Immersive Telepresence and Remote Collaboration using Mobile and Wearable Devices

Jacob Young, *Student Member, IEEE*, Tobias Langlotz, *Member, IEEE*, Matthew Cook, Steven Mills, and Holger Regenbrecht, *Member, IEEE*

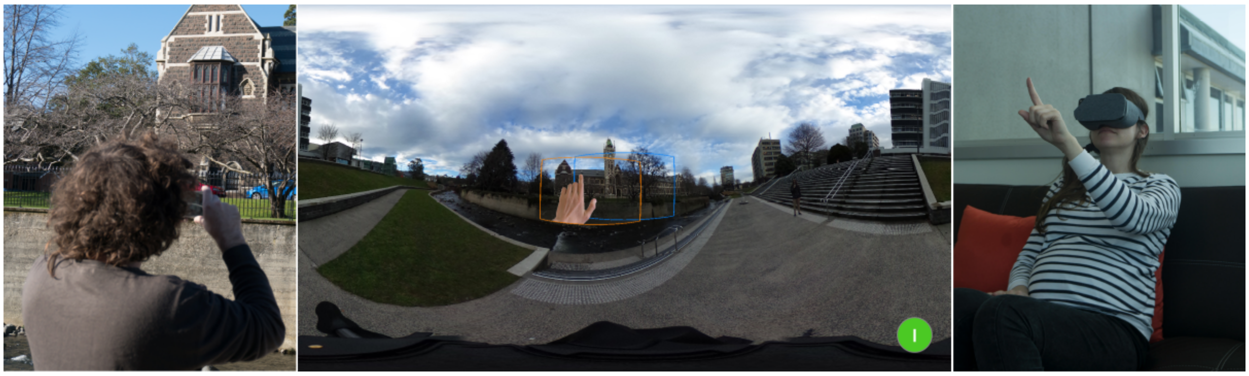

Fig. 1. Our approach for immersive mobile telepresence. (Left): Using their mobile phone, the *local user* creates a spherical mapping of their environment using one of several proposed methods. (Right): A *remote user* views the shared environment through their own mobile phone (with optional head-mounted display) with their viewpoint independent from the direction of the local user's camera. The remote user can point to objects of interest in this environment and have these gestures shown to both users with correct spatial context. (Center): The panoramic environment shared by the two users. Coloured outlines indicate the field of view of each user so they know where the other is looking.

Abstract—The mobility and ubiquity of mobile head-mounted displays make them a promising platform for telepresence research as they allow for spontaneous and remote use cases not possible with stationary hardware. In this work we present a system that provides immersive telepresence and remote collaboration on mobile and wearable devices by building a live spherical panoramic representation of a user's environment that can be viewed in real time by a remote user who can independently choose the viewing direction. The remote user can then interact with this environment as if they were actually there through intuitive gesture-based interaction. Each user can obtain independent views within this environment by rotating their device, and their current field of view is shared to allow for simple coordination of viewpoints. We present several different approaches to create this shared live environment and discuss their implementation details, individual challenges, and performance on modern mobile hardware; by doing so we provide key insights into the design and implementation of next generation mobile telepresence systems, guiding future research in this domain. The results of a preliminary user study confirm the ability of our system to induce the desired sense of presence in its users.

Index Terms—Telepresence, Remote Collaboration, CSCW

1 Introduction

The increase in computational capability of mobile devices has allowed them to power mobile *Head-Mounted Displays* (HMDs) such as Samsung's GearVR or Google's Daydream, which provide immersive *Virtual Reality* (VR) experiences with a unique portability that traditional VR systems are unable to replicate. However, most previous efforts in mobile VR have focused on gaming or 360° video, which do not fully exploit this mobility and ignore the device's original use as a means of communication.

In this work we present a mobile telepresence system that allows distant users to feel *spatially present* (the sense of "being there" [31]) within a shared real-world panoramic environment. Each user has an independent view of this environment based on the orientation of their device, and their current *field of view* (FoV) is shown to their communication partner to aid in the coordination of viewpoints. Either user may optionally wear a mobile HMD to further immerse themselves within the space. We also aim to induce *co-presence* (mutual entrainment [6]—the sense of "being together") between users by supporting natural conversational cues; each user's hands are captured by their device's inbuilt camera and shown in the shared space to allow for intuitive gesture-based conversation. Our use of purely mobile devices ensures this experience can be enjoyed anywhere by anyone, preserving the ubiquity users expect from existing mobile solutions. We see this as a natural extension of these devices' current use as a communication platform and one that would be impractical on desktop systems.

We explore five different approaches to constructing this shared panoramic environment. Each exposes varying levels of view independence between users and between them sample the full continuum of techniques possible in such a panoramic space. The implementation and individual challenges of each will be described in detail as well

- *Jacob Young is with the Department of Information Science at the University of Otago. Email: jacobyoung.research@gmail.com.*
- *Tobias Langlotz is with the Department of Information Science at the University of Otago. Email: tobias.langlotz@otago.ac.nz.*
- *Matthew Cook is with the Department of Information Science at the University of Otago. Email: matthew@cook.run.*
- *Steven Mills is with the Department of Computer Science at the University of Otago. Email: steven@cs.otago.ac.nz.*
- *Holger Regenbrecht is with the Department of Information Science at the University of Otago. Email: holger.regenbrecht@otago.ac.nz.*

Manuscript received 10 Sept. 2018; accepted 7 Feb. 2019.
Date of publication 17 Feb. 2019; date of current version 27 Mar. 2019.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2019.2898737

as an evaluation of their performance on modern consumer devices, allowing for our work to be used as a framework for future research. The results of a preliminary user study are also detailed which confirm the ability of our system to induce the desired sense of presence in its users. This is a culmination of prior research [14, 26, 28, 33] as well as an attempt to overcome these systems' shortfalls and introduce them into a purely mobile setting.

Overall, the contributions of this paper are:

- *A comprehensive discussion and implementation details of approaches to building, updating and sharing a live panoramic representation of a physical environment* in real time on purely mobile devices.
- An approach for incorporating *gestures and mutual field of view awareness* into remote conversation to support more natural interaction between communication partners.
- A *technical evaluation* of each presented approach on modern consumer hardware.
- The results of a *preliminary user study* that confirms the ability of our system to induce spatial and co-presence within the shared environment.
- A *summary of our findings and lessons learned*, providing directions for future research on mobile telepresence systems.

These contributions will be of relevance to the fields of mobile telepresence and mobile virtual environments, which are so far relatively under-explored compared to the large body of work on desktop-based telepresence systems.

2 Related Work

Several attempts have been made to create such an immersive telepresence system, though most tether one or both users to a desktop computer or require specialised hardware not viable for most consumers. In the following we identify what requirements a system must meet to induce a sense of presence between users and evaluate past efforts to determine whether these requirements have been met.

2.1 Requirements for Effective Collaboration

The degree of presence a user achieves within a virtual environment determines how they will perceive and interact with it. Once sufficient presence is achieved, a user's mental model of the virtual environment shifts from one on a screen to one within it, and they begin thinking not in terms of actions performed on an interface but in terms of how these actions affect the virtual elements contained within [31]. Two users achieving presence within a shared environment would feel that they are truly together rather than communicating through some device.

Much work has been done to identify which characteristics contribute to this shared sense of presence. Luff *et al.* [25] believe that social interaction is accomplished largely through objects in the environment, and so users must have an intuitive and effective means of interacting with them for effective communication to take place; this places less emphasis on the face-to-face view employed by most existing systems. This idea is supported by Fussel *et al.* [10], who found that views of the task space and each user's actions and direction of attention within it are more effective for conversational grounding than views of the communication partner's face.

It is important to consider how this task space will be presented to users. Fixing the remote user's view to the direction of the local user's camera is detrimental to collaborative task performance [9] as reorientation within the scene must be done through verbal instruction. Providing the remote user with an independent view of the task space through manipulation of a physical or virtual camera is preferred [11, 12, 14] as it allows them to perform this reorientation themselves, increasing their sense of spatial presence within the environment [14, 26] and significantly reducing the time taken to complete collaborative tasks while achieving similar or increased accuracy [14, 28]. Remote users often utilise this freedom of viewpoint to focus on areas of the task space not seen by the local user [15, 32], allowing them to direct the focus of attention or describe the greater context of the environment without local intervention [33], resulting in greater confidence in the results of collaborative tasks [15, 21].

Coordination of users' viewpoints becomes an issue in such a space. Tang *et al.* [33] and Kuzuoka *et al.* [24] found that participants' inability to see the other's FoV led to frustration as important context was removed from instructions. Reorientation required complex verbal negotiation, and consequently participants would negate the benefits of independent view control by keeping their viewpoints close.

Perhaps the most important aspect of telepresence systems is how they allow users to interact. Gesture-based systems are preferred to traditional ones [12, 18] as they make users feel more coordinated [9] and significantly decrease completion time [10–12] of collaborative tasks with no loss in accuracy [19]. Gestures make identification of task objects easier by reducing ambiguity of deictic ("that one there") references [9] which can lead to verbal communication becoming unnecessary and inefficient [4]. Gestural interaction is thus commonly described as a requirement in telepresence applications [9, 23, 24, 33] and failure to incorporate gestures can lead to frustration as users will attempt to use them even knowing they are not visible [33].

We thus identify three key requirements a telepresence system must fulfil to provide rich communication, collaboration and a sufficient sense of presence between users:

- It must allow users to share a representation of the physical environment that preserves the spatial relationship between objects and users.
- Users must be able to obtain independent viewpoints within the environment to prevent local parties from dictating the direction of attention.
- Conversational cues such as voice, unmediated gestures and FoV awareness must be shared to allow for natural communication.

Fulfilling these requirements will arguably increase the sense of co-presence between users and provide a social or collaborative experience similar to that of side-by-side communication. In addition, we argue that the use of purely ubiquitous and portable hardware is important in realising such a system as this would allow for spontaneous or outdoor use, providing a greater range of use cases and conversational grounding cues than stationary systems [3] while making it available to the average consumer.

2.2 Spatial Videoconferencing Solutions

Several attempts have been made to incorporate a shared environment, view independence, and natural conversational cues into one system, however most are not fully mobile or cannot fulfil some of the above requirements.

Chili [14] allows partial view independence by projecting the users' camera streams into a spherical environment based on their devices' orientation, providing spatial context to images. Jo *et al.* found that this provides a greater sense of spatial presence than traditional videoconferencing systems, however nothing is visible outside of the other user's FoV and interaction between users is limited to drawn annotations.

PanoVC [26] provides users independent views by incrementally building and sharing a live panoramic representation of the local environment. This was shown to increase remote user's sense of spatial presence within the shared space, however co-presence was limited by insufficient representation of the remote user.

Similarly, *PanoInserts* [28] shares a real environment between users by overlaying live video onto a pre-captured static panorama through the use of tracking markers. Requiring these markers and a pre-captured environment makes the system unsuitable for spontaneous use, and the complex stitching results in low frame rates, even on desktop systems.

Polly [22] provides full view independence between users by allowing the remote user to physically reorient the local camera through use of a mechanical gimbal on the local user's shoulder. This positioning

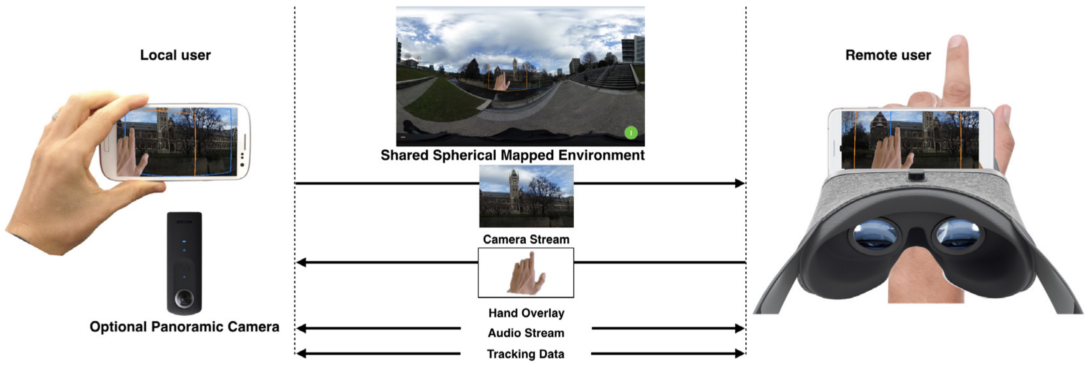

Fig. 2. An overview of the presented panoramic immersive mobile telepresence system. A local user sends their camera stream and device orientation to a remote user where the received data can be used to render different panoramic representations of the shared environment. The remote user can perform hand-based gestures, which are captured and sent to the local user along with their orientation so they can be spatially mapped onto the physical environment. Both sides also exchange audio, allowing for natural conversations as if the two parties are spatially present. Either user may also increase immersion within the environment by viewing it through a mobile HMD.

could make spatially mapping the remote user's actions within the environment difficult [20]; Kratz *et al.* found that this could be mitigated by displaying the remote user's current FoV in a see-through HMD.

2.3 Interaction within Shared Environments

Some telepresence systems place more emphasis on user interaction than how the environment is displayed. *BeThere* [32] provides gesture-based interaction between users by scanning the remote user's hand with a tablet-mounted depth sensor and displaying it within the shared environment; unfortunately the sensor's technical limitations make the system impractical for mobile or outdoor use.

JackIn [15] provides full view independence within a SLAM-tracked real-world panoramic environment. The remote user can point to objects of interest through use of a Leap Motion Controller[1] with the target shown to the local user in an optical see-through HMD. Again, this depth sensor restricts the system to stationary indoor use.

Several systems provide interaction through unmediated video of the user's hands, though these also have their limitations. *HandsinAir* [13] displays these gestures to the local user in a video see-through HMD, though the users' views are coupled so movement of the camera during a gesture could result in a misunderstanding of its intent.

Gauglitz *et al.* improve upon this by allowing for full view independence using homographies [11] or SLAM [12] and overlaying gestures on the remote user's view. However, these techniques are computationally expensive and so rely on more powerful stationary hardware.

These systems aim to provide an immersive telepresence experience by providing independent views, FoV awareness, or gesture-based interaction between users, however most focus on only one of these aspects and nearly all require stationary hardware on one or both ends of the connection. We argue that an increasingly mobile world requires telepresence solutions that allow for point to point communication independent of non-ubiquitous hardware. To the best of our knowledge, our approach is the first that successfully demonstrates how all of these features can be combined into one system and be experienced in real time on mobile devices. As such, our system can achieve levels of presence previously restricted to stationary systems while providing superior portability and ubiquity, potentially enabling many new application scenarios.

3 A Framework for Immersive Mobile Telepresence

We present a framework for an immersive telepresence system that fulfils the previously identified requirements by allowing users to interact within a shared panoramic representation of a real-world environment as if they were spatially present within it. Our solution uses off-the-shelf mobile hardware for both clients and supports communication from anywhere as long as there is sufficient network connectivity.

[1]https://www.leapmotion.com/

The *local user* captures their surroundings, using either their device's integrated camera or an optional external panoramic one, and shares it with a *remote user*, who can obtain independent viewpoints within it by reorienting their device. The two users can then communicate through voice, gestures and shared FoV awareness to allow for natural conversation between them, as illustrated in Fig. 2. The system supports several ways to create this environment, called our *modes of interaction*, which between them sample the full continuum of techniques possible in such a panoramic space. We present these in order of increasing view independence:

- *Live Video Calling:* The local user's camera feed is shown directly to the remote user. This mode functions almost identically to conventional video-calling solutions such as Skype[2] and is included as a point of comparison.

- *Live Spatial Video Calling:* Both users are situated at the center of a virtual sphere, and the local user's camera feed is projected onto its inside surface based on the orientation of their device. Each user can then independently control their viewpoint by reorienting their device, but the remote user still requires that any areas they wish to see are within the local user's FoV.

- *Incremental Panoramic Calling:* Similar to Live Spatial Video Calling, however each frame projected to the inside of the sphere from the local camera is recorded there, which over time creates a panoramic representation of the space. This allows the remote user to view previously seen areas at their leisure, providing static context to the live focus in the local user's FoV.

- *Panoramic Calling with Live Inserts:* A full panorama of the local user's environment is captured and shared ahead of time, allowing the remote user to see content in areas the remote user has not yet visited. We make use of a focus and context technique [7] by projecting the live view from the local user's camera onto this pre-recorded panorama.

- *Live Panoramic Video Calling:* An external 360° camera is used to capture the full environment in real-time. This allows for full view independence, however requiring the panoramic camera reduces this mode's potential for spontaneous use. The local user may wear the camera around their neck, which will give the remote user a more accurate representation of their viewpoint at the cost of obscuring half of the environment, or use a mount such as the one used by Tang *et al.* [33] to capture the full environment.

[2]https://www.skype.com/en/new/

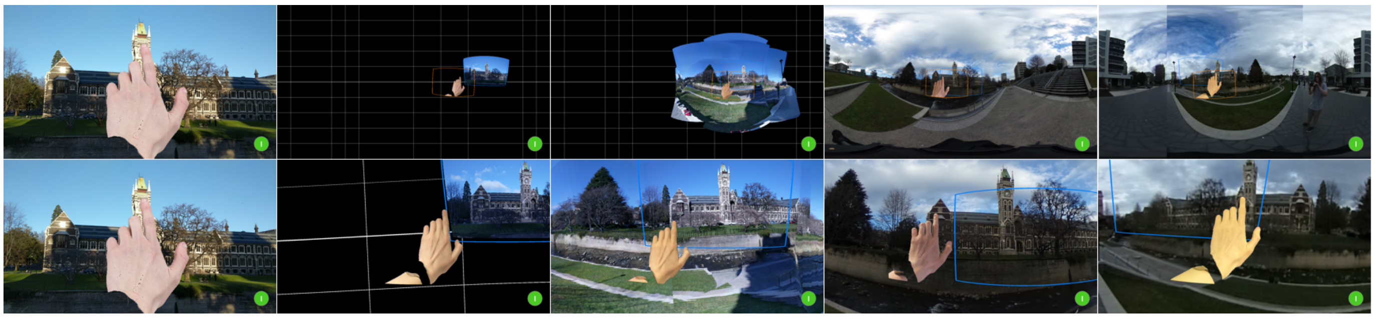

Fig. 3. Each of our modes of interaction in use, from the leftmost column to the rightmost: Live Video Calling, Live Spatial Video Calling, Incremental Panoramic Calling, Panoramic Calling with Live Inserts, Live Panoramic Video Calling. The top row shows an equirectangular projection of the panoramic environment constructed by that mode, with the bottom row showing how it would be seen by the remote user. Both views are the same in Live Video Calling as the two users' views are coupled. The FoV of the local and remote user are shown as a blue and orange outline, respectively.

For ease of reference each mode other than Live Video Calling shall be referred to as our *spatial modes* throughout the rest of this work. An environment constructed using each is shown in Fig. 3. All modes allow the user to view the environment through their mobile phone's display or an optional mobile HMD. An equirectangular projection of the whole environment may also be seen if desired. Each of these representations is shown in Fig. 4.

Since each user can obtain an independent view of the environment, we spatially map their current FoV as a coloured outline to allow for these viewpoints to be coordinated. If either user is outside the other's FoV, the edge of their screen is coloured to show the direction they have turned, allowing each to easily respond to spatially sensitive observations or instructions.

Previous work has shown that gestures are an integral part of everyday conversation [10, 11]. Consequently, we have allowed both users to incorporate their hands into conversation in a way that does not interfere with the representation of the shared environment. The remote user's hands are captured by their device's integrated camera, isolated from the background and projected into the environment within their current FoV. This projection is shared with the local user, allowing the remote user to point to objects of interest or perform other natural conversational cues. Verbal communication is also possible through inbuilt VoIP capabilities.

4 Technical Foundation

Our application is designed to operate on any Android device. To allow for run-time configuration all modes of interaction share the same low-level subsystems which we describe in the following sections. The interactions between these are illustrated in Fig. 5.

4.1 Camera Access

Camera access is performed via Google's Camera2 API which grants direct access to the raw camera stream. Parameters such as focus, exposure and white balance can be controlled at run-time, which is crucial for building panoramas in Incremental Panoramic Calling with consistent illumination and ensuring lighting-independent segmentation of the user's hands.

Images are captured at a resolution of 1280×720 in all modes to reduce bandwidth requirements. As each frame is captured it is stored in two buffers; one is CPU-controlled and passes images to the networking module to be sent to the remote peer, while the other is backed by an OpenGL texture and is used to display the camera stream locally. Frames are captured in NV12 format so require conversion to I420 for the network module and RGBA for rendering. All operations on these images are performed using either OpenCV[3] or OpenGL[4].

[3]https://opencv.org

[4]https://www.opengl.org

4.2 Orientation Estimation

To allow for independent views within the environment we must implement some way to track the orientation of each user's device. Existing solutions such as ARCore[5] and ARKit[6] were considered, though these would limit the application's use to compatible devices and thus compromise its ubiquity.

Two orientation estimation methods are supported that can be used interchangeably at run-time. Both produce a three-dimensional rotation matrix which is encoded in each frame so that the users' FoV and camera images may be positioned correctly within the environment. We restrict movement to 3D rotation as the panoramic representation of the environment is position invariant, which is reasonable for the envisioned use case.

The simplest but fastest method is a fusion of the device's inbuilt sensors and Kalman filtering, as described by Pacha [27]. For most modes of interaction this is acceptable as the calculated orientation does not have to accurately represent reality as long as it is internally consistent. However, sensor drift can cause visible seams in the constructed panorama in Incremental Panoramic Calling; for this reason a vision-based approach is supported which uses feature detection and matching to estimate the device's orientation with higher precision.

For the first frame we assume its rotation to be a combination of the pitch and yaw estimated by the sensor fusion approach and use this as our absolute orientation R_{a-1}. We also find the set of feature points f_{t-1} within the latest camera image using the ORB method [30]. For each subsequent frame we detect its feature set f_t using the same method and find matches between the two sets through a brute force approach, comparing every feature point in f_{t-1} to each in f_t and assigning a similarity score to each pair based on the immediate neighbourhood of their keypoints. The pair with the highest similarity score for each feature is kept and added to a set of matches m if its similarity score is above 70% of the highest possible and is at least double the next best matching pair. Once m is found, we use a homography-based estimator to calculate the relative rotation matrix R_r between the two images, which is then refined through bundle adjustment. We check this estimated rotation against the rotation R_s estimated by our sensor fusion approach by calculating their difference $D = R_r R_s^{-1}$ and discarding the result and its corresponding frame if a discrepancy of more than 5° is found; this is needed since recovering from an erroneous estimate is difficult. Once a relative rotation between the two frames is found, we calculate the absolute rotation of the new frame $R_a = R_{a-1} R_r$, which is used as the device's orientation matrix R.

4.3 Networking and Synchronisation

To facilitate the connection between users we use Google's open-source implementation of WebRTC[7], which allows for efficient sharing of

[5]https://developers.google.com/ar/

[6]https://developer.apple.com/arkit/

[7]https://webrtc.org

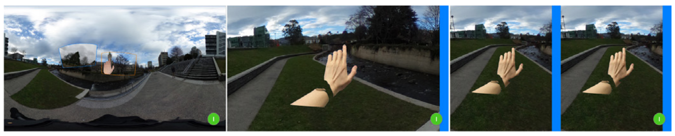

Fig. 4. The ways in which a user may view the shared environment. (Left): An equirectangular projection of the entire panorama. (Centre): A first-person view of the environment based on the orientation of the user's device. (Right): A pseudo-stereoscopic view for mobile HMDs.

audio and video with minimal configuration. Matchmaking requires a central server but communication is completely peer-to-peer.

When users connect to each other, three separate channels are created for each: one for audio encrypted using the Opus codec, one for encrypted video using VP8, and one for data, which is used to send each device's orientation to the communication partner. Since WebRTC provides no means of synchronising video and data packets, we encode an identification tag into two 8×8 regions of pixels in the top left corner of each frame, with the bits encoded in a way that ensures significant difference in luminosity between frames despite the tags being consecutive integers. When an image arrives over the connection, the client waits for the accompanying data packet and reconstructs the frame with the missing data before passing it to the rendering pipeline.

5 Constructing the Shared Environment

Now that we have low-level functionality we can begin constructing and sharing the environment. How this is done differs between modes, though to ensure modularity each is built upon the same foundation which we describe here along with methods for interacting with and viewing the resulting environment.

5.1 Representation of the Shared Environment

Though the various spatial modes differ in terms of how the environment is created, all can be generalised as to provide some virtual spherical space for users to interact within. To ease implementation and allow switching between modes at run-time we store the various environments in exactly the same manner: as a panorama that is updated differently depending on the desired mode.

We store this panorama as a simple two-dimensional texture rather than a more complicated representation such as a point cloud or a 3D mesh due to much lower requirements for memory and computation time. Previous approaches store such a panorama using a cylindrical model [1, 26], which is computationally simple but results in a reduced vertical field of view due to missing information at the latitudinal poles; this is a problem when displaying the panorama in an HMD and may affect the sense of presence. We instead use a spherical model, which allows these poles to be visible. We assume that all rotations of the device are performed with the camera's optical centre as the origin with no movement of the device itself; this creates a discrepancy in the predicted and actual rotation of the environment, though this will be negligible for sufficiently distant objects [8]. For this reason we conceptualise the sphere as infinitely large such that all points on the sphere are at an infinite distance from the origin, making the environment position-insensitive.

The panorama is stored as an equirectangular projection in a texture-backed OpenGL framebuffer. Since the maximum size of any side of a buffer in OpenGL is 4096 texels we use this as its width, and the equirectangular model requires its height to be half of this, giving us a total resolution of 4096×2048. A higher resolution is possible but would require splitting the panorama across multiple framebuffers; the slight increase in quality was not deemed to be worth the extra computational load. All operations on this framebuffer are performed through fragment shaders to maximise performance.

5.2 Projection into Panorama Space

Projecting images into the panorama is a vital part of our application as it allows each user's camera stream to be shown in the correct position

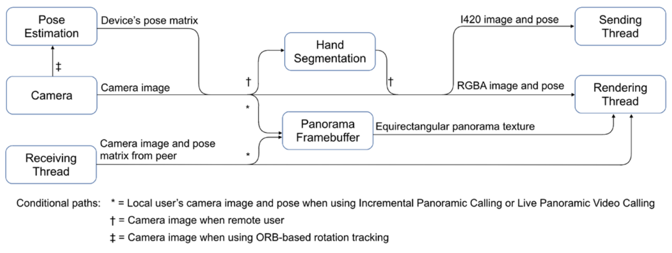
Pose Estimation
Device's pose matrix
Camera
Camera image
Hand Segmentation
I420 image and pose
Sending Thread
RGBA image and pose
Rendering Thread
Panorama Framebuffer
Equirectangular panorama texture
Receiving Thread
Camera image and pose matrix from peer
Conditional paths: * = Local user's camera image and pose when using Incremental Panoramic Calling or Live Panoramic Video Calling
† = Camera image when remote user
‡ = Camera image when using ORB-based rotation tracking

Fig. 5. An overview of the system's shared modules and how they interact. Paths marked $*$, $\dagger$, or $\ddagger$ are only conditionally followed, for example when a specific mode of operation is in use.

in the environment. Projection is performed in a fragment shader during rendering to ensure optimal performance.

We first transform the panorama's texture coordinates $\mathbf{t} = (t_x, t_y)$ to the equirectangular sphere map coordinates

$$(\theta, \phi) = \left(2\pi\left(t_x - \tfrac{1}{2}\right),\ \ \pi t_y\right), \tag{1}$$

where t_x, $t_y \in [0,1]$ and $\theta \in [-\pi, \pi]$, $\phi \in [\frac{-\pi}{2}, \frac{\pi}{2}]$ are the azimuth and inclination on the environment sphere respectively. These are then projected to a unit vector $\mathbf{u} = (u_x, u_y, u_z)$ on this sphere such that

$$\mathbf{u} = \begin{bmatrix} \sin(\phi)\sin(\theta) \\ \cos(\phi) \\ \sin(\phi)\cos(\theta) \end{bmatrix} \tag{2}$$

We then rotate the sphere-space projection based on the device's estimated orientation matrix R, followed by a projection to the camera's sensor space using its intrinsic matrix K. Thus the camera coordinate $\mathbf{u}'$ for a given pixel is given by

$$\mathbf{u}' = KR^T\mathbf{u}. \tag{3}$$

Since pixels near the poles may appear in the camera's forward and back projection, we choose not to render a pixel if the calculated $u'_z \leq 0$. We then normalise the calculated coordinates across the camera's pixel coordinates, giving the final texel coordinates $\mathbf{v} = (v_x, v_y)$ such that

$$\mathbf{v} = \left(\frac{u'_x}{r_x u'_z}, \frac{u'_y}{r_y u'_z}\right) \tag{4}$$

where $(r_x, r_y) = (1280, 720)$ is the resolution of the camera. If both v_x, $v_y \in [0, 1)$ then the texel at $\mathbf{v}$ in the input image is displayed at the current fragment.

5.3 Field of View Awareness

To show each user's current FoV, we colour a fragment blue (for the local user) or orange (for the remote user) if its v_x or v_y, calculated at the end of the projection step, are within some small threshold to 0 or 1. We also colour the fragment if it is near the edge of the screen and the calculated v_x, $v_y \notin [0, 1]$ imply that the area projected to is outside that edge to indicate which direction the user needs to turn in order to see their partner's current FoV.

5.4 View Unprojection

Now that we can store the environment we must implement some way of viewing it. For Live Video Calling we can simply display the local user's camera stream, but for the other modes more sophisticated methods are needed.

The first way of viewing the environment is to simply map the panorama texture to the device's screen, allowing the entire space to be viewed at once. We can alternatively obtain a novel view of the environment for each user by unprojecting a region of the panorama based on the current orientation of their device. To do this, we create a virtual camera with an 82° FoV and intrinsic matrix K and retrieve the user's estimated orientation matrix R. Then for each fragment we cast a unit vector $\mathbf{m}'$ from the centre of the sphere M such that

$$\mathbf{m}' = K^T R^T M, \tag{5}$$

which we normalise to obtain the coordinate $m = \frac{\mathbf{m}'}{\mathbf{m}'_z}$ in the panorama texture to sample from.

For increased immersion within the environment either user may optionally view it through a mobile HMD. Since no depth data is stored we simply split our unprojected view in half vertically and show the same image to each eye. With our intended outdoor use case (and even in most indoor scenarios) most objects are too distant for this lack of depth to be noticeable, and its absence has been shown to have no detrimental effect on collaborative performance [21]. These unprojection techniques are performed within the rendering shader after each user's camera frame has been projected into the environment.

5.5 Hand Segmentation

While the local user's gestures are visible within the existing camera stream, to allow the remote user to perform them their hands are captured using their device's camera, isolated from the background and projected into the panorama using their current orientation. As this segmentation process needs to be robust enough to work in most environments while still fast enough to not affect application performance we use a simple colour-based approach.

Whenever a frame is captured from the remote user's camera, each pixel's colour is checked and is assumed to not belong to a hand if its YUV and corresponding RGB values satisfy the following conditions proposed by Al-Tairi *et al.* [2]:

$$u \in (80,130),\ v \in (136,200),\ r > 80,\ g > 30,\ b > 15,\ |r-g| > 15.$$

Since human skin has similar hue and mostly only differs in lightness [34], ignoring the Y channel ensures compatibility with a range of different skin colours.

To reduce the risk of other skin-coloured objects being included in the results, we remove pixels if their Euclidean distance to the nearest non-skin coloured pixel is below some small threshold. Large skin-coloured regions will still be visible, but as seen in Fig. 6 this method can still give acceptable results given our real-time constraint. More advanced techniques such as GrabCut [29] were considered but were found to be infeasible for real-time use on current mobile hardware.

6 Modes of Interaction

Here we describe our implementation of each of the proposed modes of interaction. All but Live Video Calling utilise the previously described low-level subsystems.

6.1 Live Video Calling

Live Video Calling is the simplest of our modes to implement as it displays the local user's camera stream directly. Once a network connection has been established, each frame captured by each user's camera is sent to their communication partner; to allow the remote user to perform hand gestures their image will first pass through the hand segmentation module. When rendering we then simply sample from the remote user's image if the current texel is determined to belong to a hand and otherwise sample from the local user's image, resulting in the remote user's gestures always being visible.

6.2 Live Spatial Video Calling

Live Spatial Video Calling is more complex as each user's camera image must be correctly mapped within the environment to give it spatial context and so both users' estimated device orientation matrix must also be sent along with each captured camera image. The previously outlined projection step is then used by the rendering shader to project the images and FoV indicators onto the panorama framebuffer, and the unprojection step is then performed to render the environment to the users' screens. Both users are shown a spherical grid in otherwise empty areas to provide them with a consistent sense of orientation.

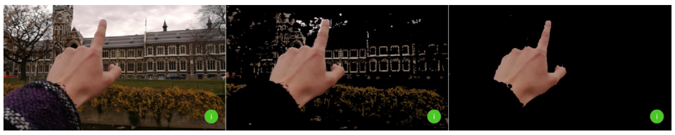
Fig. 6. An example of the segmentation accuracy we can achieve in real-time. (Left): The original image. (Centre): The image after colour-based segmentation. (Right): The segmented image after distance-based thresholding is applied.

While similar to the Chili system [14], we use gesture-based interaction which is preferred to the annotations employed by Jo *et al.* [10, 15] which should increase the system's potential as a collaborative tool [19] and make it more enjoyable for users.

6.3 Incremental Panoramic Calling

Incremental Panoramic Calling builds upon Live Spatial Video Calling by recording each of the local user's camera frames whenever they are projected into the environment to provide additional static context. Over time this constructs a panorama of the local user's surroundings, allowing the remote user to view areas the local user has previously visited at their leisure.

To do this we require an additional step in our application pipeline. Each time a new camera frame arrives from the local user we project it into the environment in a separate shader before the rendering step with the panorama framebuffer as the output rather than the screen. To ensure that the panorama always reflects the most recent view of the space we overwrite each affected texel each frame; due to our real-time constraint we assume that the calculated device orientation is accurate and do not perform any stitching.

To prevent the local user's hand gestures being permanently recorded in the panorama and thus occluding the background it is possible to perform the previously described hand segmentation process during this projection. Each pixel is prevented from being recorded if it is determined to belong to a hand, thus leaving the occluded background information intact. The original image is unaffected and so the gesture will still be visible after rendering.

Because the panorama is used as the output for this step, the shader would usually operate on all 4096×2048 fragments in the framebuffer even though most will not be affected by the projection. To improve performance we overlay the panorama with 144 culling quads and only run the shader on each if it will be affected by the projection. For a camera with diagonal field of view F we can determine if a culling quad will be fully or partially inside the area being projected to if any of its sphere space corners $\mathbf{c}$ satisfy

$$R\mathbf{z} \cdot \mathbf{c} \geq \cos\left(\tfrac{F}{2}\right), \tag{6}$$

where $\mathbf{z}$ is the unit vector along the z axis and R the orientation matrix associated with this frame.

While reminiscent of PanoVC [26], our system differs in a few key areas. We use a spherical panorama rather than a cylindrical one to better accommodate HMD use, and each user only calculates their own orientation, ensuring the environment remains consistent between users and allowing for scalability beyond two parties. Our use of gestures also solves PanoVC's inability to induce significant co-presence between users due to a lack of representation of the remote user.

6.4 Panoramic Calling with Live Inserts

While the device's inbuilt camera can create convincing representations of the environment, the quality of this representation depends on the accuracy of the device's orientation estimates and is also sensitive to temporal changes in the environment. Sensor-based tracking is susceptible to drift over time, and vision-based approaches assume no movement in the environment, so artefacts will likely be present in the panorama in imperfect conditions. Panoramic Calling with Live Inserts

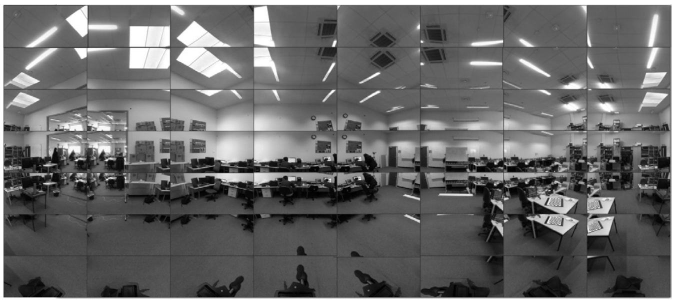
Fig. 7. A pre-captured environment split and unprojected into multiple segments for feature matching. The relative rotation between new camera frames and the segment with the most matches to the new frame is combined with that segment's rotation to calculate the absolute orientation of the user's device.

avoids these shortcomings by constructing the environment before communication starts, allowing for slower, more accurate stitching to be used at the expense of spontaneity.

When the application starts, the pre-captured panorama is loaded from disk into the panorama framebuffer. We assume that the panorama is already equirectangularly projected so this is done with a bitwise copy. Other than this initial step the implementation is identical to Live Spatial Video Calling.

To ensure that objects in the local user's camera stream align with their position in the static environment we must determine where these objects are; to do this, we have implemented an additional orientation estimation method that calculates the relative rotation between the new frame and pre-defined segments of the existing panorama.

Once the panorama is loaded we rotate around the environment sphere at set angles and at each rotation obtain an unprojected view of the panorama. This results in a two-dimensional array of images such as in Fig. 7 which between them show the entire environment. We then detect and store the features and absolute rotation of each of these segments using the ORB method [30].

Each time a new frame arrives from the local user's camera we detect its feature set, again using the ORB method. Since the pitch of an object in the pre-captured environment is likely to match its relative pitch in the real world, we use the pitch estimated through the sensor fusion approach to find a likely row of segments the new image could match. We then match the new image against each segment in this row, take the segment with the most matches and calculate the relative rotation matrix between it and the camera frame, which is multiplied by the absolute rotation matrix of the chosen segment to find the absolute orientation of the device. Matching image features against a full panorama is predictably slow, taking 453ms on average for each frame; for this reason we only perform this absolute tracking every 500 frames and use the calculated orientation as an offset for sensor-based tracking.

Despite the computational limitations of mobile phones, this mode achieves a similar experience to PanoInserts [28] at a much higher frame rate, resulting in a much more comfortable and portable experience for users. We additionally allow for more natural interaction via unmediated gestures, providing an interaction method and representation of the remote user that PanoInserts lacks.

6.5 Live Panoramic Video Calling

While Panoramic Calling with Live Inserts allows the entire space to be shared, its static nature means that the remote user will not be able to see temporal changes in the environment. Live Panoramic Video Calling enables this by utilising an external 360° camera to share the full environment in real time. We use a Ricoh Theta S[8], a portable and inexpensive camera designed for use with mobile phones. The Theta provides its own API via remote HTTP requests, however this requires connecting the mobile device to a dedicated wireless network broadcast by the camera that only accepts one client, making communication with the remote peer impossible. We instead connect to the camera via USB using the UVCCamera library[9].

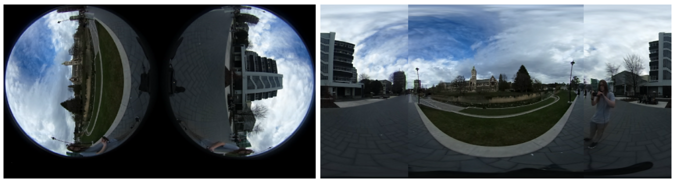
Fig. 8. An example of the 360° images we obtain from the Ricoh camera. (Left): The raw dual-fisheye image provided by the camera. (Right): The result of projecting this image into our equirectangular panorama.

Frames are retrieved from the camera and sent to the remote user in the dual fisheye format seen in Fig. 8. This is then converted to an equirectangular format and projected into the panorama using the following method. As with Incremental Panoramic Calling, this is performed in a fragment shader before the rendering step each frame with the panorama framebuffer as the target.

For each fragment in the panorama framebuffer we calculate its latitude φ and longitude λ

$$(\varphi, \lambda) = \left(\frac{-\pi}{2} + \frac{\pi t_y}{r_y}, -\pi + \frac{2\pi t_x}{r_x} \right) \tag{7}$$

where (t_x, t_y) is the coordinate of the current fragment and $(r_x, r_y) = (4096, 2048)$ the resolution of the panorama framebuffer. We project these to a unit vector $\mathbf{m}$ on the environment sphere, which is rotated so that it aligns with the coordinate space of our panorama:

$$\mathbf{m} = \begin{bmatrix} 0 & 0 & -1 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \cos(\varphi)\cos(\lambda) \\ \sin(\varphi) \\ -\cos(\varphi)\sin(\lambda) \end{bmatrix} \tag{8}$$

From here we calculate the spherical equirectangular coordinates

$$\theta = \begin{cases} 0 & \text{if } |\mathbf{m}_y| \geq 1, \\ \cos^{-1}\left(|\mathbf{m}_y|\right) & \text{if } |\mathbf{m}_y| < 1 \end{cases}$$
$$\phi = \begin{cases} \tan^{-1}\left(\frac{\mathbf{m}_x}{\mathbf{m}_z}\right) & \text{if } \mathbf{m}_y < 0, \\ \tan^{-1}\left(\frac{-\mathbf{m}_x}{\mathbf{m}_z}\right) & \text{if } \mathbf{m}_y \geq 0 \end{cases} \tag{9}$$

where θ is the azimuth and ϕ the inclination of the environment sphere, and find their corresponding texel coordinates (u, v) in the fisheye texture

$$(u, v) = (t_x, t_y) \cdot \begin{bmatrix} c_x + \frac{r\pi}{2}\cos(\phi) \left| \sin\left(\frac{\theta}{2}\right) \right| \\ c_y + \frac{r\pi}{2}\sin(\phi) \left| \sin\left(\frac{\theta}{2}\right) \right| \end{bmatrix}. \tag{10}$$

Here (c_x, c_y) is the centre point of one of the fisheye images seen in Fig. 8 and r its radius. If $\mathbf{m}_y < 0$ we take the texel from the left image, otherwise we take it from the right. Since each lens of the Ricoh covers slightly more than 180° and its intrinsics are not publicly available we cannot perform a perfect projection without computationally expensive stitching, causing visible seams where the two images meet.

Our implementation solves a few key problems Tang *et al.* [33] found with their 360° videoconferencing system. Their remote users often felt frustrated that the local user could not see their gestures or FoV, and despite this often used them when giving instructions anyway. We allow for both of these interaction methods to be fully utilised, increasing the system's collaborative potential and decreasing user frustration.

7 Performance

For this shared experience to provide a real sense of presence we require that it performs in real time with a high frame rate and low latency so that actions are shown as they would be in the real world.

[8]https://theta360.com/en/about/theta/s.html

[9]https://github.com/saki4510t/UVCCamera

In our testing we use two Google Pixels with Android 7.1.2 connected via Wi-Fi paired with Google's Daydream HMD. The latency of these devices' cameras was measured to be at most 128ms. We similarly measured the network latency to be 128ms at worst. The Pixel's camera operates at 30 *frames per second* (fps), and its screen has a refresh rate of 60Hz. The Ricoh Theta we use for panoramic image capture has a worst-case latency of 256ms and is only capable of streaming to third-party applications at 15fps.

The average end-to-end latency for each frame is shown in Fig. 9 with the average frame rate and time required to process each frame shown in Fig. 10. These results proved constant over several measures and so standard deviations were negligible. The application's average frame rate is 60fps in most modes, which is the highest attainable on these devices due to Android's enforced VSync. Incremental Panoramic Calling sees slightly lower performance, averaging 50fps and 48fps for the local and remote user respectively due to the extra projection required for building the panorama. The local user will see their camera image only 141ms after it is captured in most cases, and it will be shown to the remote user in only 302.34ms on average for images from the inbuilt camera or 351.54ms for images from the panoramic camera. The remote user can expect to wait 175ms to see their own hands, which will be visible to the local user in 246.19ms on average. This latency is lower than was achieved in similar systems with no ill effects reported by the authors [26, 33], and almost all is unavoidable due to camera and network constraints and may be removed almost completely with future hardware revisions.

Our application is comprised of three main threads: the *rendering thread*, which determines the refresh rate of the display, the *sending thread*, which determines how often local frames are displayed and sent to the remote peer, and the *receiving thread*, which determines how often the remote peer's frames are displayed. In the following sections we evaluate the performance of each, including a description of what tasks they are responsible for and the time taken to perform them.

7.1 Rendering

The rendering thread is responsible for constructing and rendering the shared environment and performs the following tasks each frame:

1. Retrieves the latest frames from both the local camera and the receiving thread. The time taken for this is negligible, requiring much less than a millisecond.

2. In Incremental Panoramic Calling or Live Panoramic Video Calling, the local user's camera frame is projected into the shared environment. This takes 8.42ms for standard images and 12.00ms for images from the panoramic camera. The local user's hands may also be removed from the image during this process to avoid occluding the environment, requiring a further 34ms.

3. Renders views to the device display. This takes 14.05ms, 13.75ms or 13.97ms to display the full panorama, the oriented unprojected view or the HMD view, respectively.

The frame rate of this thread determines the overall frame rate for the application and as such is the most important to optimise. A low frame rate would affect the update rate for local orientation tracking, and high latency could result in motion sickness when using an HMD as head rotations would not provide immediate feedback; fortunately this thread runs at 60fps in most cases. Performance is slightly lower on the remote user's device in both Incremental Panoramic Calling and Live Panoramic Video Calling; this can be attributed to increased contention for resources caused by hand segmentation and the extra projection.

7.2 Sending Frames

The sending thread is responsible for processing frames from the local camera and passing them to the network module so that they can be sent to the remote peer. It performs the following tasks each frame:

1. Captures an image from the camera. This takes at most 128ms for the integrated camera and 256ms for the panoramic camera.

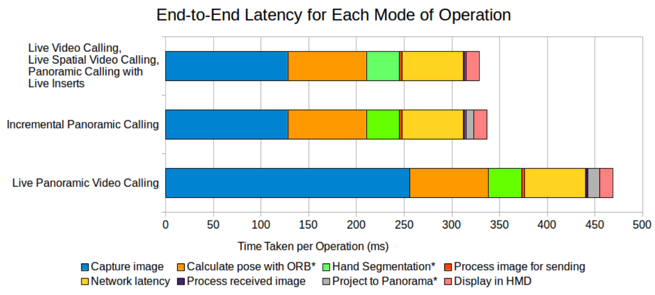

Fig. 9. The average end-to-end latency of each mode. This assumes a typical use case where the panoramic camera is not used and the user views the environment in the unprojected view. Note that ORB tracking is optional, projecting to the panorama is only required in Incremental Panoramic Calling and Live Panoramic Video Calling, and hand segmentation usually only occurs on the remote user's device.

2. Hand segmentation is performed on the remote user's device, which takes 34ms on average.

3. The user's orientation is estimated using either sensor fusion, which takes less than 1ms, or using our vision-based approach, which takes 82.38ms and reduces this thread's frame rate to 14fps.

4. The image is then processed for sending, including encoding of an identification tag into the frame. This takes 3ms.

This thread's performance is restricted by the hardware limitations of the cameras, resulting in a best-case latency of 128ms or 256ms and a best-case frame rate of 30fps or 15fps for the integrated or panoramic camera respectively. This is much less frames than the application is capable of rendering so immediate benefits will be seen once mobile phones integrate more capable cameras.

7.3 Receiving Frames

The receiving thread is responsible for receiving frames from the remote peer and processing them so that they can be displayed within the environment. It performs the following tasks each frame:

1. Receives a new image from the remote peer. Network latency is at worst 128ms.

2. Reconstructs the received image with the missing pixels and pairs it with its associated data packet, which will contain the remote peer's orientation at the time the image was taken. This takes 2.57ms on average.

Much like the sending thread, this thread's latency is largely limited by the network, and its frame rate is limited by the capture rate of the remote peer's camera. It has the least processing to perform of our threads, allowing it to easily keep up with demand, and spawning a thread for each new frame ensures that network latency does not affect the overall frame rate.

8 User Evaluation

A preliminary study was conducted to gauge the application's ability to induce a sense of spatial or co-presence within the shared environment. We hypothesised that there would be a correlation between the degree of view independence and the sense of spatial presence felt within the environment, and a similar correlation between the degree of view independence and the co-presence felt between users. Due to the preliminary nature of this study we limited the number of participants recruited and thus the number of conditions evaluated. Live Video Calling was excluded as it does not lend itself to HMD use, and Panoramic Calling with Live Inserts was similarly excluded as it essentially provides a best-case scenario for Incremental Panoramic Calling.

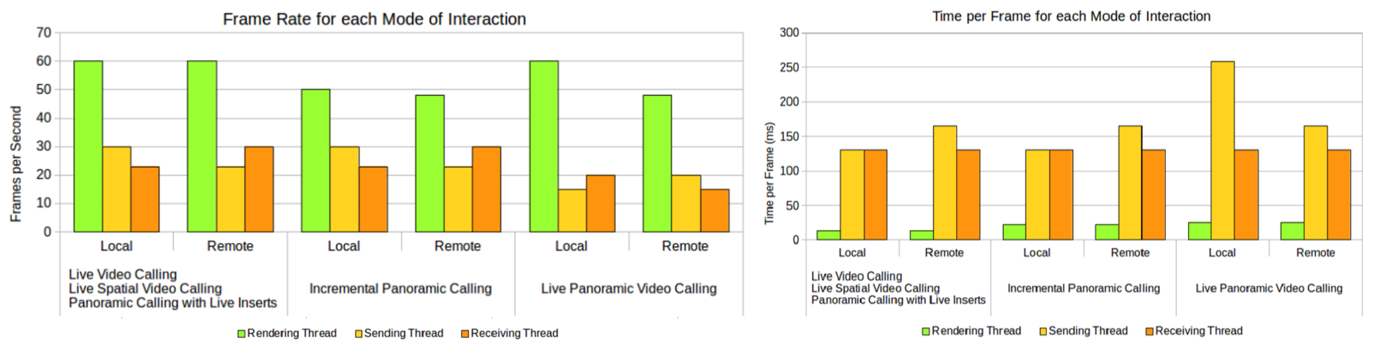

Fig. 10. The average frame rate (left) and required time per frame (right) for each of our modes of interaction for the local and remote user. Each of our main threads executes asynchronously and so their performance is evaluated separately. These results assume that vision-based tracking is not used. The standard deviations for each result were negligible and thus are omitted.

8.1 Study Design

19 participants were recruited between the ages of 18-65 that self-identified as having no history of simulator sickness. Only seven had prior experience with virtual reality. The study followed a within-subjects design with the independent variable being the mode of interaction used. A Google Daydream HMD was used to view the shared environment for all conditions. Spatial and co-presence were evaluated for each condition via modified versions of questionnaires by Schubert *et al.* [31] and Biocca *et al.* [5] respectively that comprised of statements about the participant's experience within the virtual environment, with the participant noting the degree to which they agreed with each statement via a 7-point Likert scale (1 = strongly disagree, 7 = strongly agree). Statements were presented in a way that higher scores indicated a higher sense of presence. Participants could also write free-form comments about their experience, and any simulator sickness felt was evaluated using the questionnaire designed by Kennedy *et al.* [16].

The order of conditions was randomised for each participant to reduce any potential learning effects. For each condition, participants were instructed on how to use that mode of interaction, and were then allowed to familiarise themselves with the system for two minutes within a pre-recorded environment; this was different than the environment used for evaluation and was done without a communication partner to encourage experimentation. They were then connected to the study mediator and placed within a hill-top, urban, outdoor scene where they were asked to have a free-form discussion with the mediator. Participants were shown various landmarks and were encouraged to point out and ask questions about any others they could see. Once two and a half minutes had elapsed they were asked to complete the presence questionnaires. This was repeated for the remaining conditions, after which participants were asked to complete the simulator sickness questionnaire and were gifted a $20 voucher.

To ensure that the presented environment remained consistent between conditions and participants the study mediator's video was recorded ahead of time. Each frame was stored along with its timestamp and orientation matrix so that it could be played back as if it were arriving from a real remote partner, allowing us to control unpredictable factors such as noise and the weather; removing these important environmental cues could limit environmental immersion but prevents extreme conditions from interrupting the experiment. Participants were informed of this at the beginning of the study. Their own video was still broadcast live to the study mediator over Wi-Fi so that their gestures and FoV could be seen in relation to the environment. Both the participant and the mediator were situated within the same room so could hear each other speak without using the application's inbuilt VoIP functionality. The same mediator and environment were used for all participants and conditions. As it is difficult to place a panoramic camera without the local user being visible, the environment was limited to $180°$ to ensure a consistent first-person view between conditions.

As simulating a communication partner may have some effect on co-presence, the study was repeated with live video to determine whether the system would behave similarly in a real-life scenario. Seven participants took part, with a second study mediator placed within the same environment as in the pre-recorded study and connected to the participant over Wi-Fi using the system's VoIP capabilities.

8.2 Results

For each condition, its spatial and co-presence scores, as shown in Fig. 11, were determined by the mean of the relevant questionnaire responses. Live Spatial Video Calling (C1) scored the lowest of our tested modes for both forms of presence in the pre-recorded environment with mean scores of 4.56 and 4.98 for spatial and co-presence respectively. Incremental Panoramic Calling (C2) was higher rated, with mean scores of 5.22 and 5.35 , and Live Panoramic Video Calling (C3) achieved similar results with mean scores of 5.19 and 5.54 . Friedman tests showed a significant difference between conditions ($p \leq 0.05$) in both spatial presence ($p = 0.019$) and co-presence ($p = 0.015$). Wilcoxon signed-rank tests ($N = 19$) revealed that C1 induced significantly lower spatial presence than both C2 ($p = 0.012$) and C3 ($p = 0.019$), with no significant difference between C2 and C3 ($p = 0.917$). Co-presence was similarly distributed, with C3 scoring significantly higher than C1 ($p = 0.008$) but with no significant difference between C1 and C2 ($p = 0.343$) or between C2 and C3 ($p = 0.586$).

The results of the live study were similar to the first. C1 was rated the lowest for both forms of presence, with mean scores of 3.99 and 5.39 for spatial and co-presence respectively. C2 and C3 were again similar, with C2 achieving mean scores of 4.70 and 5.71 and C3 scoring 4.94 and 5.57 . Friedman tests again showed a significant difference in induced spatial presence between conditions ($p = 0.030$), but this time no significant difference in co-presence was found ($p = 0.368$). Wilcoxon signed-rank tests ($N = 7$) showed that participants felt significantly more spatially present within the environment in C3 than in C1 ($p = 0.035$), but no significant difference was found between C1 and C2 ($p = 0.051$) or between C2 and C3 ($p = 1.00$).

Reported simulator sickness was low across all participants. Responses to each symptom were coded to allow for numerical analysis ("None" = 0, "Severe" = 3), resulting in an average response of 0.313 ($\sigma = 0.192$) across all symptoms. Participants were informed that they may conclude the experiment at any time if they felt any severe symptoms, though none felt it necessary to do so.

8.3 Discussion

We believe that view independence is the single most important factor for increasing spatial presence within the shared environment, and that per our first hypothesis there would be a correlation between the two. This was partially supported by our experiments; both Incremental Panoramic Calling and Live Panoramic Video Calling induced significantly higher spatial presence than Live Spatial Video Calling, suggesting that allowing the remote user to view areas outside of the local user's field of view could be beneficial to providing a sense of presence within that space. This view was shared by many of our par-

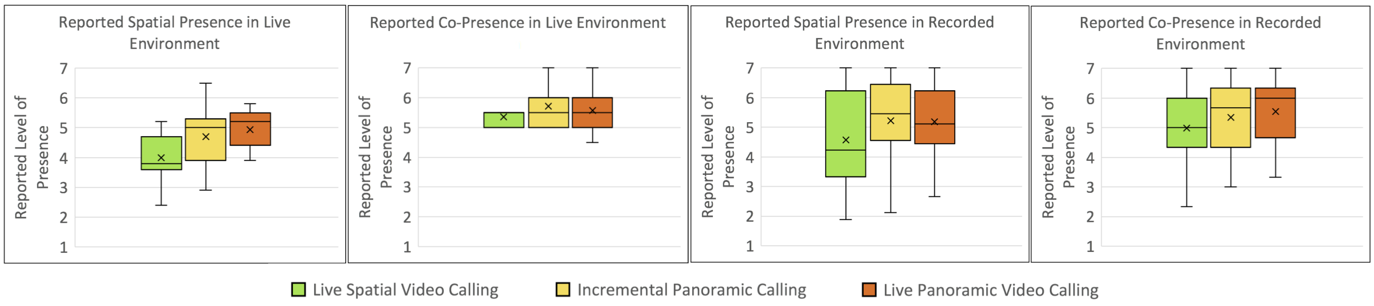

Fig. 11. Participants' reported levels of spatial presence and co-presence within the live environment (left) and pre-recorded environment (right) for each of the evaluated modes of interaction, as indicated by 7-point Likert scales.

ticipants, who made comments such as "[C3] felt more immersive [than C1] as you can see the whole area", "Looking through a small window [in C1] made it harder to fully immerse", and "I felt more immersed with being able to see outside the blue square".

Our preliminary results indicate that there is little difference in induced spatial presence between C2 and C3. Based on this observation, we suggest that the full environment presented in Live Panoramic Video Calling could provide no additional benefit over Incremental Panoramic Calling's partial panorama. This could potentially be due to the low resolution of the panoramic camera, which was noticed by almost all participants with comments such as "The lower resolution made me feel slightly dizzy", "I felt the resolution on this really impacted how immersed I felt", and "because the image was so grainy I felt like I was in a game rather than a real location". Though the resolution of panoramic cameras may improve in future, we believe this will remain an issue; network limitations will always constrain the resolution of images able to be streamed in real time, and using all of that resolution over a smaller area as in Incremental Panoramic Calling rather than over the full panorama will always result in higher quality environments (assuming perfect stitching). We assumed that the panoramic camera's low frame rate would have similar detrimental effects, however none of the participants mentioned this as being an issue and the performance of the system as a whole proved satisfactory for participants. Only one reported any performance issues, which were caused by unscheduled background processing and not experienced in subsequent conditions.

The ability to gesture within the environment was well received by participants. Deictic references were used frequently, and representational gestures were also often used for tasks such as tracing the path of a river. At least one participant found that "being able to see the movements of my own hands creates a much higher sense of engagement than if this feature was not included, and drives most of the sense of 'being there'", suggesting that this unmediated view of their gestures could induce spatial presence within the environment. Despite our segmentation algorithm's prioritisation of performance over quality, no participants reported any issues arising from incorrect segmentation, even in the unprepared room they were situated within.

Our hypothesis that there would be a correlation between the degree of view independence provided and co-presence induced was not supported by our results; there is a small correlation, though not enough for significant differences to present themselves. Co-presence was rated highly across all conditions, which was particularly surprising for the pre-recorded environment given that the study mediator could not properly react to the participants' actions. Bias may have been introduced by the mediator's presence within the same room as participants, however co-presence scoring similarly in the live conditions where the mediator was remotely located suggests no such bias exists.

It is also worth mentioning that the results of Incremental Panoramic Calling mode are in agreement with experimental results found by Müller *et al.* [26], which validates our general study design and the shown results as well as the potential of our system.

Overall, the system was very well received by participants, with even those unfamiliar with VR experiencing little simulator sickness. This implies that the system could be used by the wider public without issue, though it is unsure whether this would hold true for longer exposure times [17]. Many saw the system's potential as the future of telecommunications, with some commenting "After I took [the HMD] off I had forgotten exactly where I was... really did feel like I was there" and "makes one actually feel they are really in the same place with the other person. Taking communication to another level!".

9 Conclusion and Future Work

We presented several approaches for enabling real-time immersive telepresence using mobile and wearable devices. Each utilised a shared spherical mapping of the local environment that can be created and updated on the fly. This shared environment can be viewed independently by each user, and shared mutual FoV awareness and support for gesture-based interaction allow for natural interaction between communicating parties. To the best of our knowledge, ours is the first system to provide these features using purely mobile devices. A preliminary study showed this system was well received by novice users, who found it induced a sense of presence within the shared space. A full evaluation of all factors (e.g. resolution, frame-rate) and their contribution to the sense of presence, as well as a full evaluation of each mode of interaction, will be the subject of future work.

Given the extensive literature on desktop-based telepresence solutions, mobile telepresence research seems to be in its infancy. However, as shown in this work, modern mobile hardware has now reached the point where it is capable of supporting real-time immersive telepresence experiences in unprepared outdoor environments. We argue that this is a big step forward for mobile telepresence research as it allows for spontaneous experiences independent of location, supporting new application scenarios not possible in existing stationary solutions. This required compromises in tracking and stitching quality and thus still poses a significant engineering challenge, however even with these constraints we have shown that such systems are viable and are encouraging further research in this domain.

We believe these insights and results are beneficial for future research in immersive mobile telepresence systems. We argue that systems like this have the potential to change activities such as maintenance scenarios, remote collaboration and tele-tourism. Future advances in mobile technology will make such systems more viable, and we believe it is inevitable that users will seek more immersive experiences such as this when traditional systems fail to provide them the collaborative and social experiences they desire.

Acknowledgments

The authors wish to thank Oliver Reid for his help in conducting the user study, Katie Tong for her constant support, and the rest of the HCI lab for their valuable input throughout the duration of this research.

References

[1] A. Agarwala, K. C. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski. Panoramic Video Textures. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pp. 821–827. ACM, New York, NY, USA, 2005. doi: 10.1145/1186822.1073268

[2] Z. Al-Tairi, R. Rahmat, M. Iqbal Saripan, and P. S. Sulaiman. Skin Segmentation Using YUV and RGB Color Spaces. *Journal of Information Processing Systems*, 10(2):283–299, 2014. doi: 10.3745/JIPS.02.0002

[3] I. Arminen and A. Weilenmann. Mobile Presence and Intimacy - Reshaping Social Actions in Mobile Contextual Configuration. *Journal of Pragmatics*, 41(10):1905–1923, oct 2009. doi: 10.1016/j.pragma.2008.09.016

[4] M. Bauer, G. Kortuem, and Z. Segall. "Where Are You Pointing At? A Study of Remote Collaboration in a Wearable Videoconference System. In *In: Proceedings of the 3rd International Symposium on Wearable Computers*, pp. 151–158. IEEE, San Francisco, California, 1999. doi: 10.1109/ISWC.1999.806696

[5] F. Biocca, C. Harms, and J. Gregg. The Networked Minds Measure of Social Presence: Pilot Test of the Factor Structure and Concurrent Validity. In *4th Annual International Workshop on Presence*, pp. 1–9, 2001.

[6] C. Campos-Castillo and S. Hitlin. Copresence: Revisiting a building block for social interaction theories. *Sociological Theory*, 31(2):168–192, 2013. doi: 10.1177/0735275113489811

[7] A. Cockburn, A. Karlson, and B. B. Bederson. A Review of Overview+Detail, Zooming, and Focus+Context Interfaces. *ACM Comput. Surv.*, 41(1):2:1–2:31, Jan. 2009. doi: 10.1145/1456650.1456652

[8] S. Diverdi, J. Wither, and T. Hllerert. Envisor: Online Environment Map Construction for Mixed Reality. In *Virtual Reality Conference, 2008. VR '08. IEEE*, pp. 19–26. IEEE, Reno, NE, USA, 2008. doi: 10.1109/VR.2008.4480745

[9] S. R. Fussell, R. E. Kraut, and J. Siegel. Coordination of Communication: Effects of Shared Visual Context on Collaborative Work. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, pp. 21–30, 2000. doi: 10.1145/358916.358947

[10] S. R. Fussell, L. D. Setlock, J. Yang, J. Ou, E. Mauer, and A. D. I. Kramer. Gestures over Video Streams to Support Remote Collaboration on Physical Tasks. *Human-Computer Interaction*, 19(3):273–309, Sept. 2004. doi: 10.1207/s15327051hci1903_3

[11] S. Gauglitz, C. Lee, M. Turk, and T. Höllerer. Integrating the Physical Environment into Mobile Remote Collaboration. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '12, pp. 241–250. ACM, New York, NY, USA, 2012. doi: 10.1145/2371574.2371610

[12] S. Gauglitz, B. Nuernberger, M. Turk, and T. Höllerer. World-stabilized Annotations and Virtual Scene Navigation for Remote Collaboration. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pp. 449–459. ACM, New York, NY, USA, 2014. doi: 10.1145/2642918.2647372

[13] W. Huang and L. Alem. HandsinAir: A Wearable System for Remote Collaboration on Physical Tasks. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work Companion*, CSCW '13, pp. 153–156. ACM, New York, NY, USA, 2013. doi: 10.1145/2441955.2441994

[14] H. Jo and S. Hwang. Chili: Viewpoint Control and On-Video Drawing for Mobile Video Calls. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems on - CHI EA '13*, pp. 1425–1430. ACM Press, New York, New York, USA, 2013.

[15] S. Kasahara and J. Rekimoto. JackIn: Integrating First-Person View with Out-of-Body Vision Generation for Human-Human Augmentation. In *Proceedings of the 5th Augmented Human International Conference*, pp. 46:1–46:8. ACM, Kobe, Japan, 2014. doi: 10.1145/2582051.2582097

[16] R. S. Kennedy, N. E. Lane, K. S. Berbaum, and M. G. Lilienthal. Simulator Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness. *The International Journal of Aviation Psychology*, 3(3):203–220, 1993.

[17] R. S. Kennedy, K. M. Stanney, and W. P. Dunlap. Duration and Exposure to Virtual Environments: Sickness Curves During and Across Sessions. *Presence*, 9(5):463–472, 2000. doi: 10.1162/105474600566952

[18] S. Kim, G. Lee, N. Sakata, and M. Billinghurst. Improving Co-Presence with Augmented Visual Communication Cues for Sharing Experience through Video Conference. In *ISMAR 2014 - IEEE International Symposium on Mixed and Augmented Reality - Science and Technology 2014, Proceedings*, pp. 83–92, 2014. doi: 10.1109/ISMAR.2014.6948412

[19] D. Kirk and D. Stanton Fraser. Comparing Remote Gesture Technologies for Supporting Collaborative Physical Tasks. In *Proceedings of the SIGCHI conference on Human Factors in computing systems - CHI '06*, pp. 1191–1200, 2006. doi: 10.1145/1124772.1124951

[20] S. Kratz, D. Avrahami, D. Kimber, J. Vaughan, P. Proppe, and D. Severns. Polly Wanna Show You: Examining Viewpoint-Conveyance Techniques for a Shoulder-Worn Telepresence System. In *MobileHCI 2015 - Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, pp. 567–575. Toronto, Canada, 2015. doi: 10.1145/2786567.2787134

[21] S. Kratz and F. Ferreira. Immersed Remotely: Evaluating the Use of Head Mounted Devices for Remote Collaboration in Robotic Telepresence. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 638–645. IEEE, August 2016. doi: 10.1109/ROMAN.2016.7745185

[22] S. Kratz, D. Kimber, W. Su, G. Gordon, and D. Severns. Polly. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices and services - MobileHCI '14*, pp. 625–630. ACM Press, New York, New York, USA, sep 2014. doi: 10.1145/2628363.2628430

[23] H. Kuzuoka. Spatial Workspace Collaboration: A SharedView Video Support System for Remote Collaboration Capability. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, vol. Monterey,, pp. 533–540, 1992. doi: 10.1145/142750.142980

[24] H. Kuzuoka, S. Oyama, K. Yamazaki, K. Suzuki, and M. Mitsuishi. GestureMan. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work - CSCW '00*, pp. 155–162. ACM Press, New York, New York, USA, dec 2000. doi: 10.1145/358916.358986

[25] P. Luff, C. Heath, H. Kuzuoka, J. Hindmarsh, and S. Oyama. Fractured Ecologies: Creating Environments for Collaboration. *Human-Computer Interaction*, 18(1):51–84, 2003. doi: 10.1207/S15327051HCI1812_3

[26] J. Müller, T. Langlotz, and H. Regenbrecht. PanoVC: Pervasive Telepresence using Mobile Phones. In *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–10, March 2016. doi: 10.1109/PERCOM.2016.7456508

[27] Pacha, Alexander. *Sensor Fusion for Robust Outdoor Augmented Reality Tracking on Mobile Devices*. Diploma thesis, University of Augsburg (Institut für Software & Systems Engineering), 2013.

[28] F. Pece, W. Steptoe, F. Wanner, S. Julier, T. Weyrich, J. Kautz, and A. Steed. Panoinserts: Mobile Spatial Teleconferencing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems SE - CHI '13*, pp. 1319–1328, 2013. doi: 10.1145/2470654.2466173

[29] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts. *ACM Transactions on Graphics*, 23(3):309, 2004. doi: 10.1145/1015706.1015720

[30] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2564–2571, 2011. doi: 10.1109/ICCV.2011.6126544

[31] T. Schubert, F. Friedmann, and H. Regenbrecht. The Experience of Presence: Factor Analytic Insights. *Presence: Teleoperators and Virtual Environments*, 10(3):266–281, 2001. doi: 10.1162/105474601300343603

[32] R. S. Sodhi, B. R. Jones, D. Forsyth, B. P. Bailey, and G. Maciocci. BeThere: 3D Mobile Collaboration with Spatial Input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pp. 179–188. ACM, New York, NY, USA, 2013. doi: 10.1145/2470654.2470679

[33] A. Tang, O. Fakourfar, C. Neustaedter, and S. Bateman. Collaboration in 360 Videochat: Challenges and Opportunities. In *Proceedings of the 2017 Conference on Designing Interactive Systems*, pp. 1327–1339. ACM, Edinburgh, United Kingdom, 2017. doi: 10.1145/3064663.3064707

[34] J. Yang and A. Waibel. A Real-Time Face Tracker. In *Proceedings Third IEEE Workshop on Applications of Computer Vision. WACV'96*, pp. 142–147, 1996. doi: 10.1109/ACV.1996.572043