3GPP TSG-SA WG4 Meeting #132S4-250978

Japan, Fukuoka, 19 – 23 May 2025

**Source: InterDigital Canada, Orange**

**Title: [FS\_ARSpatial] Pseudo-CR on Remote Processing of Spatial Computing Functions**

**Spec: 3GPP TR 26.819 v0.4.0**

**Agenda item: 9.8**

**Document for: Agreement**

**1. Introduction**

The Study on Spatial Computing for AR Services (FS\_ARSpatial) was approved during SA#104 meeting. The objectives of the study include identifying where spatial computing functions run and which media, metadata, and description formats are used for exchange between these elements based on the architecture defined in the TS 26.506, notably in split processing scenarios.

This document adds details with examples about when it can be relevant to execute spatial computing functions on the cloud or on the edge on remote spatial computing servers.

**2. Reason for Change**

One of the main objectives of the study is to “Study how spatial computing functions such as relocalization, mapping, and semantic perception are realized and identify the necessary set of spatial mapping information.”. This contribution addresses this objective by updating the section on the relocalization function.

Spatial computing functions can be executed locally on the device or can run remotely on the cloud or on the edge in a spatial computing server as detailed in TR 26.298. Additional examples about remote spatial computing functions are provided to motivate the need for remote processing.

**3. Proposal**

It is proposed to agree the following changes to 3GPP TR 26.819 v0.4.0.

\* \* \* First Change \* \* \* \*

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non‑specific.

- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TR 26.928: "Extended Reality (XR) in 5G".

[3] 3GPP TR 26.998: "Support of 5G glass-type Augmented Reality / Mixed Reality (AR/MR) devices".

[4] ARCore SLAM, https://developers.google.com/ar/develop/fundamentals

[5] ARKit VIO, https://developer.apple.com/documentation/arkit/arkit\_in\_ios/configuration\_objects/understanding\_world\_tracking

[6] ARCore Cloud Anchor, <https://codelabs.developers.google.com/codelabs/arcore-cloud-anchors#0>

[7] ARKit World Map, <https://developer.apple.com/documentation/arkit/arworldmap>

[8] Meta Quest Spatial Anchor, <https://developer.oculus.com/documentation/unity/unity-spatial-anchors-overview/>

[9] HoloLens, https://learn.microsoft.com/en-us/windows/mixed-reality/design/spatial-mapping

[10] Meta Space Setup, <https://www.uploadvr.com/quest-v64-undocumented-features-furniture-recognition-multimodal/>

[11] Apple RoomPlan, <https://developer.apple.com/augmented-reality/roomplan/>

[12] Google Scene Semantics, <https://developers.google.com/ar/develop/scene-semantics>

[13] Unity Documentation - Collision, <https://docs.unity3d.com/2023.1/Documentation/Manual/collision-section.html>

[14] Blender Documentation – Collisions, https://docs.blender.org/manual/en/latest/physics/rigid\_body/properties/collisions.html

[15] Unity Documentation - Compound Colliders, <https://docs.unity3d.com/Manual/compound-colliders-introduction.html>

[16] Unity Documentation - Mesh Colliders, <https://docs.unity.cn/Manual/mesh-colliders-introduction.html>.

[17] ARCore Ligthing Estimation, <https://developers.google.com/ar/develop/lighting-estimation>

[18] Cast Shadow <https://ieeexplore.ieee.org/document/9018202>

[19] Macario Barros, A., Michel, M., Moline, Y., Corre, G., & Carrel, F. (2022). A comprehensive survey of visual slam algorithms. Robotics, 11(1), 24.

[20] Khronos OpenXR: https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html

[21] ETSI Industry Specification Group AR Framework ([ISG ARF](https://www.etsi.org/committee/1420-arf)), <https://www.etsi.org/committee/1420-arf>

[22] ETSI GS ARF 004-2: “Augmented Reality Framework (ARF); Interoperability Requirements for AR components, systems and services - Part 2: World Storage and AR Authoring functions”.

[23] ETSI GS ARF 005: “Augmented Reality Framework (ARF); Open APIs for the Creation and Management of the World Representation”.

[24] ISO/IEC 23090-14, Text of ISO/IEC FDIS 23090-14 2nd edition Scene description, April 2024.

[25] 3GPP TS 26.119: “Media Capabilities for Augmented Reality”

[26] 3GPP TS 26.143: “Messaging Media Profiles”

[27] 3GPP TS 26.264: “IMS-based AR Real-Time Communication”

[28] 3GPP TR 26.812: “QoE metrics for AR/MR services”

[29] 3GPP TR 23.700-21: Study on Application architecture for enabling mobile metaverse applications.

[30] Open Geospatial Consortium: https://www.ogc.org/

[31] Open AR Cloud: <https://www.openarcloud.org/>

[32] ETSI GS ARF 004-4: “Augmented Reality Framework (ARF); Interoperability Requirements for AR components, systems and services - Part 4: World Analysis, World Storage and Scene Management functions”.

[33] OGC GeoPose 1.0 Data Exchange Draft Standard, https://docs.ogc.org/dis/21-056r10/21-056r10.html.

[34] OGC Points of Interest Conceptual Model Standard (draft document), <https://docs.ogc.org/DRAFTS/21-049.html>

[35] 3GPP TR 22.856: "Feasibility Study on Localized Mobile Metaverse Services".

[36] SIFT, <https://www.vlfeat.org/api/sift.html>

[37] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, Barcelona, Spain, 2011, pp. 2564-2571.

[38] A. Alahi, R. Ortiz and P. Vandergheynst, "FREAK: Fast Retina Keypoint," 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 2012, pp. 510-517.

[39] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha and P. Fua, "BRIEF: Computing a Local Binary Descriptor Very Fast," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 7, pp. 1281-1298, July 2012.

[40] OBJ, https://en.wikipedia.org/wiki/Wavefront\_.obj\_file

[41] PLY, <http://gamma.cs.unc.edu/POWERPLANT/papers/ply.pdf>

[42] FBX, <https://www.autodesk.com/products/fbx/overview>

[43] STL, <https://docs.fileformat.com/cad/stl/>

[44] glTF, https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.pdf

[45] USD, <https://openusd.org/dev/api/class_usd_object.html>

[46] Unity, <https://docs.unity3d.com/Manual/mesh-colliders-introduction.html>

[47] ISO/IEC 23090-14, Text of ISO/IEC FDIS 23090-14 2nd edition Scene description, April 2024.

[48] FAST, <http://www.edwardrosten.com/work/rosten_2006_machine.pdf>

[49] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool, Speeded-Up Robust Features (SURF), Computer Vision and Image Understanding, Vol. 110, Issue 3, 2008, pp. 346-359.

[50] 3GPP TS 23.558: “Architecture for enabling Edge Applications”

[51] 3GPP TS 26.501: “5G Media Streaming (5GMS); General description and architecture”

[52] 3GPP TS 23.548: “5G System Enhancements for Edge Computing; Stage 2”

[53] 3GPP TS 23.501: “System architecture for the 5G System (5GS)”

[54] W3C WebXR Device API: <https://immersive-web.github.io/webxr>

[55] 3GPP TS 23.503: “Policy and charging control framework for the 5G System (5GS); Stage 2”

[Y1] Niantic Visual Positioning System: https://www.nianticspatial.com/products/visual-positioning-system

[X1] Niantic Lightship VPS: [https://lightship.dev/docs/ardk/features/lightship\_vps/](https://lightship.dev/docs/ardk/features/lightship_vps/" \o "https://lightship.dev/docs/ardk/features/lightship_vps/" \t "_blank)

[X2] Augmented City: [Augmented City API](https://developer.augmented.city/doc/v2)

[X3] Google Geosptial API: <https://developers.google.com/ar/develop/geospatial>

[X4] bcom<>\*Overview\*: <https://b-com.com/en/overview>

[X5] Niantic Lighship Localizing with VPS: <https://www.lightship.games/docs/ardk/vps/vps_localization.html>

[X6] Magic Leap AR Cloud: http://magicleap.care/hc/en-us/articles/9312806819597-AR-Cloud

[x] <doctype> <#>[ ([up to and including]{yyyy[-mm]|V<a[.b[.c]]>}[onwards])]: "<Title>".

\* \* \* Next Change \* \* \* \*

### 4.2.3 Relocalization

Relocalization is a function that estimates the pose of the AR device according to a known real environment. It is used to estimate the pose of the AR device at initialization, when tracking is lost, or regularly to correct the drift of the tracking (TR 26.998 [3] clause 4.2.3). Cameras capture the real world, while sensors (accelerometers, gyroscopes, and depth sensors) contribute additional data for mapping and positioning. Computer vision algorithms process this data to determine the location and orientation of the device relative to its environment.

SLAM, Visual Localization, e.g., Visual SLAM (vSLAM), or Visual Positioning System (VPS) are all algorithms that can be used for mapping unknown environments while also maintaining the localization of the device/user within that environment, as explained in TR 26.928 [2] clause 4.1.4.

Visual Positioning System (VPS), an example of relocalization, relies on network services to provide the mapping information. Such functions may be executed in the network for computing and storage capability reasons. In such a case, the VPS relies on pre-existing mapping of the real environment. The previously mentioned sensor data can be sent to a server which then computes and provides the pose of the AR device in the reference coordinate system of a known real environment. In an alternative implementation, the AR device retrieves a local map from the server based on this GPS position and then performs the relocalization computation locally.

Combining a spatial description of the real world with the estimation of the pose of different trackables can also be used for localization as proposed with the World Storage and World Analysis components of the ETSI ARF architecture [21].

For relocalization, the following input data may be used:

- Sensor data:

- images (for SLAM)

- depth maps

- GNSS data

- local pose of AR Device

- IMU data

- Camera calibration information

- Derived data

- extracted feature vectors

The output of the relocalization function is a pose, which is defined by a position and an orientation.

Examples of VPS include Google GeoSpatial [X3] and Niantic Lightship [Y1] APIs. Standardized interfaces between AR application and remote relocalization services have been proposed by ETSI ARF as described in Section 5.2.2 and by the Open Spatial Computing Platform (OSCP) with the GeoPoseProtocol API. The GeoPoseProtocol API provides an exhaustive list of these exchanged data between an AR client and a remote relocalization server.

Regarding QoS, the two important aspects are:

* The network capacity to transfer high resolution images from the client to the server at regular intervals. For instance, transferring two 1280×720 images (depth and RGB) per second in a JPEG format could require between 8 and 16 Mbps of bandwidth. Considering that the AR device has reliable World Tracking system, relocalization can be done at larger intervals releasing the need for a high bandwidth. As an example, Niantic Lighship VPS requires the client to send around 300Kb of data to the server every 1-3 seconds [X5] therefore requiring a bandwidth between 0.82 and 2.46 Mbps.
* The end-to-end latency of the relocalization process includes the capture of the data from the device sensors, their transmission through the network, their processing on the server, and the transmission of the pose from the server to the user device. With XR devices, motion-to-photon latency is expected to be between 10 and 15 milliseconds to avoid any cybersickness effect. However, with a World Tracking system running locally on the AR device of the client, there is no strong latency constraint, as the pose received from the VPS can be corrected thanks to the difference between the local poses obtained when sending data and when receiving relocalization data. Keeping latency as low as possible (<100ms) still limits potential drifting issues.

#### 4.2.3.1 Example: Lightship (Niantic)

Lightship is Niantic's toolkit for creating immersive location-based experiences. Niantic's Visual Positioning System (VPS) ([X1]) is a cloud-based system that allows devices to know where they are relative to the Lightship map. By determining the six-degrees-of-freedom pose of a device relative to the map, you can place persistent virtual objects that stay aligned with the real world.

The Lightship VPS map is built using scans submitted by users and is organized by the coverage of the locations in the scans. When enough scans have been submitted for a location, the location becomes "VPS-Activated". The highest-quality space of a VPS-Activated location will be saved as the default space for that location with each default space having a default anchor associated with it.

The VPS Coverage API provides functionality for discovering AR Locations at runtime to display them on maps and use them as localization targets. The VPS Coverage API provides the default anchor as part of the LocalizationTarget.

To localize, the user needs to point their device at the real-world location associated with that anchor. Lightship then sends the camera frames to the cloud and looks for a match.

#### 4.2.3.2 Example: Augmented City

Augmented.City ([X2]) is an augmented reality cloud and platform ecosystem that allows users to capture, enrich with data, and visualize it on location in basically any device.

With the Augmented.City platform, users can easily create city-scale AR Clouds, populate them with information, and instantaneously localize inside them. The information could be visualized and shared with any mobile device.

To get started, the user needs to localize in a previously scanned location, point the camera at the nearest facade and start AR mode.

The user sends a request with sensor data and receives a pose. An example request and response are given in Table 4.2.3.2-1.

Table 4.2.3.2-1: Augmented.City relocalization request-response example.

|  |  |
| --- | --- |
| **Sample Request** | **Sample Response** |
| {  "id": "string",  "timestamp": 0,  "type": "string",  "sensors": [{}],  "sensorReadings": [{}],  "priorPoses": [{}],  "hint": {  "reconstructions": [],  "hint\_only": false  }  } | {  "id": "string",  "timestamp": 0,  "accuracy": {  "position": 0,  "orientation": 0  },  "type": "string",  "geopose": {  "position": {},  "quaternion": {}  },  "ecefPose": {  "position": {},  "orientation": {}  },  "localPose": {  "position": {},  "orientation": {}  },  "reconstruction\_id": 390  } |

#### 4.2.3.3 Example: ARCore Geospatial API (Google)

The **ARCore Geospatial API** ([X3]) enables the user to remotely attach content to any area covered by Google [Street View](https://www.google.com/streetview/" \o "https://www.google.com/streetview/" \t "_blank) and create AR experiences on a global scale. It uses device sensor and GPS data to detect the device's environment, then matches the recognizable parts of that environment to a localization model provided by Google’s Visual Positioning System (VPS) to determine the precise location of a user’s device.

Street View images from Google Maps, which have been captured around the globe for more than 15 years, are the foundation of VPS. Deep neural networks identify and describe parts of the images that are likely to be recognizable over long periods of time. Those parts are then combined across tens of billions of images to compute a 3D point cloud of the global environment. This localization model consists of trillions of points and spans nearly all countries, with future coverage.

When the user’s device makes a request to the Geospatial API, a neural network processes the pixels to find recognizable parts of the user’s environment and matches them to the VPS localization model. Computer vision algorithms then compute the position and orientation of the device, offering a location that is much more accurate than what was previously possible with GPS alone.

#### 4.2.3.4 Example: b<>com \*Overview\*

#### b<>com \*Overview\* ([X4]) enables secure and accurate augmented reality experiences running on any device. It consists of a set of software services able to map the real environment and locate in 3D AR devices (smartphones, tablets, and AR glasses) at a building, construction site, or factory scale.

Spatial computing services supported by b<>com \*Overview\* run on GPU servers. For relocalization, b<>com \*Overview\* uses a single visual positioning system. Using a Unity plugin, an AR Client can connect an application to the b<>com \***Overview**\* services.

\* \* \* Next Change \* \* \* \*

### 4.2.5 3D model reconstruction

Spatial computing enables the creation of accurate 3D models of surrounding space. It accurately captures real-world scenes and objects using 3D scanning techniques or photogrammetry. These 3D models can be displayed in immersive 3D environments in real-time to provide users with a sense of interactivity and presence. When built in real-time, they can, for example, be used to manage occlusion and physics behaviors between real and virtual objects. Such 3D models can also be used offline for authoring AR experiences, and for completing or correcting a digital twin of a real environment.

The 3D model of a real-world environment may also be constructed collectively by aggregating meshes captured by an AR device.

HoloLens (Microsoft) and AR SDKs from Apple and Meta can all build a 3D model of the surrounding environment in real-time.

Achieving highly realistic and accurate 3D model reconstruction can require significant computing capabilities, especially when dealing with multiple users capturing the same space, and when using advanced structure-from-motion algorithms or radiance-field-based representations, such as Gaussian splatting. This can be an issue for some devices. These algorithms can then be executed offline on remote servers using sensor data provided by one or multiple AR devices, with the reconstructed models sent back upon completion.

To build the 3D model, the following input data can be used:

* Sensor data:
* Images captured by AR Device
* pose of AR Device
* Depth map (image or texture)
* Mesh captured by AR Device

The output of the 3D Model construction is a 3D Model. A 3D model is a broader term that encompasses the complete representation of a 3D object, including its geometry (which may be represented by a mesh), texture, and materials.

When using a remote computing server, streaming all these data from several AR client can require a high bandwidth. As an example, considering an RGB image and depth map with a resolution of 1280×720 compressed in JPEG format, and an uncompressed mesh composed of 200000 vertices (encoded as three 32-bit floating point coordinates per vertex), this would require approximately a bandwidth of 1 Gbps. This corresponds to the requirements of the Magic Leap AR Cloud platform [X6] (mentioning between 480 Mbps and 1 Gbps).

#### 4.2.5.1 Example: Magic Leap AR Cloud

AR Cloud [X6] is a connected service that enables Magic Leap 2 devices to access extensive spatial data. It allows digital twin administrators to create Shared Spaces and manage device-side scans through a web-based console. A key feature of AR Cloud is its ability to merge scans from different users into a unified 3D model reconstruction accessible from each client. The service can run on the cloud on providers such as AWS, GCP, and Azure.

\* \* \* End of Changes \* \* \* \*