PanoVC: Pervasive Telepresence using Mobile Phones

Jörg H. Müller
Advanced Concepts Team
European Space Agency
Noordwijk, The Netherlands
joerg.mueller@esa.int

Tobias Langlotz, Holger Regenbrecht
Department of Information Science
University of Otago
Dunedin, New Zealand
{tobias.langlotz, holger.regenbrecht}@otago.ac.nz

Fig. 1. Conceptual illustration of our implemented PanoVC prototype. (Left) A user of our PanoVC system (the local user) shares the environment by capturing it with the mobile phone, (Middle) a distant user (the remote user) receives a camera stream, and builds and updates a panoramic representation of the distant environment. Using orientation tracking the phone becomes a window into the distant environment as both users can independently control their current view. (Right) By providing a window into the distant environment, users of PanoVC experience the feeling of presence as they are virtually "being there together".

***Abstract*—We are presenting PanoVC - a mobile telepresence system based on continuously updated panoramic images. We are showing that the experience of telepresence, i.e. the sense of "being there together" at a distant location can be achieved with standard state-of-the-art mobile phones. Because mobile phones are always on hand users can share their environments with others in a pervasive way. Our approach is opening up the pathway for applications in a variety of domains such as the exploration of remote environments or novel forms of videoconferencing. We present implementation details, technical evaluation results, and the findings of a user study of an indoor-outdoor environments sharing task as proof of concept.**

Keywords—telepresence; mobile computing; panorama; presence; pervasive computing; user study; immersive communication; mobile phones

I. INTRODUCTION

Most of us use videoconferencing almost every day and even less technically minded people would use it occasionally or at least would have heard of it. A laptop computer, tablet, or mobile phone with a front-facing camera and software products like Skype or Facetime allow for effective real-time, audio-visual communication. Sometimes, for instance when we are in a new location like on holidays or business trips, we also want to share our environment with others. To do so we either describe the environment to the other person or we turn around our computer or use the phones back-facing camera and augment the description with real-time video impressions. While this "pointing around" video streaming can be effective in terms of giving the other person a certain impression of the environment or at least parts of it, it most likely fails to develop a sense of *social presence*, i.e. the feeling of being together with another [1], a sense of being together in one place, also called *co-presence*, or of being in that other place, also known as a sense of *spatial presence* or simply presence. One could argue that in remote communication and collaboration systems the combination of a sense of (spatial) presence and of social presence allows for the development of true telepresence.

Wouldn't it be desirable to have such an environment-sharing capability on your mobile phone? Such a mobile telepresence system would not only allow for a richer communication but also for effective support for certain professions, like supporting technicians working in remote locations or police examining a scene while communicating in real-time with their expert counterparts at other locations.

We are proposing a novel approach and implementation, which allows for mobile telepresence using panoramic picture and video techniques in combination with a suitable human-computer interface. Our system implementation coined PanoVC is based on standard videoconferencing techniques and extends them towards pervasive telepresence on mobile phones.

Jörg Müller's research internship at the University of Otago was partially supported with a travel and hardware scholarship awarded from Graz University of Technology. This work was partially supported by InternetNZ's Conference Attendance fund.

In a controlled user study we can show that spatial presence and social presence significantly improve over to-dates "pointing around" techniques. With this, mobile communication develops into pervasive telepresence. A multitude of sensors is used to realize this: e.g. we sense features in the surrounding environment and build a panoramic environment which also allows us to track the users' look-around movement in space using computer vision (see Figure 1). We further use built-in sensors in mobile phones to allow for interactive exploration and to improve stability and precision of the vision-based tracking.

We present the system architecture and relevant implementation details of our novel mobile telepresence prototype PanoVC and relate those to previous work in the field. We also describe an evaluation study to prove our concept. To our best knowledge, PanoVC is the first system, which allows for pervasive telepresence on off-the-shelf mobile phones.

II. Related Work

Telepresence can and has to be viewed in two ways: technological and experiential. The experience of being present in another place is supported by a set of appropriate hard- and software components. If the sense of presence, i.e. the sense of "being there" [2] in a virtual or remote natural environment is successfully combined with social presence [3], i.e. the sense of "being together", a sense of "being there together" can be achieved - a defining element for usable and useful telepresence. Our focus here is on considering technological approaches to achieve pervasive telepresence.

We will first briefly look into systems and studies in office or meeting room environments, which we characterize as static telepresence systems. In opposition, we look at mobile telepresence. Historically, telepresence research stems from questions on controlling remote robotic systems. There is a huge body of work on telepresence and robots and we will briefly discuss those works with respect to relevance to our mobile, pervasive telepresence system. Finally, more recently, researchers focused on actually using mobile devices to achieve the experience of telepresence. Those works will be discussed with the most attention.

A. Static Telepresence

The experience of telepresence can be achieved in videoconferencing environments, if cameras, microphones, displays, participants, and furniture and lighting are combined with specialized teleconferencing systems.

For instance, a complex but effective system is presented by Schreer et al. [4] where three participants are placed in a virtual world giving the impression of sitting around a table. Four cameras on each system are used to reconstruct the upper bodies of the participants and are synthesized into one telepresence experience, including eye-to-eye contact. Similar experiences are provided by systems like HoloPort [5] or MultiView [6]. Some commercial systems, like Cisco TelePresence[1] implement telepresence in meeting rooms.

While all those approaches and systems deliver high technical fidelity and can lead to the experience of telepresence, they are restricted to the (indoor-) environments they are installed in and are designed to capture the participants and not the environment. The static environment and system setup predetermines in particular the spatial and co-presence aspects of the experience.

B. Mobile Telepresence

To overcome some of the restrictions of static telepresence systems, telepresence systems needs to become mobile. One possibility is to use teleoperated robots, which have been extensively studied in the context of telepresence. Jouppi for example presents a system remotely showing the user's head from four sides on the robot [7]. The four displays show videos streamed from four cameras placed around the user's head. However, the robot was still constrained to indoor usage. The miniaturisation of teleoperated robots such as presented by Kratz et al. allowed them to take the robot outside [8]. Here, a mobile phone is mounted on a gimbal next to the shoulder of the carrier. The remote user can control his view using the gimbal. An important conclusion by Kratz et al. is that the free choice of view enhances the engagement of the remote user however they do not report on presence. While the use of remote controlled robots enables many beneficial possibilities for telepresence, the hardware requirements, costs and limitation to indoor usage prohibit the casual use of such a system for many application areas. Furthermore, switching the environment is only possible when both sides have a robot adding additional costs.

Jo et al. presented Chili [11], a mobile video calling system allowing people to request a specific view without the need of special hardware by using mobile phones. Here remote users can indicate which part of the environment they want to see by rotating their phone into the desired direction. This is visually displayed to the local users who can then point their camera in this direction to share the view. While this system is related to ours, the users views are still coupled as well as there is no study to reveal if any effects on presence can be measured. Gauglitz et al. [12] have shown that *Simultaneous Localization And Mapping* (SLAM) can be used to create a virtual representation of the remote environment in a telepresence system. JackIn by Kasahara et al. [13] uses this approach to create an environment for a stationary user based on the exploration of a mobile user who wears a *head-mounted display* (HMD). The problem of SLAM algorithm as discussed by the authors is, that it needs a big enough parallax to create a 3D point cloud of the environment and it needs comparably high processing power. Therefore, in their subsequent work, Kasahara et al. [14] use LiveSphere, a head-mounted device with multiple cameras, to create a panoramic video for the spectator to overcome the problems of SLAM. While these systems allow for free view control they only have one mobile user (the one sharing the environment), as the techniques need enough processing power on the stationary side. Consequently, they don't allow for switching worlds, require expensive hardware, and one user is constrained to indoor environments.

Besides Kratz et al. and Kasahara et. al. also other groups investigated free view control in video calling applications. Free view control has for example been concluded as a major possibility for improvement by Jones et al. from an observational study of mobile collaborative tasks using video calling applications [9]. Additionally Jones et al. found that better support for interactions in video calling applications

[1] https://ciscomeetingonus.com

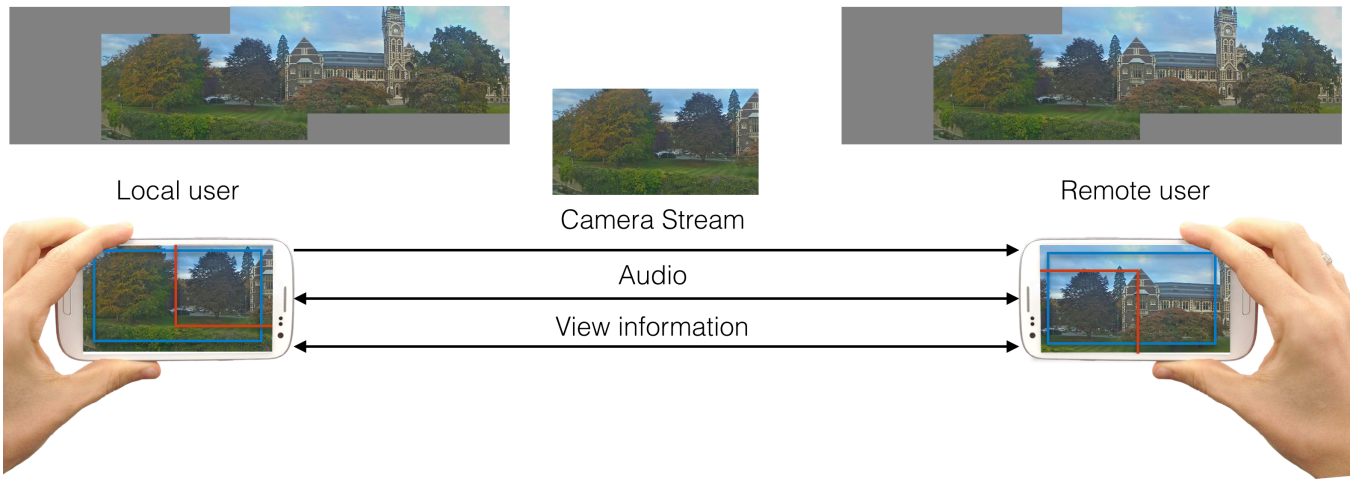

Fig. 2. Overview on our PanoVC system. The local user creates a panoramic image from the camera stream which is also used for vision-based tracking. The camera stream is sent to the remote user where it is used to create and update a panoramic image of the distant environment. Additionally, the user exchange audio and view information. View information allows both users to see where they are currently looking.

would counter problems found during their study. One example of such an advanced interaction technique is a collaborative drawing interface as shown by Fussel et al. that greatly aids the interaction between participants [10]. Billinghurst et al. [15] conceptually explored the application of shared panoramas using Android tablets or Google Glass. However, the concept was only demonstrated with pre-recorded panoramas and not as part of a working system.

III. System Overview

Our system, PanoVC, is based around the requirement to decouple the views of the communication partners and give the partners free viewpoint control while also using mobile phones on both sides of the telepresence system. Thus the *local user* (the user sharing the environment) and the *remote user* (the user receiving the distant environment) can look into different regions of the shared environment. As such our work is similar to the work by Kasahara et al. [13,14]. However, we want to achieve this without requiring desktop hardware or expensive 360º cameras such as those used by Kasahara et al. [14]. We will show that by doing so we truly enable pervasive telepresence. Because both sides, local user and remote user, have the same mobile hardware, we also allow for switching the environment as well as both users can use our system in a mobile outdoor scenario. We argue that this is an important requirement to a truly pervasive telepresence system.

To achieve our goals, we propose that the users of our system do not see the remote environment directly using the camera stream but look at the environment via a panorama of the environment that is built on the fly while also being constantly updated. Users can control their view into the environment by rotating the phone providing a feeling of a window into a distant world. Instead of building the panorama on the local user's device and streaming it to the remote user, both users build their own version of the panoramic map using latest frames from the camera stream (see Figure 2). The main reason for this approach is to reduce required bandwidth. Dependent on the used camera resolution, the panorama that would need to be streamed might be impractical to be transmitted and processed. Instead, in our system the local user builds the panorama from the camera stream but also sends the camera stream to the remote user via the communication component of our system. There, the camera stream is used to recreate and update the panorama based on the local user's environment. We use the communication component also for bidirectional audio communication and can furthermore exchange additional information, such as touch events and view information (where is each user looking at) in real-time. The latter one is needed as once we are able to decouple the views (users can look around independently), we want to also support gaze awareness, indicating to each user the current view of the other person. The view information is computed for both users in the tracking component by precisely tracking the device orientation relative to the surrounding world.

In the following, we present each component of our PanoVC system. These components are (1) the communication component for data exchange between the communication partners, (2) the environment mapping component, for creating and updating the panorama in real-time, and (3) the tracking component for tracking the user's orientation and view into the environment. We integrate these components with a user interface that is responsive and runs interactively on a mobile phone. Based on our experience, we are aware that even on the most recent mobile phones running all these components will be a challenge in terms of available processing power, if possible at all. We therefore planned our initial design in a way that these described components are as modular as possible and as loosely coupled as possible to not block each other but use the existing resources most efficiently. We therefore implemented each of the described components in their own threads to exploit the parallel processing power of today's mobile phones.

IV. Communication

For implementing the real-time communication component, we require a protocol that allows us to connect the devices directly and transmit the data peer-to-peer to allow for the lowest possible latency. Apart from standardized protocols for real-time communication and video conferencing software, there are video call solutions such as Skype that have their own proprietary implementation. For our work we were looking for a protocol and implementation that fulfils our requirements for

low latency, high-quality audio and video transmission, extra data channels for transmitting tracking and other meta data, and most importantly, an implementation that can be used on mobile devices.

After a careful review of existing solutions including solutions that build on top of H.323, *Session Initiation Protocol* (SIP), *Extensible Messaging and Presence Protocol* (XMPP) + Jingle, and *Session Description Protocol* (SDP), we decided for Google's WebRTC (Web Real-Time Communication) implementation that uses SDP [16]. WebRTC fulfils our requirements, is actively developed and a state-of-the-art technology. The WebRTC API was originally specified by Global IP Solutions, a company that has been bought by Google Inc. Since then it has been adopted by the *World Wide Web Consortium* (W3C) to create a standardized real-time communication API for browsers [17]. Google's implementation can also be used to implement a native client running on mobiles, in particular on Android. WebRTC uses several standardized protocols to accomplish its tasks, including SDP and ICE. WebRTC uses the *Secure Real-time Transport Protocol* (SRTP) to transfer the audio, video and data streams encoded in Opus and H.264 or other commonly used codecs, like VP8.

Based on Googles WebRTC library, we implemented our own WebRTC client that serves us the purpose of connecting the users of our system and transmitting all data between the users. To setup the connection between the two clients, a server is required that simply forwards the signalling data provided by WebRTC. As WebRTC doesn't specify, how exactly the signalling data has to be transmitted, we wrote a simple signalling server in Go using websockets [18]. The server waits for two clients to connect and simply forwards the data between them. Once the clients established a direct connection, they disconnect from the server, as they can now communicate using the direct connection at the lowest possible latency.

V. Environment Mapping

Giving remote parties free view control requires building a representation of the environment to overcome the restrictions of the constrained camera view. There are several ways to achieve this: 2D environment mapping or 3D environment mapping. The later one builds a 3-dimensional model of the environment using dedicated depth cameras (e.g. Microsoft Kinect) or by using stereo-vision. Unfortunately, both approaches and consequently 3D environment mapping seemed not feasible for us. While there is a prototypical version of depth-cameras equipped mobile phones (e.g., Google Tango), they are based on infrared light making them only work indoors. SLAM, an approach that can be used to build a 3D model of the environment only using a monocular camera instead of full stereo vision, is very computational expensive and requires a wide baseline (large parallax) in the captured image material for being able to work. Consequently, we decided for 2D environment mapping by building a panoramic representation of the environment.

Our approach creates a panorama on the fly and updates it at each frame rather than stitching a static panorama beforehand. This allows our application to start without time-consuming panorama preparation and the constant update of the panorama allows capturing dynamic environments. However, this comes at relatively high computational costs. In the following we briefly present our approach that runs in real-time despite the high computational requirements.

Our method is based on the approach by Wagner et al. [19]. They introduced an implementation creating a cylindrical mapped panorama on the fly by only rotating the phones' camera. The approach can be seen as a 2D SLAM approach. The algorithm for creating the panorama starts by taking the initial camera image that is projected in the centre of the panorama. Following camera frames are added to the panorama in two steps. First, the orientation of the camera relative to the panorama is determined based on feature matching. This visual tracking step will be explained in more detail in the following section. Secondly, the tracked orientation is used to update the panorama with the camera image.

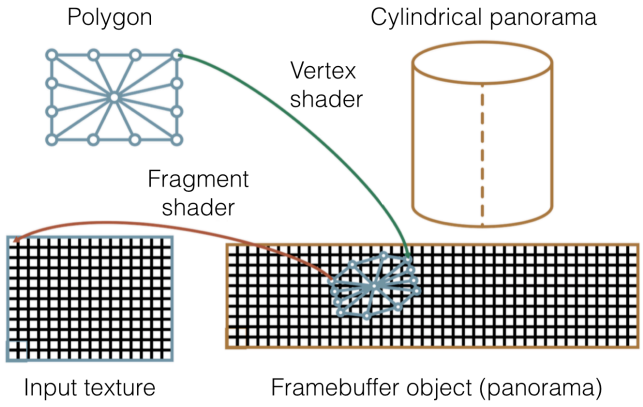

Fig. 3. Mapping the panorama on the GPU to achieve real-time generation and update of the panorama on mobile phones. The panorama is stored in a framebuffer object. For each incoming camera frame a vertex shader maps a polygon into the panorama space based on the transformation gained from visual tracking. In a second stage a fragment shader reads back colours from the inpute texture (the camera image) to the panorama.

Unfortunately, the original approach by Wagner et al. only delivered real-time performance on mobile phones by writing each pixel in the panorama once. This reduces the possible number of pixels to be written each frame as the difference between two subsequent images in a camera stream is relatively small requiring only a few pixels to be mapped, consequently increasing the speed of the approach. However, this comes with the disadvantage that the panorama is not continuously updated as already mapped pixels are not updated. So changes in the environment are not added to the panorama. However, in our approach we want to update the panorama at each frame using the latest camera image.

We achieved this by changing the original approach and mapping the panorama on the integrated GPU of the mobile phone instead of the CPU. We wrote a GPU shader program that in the first stage (vertex shader) takes in vertices of a subdivided rectangle representing the camera frame. These vertices are projected into the panorama space utilizing the transformation data from the keypoint matching (see Figure 3). The panorama to be generated is stored in a *framebuffer object* (FBO), which is set as a render target. In our current implementation we use a panorama size of 2048 × 512 pixels. In the second stage, we apply a fragment shader that projects back from the panorama space to the camera space to determine the colour to be drawn in the panorama. This

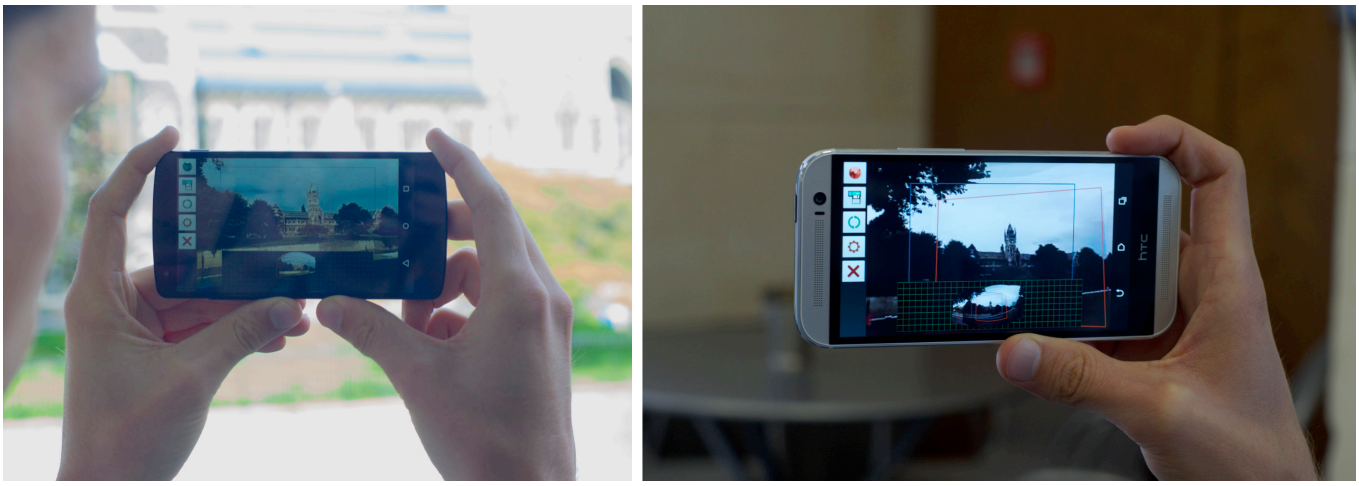

Fig. 4. The final application as used in the user study. (Left) The local user starts our app and calls a remote user. Once the connection is established, our application automatically captures the environment and streams the camera feed to the remote user. The local user sees the current camera view and a visualises where the remote user is looking at. A blue polygon shows the own view while a red polygon outlines the view of the remote user. (Right) The remote user looks at the panorama of the distant environment. Rotating the phone allows to select the current view while the red and blue polygons highlights each others view. Both users see a minature version of the full panorama to provide context information.

approach allows us to update the panorama with the latest camera image while still delivering real-time performance on current generation mobile phones, as the first stage prevents pixels of the panorama that are outside the camera frame to be processed. One drawback of this approach is that the panorama might show inconsistencies caused by moving objects in the scene but this can only be avoided if a 360º camera is used.

VI. Tracking

Tracking the phones' pose is essential to our approach and the tracked pose is used for various purposes. Firstly, we use the pose information to stitch and update the panorama for both users. Both users stitch the panorama using the camera feed (local users the local camera feed, remote users using the camera feed from the local user transmitted via WebRTC) and have to compute the pose first using vision-based tracking on the camera feed. Secondly, we track the pose to control the current view in the panorama. This only applies to the remote users as they see the environment through the panorama. Here we have two options for tracking which can be switched in the settings: We can use a vision-based tracking analyzing the remote users environment as captured by the integrated camera or we can use the phone-integrated sensors in case the current environment does not offer good visual features (e.g., dark room or white walls). The latter one offers less accuracy but works in nearly all cases [19]. Finally, we also share the pose information with our communication partner to visualize each other's view direction showing where we are currently looking at in the shared environment. Common to all approaches is that we only track the orientation of the phone with *3 degrees of freedom* (3DoF). Our approach relies on the panorama therefore we only have a rotational movement that we need to track, as a larger translational movement would not permit for panorama creation. This goes hand-in-hand with our envisioned scenario of keeping a fixed position while rotating the phone to browse the environment.

A. Vision-based tracking

Vision-based tracking is known to be more accurate compared to the error-prone sensors integrated in mobile phones as described by Wagner et al. [19]. Their algorithm first analyses camera frames for keypoints using the FAST keypoint detector [20]. The computed keypoints are later matched against keypoints in the panorama. The keypoints of the current camera image are matched with keypoints within the panorama using a constant-velocity motion model predicting the position of the image feature within the panorama and template matching using *normalized cross correlation* (NCC). This matching of the keypoints gives the transformation between the existing panorama and the current camera frame and allows us to extend or update the panorama by mapping the camera frame into the panorama with high accuracy. This vision-based tracking approach shows only a small error of ~ 1º for the horizontal axis and is also able to relocalize in case the tracking is lost. Unfortunately, vision-based tracking is computational expensive, but we need to track the phone's rotation to build and update the panorama as the sensor based tracking is too inaccurate.

B. Sensor Fusion

The remote user's phone builds the distant panorama based on visual tracking using the camera feed obtained via WebRTC. However, we also need to compute the remote users own orientation. We can achieve this using visual tracking as described before but on the camera stream of the remote user. However, sometime there are not enough features in the environment or on slow phones we can increase the performance of the application by using sensor-based orientation. While not being as accurate, the sensor-based orientation has the advantage of having a higher update rate and lower latency. This enables the user to enjoy an interactive view, as the orientation sensors are accurate enough for looking around in the remote user's panorama. Additionally, in the rare case when the vision-based tracking is lost, we use the integrated orientation sensors as a backup. We compute the phone's orientation using the built-in sensors, namely the accelerometer, the magnetometer and the gyroscope by fusing together their individual results. The magnetometer and accelerometer are used to get an absolute orientation, which is noisy and has a slow update rate. Therefore, we use it in combination with the gyroscope, which is precise and quick, but drifts over time. We fuse together the results by using a

low-pass filter on the former to compensate for the noise and a high-pass filter on the latter to compensate for the drift before adding the two signals as outlined in [21].

VII. Application and Interface

In the following we present a brief summary of the implemented interface components. The central interface element of our developed application prototype shows the view into the environment. Assuming that we currently share our environment with the remote user, the remote user would browse the environment by rotating its phone to select the portion of the panorama to be displayed.

We decided that the local user (the user sharing the environment) is not just seeing the current camera frame but is provided with a bigger field of view. We can display a wider field of view based on the locally built panorama (see Figure 4). Using the panorama for both users ensures that they have a similar interface for the shared environment.

We also decided that we need to provide gaze awareness to indicate where each user is looking at. We achieve this in two ways: Firstly, we draw a red frame in the user's view indicating the current view of the other user. However, if the view differs too much (e.g. the other user looks exactly in the opposite direction) the frame is outside of the user's view frustum. We therefore also added a mini-map depicting a 360º view of the environment. It is basically a small version of the panorama, which is also constantly refreshed to show the latest update of the panorama. Similarly to the main view we draw small frames to indicate the view of the remote users. As the small panorama always covers 360º we see the current view of the remote person even if it is outside of our current view frustum and with this support gaze awareness in all cases.

In addition to visualising each other's current view, we also integrated pointing gestures using the touch screen. Both users can draw or highlight objects in the environment by touching the screen at the corresponding position. We track the finger's coordinate and send it to the other user using WebRTC's data channel. Based on the finger's coordinate and the tracking information of the device we can map the remote user's touch events (and consequently the drawn graphics) into the local user's screen.

Figure 4 shows the final implemented. The control buttons allow the users to manually restart the panoramic environment mapping, to save the panorama, to open the settings, and to switch roles. The latter allows the users to switch from seeing their environment to browsing the remote environment. For simplicity, we synchronized both users so switching one role automatically switches the other user's view as well. This should guarantee that the users of our systems always stay in the same environment (in the local user's or remote user's environment). This however, was a design decision and the implementation supports different combinations. The settings screen allows customization of the application including switching user interface components on or off (e.g., mini-map, frames for gaze awareness).

VIII. Evaluation

We decided to evaluate our initial prototype for pervasive telepresence using a technical evaluation and a user evaluation. The technical evaluation should inform us about the performance that can be achieved using our prototype and the bottlenecks of our pipeline. The user study aims to provide feedback on how our prototype achieves the desired goal of telepresence.

A. Technical evaluation

The performance of the system is dependent on many components in the system. Foremost the camera performance: We tried several current Android devices, all with the most recent Android version (Android 4.4 to Android 5.1 depending on the device). The camera performance when accessing the camera using WebRTC (which uses the official Android API) ranged from 10 frames per second (fps) on a Google Nexus 4 to consistent 30fps on a Google Nexus 5. Other devices (e.g. Samsung Galaxy S3, Samsung Galaxy S5, HTC One M8, LG G2) achieved a camera frame rate of 20-28fps. Based on these results we decided to use the Google Nexus 4 and Nexus 5 as worst and best case devices of all tested devices. The camera frames are delivered to us from WebRTC via a callback at which moment we start measuring the time spent in our pipeline. The first step is simply copying the camera frame into a queue and return from the callback to not stall WebRTC. This takes far less than 1ms for all tested phones (see Figure 5).

The next step required after acquiring the camera image is the visual tracking, which runs in a separate thread. This thread constantly waits for new threads to be delivered and picks them up within less than 1ms on all devices (Queue 1). Tracking is the most time consuming step in the pipeline. The tracking performance is affected by the used CPU but interestingly also by the camera frame rate. The higher the camera frame rate is, the smaller is the difference between two consecutive frames. This improves the tracking speed, as it is easier for the tracker to find matching features if the difference between frames is small. This can easily be seen when looking at the tracking time needed for the Nexus 4, which has the lowest camera frame rate (see Figure 5). When the tracking is finished, the frame is put in a second queue (Queue 2).

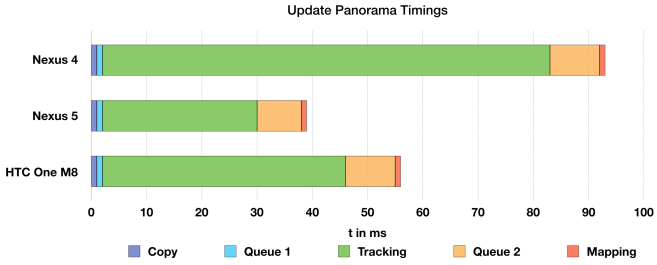

Fig. 5. Average timings for updating the panorama on different phones. Copy frame is the time that is required for copying the image data into memory, track frame is the time required for determing the tracking information using the provided frame. The Queue 1 and Queue 2 timings indicate the time the data is waiting to be picked up by the next thread. Mapping is the time it takes to update the panoramic map on the GPU using the provided frame and tracking data. Copy, Queue 1 and Mapping time have been rounded up to 1ms to be visible in the graph.

Finally, the last step in the pipeline is the panorama generation which is done in the rendering thread. This thread runs in intervals depending on the frame rate. Consequently, the time until the frame for the panorama is picked up by the rendering thread is on average theoretically half the average time of this interval. As rendering works asynchronously on the GPU, measuring the time spent rendering is difficult as

OpenGL function calls return before the rendering finished. The mapping time shown in Figure 5 shows only the time spent until the OpenGL calls to update the panorama are finished. Afterwards the user's view and the graphical user interface are rendered, which are not included in the figures.

In sum, we measured the time the system takes from accessing the camera frame to updating of the panorama. On the Nexus 5 it took on average 38ms, on the HTC One M8 it took 55ms, and on the Nexus 4 it took 92ms (see Figure 5). These timings were measured on a local client and while the connection with a remote phone was running. They give us an idea on how much more latency we can expect from our system compared to a system that simply renders the frames after they have been received by WebRTC.

Regardless of the phone our algorithm takes a bit longer during startup of the application when the first frame arrives as the system is initialized, which is in particular visible on the less powerful Nexus 4 (see Figure 6). We also evaluated the performance of the mapping algorithm in case the camera frames cannot be tracked. This means not enough image features can be matched between the camera frame and the panorama and this is usually a result of motion blur in the camera image caused by fast motions, non textured environments showing no image features (e.g. white walls), or bad connections resulting in high compressed camera frames from WebRTC. However, also in these extreme cases the algorithm performs only minimally worse and panorama mapping is skipped (see Figure 6).

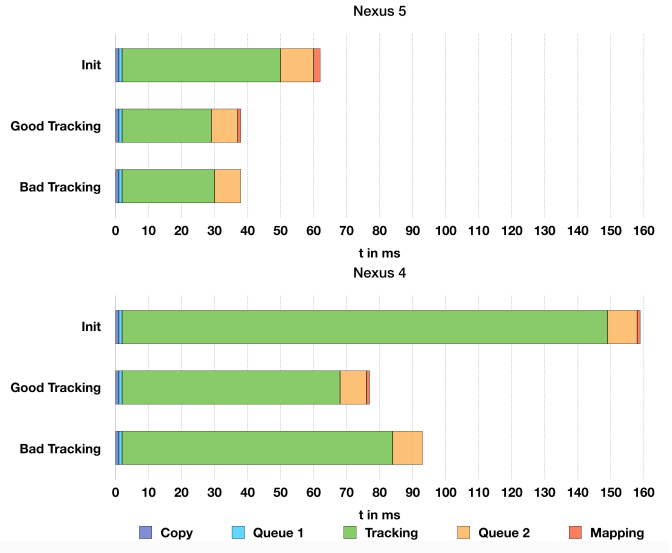

Fig. 6. Average timings for updating the panorama during different stages. For each device we measured the average time to process an incoming camera frame for the first 10 frames (init), a steady movement with panorama generation (good) and a jerky movement with bad tracking results where the tracking fails (bad). In the latter case no mapping is performed.

Looking at the overall frame rate of our system, we find, that due to the parallelized architecture, the application runs efficient enough without having to drop any frames. As the two queues involved are limited to two entries, dropping frames would happen as soon as the pipeline gets too slow. We are still able to update the panorama with 30fps (Nexus 5) or 10fps on the slowest device (Nexus 4) once the entire system is initialized.

Rendering the user interface, which includes rendering the panoramic map and user interface elements such as buttons or touch events in the panorama, is between 35fps and 55fps on all the tested devices. This thread runs in parallel to the other threads, thus we have interactivity even on the slower devices while only the panorama update is not interactive on the slower phones.

So far we did not consider the time it takes to send the data to the remote phone using WebRTC. We measured the time it took to capture a frame and send it to the remote client where the image is tracked and mapped into the panorama before it is displayed (end-to-end latency). This total time varies between 470ms and 550ms (Nexus 5 to Nexus 5) and 700-800ms (Nexus 4 to Nexus 4). The latency for only transmitting the image is between 400-500ms, which is in accordance with the latency of 300-500ms reported by WebRTC developers[2]. This is the time between the rendered result appearing on the local phone compared to the remote phone. We should also note that WebRTC adapts the coding based on the available bandwidth, which might lead to slightly different measurements. However, we repeated the measurements several times within 802.11g network with similar results.

Overall, we can show that our application is interactive in real-time (>30fps) on all tested phones while we were also able to show that even the panorama/environment update running in a separate thread is nearly interactive in real-time (20-30fps) on the faster of the tested phones while we measured a latency of max. 500ms between the communication partners.

B. *User evaluation*

In addition to the technical evaluation we conducted a user evaluation to receive feedback from novice users. We picked novice users, as this is a common form for investigating the feeling of presence in particular when using well-explored questionnaires [1,22,23]. We were in particular interested in how our system affects the experienced sense of presence. More particular, we aimed for investigating the effect of our system with regards to spatial presence ("sense of being there"), co-presence ("sense of being near"), and social presence ("sense of being together") [24]. To investigate these effects we decided for a study comparing our approach using a live updated panoramic representation ("panorama"), with a conventional system only showing the current camera image of the remote user ("non-panorama"). Our main hypotheses when comparing these two systems are stating that for the local user, i.e. the one who shares the environment (aka producer) the spatial presence will be unaffected by the panorama function, but social and co-presence will significantly increase with the panorama mode. For the remote user, i.e. the one experiencing the local user's environment (aka consumer) spatial and co-presence will significantly increase with the panorama mode, but social presence will be unaffected.

1) Study design

We decided against using existing systems (e.g. Skype) for the non-panorama mode as they use video codecs offering a different video quality in combination with a different user interface, which would introduce too many side effects. Instead we modified our system to just show the live camera feed without creating a panorama, and not displaying gaze awareness or pointing information (in the following called

[2] https://www.facebook.com/WebRTC/posts/726032664137227

VC). This modified version of our software shares many implementation details with our PanoVC prototype (e.g. similar latency and video quality, similar user interface).

We recruited 17 participants (15 in age group 19-29, 2 in 30-49 age group, 5 female, 12 male) through word of mouth. After filling out an agreement form we asked the participants about personal details including familiarity with mobile video call applications such as Skype or Facetime. This was followed by a short presentation of our prototypes (PanoVC and VC modes). The participants had a couple of minutes to try the prototype themselves and to ask questions about the usage. As our prototype at the current stage is not designed as a productive system we did not ask usability related questions. After that, we started the actual study. We asked the participants to place a video call with the remote person. We had two scenarios with two conditions each. We randomized the order of scenarios and conditions to avoid effects resulting from having the same order. One scenario is to share the environment with the remote user using each condition – using PanoVC and using the system only showing the video feed (VC). The other scenario is to browse the environment shared by the remote users again using each condition, PanoVC and VC only. We randomized scenarios and conditions beforehand, and we changed location so that the participants didn't see the environment twice. The other person was always a moderator to avoid effects resulting from participants knowing or not knowing each other as well as to guarantee a controlled behaviour from the communication partner. We always showed the frames for supporting gaze awareness allowing both users to see where the other users is looking at.

After each condition we asked the participants to fill out a short questionnaire to receive feedback on spatial presence, co-presence, and social presence. The questionnaire is based on 7-point Likert-like scale questions. For each sub-questionnaire we used existing verified questionnaires from the literature and only modified the wording if necessary. For spatial presence, we used the questionnaire by Schubert et al. [23]. For co-presence we used items from Biocca et al. [1] and adapted it to fit our purposes (changed 5-point to 7-point Likert-like scales to match the other questions and replaced "room" with "environment"). Finally, for social presence we applied the differential semantic measure by Short et al. [22]. Overall the questionnaire had 18 items that had to be answered using 7-point Likert-like scales.

After finishing the two scenarios and each condition, we asked the participants to fill out a final questionnaire where they had the chance to indicate their preference – PanoVC vs VC – in addition participants could provide other feedback.

2) Results

The sample size as well as the type of the scales lend themselves to non-parametric testing. This is the more conservative form of analysis for this type of study. The remote user was an indoor consumer of the panorama and non-panorama conditions. The local user (producer) was sharing the environment from outdoors locations.

A Wilcoxon Signed Rank Test (N=17) revealed a statistically significant increase in reported spatial presence of statistics for the indoor consumer, $z=-2.854$, $p < 0.05$, with a medium size ($r=0.488$). The mean score increased from non-panorama M=3.961 (SD=1.265) to panorama M=4.961 (SD=1.006).

Another Wilcoxon Signed Rank Test (N=17) revealed a statistically significant increase in reported social presence of statistics for the indoor consumer, $z=-2.57$, $p < 0.05$, with a medium effect size ($r=0.441$). The mean score increased from non-panorama M=4.578 (SD=0.959) to panorama M=5.108 (SD=0.931).

All other Wilcoxon Signed Rank Tests (N=17) did not reveal any statistically significant differences in the reported co-presence for the indoor consumer and also no significant differences in spatial, social, or co-presence for the outdoor producer.

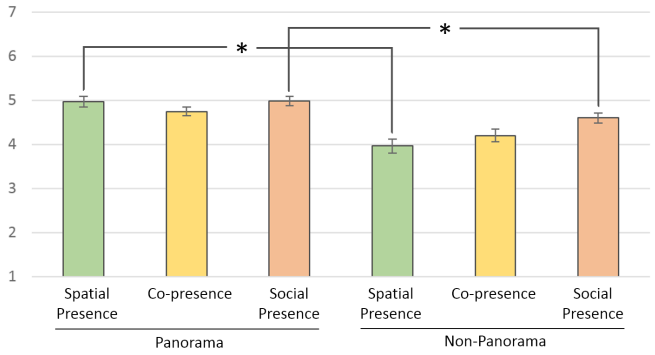

Fig. 7. Likert-like (1..7) scale responses for the three presence categories for the panorama (PanoVC) and non-panorama (VC) conditions for the indoor consumer. An asterisk (*) marks a significant difference.

Finally, the participants preferred the PanoVC over the VC mode (significance test against mid-point of scale, $p<0.005$). Hence, our results partially supported our hypotheses, but also rejected some.

3) Discussion

According to our results users experiencing our pervasive telepresence system feel that they are spatially "transported" to the environment of the sharing local user. They have a feeling of "being there". They also feel "being together" with that other person, also known as social presence. Both of those experiences are significantly improved with our PanoVC system. While our users also feel being near the other person (co-presence), this feeling is not significantly improved with PanoVC when compared to a standard videoconferencing solution (VC). However, the calculated p-value of $p=0.073$ was not too far from the significance level of 0.05 and a higher sample size might reveal a significant difference. Also, perhaps missing representations of the users themselves (e.g., by way of avatars or other video representations) have led to such a finding.

It is worth noting that while temporal issues in the panoramic representation are likely to have happened, none of the participants reported any problems with it. So we assume that the participants haven't noticed it or they updated the critical areas in the panorama removing the temporal artifacts.

IX. Conclusion

Telepresence was so far restricted to static setups where stationary hardware in dedicated rooms was required to achieve the feeling of presence by virtually bridging the distance between remote groups. Other solutions made use of custom robots that explore remote environments and share their view or custom hardware such as expensive panoramic cameras. In

this work we presented PanoVC a first approach for pervasive telepresence using only of-the-shelf mobile phones. PanoVC builds and updates a panorama on the fly and shares it with remote users. As remote users see the distant environment via the panorama they can manage their own view by rotating their phone. We described the implementation of PanoVC including the panorama mapping, the orientation sensing for both parties and the communication. While being demanding on the hardware, our evaluation shows that our approach runs interactively on current generation mobile phones and even more importantly, increases the feeling of spatial presence and social presence for users of our system. Furthermore, we also allow for instant switching of the roles as both communication partners use the same mobile hardware. To our best knowledge PanoVC is the first approach investigating telepresence between mobile clients and we believe it offers a substantial contribution to academic literature in the areas of telepresence, human-computer interaction and pervasive immersive communication. By providing new ways in terms of size, mobility and accessibility of telepresence solutions we open many new application areas including personal and professional communication and collaboration. Example key application areas are maintenance scenarios, where remote experts can support local workers by sharing the same environment or tourism settings, where people can share their location and experience with relatives and friends at home.

A. Limitations and Future Work

We see many promising future research directions including support for multiparty video or investigating wearable hardware. Currently our systems architecture is built for one-to-one communication and does not scale well phones connect directly to each other. Future iterations of the architecture could use a server who is responsible of distributing the video data to all connected clients to allow for group calls. Another limitation of our approach is the sensitivity to temporal changes when creating and updating the panoramic representation. Unfortunately, only using less pervasive and additional hardware such as panoramic cameras or wide-angle lenses would solve this problem. While our focus was set on enabling pervasive telepresence on mobile phones, future research should address better support for co-operation and collaboration by way of providing new forms of interaction.

ACKNOWLEDGMENT

We would like to thank Samuel O'Connell and Braden Hart for assisting with running the user study. We further thank all users participating in the experiment as well as all members of the Human-Computer Interaction Group of the University of Otago for their input and valuable discussions.

REFERENCES

[1] F. Biocca, C. Harms, and J. Gregg, "The networked minds measure of social presence: Pilot test of the factor structure and concurrent validity," in *Proceedings of the 4th International Workshop on Presence*, 2001, pp. 1-9.

[2] M. Lombard and T. Ditton, "At the heart of it all: The concept of presence," in Journal of Computer-Mediated Communication, vol. 3, no. 2, 1997.

[3] P. de Greef, W. IJsselsteijn, "Social Presence in the PhotoShare Tele-Application," in *Proceedings of Presence 2000 - 3rd International Workshop on Presence*, 2000.

[4] O. Schreer, I. Feldmann, N. Atzpadin, P. Eisert, P. Kauff, and H. J. W. Belt, "3D Presence - A System Concept for Multi-User and Multi-Party Immersive 3D Videoconferencing," in *Visual Media Production (CVMP 2008), 5th European Conference on*, 2008, pp. 1-8.

[5] M. Kuechler and A. Kunz, "HoloPort - A Device for Simultaneous Video and Data Conferencing Featuring Gaze Awareness," in *Virtual Reality Conference*, 2006, pp. 81-88.

[6] D. T. Nguyen and J. Canny, "Multiview: improving trust in group video conferencing through spatial faithfulness," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2007, pp. 1465-1474.

[7] N. P. Jouppi, "First steps towards mutually-immersive mobile telepresence," in *Proceedings of the 2002 ACM conference on Computer supported cooperative work - CSCW '02*, 2002, pp. 354.

[8] S. Katz, D. Kimber, W. Su, G. Gordon, and D. Severns, "Polly," in *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services - MobileHCI '14*, 2014, pp. 625-630.

[9] B. Jones, A. Witcraft, S. Bateman, C. Neustaedter, and A. Tang, "Mechanics of Camera Work in Mobile Video Collaboration," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*, 2015, pp. 957-966.

[10] S. Fussell, L. Setlock, J. Yang, J. Ou, E. Mauer, and A. Kramer, "Gestures Over Video Streams to Support Remote Collaboration on Physical Tasks," in *Human-Computer Interaction*, vol. 19, no. 3, 2004, pp. 273–309.

[11] 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*, 2013, pp. 1425-1430.

[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*, 2014, pp. 449-459.

[13] S. Kasahara and J. Rekimoto, "JackIn", in *Proceedings of the 5th Augmented Human International Conference on - AH '14*, 2014, pp. 1-8.

[14] S. Kasahara, S. Nagai, and J. Rekimoto, "First Person Omnidirectional Video: System Design and Implications for Immersive Experience," in *Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video*, 2015, pp. 33-42.

[15] M. Billinghurst, A. Nassani, and C. Reichherzer, "Social panoramas: using wearable computers to share experiences, " in *SIGGRAPH Asia 2014 Mobile Graphics and Interactive Applications*, 2014, p. 25.

[16] M. Handley, C. Perkins, and V. Jacobson, (2006, July). SDP: session description protocol [Online]. Available: http://tools.ietf.org/html/rfc4566

[17] W3C (2015, September 29). WebRTC 1.0: Real-time Communication Between Browsers [Online]. Available: http://w3c.github.io/webrtc-pc/

[18] I. Fette and A. Melnikov, (2011, December). The websocket protocol [Online]. Available: https://tools.ietf.org/html/rfc6455

[19] D. Wagner, A. Mulloni, T. Langlotz, and D. Schmalstieg, "Real-time panoramic mapping and tracking on mobile phones," in *Virtual Reality Conference (VR)*, 2010 IEEE, 2010, pp. 211-218.

[20] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2005, pp. 1508-1515.

[21] S. Colton, (2007). The balance filter: a simple solution for integrating accelerometer and gyroscope measurements for a balancing platform [Online]. Available: http://burt.googlecode.com/svn-history/r121/trunk/Hardware/Sensors/filter.pdf

[22] J. Short, E. Williams, and B. Christie, "The social psychology of telecommunications". Wiley, London, 1976.

[23] T. Schubert, F. Friedmann and H. Regenbrecht, "The experience of presence: Factor analytic insights". in *Presence: Teleoperators and virtual environments*, 10(3), MIT Press, Cambridge/MA, USA. 2001, 266-281

[24] K. Nowak, "Defining and Differentiating Copresence, Social presence and Presence as Transportation". Presence Workshop, Philadelphia, 2001, 1-23