**3GPP TSG-SA WG4 Meeting #127** **S4-240219\_r01**

**Sophia-Antipolis, France, 29 January - 2 February 2024** *revision of S4aV230129*

**Source: Interdigital Finland Oy**

**Title:** **[FS\_ARMRQoE] Editor’s note on accuracy level computation**

**Spec: 3GPP TR 26.812 v1.0.0**

**Agenda item: 9.7**

**Document for: Discussion and Agreement**

**1. Introduction**

The pose error and time error metrics for the predicted pose for rendering are defined in the technical report. In the measurement procedure in clause 6.3.5.3, the XR application may use the status information returned with the pose estimation from the XR runtime to compute an accuracy level of that pose estimation.

A Note following the measurement procedure details how to get the status information related to the pose estimation with the Kronos OpenXR API.

An editor’s note remains in clause 6.3.5.3 about the computation of the accuracy level using the XrViewStateFlags is FFS.

The contribution S4aV230129 was presented at the 3GPP SA4 Video SWG Telco on January 16, 2024.

Discussion:

* Liangping: How do you derive the accuracy level? From OpenXR?
  + Stephane: very simple formula. We have 8 different cases, no specific mathematics. Maybe we could have something better
  + Gilles: concur with Liangping’s conclusion. Maybe we can have some level providing a percentage?
  + Stephane: could also be a possibility, kind of a different proposal
* Imed: We should add an exact measurement, and not an accuracy definition. These bits just say whether you can use them or not. Should not codify as a value, just convey them.
  + In 3DoF, orientation more important than position !
  + Gilles: seems to be better addressed in offline. Accuracy level not good.

**2. Reason for Changes**

The contribution proposes to address the editor’s note in clause 6.3.5.3 “the computation of the accuracy level using the XrViewStateFlags is FFS.”

We propose an example of use of the status information related to the pose estimation to deduce an accuracy level (renamed confidence status) of the calculated Viewer Pose Predicted Error with Kronos OpenXR API.

The Confidence status may be reported in the Viewer Pose Predicted Error metric.

**2.1 Kronos OpenXR specification on xrLocateViews**

The XR application queries for a predicted pose on steps 5 and 15 in the measurement procedure. Those can be achieved by calling the function xrLocateViews with the Kronos OpenXR API.

In the Kronos OpenXR documentation (<https://registry.khronos.org/OpenXR/specs/1.0/man/html/xrLocateViews.html>), the xrLocateViews function is defined as:

|  |
| --- |
| // Provided by XR\_VERSION\_1\_0  XrResult xrLocateViews(  XrSession session,  const XrViewLocateInfo\* viewLocateInfo,  **XrViewState\* viewState,**  uint32\_t viewCapacityInput,  uint32\_t\* viewCountOutput,  XrView\* views); |

Parameters Descriptions :

|  |
| --- |
| * *session* is a handle to the provided [*XrSession*](https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrSession.html). * *viewLocateInfo* is a pointer to a valid [*XrViewLocateInfo*](https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrViewLocateInfo.html) structure. * ***viewState* is the output structure with the viewer state information.** * *viewCapacityInput* is an input parameter which specifies the capacity of the views array. The required capacity must be same as defined by the corresponding [*XrViewConfigurationType*](https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrViewConfigurationType.html). * *viewCountOutput* is an output parameter which identifies the valid count of views. * *views* is an array of [*XrView*](https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrView.html). * See [*Buffer Size Parameters*](https://www.khronos.org/registry/OpenXR/specs/1.0/html/xrspec.html#buffer-size-parameters) chapter for a detailed description of retrieving the required *views size*. |

The XrViewState structure contains additional state data.

|  |
| --- |
| typedef struct XrViewState {  XrStructureType type;  void\* next;  **XrViewStateFlags viewStateFlags;**  } XrViewState; |

The structure *XrViewState* contains a field *XRViewStateFlags*

In *XrViewStateFlags* field, it contains a bitmask of [*XrViewStateFlagBits*](https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrViewStateFlagBits.html) indicating state for all views.

|  |
| --- |
| // Flag bits for XrViewStateFlags  static const XrViewStateFlags XR\_VIEW\_STATE\_ORIENTATION\_VALID\_BIT = 0x00000001;  static const XrViewStateFlags XR\_VIEW\_STATE\_POSITION\_VALID\_BIT = 0x00000002;  static const XrViewStateFlags XR\_VIEW\_STATE\_ORIENTATION\_TRACKED\_BIT = 0x00000004;  static const XrViewStateFlags XR\_VIEW\_STATE\_POSITION\_TRACKED\_BIT = 0x00000008; |

The definition of state values returns for views are the following:

|  |
| --- |
| Flag Descriptions   * XR\_VIEW\_STATE\_ORIENTATION\_VALID\_BIT indicates whether all [XrView](https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrView.html) orientations contain valid data. Applications **must** not read any of the [XrView](https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrView.html) pose orientation fields if this flag is unset. XR\_VIEW\_STATE\_ORIENTATION\_TRACKED\_BIT **should** generally remain set when this bit is set for views on a tracked headset or handheld device. * XR\_VIEW\_STATE\_POSITION\_VALID\_BIT indicates whether all [XrView](https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrView.html) positions contain valid data. Applications **must** not read any of the [XrView](https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrView.html)::pose position fields if this flag is unset. When a view loses tracking, runtimes **should** continue to provide valid but untracked view position values that are inferred or last-known, so long as it’s still meaningful for the application to render content using that position, clearing XR\_VIEW\_STATE\_POSITION\_TRACKED\_BIT until tracking is recovered. * XR\_VIEW\_STATE\_ORIENTATION\_TRACKED\_BIT indicates whether all [XrView](https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrView.html) orientations represent an actively tracked orientation. This bit **should** generally remain set when XR\_VIEW\_STATE\_ORIENTATION\_VALID\_BIT is set for views on a tracked headset or handheld device. * XR\_VIEW\_STATE\_POSITION\_TRACKED\_BIT indicates whether all [XrView](https://registry.khronos.org/OpenXR/specs/1.0/man/html/XrView.html) positions represent an actively tracked position. When a view loses tracking, runtimes **should** continue to provide valid but untracked view position values that are inferred or last-known, e.g. based on neck model updates, inertial dead reckoning, or a last-known position, so long as it’s still meaningful for the application to render content using that position. |

**2.2 Validity and tracking status of the predicted pose**

As described above, the following XrViewStateFlags flags in the XrViewState provide information on the validity and the tracking of both components (position and orientation) of the predicted/estimated pose information.

- XR\_VIEW\_STATE\_ORIENTATION\_VALID\_BIT

- XR\_VIEW\_STATE\_POSITION\_VALID\_BIT

- XR\_VIEW\_STATE\_POSITION\_TRACKED\_BIT

- XR\_VIEW\_STATE\_ORIENTATION\_TRACKED\_BIT

The status flags information must be checked in steps 7bis and 18 of the measurement procedure before using the predicted pose. The status flags information may be transmitted with the pose information in step 8.

**2.3 Confidence status of the Pose Error**

In step 18 of the measurement procedure, the XR Application computes a pose error using the predicted poses of the step 7 and 17 (P.predicted1 and P.predicted2).

In Table 6.3.5.2-1 - Viewer Pose Prediction Error of TR 26.812, the Viewer Pose Prediction Error reports separately the errors on position and orientation components. Therefore, the validity and tracking status can be independently checked on the two components (position and orientation) to get a confidence on the error per component.

**Table 6.3.5.2-1: Viewer Pose Prediction Error**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Key** | | | | | | **Type** | **Description** |
| ViewerPosePredictionErrorSet | | | | | | Set | Set of viewer pose prediction errors. |
|  | *Entry* | | | | | Object |  |
|  |  | Time | | | | Integer | The time when the predicted viewer pose is used for. |
|  |  |  | view | | | Integer | The view index (0 for left eye and 1 for right eye) |
|  |  |  |  | Pose prediction error | | Set | The deviation between the actual and predicted pose. |
|  |  |  |  |  | Position prediction error | Vector | Vector distance between the actual and predicted position |
|  |  |  |  |  | Orientation prediction  error | Vector | Quaternion distance between the actual and predicted position |
|  |  |  |  | FoV prediction error | | Set | The deviation between the actual and predicted FoV. |
|  |  |  |  |  | Left error | float | Difference between the actual and predicted left angle of FoV |
|  |  |  |  |  | Right error | float | Difference between the actual and predicted right angle of FoV |
|  |  |  |  |  | Up error | float | Difference between the actual and predicted Up angle of FoV |
|  |  |  |  |  | Down error | float | Difference between the actual and predicted Down angle of FOV |

The confidence of the position and orientation error depends on the validity and tracking status of the two predicted poses (P.predicted1, P.predicted2). For example, if the orientation of the first pose P.predicted1 is tracked and the orientation of the second pose P.predicted2 is untracked, then the resulting orientation error (P.predicted1, P.predicte2) is uncertain or with a low level of confidence.

Combining the validity and tracking status of the two predicted poses give the confidence of the position and orientation error listed in table 2 and table 3. Combining the validity and tracking status is performed on each component (position and orientation) of the poses to get a confidence status on the pose error per component.

Table 2 – Confidence status in the Position component of the Pose prediction Error.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| First Pose (P.predicted1) | | | Second Pose (P.predicted2) | | Confidence status in the Position component of the Pose prediction Error |
| Position status | | | Position status | |
| VALID\_BIT | TRACKED\_BIT | VALID\_BIT | | TRACKED\_BIT |  |
| 0 | X | X | | X | The Pose prediction error for the Position component cannot be estimated. |
| X | X | 0 | | X | The Pose prediction Error for the Position component cannot be estimated. |
| 1 | 0 | 1 | | 0 | The Pose prediction Error for the Position component is uncertain. Low level of confidence. Tracking is lost for both predicted poses used to calculate the pose prediction error. |
| 1 | 0 | 1 | | 1 | The Pose prediction Error for the Position component is uncertain. Low level of confidence. Tracking is lost for the first predicted pose used to calculate the pose prediction error. |
| 1 | 1 | 1 | | 0 | The Pose prediction Error for the Position component is uncertain. Low level of confidence.  Tracking is lost for the second predicted pose used to calculate the pose prediction error. |
| 1 | 1 | 1 | | 1 | The Pose prediction Error for the Position component is OK. High level of confidence. |

“X” means “0 or 1”

Table 3 – Confidence status in the Orientation component of the Pose prediction Error.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| First Pose (P.predicted1) | | | Second Pose (P.predicted2) | | Confidence status in the Orientation component of the Pose prediction Error |
| Orientation status | | | Orientation status | |
| VALID\_BIT | TRACKED\_BIT | VALID\_BIT | | TRACKED\_BIT |  |
| 0 | X | X | | X | The Pose prediction error for the Orientation component cannot be estimated. |
| X | X | 0 | | X | The Pose prediction Error for the Position/Orientation component cannot be estimated. |
| 1 | 0 | 1 | | 0 | The Pose prediction Error for the Orientation component is uncertain. Low level of confidence.  Tracking is lost for both predicted poses used to calculate the pose prediction error. |
| 1 | 0 | 1 | | 1 | The Pose prediction Error for the Orientation component is uncertain. Low level of confidence.  Tracking is lost for the first predicted pose used to calculate the pose prediction error. |
| 1 | 1 | 1 | | 0 | The Pose prediction Error for the Orientation component is uncertain. Low level of confidence.  Tracking is lost for the second predicted pose used to calculate the pose prediction error. |
| 1 | 1 | 1 | | 1 | The Pose prediction Error for the Orientation component is OK. High level of confidence. |

“X” means “0 or 1”

**3. Conclusion**

We propose to rename the accuracy level to confidence status to better reflect its purpose.

And we propose an example to derive the confidence status of the pose error using Kronos OpenXR API to address the editor’s note in clause 6.3.5.3 “the computation of the accuracy level using the XrViewStateFlags is FFS.”

**4. Proposal**

Based on the above discussion, it is proposed to agree the following changes to 3GPP TR 26.812.

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

### 6.3.5 Pose error and time error

#### 6.3.5.1 Background

The rendering process may use a predicted pose for rendering. The pose error (the difference between the pose used for rendering and the pose at the actual display time) affects the match can cause motion sickness, although the XR Runtime can mitigate the impact of pose errors to some extent by reprojection. Thus the pose error is a relevant metric for QoE.

The pose error depends on the time error (for a rendered frame, how much the predicted display time is off from the actual display time). The time error can be used as a control knob by the rendering process and the communication network to adjust the respective delays in optimizing the QoE. Therefore, the time error is a relevant metric for QoE optimization.

#### 6.3.5.2 Metric description

As described in clause 6.2.1.1, a pose can be described by a *position* and an *orientation* in space relative to an XR Space. Viewer Pose Prediction Error QoE metric is defined in the below table 6.3.5.2-1.

Table 6.3.5.2-1: Viewer Pose Prediction Error



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Key | | | | | | Type | Description |
| ViewerPosePredictionErrorSet | | | | | | Set | Set of viewer pose prediction errors. |
|  | *Entry* | | | | | Object |  |
|  |  | Time | | | | Integer | The time when the predicted viewer pose is used for. |
|  |  |  | view | | | Integer | The view index (0 for left eye and 1 for right eye) |
|  |  |  |  | Pose prediction error | | Set | The deviation between the actual and predicted pose. |
|  |  |  |  |  | Position prediction error | Vector | Vector distance between the actual and predicted position |
|  |  |  |  |  | Orientation prediction  error | Vector | Quaternion distance between the actual and predicted position |
|  |  |  |  |  | Confidence in Position prediction error | enum | Confidence status on the position predicted error.  (NOT\_VALID, UNCERTAIN, OK) |
|  |  |  |  |  | Confidence in Orientation prediction  error | enum | Confidence status on the orientation predicted error.  (NOT\_VALID, UNCERTAIN, OK) |
|  |  |  |  | FoV prediction error | | Set | The deviation between the actual and predicted FoV. |
|  |  |  |  |  | Left error | float | Difference between the actual and predicted left angle of FoV |
|  |  |  |  |  | Right error | float | Difference between the actual and predicted right angle of FoV |
|  |  |  |  |  | Up error | float | Difference between the actual and predicted Up angle of FoV |
|  |  |  |  |  | Down error | float | Difference between the actual and predicted Down angle of FOV |

An example of Confidence status in the position, respectively orientation, component of the Viewer Pose Prediction Error is listed in table 6.3.5.3-1, respectively 6.3.5.3-2.

The view is an integer value specifying left or right eye. In OpenXR this corresponds to view index in XrViewConfigurationProperties and XrCompositionLayerProjection. As an example, the deviation of actual and predicated pose information can be summarized into a single metirc as formula 6.3-1. In this formula, *DevposPredError* means the deviation of actual and predicated pose information. *α* and *β* represent the weights of the deviation of position and orientation respectively, the weights may be set based on the implementation or application. *PA*, *PP* refer to the actual position and the predicted position respectively, with (*x,y,z*) indicating their respective Cartesian coordinates , *and QA* ,*QP* refer to the actual orientation and the predicted orientation respectively, expressed as unit quaternions and Q-1 indicates the quaternion conjugation operation.

6.3-1



Note that the actual pose may not be known during an XR session.

6.3.5.3 Measurement procedure

A measurement procedure for the scenario of cloud-based rendering is shown in Figure 5.2.3-1. The XR Runtime and the XR Application may be on a same device such as a UE, or on difference devices such as an AR glasses (which hosts the XR Runtime) and a UE (which hosts the XR Application). The steps are as follows:

1) The XR Application estimates the round-trip time (RTT) between the XR application and the Edge Application Server (EAS).

2) The XR Application queries for the next display time. This (and step 3) can be achieved by calling the xrWaitFrame function in OpenXR.

3) The XR Runtime replies with the next display time.

4) The XR application predicts a display time – an initial prediction – and the use of initial is because a second prediction/estimation will be made later. This predicted display time is called T2.predicted1.

5) The XR application queries for a predicted pose at the initial predicted display time T2.predicted1. Calling the function xrLocateViews in OpenXR can achieve this step and step 7.

6) The XR Runtime predicts the pose, and the prediction occurs at time T1.

7) The XR Runtime returns the predicted pose (P.predicted1) including status flags information.

7bis) The XR application checks the status flags information (F.predicted1) related to the predicted pose . If the pose is not valid on position and/or orientation, the XR application may need to go to step 5 to query for a new predicted pose.

8) The XR application sends the predicted pose (P.predicted1) and the associated initial predicted display time (T2.predicted1) to the EAS with the status flags.

9) The EAS renders for the predicted pose (P.predicted1), and compresses the rendered frame.

10) The EAS returns the rendered frame along with the initial predicted display time (T2.predicted1) to the XR Application.

11) The XR Application sends the rendered frame to the XR Runtine, e.g., via swapchain. This can be achieved by calling the xrReleaseSwapchainImage function in OpenXR. The XR Application passes the display time used for the rendering the frame, and this can be achieved by calling the xrEndFrame function in OpenXR.

12) The XR Application queries for the predicted display time. This is intended to get a more accurate prediction of the display time than the one in step 4, because there is less time to predict into the future at this moment.

13) The XR Runtime returns an updated prediction of the display time (T2.predicted2).

14) The XR Runtime performs reprojection for pose correction. The actual display play time is called T2.actual.

15) The XR Application queries for the pose associated with the updated prediction of the display time (T2.predicted2). This can be achieved by calling the xrLocateViews function in OpenXR.

16) The XR Runtime does pose estimation.

17) The XR Runtime returns a pose estimate (P.predicted2) including status flags information.

18) The XR Application checks the confidence status (C.predicted2) using the status information related to the pose (P.predicted2) in step 17 and the status flags information (F.predicted1) in step 7bis. Then the XR Application computes a pose error estimate (P.predicted1 – P.predicted2) and a time error estimate(T2.predicted1 – T2.predicted2) according to the confidence status (C.predicted2).

****

**Figure 6.3.5.3-1: The procedure for measuring the pose error and time error in pose prediction**

Note that two queries are used to predict the display time of a same frame. The first query occurs in step 2, and the query result is used to determine a target display time for the rendering process in step 4. The second query occurs much closer to the actual display time, as shown in steps 12-13, and thus provides higher accuracy. This is shown in Figure 6.3.5.3-2.

****

**Figure 6.3.5.3-2: The use of a second prediction (T2.predicted2) of the display time for better accuracy**

Note: to derive the confidence status in 18 with the Khronos OpenXR API [22], the xrLocateViews function returns the status flags information related to the predicted/estimated pose in the XrViewState structure. XrViewStateFlags in the XrViewState are flags that give information validity and tracking of position and orientation:

- XR\_VIEW\_STATE\_ORIENTATION\_VALID\_BIT

- XR\_VIEW\_STATE\_POSITION\_VALID\_BIT

- XR\_VIEW\_STATE\_POSITION\_TRACKED\_BIT

- XR\_VIEW\_STATE\_ORIENTATION\_TRACKED\_BIT

XrViewStateFlags should be checked in steps 7bis and 18 of the measurement procedure before using the predicted/estimated pose.

XrViewStateFlags of the two predicted poses may be combined to derive the confidence status of the position and/or orientation error. The XrViewStateFlags can be independently checked on the two pose components (position and orientation) to get a confidence status on the error per component.

Combining the XrViewStateFlags of the two predicted poses (P.predicted1, P.predicte2) give the confidence status of the position and/or orientation error. The Confidence may be reported with the Viewer Pose Prediction Error in table 6.3.5.2-1

An example of Confidence status in the position, respectively orientation, component of the Viewer Pose Prediction Error is listed in table 6.3.5.3-1, respectively 6.3.5.3-2.

Table 6.3.5.3-1 – Example of Confidence status in the Position of the Pose prediction error.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| First Pose (P.predicted1) | | | Second Pose (P.predicted2) | | Confidence status in the Position of the Pose prediction Error | |
| Position status | | | Position status | |
| VALID\_BIT | TRACKED\_BIT | VALID\_BIT | | TRACKED\_BIT |
| 0 | X | X | | X | NOT\_VALID | The Pose prediction Error for the Position component cannot be estimated. |
| X | X | 0 | | X | NOT\_VALID | The Pose prediction Error for the Position component cannot be estimated. |
| 1 | 0 | 1 | | 0 | UNCERTAIN | The Pose prediction Error for the Position component is uncertain. |
| 1 | 0 | 1 | | 1 | UNCERTAIN | The Pose prediction Error for the Position component is uncertain. |
| 1 | 1 | 1 | | 0 | UNCERTAIN | The Pose prediction Error for the Position component is uncertain. |
| 1 | 1 | 1 | | 1 | OK | The Pose prediction Error for the Position component is OK. |

“X” means “0 or 1”

Table 6.3.5.3-2 – Example of Confidence status in the Orientation of the Pose prediction error.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| First Pose (P.predicted1) | | | Second Pose (P.predicted2) | | Confidence status in the Orientation of the Pose prediction Error | |
| Orientation status | | | Orientation status | |
| VALID\_BIT | TRACKED\_BIT | VALID\_BIT | | TRACKED\_BIT |
| 0 | X | X | | X | NOT\_VALID | The Pose prediction Error for the Orientation component cannot be estimated. |
| X | X | 0 | | X | NOT\_VALID | The Pose prediction Error for the Orientation component cannot be estimated. |
| 1 | 0 | 1 | | 0 | UNCERTAIN | The Pose prediction Error for the Orientation component is uncertain. |
| 1 | 0 | 1 | | 1 | UNCERTAIN | The Pose prediction Error for the Orientation component is uncertain. |
| 1 | 1 | 1 | | 0 | UNCERTAIN | The Pose prediction Error for the Orientation component is uncertain. |
| 1 | 1 | 1 | | 1 | OK | The Pose prediction Error for the Orientation component is OK. |

“X” means “0 or 1”

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