November 2022

PROVIDING ENHANCED AUGMENTED REALITY EXPERIENCES USING RELATIONSHIP AND CONTEXTUAL DATA

Ram Mohan R

Faisal Siyavudeen

Rajarshee Dhar

PROVIDING ENHANCED AUGMENTED REALITY EXPERIENCES USING RELATIONSHIP AND CONTEXTUAL DATA

AUTHORS:
Ram Mohan R
Faisal Siyavudeen
Rajarshee Dhar

ABSTRACT

Techniques are provided herein for using relationship intelligence of the participants in the meeting to provide a customized Augmented Reality (AR) experiences for each participant of that meeting whose meeting client is AR-ready. These techniques may allow developers to provide their own relationship information and integrate that information with the meeting platform by proposing a set of Communications Platform as a Service (CPaaS) Software Development Kit (SDK) enhancements described herein.

DETAILED DESCRIPTION

Augmented Reality (AR) has been around for some time. Many meeting providers offer various AR experiences for live meetings. Current AR experiences have the following limitations:

Problem 1:

Current AR experiences cannot provide customized AR experiences. For example, they lack the ability to perform AR operations on a participant's video locally before rendering the video to the user based on that user's relation/association with other participants in the bridge.

In an enterprise setup, the meeting client can use personal insights from the meeting provider and/or third-party relationship data to find the top collaborators of a participant who are also in the current meeting and provide AR experiences. This may be considered a customized AR experience because, for each participant in the bridge, their top collaborators may not be same set of participants. The meeting client may use the personal

insights to identify the top collaborators for that participant and render an additional AR experience. For example, the client may superimpose additional content (e.g., expertise relevant to the context of the meeting, recent projects done in that context (e.g., links, images, etc.)) on their video that is relevant to the context of that meeting.

Problem 2:

Cannot provide AR experiences based on contextual data of the meeting or trigger AR experiences based on user actions (e.g., the user clicking or hovering over a participant's video or other triggers tom show AR data about that participant relevant to the meeting).

Problem 3:

Cannot allow third party developers to bring their own AR devices or their own relationship data, which can help enrich meeting experiences.

An example flow is provided below with the following high-level steps. In this example, the user is a participant who joins a meeting. The user is provided with an AR experience using relationship intelligence and contextual data.

STEP 1. Users join the meeting from their personal meeting clients (e.g., application, desk-phone, video endpoints, etc.).

STEP 2. The user is prompted to enable an AR experience for that meeting. The user can choose to accept this (and optionally allow the meeting system to use that as a default setting for future meetings of same series, or all meetings, as selected by the user).

STEP 3. The user's meeting client may subscribe and obtain "roster" information, which may identify the list of participants in the meeting.

STEP 4. The user's meeting client may use personal insights to discover the relationship between the participants of the meeting and the current user for which the query is made. Various relationships may be considered. Some examples may include:
-participants in the roster who collaborate often with the user
-participants in the roster who are in same team/group/Business Unit (BU), at various levels, as the user
-participants in the roster who are leaders/managers of the user
-common expertise between the participants in the meeting
-common projects worked

STEP 4.1: The meeting client may also use the participant relationship to fetch related information, such as topics that the participants worked on together, based on their emails, text messages, previous meeting notes, etc. These in turn can help the meeting client obtain relevant AR data to render on top of the participant's video.

STEP 5. The meeting system also learns the context of the meetings. One technique involves sending live audio (both transmission and reception) to a speech to text system, obtaining a live transcript, identifying the keywords (context) based on the topic/agenda of the meeting, and performing transcript analysis (e.g., Natural Language Understanding (NLU) or other techniques).

STEP 6. Once the meeting client identifies the relationship between various participants in the roster to the user in question, it maps this relationship to the participants' streams and maintains a mapping of {user<->participants<->streams and relationships}. It also uses contextual data from step (5) to fetch additional augmented data (e.g., 3D images, articles, etc.) related to the context of the meeting being discussed and related to other participants in the meeting based on relationship knowledge.

STEP 7. Some examples of customized AR experiences that can be rendered based on relationship intelligence and contextual data include:

-If in a meeting the participants are talking about the webRTC Software Development Kit (SDK), the client uses participant relationships to learn that two participants have expertise in webRTC and recently worked on an SDK, fetch some related content, and augment that on the AR device for the user.
-The meeting client may use the relationship to superimpose additional information about participants (e.g., their expertise, recent projects (e.g., links), recent conversations of the user in question with those participants, etc.). The meeting client may do this locally by fetching the additional content to augment and mix before rendering; or this may be done on a server.
-The meeting client may use relationship information such as participants in the same team, group, etc. to show superimposition that indicates the other participants' relationships to the user in question.

STEP 8. The following steps describe briefly how the AR experience based on relationship knowledge and contextual information can be generated.
8.1) The meeting client subscribes to a service (e.g., people insight) and provides the participant list learned from roster.
8.2) The meeting client requests relationship information about a particular participant. The trigger may be the active speaker, based on the user hovering over one of the participants' videos, clicking a name, etc.
8.3) The meeting client may also use contextual data (learned from step 5) to query specific information about the participant(s) in which the user is interested (learned in the previous step).
8.4) The people insights service may fetch information about the participant (or the set of participants), as mentioned in step 4, and push it to the user as requested. Since participants of a meeting can define filters regarding what data is visible/accessible for that meeting and who can access the same, the insights service may return data as appropriate.
8.5) The meeting client receives relationship information from insights based on the requested filters. Alternatively, the insights service may provide all the data and the meeting client may filter the data based on filters discussed in steps 8.2 and 8.3.

8.6) The meeting client mixes the video with the AR data. The mixing itself may be done locally, at the client itself, or at a server as desired by the administrator. This may be defined by the client settings.

8.7) The AR data may be directly rendered (superimposed) on an AR device that the user has attached to a meeting application.

Figure 1 below illustrates an example flow showing a meeting application rendering custom AR experiences.

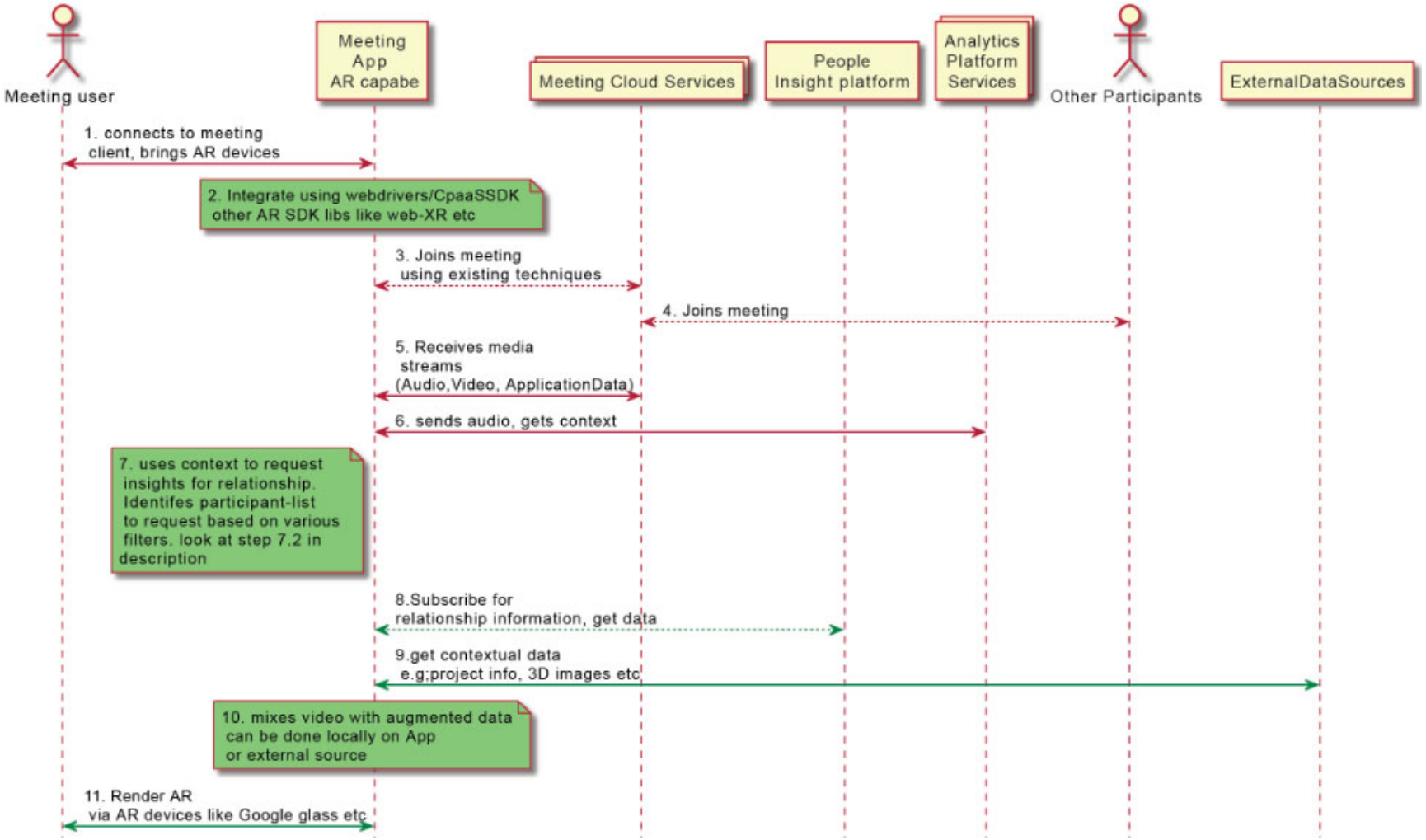
Meeting user
Meeting App AR capabe
Meeting Cloud Services
People Insight platform
Analytics Platform Services
Other Participants
ExternalDataSources
1. connects to meeting client, brings AR devices
2. Integrate using webdrivers/CpaaSSDK other AR SDK libs like web-XR etc
3. Joins meeting using existing techniques
4. Joins meeting
5. Receives media streams (Audio,Video, ApplicationData)
6. sends audio, gets context
7. uses context to request insights for relationship. Identifes participant-list to request based on various filters. look at step 7.2 in description
8.Subscribe for relationship information, get data
9.get contextual data e.g;project info, 3D images etc
10. mixes video with augmented data can be done locally on App or external source
11. Render AR via AR devices like Google glass etc

Figure 1

Communications Platform as a Service (CPaaS) SDK enhancements

The meeting platform may also enhance its CPaaS SDK offering to include the following. The CPaaS SDK may be based on iOS/Android/Desktop or a web SDK as supported based on the platform's AR support.

- A CPaaS SDK Application Programming Interface (API) set that allows developers to write applications that allow them to provide their own relationship data and integrate that data with meetings, thus enabling generation of experiences local to an organization. For example, an enterprise that has a lot of data regarding a relationship may provide and integrate that data.
- A CPaaS SDK API set that allows developers to integrate with their AR devices ("bring your own AR device") and superimpose the AR data on those devices. Depending on the platform, the CPaaS SDK layer may enable integrating with various AR devices. For example, with a web SDK, the CPaaS may use the webXR World Wide Web Consortium (W3C) API to connect to any compatible AR device. In some cases where the platform does not have a compatible driver, the browser may download the drivers for the same once the device is detected and run it as a WebAssembly (Wasm) container in the browser to enable JavaScript applications to communicate with the AR device using the webXR W3C API and render the AR data.
- A CPaaS SDK API set that allows developers to define which actions should prompt the meeting platform/client to trigger the AR data fetch and rendering. Essentially, the filters from step 8.2 may be set by the API set).

In one example, provided is a method by which a meeting client can learn a relationship between participants in a meeting and a user of that meeting client.

In another example, a method is provided by which the meeting client uses the relationship of participants to fetch the appropriate context to augment. This may include previous interactions between the user and the participants (e.g., emails, real-time chat sessions related to topics currently being discussed, documents shared related to current context, past meetings regarding the same context, etc.).

In another example, a method is provided to show an AR data rendering based on user triggers. These triggers may render AR data related to a participant on various

channels. The AR data may include any of those mentioned above. Triggers may include:
- The user moving / hovering mouse/touch/click over a participant's tile (e.g., video)
- A participant becoming active (e.g., active speaker, video on focus, sharing screen, etc.)
- Based on the context of the meeting, the participants being relevant (e.g., if the meeting agenda is to discuss WebRTC SDK design, AR data relating to participants who have prior exposure/expertise in that area may be rendered)

In another example, a method is provided to render AR data on various channels. This may involve rendering AR data on alternate channels while the participant's audio/video/share continues on existing channels. Alternate channels may include glass, other sensor devices, etc. When rendered on alternate channels, the AR data may be tagged with the participant's identity.

In another example, a CPaaS SDK API set is provided that allows developers to write applications that allow them to provide their own relationship data and integrate it with meetings. This may enable generation of experiences local to an organization. For example, an enterprise that has a lot of data regarding relationships may provide that data and integrate it.

In another example, a method is provided to allow developers to bring any AR/Virtual Reality (VR) devices and integrate them into platforms by providing a CPaaS SDK to fetch and download the drivers for the same once the device is detected and run it as a container. For example, a Wasm container with an AR/VR driver pushed to a browser on the fly may allow any web applications to communicate with any device.

In another example, a CPaaS SDK API set is provided that allows developers to define which actions should prompt the meeting platform/client to trigger the AR data fetch and rendering.

A condensed set of steps is provided as follows:

1. The user's meeting application joins the meeting, subscribes to the participant roster, and obtains the participant details.
2. The meeting application also obtains relationship intelligence from personal insight or similar platforms. The user's meeting application learns a relationship of the user to the participants in the meeting.

3. The meeting application uses the context of the meeting to fetch the appropriate augmented data. Step 4.1 above provides examples of sources from which context is fetched. Step 7 provides examples of various AR experiences that can be rendered.
4. Triggers to render AR data on a user's client may include:
- corresponding participant video/screen share when those participants are in focus (e.g., active speaker, video on focus, etc.)
- the user clicking on the participant's tile in the meeting application, hovering / clicking the mouse, touching a participant video or a name in the roster, etc.
- using context of the meeting to identify participants who have related knowledge and providing AR data on their video, tagging them, etc.
5. The AR data can render using various techniques, such as:
- mixing video with AR data as a superimposition and rendering. This may be performed at the client side by having the application fetch context from various sources.
- rendering the AR video on the meeting application, AR devices, etc.
- rendering AR data on a separate screen while the main video continues as-is (e.g., AR data can be shown on glass superimposed on the main video)
6. Developers may provide their own relationship data and/or AR/VR devices. Developers may also define the trigger points that cause AR data to be rendered.

In summary, techniques are provided herein for using relationship intelligence of the participants in the meeting to provide a customized AR experiences for each participant of that meeting whose meeting client is AR-ready. These techniques may allow developers to provide their own relationship information and integrate that information with the meeting platform by proposing a set of CPaaS SDK enhancements described herein.