–788, 2016.

[61] J. Zhang, X. Li, Z. Wan, C. Wang, and J. Liao. Fdnerf: Few-shot dynamic neural radiance fields for face reconstruction and expression editing. In *SIGGRAPH Asia 2022 Conference Papers*, SA '22, New York, NY, USA, 2022. Association for Computing Machinery.