P3X Technical Report

Eye-Contact and Gaze in Web Video Conferences

STUDENT GROUP

Group 8

He, Muchen	44638154	*mhe@ece.ubc.ca*
Xia, Kaseya	27553304	*zxia@uvm.edu*
Xiong, Beibei	13233747	*beibei971030@gmail.com*

INSTRUCTOR

Dr. Sidney Fels

Electrical and Computer Engineering, The University of British Columbia

Revision 1.3 — April 19, 2021

THE UNIVERSITY OF BRITISH COLUMBIA

Electrical and Computer Engineering

Contents

Preface

This document is intended for the instructor or students of the human-computer interface course CPEN 541 to learn and possibly reproduce the experimental setup and results. The document outlines in detail the research questions we are pursing, as well as the motivation. This document also describe the prototype and its experiments we developed to test our research questions along with our findings. Lastly, this document features the limitations as well as the future work (including what the prototype would ideally look like).

Please also refer to our other submitted work, including our conference paper, demonstrative video, and presentation slides.

We can be contacted via our emails (listed at the top of the document). All the code for this project can be found on GitHub: https://github.com/CPEN541-FutureGazer.

1 Introduction and Problem

Over the last year, we all observed and experienced first-hand at attempting to scavenge the productivity and work-ethics we once had while working from home while in a pandemic. And one of the most important work-related ritual is none other than meetings and hangouts.

As limited by the restriction and the isolation to curb the infectious virus, we sacrificed the ability to meet with people face-to-face (F2F) — whether that be getting work done, attending classes, discussing and brainstorming ideas, and even asking for help and or assistance from peers. Out of necessity, we all forced to use online web-video-conferencing (WVC) software such as Zoom, Skype, etc. as a substitute.

Despite these companies touting the technological advancements, the intuitive of its user-interfaces, the affordable cost of "free" to its users, they all had this one *glaring* (pun-intended) problem that none of these software truly provided in the absence of F2F meetings: that **eye-contact** and **gaze** are an essential part of body language and non-verbal communications in social situations.

Conventional WVC software evolved from 1-on-1 meetings, where the only other person on the screen is the one you talk to, like in a phone call. This has the few problems of modern, large meetings involving more people, since the other person's face is the only thing you look at and thus easier to make eye-contact and attention.

'insert picture of older Skype/FaceTime image'

However, seen in the image below, as we add more participants to the meeting, instead of being an intimate experience, we are often presented with a gallery, somewhat like a bookshelf view of all the participants. Each participant is placed in a uniform cell as part of a grid. One could describe it as a jail-cell, lacking in connection, organicity, and coherency. Furthermore, because everyone is looking at their own screen, towards their camera, we get a mostly static view of all the participants looking out of the screen on the receiving end. Somewhat akin to have a large audience all staring at you — even if you're one among them passively listening.

'insert gallery view of zoom' in contrast with 'image of people sitting around a table looking at each other'

2 Background and Prior Works

A large body of prior work has explored that eye contact is a critical aspect of human communication. [1, 2] Eye contact plays an important role in both in person and a WVC system. [3, 4] Therefore, it's critical and necessary to preserve eye contact in order to realistically imitate real-world communication in WVC systems. However, perceiving eye contact is difficult in existing video-conferencing systems and hence limits their effectiveness. [2] The lay-out of the camera and monitor severely restricted the support of mutual gaze. Using current WVC systems, users tend to look at the face of the person talking which is rendered in a window within the display(monitor). But the camera is typically located at the top of the screen. Thus, it's impossible to make eye contact. People who use consumer WVC systems, such as Zoom, Skype, experience this problem frequently. This problem has been around since the dawn of video conferencing in 1969 [5] and has not yet been convincingly addressed for consumer-level systems.

Some researchers aim to solve this by using custom-made hardware setups that change the position of the camera using a system of mirrors [6,7]. These setups are usually too expensive for a consumer-level system. Software algorithms solutions have also been explored by synthesizing an image from a novel

viewpoint different from that of the real camera. This method normally proceeds in two stages, first they reconstruct the geometry of the scene and in second stage, they render the geometry from the novel viewpoint. [8, 9, 10, 11, 12] Those methods usually require a number of cameras and not very practical and affordable for consumer-level. Besides, those methods also have a convoluted setup and are difficult to achieve in real-time.

Some gaze correction systems are also proposed, targeting at a peer- to-peer video conferencing model that runs in real-time on average consumer hardware and requires only one hybrid depth/color sensor such as the Kinect. [13] However, when there are more than two persons involved in a web video conference, even with gaze corrected view, users still cannot tell whether a person is looking at him or someone else in the meeting. With the gaze correction, it will create the illusion that everyone in this meeting is looking out of the screen. This could cause a serious confusion.

2.1 Eye Contact in Multi-person Conversation

Most studies of eye contact during conversations focused on two-person communication [14]. However, multi-person conversational structure becomes more complicated when a third speaker is introduced. It has long been presumed that eye contact provides critical information in conversations. Isaacs and Tang [15] performed a usability study of a group of five participants using a desktop video conferencing system. They found that during video conferencing, users addressed each other by name and started explicitly requesting individuals to start talking. In face-to-face interaction, they found people used their eye gaze to indicate whom they were addressing. Sellen [16] was one of the first to formally investigate the effects of eye contact on the turn taking process in four-person video conferencing. Unfortunately, she found no effects because the video conferencing system she implemented did not accurately convey eye contact [16]. Vertegaal et al. [17] found that without eye contact, 88% of the participants indicated they had trouble perceiving whom their partners were talking to.

Therefore, we propose our system

NOTE: ALL References can be found in Latex

‘insert existing solution‘

‘insert VR 3d chat apps‘

‘insert other research work‘

3 Research Questions and Scope

This motivated us to explore alternative ways to represent participants in WVC applications to increase engagement, interactivity, and attention.

We are set out to build a prototype WVC application to study the perceived effects of eye-contact, gaze, and head orientation in a virtual 3D space to make up the missing aspects from in-person F2F interactions.

‘insert research question‘

‘insert scope of this project‘

Our project explores whether adding eye-contact to the current WVC system will enhance the sense of interaction and presence of the users. Conventional WVC services only offer standard visual and audio communication, and they do not support intuitive and personalized eye-contact between users. Therefore, people still prefer face-to-face meetings because of the highly interactive meeting environment [P. Hart]. In distant-learning classes, for example, presenters (e.g. professors, teachers, students) often feel distracted or disengaged when there are no audiovisual feedback coming from the audience. These feedback include eye contact, gaze direction, and other body language cues.

For example, in the Gallery View in Zoom (and other WVC applications), the audience consists of black boxes and name tags, as shown in Figure 1. If the participant has video turned on, the webcam video stream replaces their box. We will refer to the space each participant takes up on the screen as their footprint. Notice that everyone has the same and uniform footprint, regardless if they are paying attention to the meeting or the active participant.

Thus, current systems neglect important cues presenters use to moderate their lecture. The lack of interactivity is one reason why online lectures are less effective than in-person lectures [T. Nygyen]. In one of the first experimental studies [D. Figlio] on the effects of traditional instruction versus online learning, students attend live lectures instead of watching the same lectures online while supplemental materials and instructions were the same. Researchers [D. Figlio] found modest evidence that the traditional format trumps the online format in engagement. Many people also think it is odd to see their faces during conversations, and it is hard to look away — significantly distracts participants during WVCs.

Lastly, some people are camera-shy and do not want to reveal themselves in WVCs. Thus we decide to explore the effect on participation and engagement from using an avatar as an alias instead of a live video.

The key metrics we want to observe in this project are: participant's attention, engagement, and the feeling of connectedness. To explore parameters that effect these metrics, we consolidate these ideas into three core research questions (hereafter will be refer to as **RQ1**, **RQ2**, and **RQ3**):

1. Can a person tell if they are being looked at in a WVC and how can 3D avatars be augmented to enhance this experience.

2. Can a person tell if other participants are looking at each other in a WVC and how using 3D avatars can be augmented to increase engagement.

3. How does a person's attention change as the avatars augmented with WVC enables eye-contact and gaze.

We intend to modify design parameters to our prototype user-interface (UI) of our mock WVC program to study the behaviour of participants.

4 The Prototype

This section describes the technical details regards to our implementation. We first outline the language and framework this prototype is developed on, then we describe the rendering, layout, and orientation calculation. Lastly we go over the configuration files, events, timed and randomized sequences, and integrated questionaries that aid us in user-experimentation.

Note: Since this project is not mainly focused on the usability study of this concept, we only implemented the minimum-viable-product to carry-out the user experiments due to time-constraints. We discuss how this prototype could be improved in future>prototype section.

4.1 Platform

The prototype is mostly developed in Processing cite (a Java-based graphic-centric language mostly used in education, prototype, and visual arts) and Unity Engine cite (a popular free-to-use game engine). The Unity engine wraps around the Processing prototype to provide user-testing interface such as integrated questionaries and videos.

We chose Processing and Unity as they have strong support of 3D environment, video and audio playback, robust mouse input, ease-of-use, and quick prototype turn-around overheads. Both frameworks are also cross-platform and can execute on Windows, macOS, and Linux. However, as we also discuss in Section limitations section, recent macOS cite macOS update added a *Gatekeeper* security feature that prevents un-sighed software from running — thus complicating the user-testing process, as distributed prototype executables to the participants cannot be opened.

4.2 3D Environment

insert figure of a typical view of the prototype in Unity app

Figure insert figure shows a typical 3D environment rendered in the prototype app. We base the user interface (UI) design — including colours, buttons, and layout — from existing WVC apps such as Zoom to present the participants with a familiar user experience. By doing so, we limit other factors that would interfere with our user experiments. However unlike traditional WVC apps, we replace the centre (where typically there would be a gallery or grid view of camera video feeds) with our prototype avatar representation.

From hereafter, we will refer to these visual representation of participants in the meeting as *Avatar Views* (*View* class). As outlined earlier in [Section on research questions](research-questions-and-scope), we propose two types of avatars to explore: 3D head avatars (*HeadView* class) and 2D eye avatars (*EyeView* class).

insert figure of head view class and the eye view class

For user experiments, we create *mock avatars* to replicate/simulate real meeting participants. This creates an illusion for the test participants as if they're real people to help us speed up the testing process without involving lots of people. However, as it is an illusion, it might affect how people perceive our prototype — as we discuss in [Section 7. Limitations](limitations).

4.2.1 Heads

show image of 3d model of the head

insert head avatar grid view

In the 3D environment, the head avatars are loaded from a free-to-use .OBJ 3D model downloaded from *TurboSquid* with standard usage license cite We load the object into the 3D scene once, and use the

same model instance for all other head avatar *HeadView* objects to save computation — since they're essentially the same 3D model, just redrawn with different transformations.

Due to time and budget constraints, we are limited to a low-polygon-count model of the head, with no texture, no rigging, no soft-body animation, and no physics simulation. The 3D model of the head, as shown in Figure TODO, has minimum features of a face. Despite the low-quality 3D models, the primitiveness of the 3D model helps maintaining a high rendering frame-rate while the prototype is running, even without writing custom shaders and performing fine-tuned optimizations — especially when there could potentially be up to 9 to 25 avatars being drawn concurrently on to the screen. This is important as we need the head avatars to transform and render in real-time as if they're real inputs in a WVC application. As we will discuss in Section TODO: future work, if given more budget and 3D talent, more work can be allocated to polishing the visual appeal of the avatars to be more inviting and friendly, and ultimately improve user-experience.

4.2.2 Eyes

The 2D eyes avatar was inspired from goggly eyes and novel desktop widgets (e.g. XEyes TODO: cite) created as general amusement. However, we decided to explore this as an alternative to the head avatars to see whether if 3D and head orientation is required to deliver eye-contact and gaze hints to the meeting participants.

insert breakdown diagram of the eye rendering

insert eye avatar view

insert eye avatar grid view

With eyes avatars, instead of a 3D object transformed to orient a direction, we simply render a pair of pupils and irises on a white background as a sub-image (Figure TODO). Then depending on where the eye should be looking at, we rendering this sub-image with a transform that translates it in x or y direction (Figure TODO). Finally, we create an eye mask that only only renders the eye itself (Figure TODO).

The result is somewhat similar to a gallery view of meeting participants in a traditional WVC application, except with only the eyes and their gaze visually represented.

4.2.3 UI Elements

As discussed in Section TODO (3d-environments) seen in Figure TODO (from 3d-environment section). The 3D head avatars and 2D eye avatars are rendered at $z = 0$. In front of that, we render the UI elements such as the bottom menu bar, as well nameplates to aid in our experiments and to provide a familiar environment to the participants.

insert image of talking halo

When a participant corresponding to an avatar is talking, we have an option to activate a yellow halo behind the avatar for indication. To do this, each avatar *View* base class has a flag isFocused that can be toggled. If this flag is on, then we draw this additional halo on a layer behind the avatars.

insert image of the point light source

While not used in our final experiments, we also added an option to highlight avatars in on the screen based on mouse cursor positions. Rather than drawing a bounding box around the avatar, we thought it was appropriate to instead model the mouse cursor effects as a point-source-light with some vibrant colour, such as seen in Figure TODO. This option can be turned on via experiment configuration file (see Section TODO: configuration file).

4.3 View Modes

To simplify the behaviour of mock avatars, we have three pre-programmed modes for the avatars to follow:

1. **NORMAL**: the avatar is in normal mode, it will follow and track a given set of target coordinates (see [Section TODO](target-coordinates)).
2. **STARE**: the avatar should stare at the active participant by gazing directly outwards from the screen.
3. **RANDOM**: the avatar randomly looks around as if they are distracted. Current implementation of the random mode uses a series of Perlin noise functions to approximate how real head moves around randomly.

Note that this would only apply to the mock avatars used in user experiments to simulate real people.

During user experiments, we can dynamically and programmatically change the modes of the avatars to simulate whether a person in the meeting is paying attention or not paying attention, and looking at the test subject participant, or anywhere else on the screen.

4.4 View Calculation

This section talks about the math and algorithms created to compute the target coordinates — the screen-space coordinates of where that avatar should be looking at. As well as the mapping between the target coordinates to a rotation (for head avatars) or a translation (for eye avatars) transformations.

4.4.1 Target Coordinates

Each avatar (*View* base class) has attributes targetX and targetY which corresponds to where the avatar should directly look at in *normal* mode. These attributes are not used in *random* and *stare* modes.

For example, if the avatar is set to track the mouse cursor's screen position, then we trivially set:

```
avatar.targetX = mouseX
avatar.targetY = mouseY
```

If an avatar A is set to look at another avatar B, we can set the target coordinates of A to the spatial coordinates of B:

```
A.targetX = B.x
A.targetY = B.y
```

4.4.2 Rotation Calculation

Once an avatar knows *where* to look (i.e. the target coordinates), it needs to compute *how* to look in that direction (i.e. compute the corresponding transforms required to show the correct visual representation).

insert image of mapping from targetXy to offset Xy

For the 2D eye avatars, this mapping from target coordinates to transformation is a simple 2D translation $(\Delta x, \Delta y)$, as seen in Figure TODO. First, a difference vector $\vec{d}$ is computed from the avatar's local origin to the target coordinates. We then scale it by some factor s to control the sensitivity of this translation. Finally, we constrain the magnitude of $s\vec{d}$ to the radius of the eyes to ensure the pupil do not go off the eye, creating a white eye.

For the 3D head avatars, the mapping is instead from the target coordinates to a set of rotations along the x, y, and z-axis. The simplest way to do this is to draw a 3D vector from the head avatar origin to the target coordinates. Then using trigonometry on the vector, we can find all angles corresponds to the x, y, and z rotations.

Notice, that up to this point, we have not established the third, z-component, of the target coordinates. This is because it depends on what the head avatar should be looking at. If the avatar is looking at another avatar, we leave $z = 0$ since all avatars are on the $z = 0$ plane. However, if the avatar were to follow the mouse or stare at the user, then we must set $z = z'$, where z' is the z-offset of the camera. The difference between with vs. Without this z-correction can be seen in Figure TODO

insert figure of heads looking straight on vs at the camera

4.4.3 Realism Approximations

For both eye and head avatars, instead of applying the transformation in the renders according to he target coordinates instantaneously, we use a linear-interpolation function to smoothen the motion and simulate a more natural response — a response with mass, inertia, and a sense of reaction time delay that would exist in real people.

We also added an option to make the head wobble controlled by some noise to make the avatars feel less robotic and more organic. If a 3D head avatar's isFocused flag is set on (such as when the participant the avatar belongs to is speaking), the wobble amplitude is slightly increased to approximate the extra motion due to mouth movements.

4.5 Configuration Files

To facilitate a series of automated, consistent, yet randomly generated user experiment scenarios, we developed a configuration framework for our prototype. Each experiment setup (see [Section TODO](experiment-setup)) is contained inside a config.json file. Each file contains all the experiment parameters such as number of avatars, type of avatars, names, and the sequence of modes, target coordinates, etc.

4.5.1 Initialization

The file can be loaded at initialization-time of the experiment. Upon which, all the parameters in the configuration file is read, and the defined avatars are populated in the scene.

4.5.2 Events

The sequence of state changes in the prototype user experiment is controlled by *events*. These events are defined in the configuration files as a single array that represents a timeline. The different types of events supported are:

- Change mode
- Change avatar target
- Set focus flags
- Play sound
- Stop experiment

For more details regarding events, please refer to the source code.

4.6 Unity Engine Wrapper

To further facilitate user testing, we wrapped the Processing prototype in a Unity engine user interface, where participants are guided without the constant supervision from us. Each experiment setup is sequenced in order and appropriate questionaries are prompted in between each setup. Finally, the questionnaire responses are automatically logged in participants' computer, making it easy for review.

5 Experiment Setup

- Recall research questions - In pursuit of these answering these questions, we proposed these experiments that study how users feel about interacting using our prototype - Describe all the different experiment configurations

6 Experiment Results and Analysis

- TODO: need to show results - TODO: what can we learn from these results?

7 Limitations

- Covid-19 working remotely, so it's hard to sync up between team members - Technical limitations (technical) - 2D and 3D avatars/views are primitive (barely colours and textures) - Not customizable, could affect how people feel about usability because it doesn't convey as much expression as originally intended - 3D head lack eyes with animation and texture (unfinished due to technical and time constraints) – also affects how people perceive eye-contact in WVC because it looks creepy (and literally have no eyes). - Primitive lighting in 3D environment (simple directional light creates uncanny environment) - Limitations in running the experiments: - User testing extremely difficult, despite the prototype framework being cross-platform, exporting/deploying apps required notarization (a security feature that only allows officially signed app to run), so we need last-minute hacks to have the user experiments run - Small sample size - Limitations in collection of data - Ideally, we want to use some gaze-tracking hardware/software to actually track and log where the users are looking at (as seen in the omitted experiment setup using

Tobii), but because none of the user experiments are performed in person, we physically cannot collect those data. - A workaround is to have the user click/move the mouse to where they're looking at, but because of aforementioned platform limitations, we can't do that either - Bias of data - Sample size could be to homogeneous (break down of first-spoken language, ethnicity, geographic location) - Time of day when running the experiment affects participant's attention, energy, and mood, and ultimately affect experiment outcomes

7.1 Prototype Development

- Unified 3D framework/engine - Apple Gatekeeper notarization - More sophisticated 3D models and animations - Add real-time gaze tracking support - More dynamic and fluid 3D environment (since right now all the heads and eyes are locked onto a grid) -

8 Future Work

This section talks about the future work

- Testing a combination of eyes and heads - Testing a large grid of eyes vs. Heads - Different combination of styles of eyes and heads

Explore how avatars can move in the space – what if the avatars can scale based on attention, what if the avatars move back and forth (z direction) based on attention? What you can physically move around in the space

Improvement to the avatar models:

- head nodding - breathing - mouth shapes - hand gestures - improved randomization - improved blinking

We would also like to see how this concept can extend to other fields: - how we can incorporate ML and facial expression recognition technology - reduce on internet bandwidth for less fortunate participants (lowers data usage) -

9 Conclusion