**3GPP TSG SA WG4#116e S4-211xxx**

**E-meeting, 10th – 19th November 2021**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *CR-Form-v12.1* | | | | | | | | |
| **PSEUDO CHANGE REQUEST** | | | | | | | | |
|  | | | | | | | | |
|  | 26.998 | **CR** |  | **rev** |  | **Current version:** | 1.0.3 |  |
|  | | | | | | | | |
| *For* [***HE******LP***](http://www.3gpp.org/3G_Specs/CRs.htm#_blank)*on using this form: comprehensive instructions can be found at* [*http://www.3gpp.org/Change-Requests*](http://www.3gpp.org/Change-Requests)*.* | | | | | | | | |
|  | | | | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Proposed change affects:*** | UICC apps |  | ME |  | Radio Access Network |  | Core Network |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | |
| ***Title:*** | [FS\_5GSTAR] On Spatial Computing | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Source to WG:*** |  | | | | | | | | | |
| ***Source to TSG:*** |  | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Work item code:*** | FS\_5GSTAR | | | | |  | ***Date:*** | | | 2021-11-02 |
|  |  | | | |  | |  | | |  |
| ***Category:*** | C |  | | | | | ***Release:*** | | | Rel-17 |
|  | *Use one of the following categories:* ***F*** *(correction)* ***A*** *(mirror corresponding to a change in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)*  Detailed explanations of the above categories can be found in 3GPP [TR 21.900](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | | | | | | | | *Use one of the following releases: Rel-8 (Release 8) Rel-9 (Release 9) Rel-10 (Release 10) Rel-11 (Release 11) … Rel-15 (Release 15) Rel-16 (Release 16) Rel-17 (Release 17) Rel-18 (Release 18)* | |
|  |  | | | | | | | | | |
| ***Reason for change:*** | |  | | | | | | | | |
|  | |  | | | | | | | | |
| ***Summary of change:*** | |  | | | | | | | | |
|  | |  | | | | | | | | |
| ***Consequences if not approved:*** | |  | | | | | | | | |
|  | |  | | | | | | | | |
| ***Clauses affected:*** | |  | | | | | | | | |
|  | |  | | | | | | | | |
|  | | **Y** | **N** |  | | | |  | | |
| ***Other specs*** | |  |  | Other core specifications | | | | TS/TR ... CR ... | | |
| ***affected:*** | |  |  | Test specifications | | | | TS/TR ... CR ... | | |
| ***(show related CRs)*** | |  |  | O&M Specifications | | | | TS/TR ... CR ... | | |
|  | |  | | | | | | | | |
| ***Other comments:*** | |  | | | | | | | | |
|  | |  | | | | | | | | |
| ***This CR's revision history:*** | |  | | | | | | | | |

**===== CHANGE =====**

3.1 Definitions

**Spatial Computing:** AR functions which process sensor data to generate information about the world 3D space surrounding the AR user. It includes functions such as SLAM for spatial mapping (creating a map of the surrounding area) and localization (establishing the position of users and objects within that space), 3D reconstruction and semantic perception. Spatial computing functions require XR spatial description and may result in updates to the XR spatial description.

**XR spatial description**: a data structure describing the spatial organisation of the real world using anchors, trackables, camera parameters and visual features. It is a meta specification of the real environment including the information for the estimation of the position and orientation (pose estimation) of AR devices for the purpose of registration, tracking and positioning, and provides a coordinate reference system in relation to the real world.

**XR spatial compute server:** is an edge or cloud server that provides spatial computing AR functions.

**XR spatial description server:** is a cloud server for storing, updating and retrieving XR spatial description.

**===== CHANGE =====**

### 4.2.3 AR Runtime

The AR Runtime is a device-resident software or firmware that implements a set of APIs to provide access to the underlying AR/MR hardware. An AR Runtime typically provides the following functions:

- System capability discovery: allows applications to discover capabilities of the AR glasses

- Session management: manages an AR session and its state

- Registration and Tracking: creates XR spaces, registers to local environment, tracks reference spaces and objects throughout the lifetime of the AR session. This information is provided to the application on request. This is an outcome of spatial computing AR functions.

- Input and Haptics: receives information about user’s actions, e.g. through usage of trackpads, and passes that information to the application. On request by the application, it may trigger haptics feedback using the AR glasses and associated hardware.

- Rendering: synchronizes the display and renders the composited frame onto the AR glasses displays.

AR runtimes are usually extensible to add support for a wide range of AR glasses and controllers that are on the market or that might be released in the future. This will allow different vendors to add custom functionality such as gaze tracking, hand control, new reference spaces, etc.

Two key representative and standardized AR runtimes are Khronos defined OpenXR [4] and W3C defined WebXR [5]. More details are provided in clause 4.6.4.

**===== CHANGE =====**

4.3 Basic Processes in an AR Session

4.3.1. Introduction

In this clause we provide a generic workflow for setting up an AR session. This consists of an AR media session for accessing the media over the network for rendering in an AR scene. In addition, there is a parallel but asynchronous spatial computing session that provides an understanding of the physical space surrounding the device to determine the device’s position and orientation and placement of AR objects in reference to the real world.

4.3.1. AR Media Session

In this clause, we provide basic processes and generic workflow description for setting up AR Media sessions for media is accessed over the network. This generic basic process may be extended to address specific applications and use cases. The call flow as shown in Figure 4.3-1 aligns with the STAR/EDGAR architecture and serves as a baseline for defining use-case specific call flows.

**Figure 4.3-1: Basic workflow for AR media sessions**

A description of the steps of the general workflow is provided as follows:

1. The application contacts the application provider to fetch the entry point for the content. The acquisition of the entry point may be performed in different ways and is considered out of scope. An entry point may for example be a URL to a scene description.

2. The application initializes the Scene Manager using the acquired entry point.

3. The Scene Manager retrieves the scene description from the scene provider based on the entry point information. It then establishes a scene session with the scene provider.

4. The Scene Manager parses the entry point and creates the immersive scene.

5. The Scene Manager requests the creation of a new AR/MR session from the AR Runtime.

6. The AR Runtime creates a new AR/MR session and performs registration with the local environment.

7. The Scene Manager will inform the MAF about its QoS and compute needs

8. The MAF will request the Media Delivery Functions, such as AF, in the network to allocate the requested resources.

9. For each component or group of components of an object/node in the scene:

a. the Scene Manager triggers the MAF to fetch the related media

b. the MAF creates a dedicated media pipeline to process the input.

c. the MAF establishes a transport session for each component of the media object.

10. The application starts the media fetching and rendering loop

a. the MAF may receive updates to the scene description from the scene provider.

b. the MAF passes the scene update to the Scene Manager.

c. the Scene Manager updates the current scene.

d. The Scene Manager acquires the latest pose information and the user’s actions

e. The Scene Manager shares that information with the AR/MR application on the server

f. For each object:

i. The media pipeline fetches the media data. It could be static, segmented, or real-time media streams

ii. The media pipeline processes the media and makes it available in buffers

g. For each object to be rendered:

i. The Scene Manager gets processed media data from the media pipeline buffers

ii. The Scene Manager reconstructs and renders the object

h. The Scene Manager passes the rendered frame to the AR/MR Runtime for display on the user’s HMD.

4.3.2. Spatial Computing Session

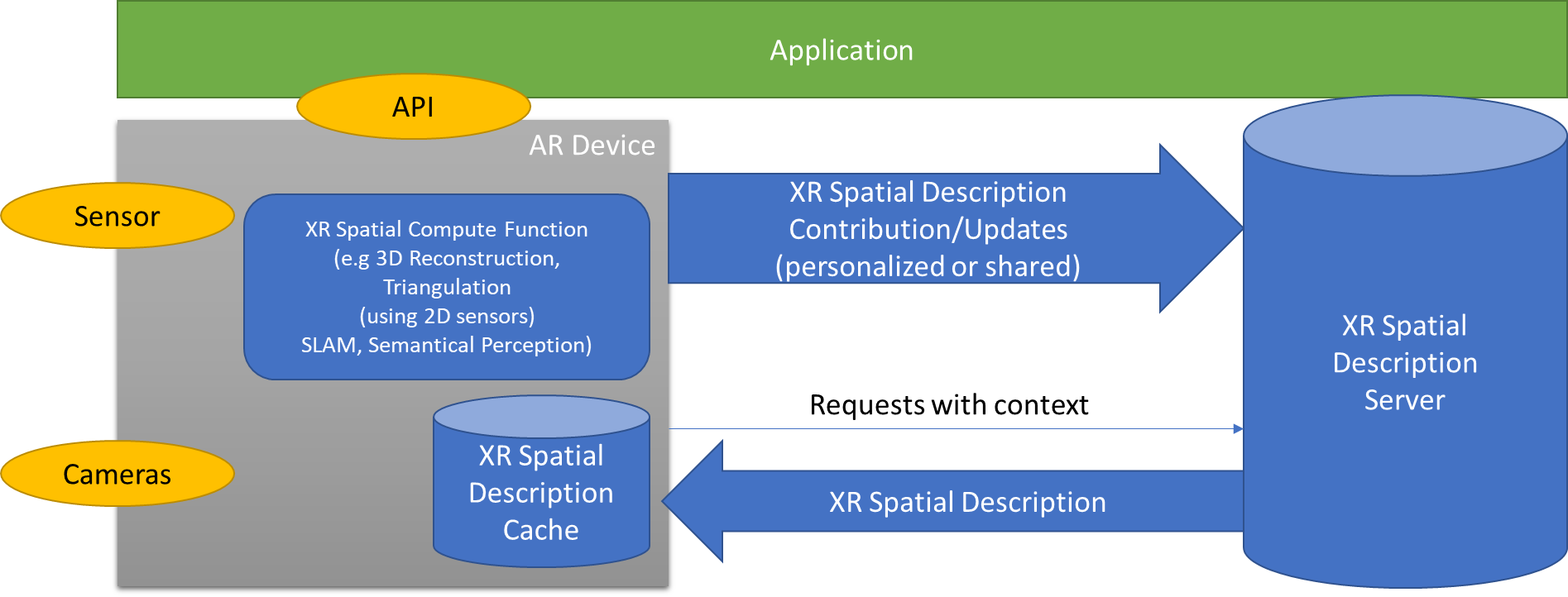
In this clause, we provide basic processes and generic workflow description for setting up spatial computing functions using the AR Runtime and possibly the network in order to provide the AR device with a continuous mapping of the scene to the real-world spaces. This requires registering with the surrounding spaces requiring spatial coordinate systems for precisely positioning and orienting 3D media objects at meaningful places in the world. Beyond the registration within a world coordinate system, additionally spatial mapping of objects is essential in order to place 3D objects on real surfaces, but also provides the ability to occlude objects behind surfaces, doing physics-based interactions based on surface properties, providing navigation functions or providing a visualization of the surface. Thirdly, for the purpose of understanding and perceiving the scene semantically, machine-learning and/or artificial intelligence may be used to provide context of the observed scene. The output of spatial computing is spatial mapping information that is organized in a data structure called the XR spatial description for storing and exchanging the information. Further details XR spatial description can be found in section 4.4.7.

The above processes may be carried out entirely on the AR device. However, it may be desirable to use an XR spatial compute edge server for offloading some or all spatial computing functions and an XR spatial description server for storage of XR spatial description and sharing it with other AR devices. Next we describe the two possible scenarios

1. where spatial computing is done on the device but an XR spatial description server is used for storage and retrieval of XR spatial description.
2. where spatial computing is done on the XR Spatial computing edge server and an XR spatial description server is used for storage and retrieval of XR spatial description.

A STAR-type device may use either of these, but an EDGAR-type device is likely to use the latter. Figure 4.3.2-1, shows functional diagram for the scenario where the device does spatial computing and an XR spatial description server is used for storage.

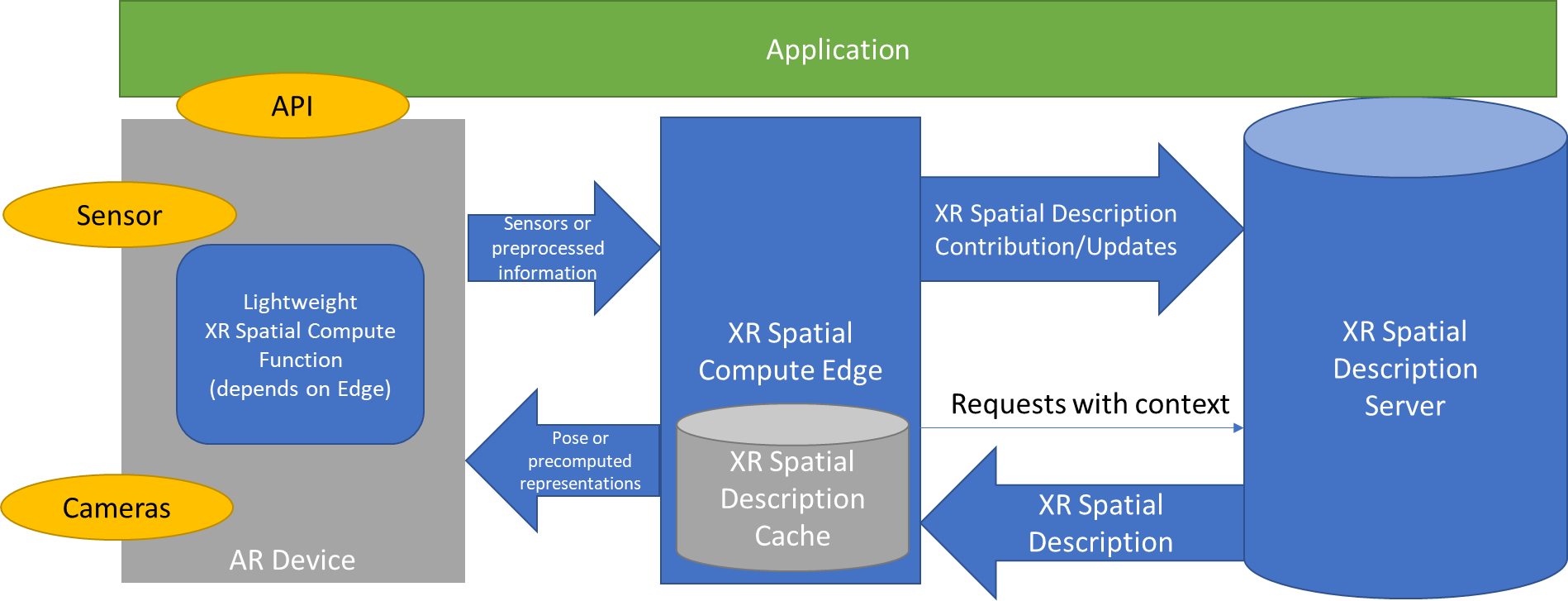
* The XR spatial description server provides a subset of the XR spatial description to the AR device. The context is derived from the anchors in scene description.
* The device performs spatial computing and maintains a local version of the XR spatial description.
* XR spatial description updates and contributions can be sent from the device to the XR spatial description server.



**Figure 4.3.2-1 Functional diagram for spatial computing**

Figure 4.3.2-1, shows the functional diagram of the case where the spatial computing AR function is handled by an XR spatial computing edge server.

* The device sends sensor data (e.g. images or visual features extracted from images for the sake of privacy) to the XR spatial compute server.
* The XR spatial compute server fetches the XR spatial description from the XR spatial description server and performs spatial computing based on device sensor data and provides pose (STAR-type) or precomputed representations (EDGAR-type) to the AR device.



**Figure 4.3.2-2 Functional diagram for spatial computing with XR spatial compute edge**

A generic call flow for with XR spatial compute edge and remote rendering is shown in Figure 4.3.2-3

Graphical user interface

Description automatically generated

**Figure 4.3.2-3 Functional diagram for spatial computing with XR spatial compute edge**

Step 1: UE starts an AR application. Steps 1-9 of Figure 4.3.1. From the scene graph, the entry point is processed.

Step 3: From the anchor references defined in the scene graph, the AR/MR Scene Manager will request for XR spatial computing from a cloud/edge based Spatial Computing Server.

Step 4,5: The Spatial Computing Server requests the XR spatial description from the World Map Server and receives it. This XR spatial description includes the anchors referenced in the scene graph as well as all the children nodes with leaves corresponding to trackables (with their features).

Step 6: The pipelines are configured for sensor data.

Step 7: The AR/MR scene is created.

Map update and rendering: The following steps run continuously during the session.

Step 8: The AR Runtime on the device will transmit images captured by the embedded vision sensors to the Spatial Computing Server, ideally at the capture frequency (~60Hz). Alternatively it may send visual features instead of full images.

Step 9: The Spatial Computing Server will receive these images and based on the XR spatial description stored in cache, it will be able to estimate the pose of the AR device using the features attached to trackables available in the world subgraph. 3D transforms between trackables and anchors will allow to provide the pose of the AR device in a common reference coordinate system of the real world.

Step 10. The Spatial Computing Server provides the spatial position (AR device pose in reference to the real world) to the remote rendering service running also into the cloud which will render the AR scene including AR assets from a viewpoint corresponding to the real pose of the AR device. If remote rendering is not used pose is sent to AR Runtime.

Step 11: The remote rendering service will transmit the rendered images according to the pose to the UE for displaying them to the end-user.

Step 12: The spatial compute server may send/receive updates in XR spatial description from the XR spatial description server.

**===== CHANGE =====**

4.4.7 XR Spatial Description

4.4.7.1 Overview

XR spatial description is a data structure (typically a scene graph) describing the spatial organisation of the real world using:

* Visual features and 3D Maps
* Spatial anchors and trackables
* camera parameters as defined in 4.4.3.2

It is a meta specification of the real environment including the information for the estimation of the position and orientation (pose estimation) of AR devices for the purpose of registration, tracking and positioning, and provides a coordinate reference system in relation to the real world. It is used for spatial computing as described in section 4.3.2.

XR spatial description data is downloaded and updated periodically from a XR Spatial Description Server, which may be an AR/MR Application provider. It can be updated by sending an XR Spatial description update (e.g., updated visual features or keyframes attached to camera parameters) from an AR device or an XR spatial compute server to the XR Spatial Description Server.

The size of the XR spatial description depends on several parameters, such as, size of the area covered by the XR application, number of supported viewpoints in the area, etc. The size may be from 10MB for a small room to several hundred MBs for a building. The frequency of updates may depend on the application.

Formats:ETSI ARF uses the term World Graph for XR spatial description. It defines the relative position of AR Assets, Trackables and World Anchors by 3D Transforms. A world graph is a kind of scene graph including trackables (embedding their features), and anchors representing the real world. Thus, the AR device can detect, recognize, and estimate its pose in relation to trackables thanks to their features, and based on the 3D transforms defined in the World graph, it can estimate its pose in relation to anchors (to which AR assets will be attached). [21]. Requirements for XR spatial description are also under consideration in MPEG for using a scene graph for real-world mapping possibly as a glTF extension. Currently only proprietary implementations exist for XR spatial description.

4.4.7.2 Camera and sensor information

2D :

LiDAR.

Depth

3D ToF

3D

Sound

4.4.7.3 Visual features and Spatial Maps

Describe details, surfaces, meshes? etc.

Visual features are characteristics of a real world element that can be searched, recognized or tracked in frames captured from an AR device visual sensor as it moves in a real environment, using Simultaneous Localization And Mapping approach (SLAM). They are the overlapping points that are recognizable in multiple images of the real environment. Visual features are extracted from frames from a single moving camera or multiple cameras in SLAM systems. A 3D Map, sparse or dense point cloud, of the real world can be generated from keyframes and their matched visual features. The keyframes must be attached to camera information defined in 4.4.3.2 to triangulate 3D points correctly from multiple cameras. This mapping process can be performed either at runtime or offline. A 3D map is then used at runtime to relocalize and thus register the AR device by matching the features extracted from the current image with the ones stored in the previously built 3D map. The mapping approach is one of well-known keyframe-based SLAM techniques [X].

Add to references [X] <https://arxiv.org/abs/1607.00470>

Therefore, a spatial map consists of spatial visual features (e.g. 3D points associated with their descriptor such as SIFT [XX], SURF [XXX], ORB [XXXX]) and additional information to match them with features extracted at runtime from the sensor data (2D or 3D depending on sensor capabilities). Note that the 2D-3D (e.g for RGB or B&W cameras) or 3D-3D (e.g. for depth sensors) feature matching is widely used to estimate the pose of the sensor (using a Perspective-n-Points algorithm), and thus of the AR device. Note that 3D-3D feature matching can also be implemented for depth sensors.

For this reason a 3D Map consists at least of:

* A spatial feature cloud, e.g. 3D points (Vector of 3 float) with their associated descriptors such as SIFT [XX], SURF, ORB. These descriptors are generally vectors of numbers (e.g vector of 128 floats for SIFT, vector of 64 floats for SURF, vector of 32 integers for ORB). Note that other features such as 3D segments can be also used.

But additionally, to speed-up the 2D-3D matching process, a 3D map generally includes:

* Information required for keyframe retrieval. For example, a keyframe retrieval can use Bag-Of-visual-Words (BoW) model. In this case, the information consists of the vocabulary of the BoW model and corresponding descriptor for each keyframe (vector of occurrence counts of a vocabulary in the keyframe). Depending on the visual descriptor used, the vocabulary size is usualy a few dozen Mb, and this vocabulary can be reused for any 3D map using the same vocabulary.
* The 2D features for each keyframes (e.g. 2D points with their associated descriptors such as SURF, SIFT, ORB represented by a vector of numbers). The number of features exracted per keyframe varies between 200 and 1000.
* The matches between 2D features of keyframes and 3D features of the spatial feature cloud.

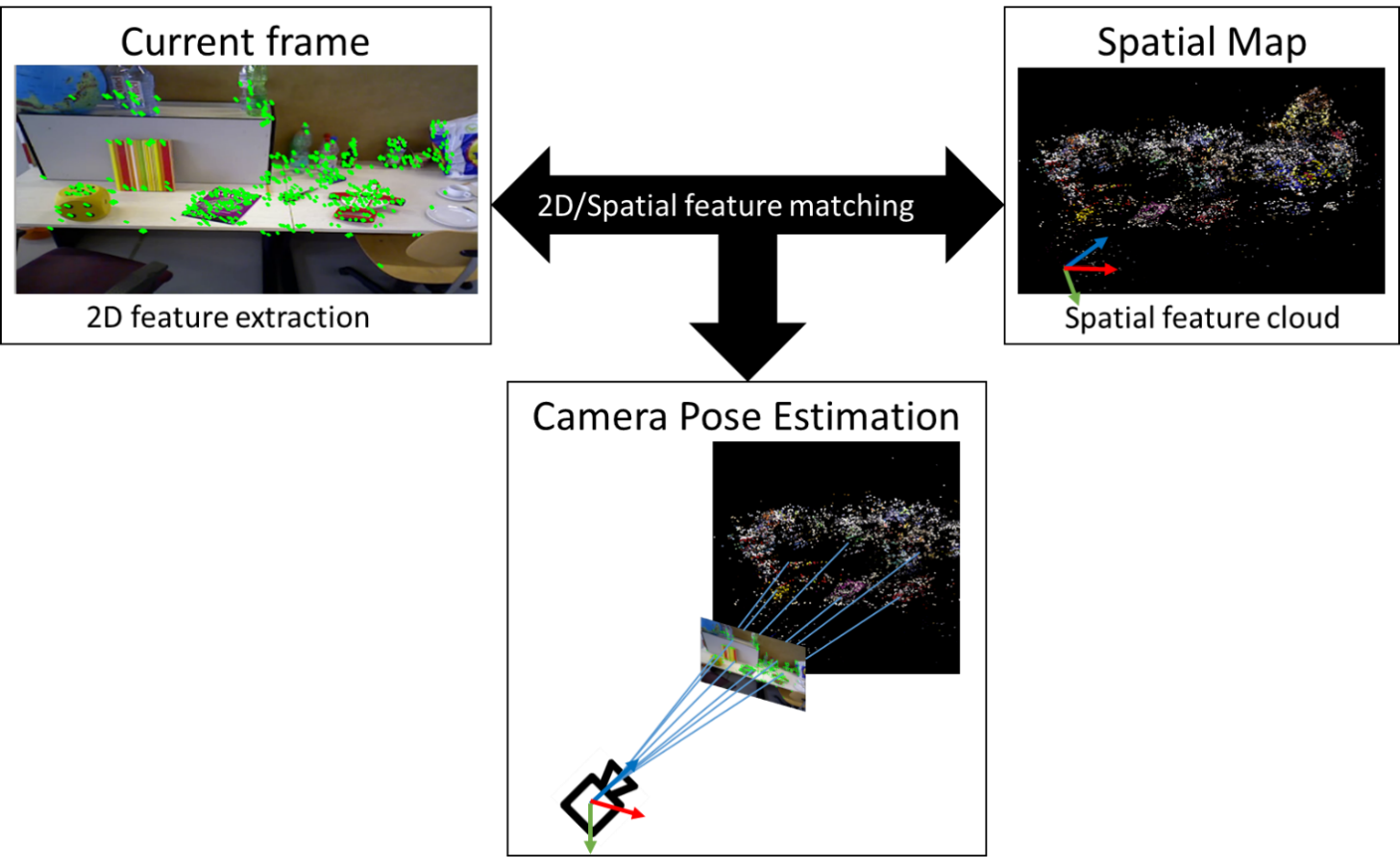
Thanks to this additional information, instead of comparing all descriptors of 2D features extracted from the current frame with all spatial feature descriptors, resulting in a very high complexity, the vision based localization system can:

* Match the closest keyframe to the current frame by retrieving it with the BoW model,
* Match the 2D features between the current frame and the retrieved keyframe,Match the 2D features between the current frame and spatial feature cloud (knowing matches between 2D features of the keyframes and 3D features of the spatial feature cloud).

Add to reference [XX] <https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf>

Add to reference [XXX] <https://people.ee.ethz.ch/~surf/eccv06.pdf>

Add to reference [XXXX] https://ieeexplore.ieee.org/document/6126544



4.4.7.4 Spatial Anchors and Trackables

AR objects can be positioned in reference to the real world (e.g., placing a vase on a table) using spatial anchors. A spatial anchor provides a fixed position and orientation in the real world based on a common frame of reference that can be used by multiple AR devices. Spatial anchors should refer to trackables for accurate positioning relative to the physical space. Spatial anchors can also be used alone (not referring to trackable) if global coordinates are used. In this case, the anchors are treated as global anchors without trackable as they have global coordinates which positions can be determined.

Trackables areelements of the real world of which features (visual or non-visual) are available and/or could be extracted. A 3D Map trackable, for instance, may define a full environment composed of a floor walls, furnitures in the real world consisting of several 3D points with visual features. However, there are other types of trackables as well. For example:

* A controller with LEDs that can be tracked by an AR headset’s vision sensor. The feature in this case is the constellation of LEDs.
* A fiducial marker that is detected as a black and white pattern by an AR device vision sensor. The feature in this case is the black and white pattern.
* Hands visible through an AR headset’s vision sensor. The feature is a learnt model for hands.

All of the above examples give a position of the trackable in reference to the position of the sensor (generally embedded in the AR headset). A spatial description data structure describing the spatial organisation of the real world using anchors, trackables, camera parameters and visual features can be used for exchanging spatial data and updates between AR Runtime and spatial compute server.

..