A picture containing drawing, food

Description automatically generated

**ISO/IEC JTC 1/SC 29/WG 11 N19375**

**ISO/IEC JTC 1/SC 29/WG 11**

**Coding of moving pictures and audio**

**Convenorship: UNI (Italy)**

**Document type: Approved WG 11 document**

**Title: OpenV3C – Multi-platform open-source implementation of the V-PCC**

**Status: Approved**

**Date of document: 2020-04-24**

**Source: 3DG**

**Expected action:**

**No. of pages:**

**Email of convenor: leonardo@chiariglione.org**

**Committee URL: mpeg.chiariglione.org**

**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 11**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC 1/SC 29/WG 11 N19375**

**Alpbach, AT, online – April 2020**

|  |  |
| --- | --- |
| **Source:** | **3DG** |
| **Title:** | **OpenV3C – Multi-platform open-source implementation of the  V-PCC** |

Introduction

OpenV3C is a cross-platform open-source framework that aims to grow the immersive media ecosystem by evangelizing immersive media delivery methods based on ISO/IEC 23090-5 standard and its future extensions. OpenV3C provides decoding and rendering capabilities for the coded representation of the dynamic point cloud sequences. This framework is developed and distributed under open design, open collaboration and open communication principles and can be used for research and commercial applications.

**OpenV3C: Multi-platform open-source implementation of the V-PCC**

# OpenV3C Community Membership

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Requirements | Responsibilities | Privileges |
| Member | Sponsor from 2 approvers, active in the community, contributed to OpenV3C | Welcome and guide new contributors | OpenV3C GitLab organization Member |
| Approver | Sponsor from 2 maintainers, has good experience and knowledge of the domain, actively contributed to code and review | Review and approve contributions from community members | Write access to specific packages in the relevant repository |
| Maintainer | Sponsor from 2 owners, shown good technical judgement in feature design/development and PR review | Participate in release planning and feature development/maintenance | Top-level write access to the relevant repository. Name entry in Maintainers file of the repository |
| Owner | Sponsor from 3 owners, helps drive the overall OpenV3C project | Drive the overall technical roadmap of the project and set priorities of activities in release planning | OpenV3C GitLab organization Admin access |

Member  
Members are active participants in the community who contribute by authoring PRs, reviewing issues/PRs or participate in community discussions on the slack/mailing list.

Requirements

* Sponsor from 2 approvers
* Enabled [two-factor authentication] on their GitLab account
* Actively contributed to the community. Contributions may include, but are not limited to:
  + Authoring PRs
  + Reviewing issues/PRs authored by other community members
  + Participating in community discussions on slack/mailing list
  + Participate in OpenV3C community meetings

Responsibilities and privileges

* Member of the OpenV3C GitLab organization
* Can be assigned to issues and PRs and community members can also request their review
* Participate in assigned issues and PRs
* Welcome new contributors
* Guide new contributors to relevant docs/files
* Help/Motivate new members in contributing to OpenV3C

Approver  
Approvers are active members who have good experience and knowledge of the domain. They have actively participated in the issue/PR reviews and have identified relevant issues during the review.

Requirements

* Sponsor from 2 maintainers
* Member for at least 2 months
* Have reviewed a good number of PRs
* Have good codebase knowledge

Responsibilities and Privileges

* Review code to maintain/improve code quality
* Acknowledge and work on review requests from community members
* May approve code contributions for acceptance related to relevant expertise
* Have 'write access' to specific packages inside a repo, enforced via bot
* Continue to contribute and guide other community members to contribute in OpenV3C project

Maintainer  
Maintainers are approvers who have shown good technical judgement in feature design/development in the past.

Requirements

* Sponsor from 2 owners
* Approver for at least 2 months
* Nominated by a project owner
* Good technical judgement in feature design/development

Responsibilities and privileges

* Participate in release planning
* Maintain project code quality
* Ensure API compatibility with forward/backward versions based on feature graduation criteria
* Analyze and propose new features/enhancements in OpenV3C project
* Demonstrate sound technical judgement
* Mentor contributors and approvers
* Have top-level write access to the relevant repository (able click Merge PR button when manual check-in is necessary)
* Name entry in Maintainers file of the repository
* Participate & Drive design/development of multiple features

Owner  
Owners are maintainers who have helped drive the overall project direction. Has a deep understanding of OpenV3C and related domain and facilitates major agreement in release planning

Requirements

* Sponsor from 3 owners
* Maintainer for at least 2 months
* Nominated by a project owner
* Not opposed by any project owner
* Helped in driving the overall project

Responsibilities and Privileges

* Make technical decisions for the overall project
* Drive the overall technical roadmap of the project
* Set priorities of activities in release planning
* Guide and mentor all other community members
* Ensure all community members are following Code of Conduct
* Although given admin access to all repositories, make sure all PRs are properly reviewed and merged
* May get admin access to the relevant repository based on the requirement
* Participate & Drive design/development of multiple features

Note: These roles are applicable only for OpenV3C GitLab organization and repositories. Currently, OpenV3C doesn't have a formal process for review and acceptance into these roles. We will come up with a process soon.

# Roadmap

OpenV3C 1.0 is our current stable branch. The roadmap below outlines new features that will be added to OpenV3C.

<http://mpegx.int-evry.fr/software/MPEG-Public/PCC/V-PCC_decoder>

*2020 Q2 Roadmap*

* Support for TMC2v9.0 compliant bitstreams.
* Support for SEI messages.
* Enhance the performance and reliability of OpenV3C.
* Improve contributor experience by defining project governance policies, release process, membership rules etc.
* Improve the performance of point cloud reconstruction.
* Improve OpenIMP installation experience

*Future*

* Support for the iOS platform.
* Support for Mixed reality headsets.
* Enable point cloud smoothing functionality.
* Support for MIV content.
* Advanced analytics for memory usage.
* Integrate bitstream compliance check.
* Support for over-the-network streaming

# OpenV3C core

The core of the OpenV3C allows the cross-platform implementation of the main functionality for the immersive media decoding and reconstruction.

The emerging dynamic point cloud compression standard enables efficient transmission of volumetric media content across the wireless networks. The volumetric media enables a variety of rich user experiences, including interacting with remote objects, telepresence, and augmented reality (AR) experiences, where the real world is augmented or enhanced by virtual objects. In this paper, we present a solution for real-time decoding and rendering of an immersive media using V-PCC compression standard on mobile devices, leveraging the mobile device hardware including a graphics processing unit (GPU) and hardware video decoder to accelerate performance.

OpenV3C is a solution for real-time decoding and rendering of the V-PCC content, leveraging the device hardware including GPU and video decoder to accelerate performance. OpenV3C supports AR experiences including the presentation of v-pcc-decoded content super-imposed on the external environment as seen through the camera.

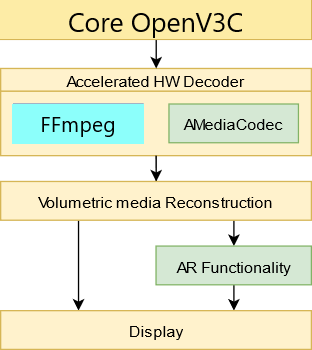


fig. 1. The architecture of the OpenV3C.

Volumetric media such as dynamic point clouds enables the viewer to freely navigate in an augmented scene with 6 degrees of freedom.

The philosophy of the V-PCC solution is to build a volumetric media delivery ecosystem based on the existing hardware infrastructure. The technology is based on creating a set of projections from 3d space to 2d space and using the variety of video coding solutions to achieve efficient compression with an addition of the auxiliary information that enables inverse transformation from 2d projections to 3d volume.

# Decoding process

Well known problem component synchronization that is present in the heterogeneous processing systems it transferring large amounts of data, such as decoded images over the but from GPU to CPU memory.

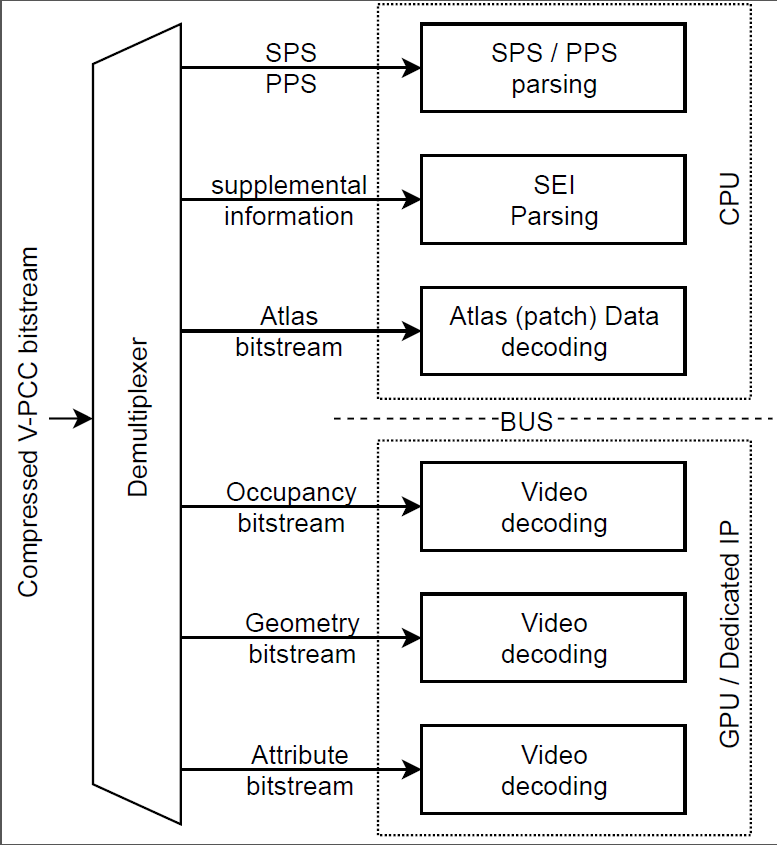


fig. 2. Diagram of the decoder pipeline of the OpenV3C.

The architecture assumes that the video components of the corresponding v-pcc sub-bitstream shall be decoded using the hardware-accelerated video decoder depending on the specific platform.

The basic reconstruction process of the dynamic point cloud data comprises of the basic 2d to 3d conversion based on atlas information. The patch in projected image space is bounded by a 2d bounding box with the coordinates defined in the atlas data, the blocks with valid samples have a corresponding non-zero value in the down-sampled occupancy map. The geometry information provides the delta depth map for the occupied samples of the patch while the attributes, in this case, add the colour to the reconstructed point cloud.

The volumetric information in a V-PCC bitstream is composed of several sub bitstreams that represent atlas data, occupancy map, geometry, attribute (colour). The packing of projected information is performed based on a block-basis.

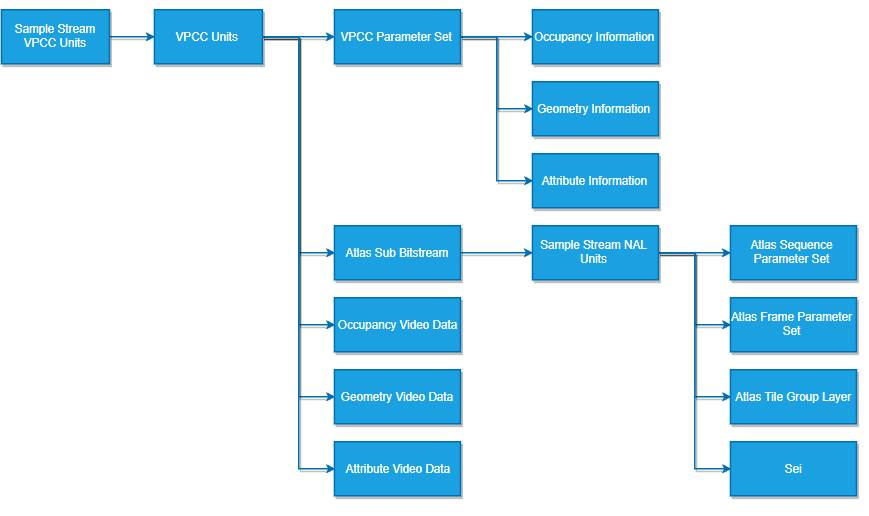


fig 3. V-PCC sample stream format diagram

The input v-pcc bitstream is parsed by the preprocessing using and the VUI, SEI, SPS, PPS and Atlas sub-bitstreams are processed using the CPU. Occupancy map, Geometry and Attribute components are decoded using the hardware module, and the result of the decoding is mapped as a video surface directly in the GPU memory.

# Reconstruction process

The AR module processes the camera images to reconstruct and create the volumetric scene, subsequently, the AR module detects and tracks planes and spatial anchor points. The tracked planes are presented on the display, enabling the user to select a spatial anchor point, which defines where to place the point cloud in the scene. After the user selects a spatial anchor, the point cloud is displayed in the scene using the anchor transformation matrix.

The v-pcc rendering pipeline is composed of the following steps:

* Initialize global variables
* Update transformation matrices
* Render the point cloud

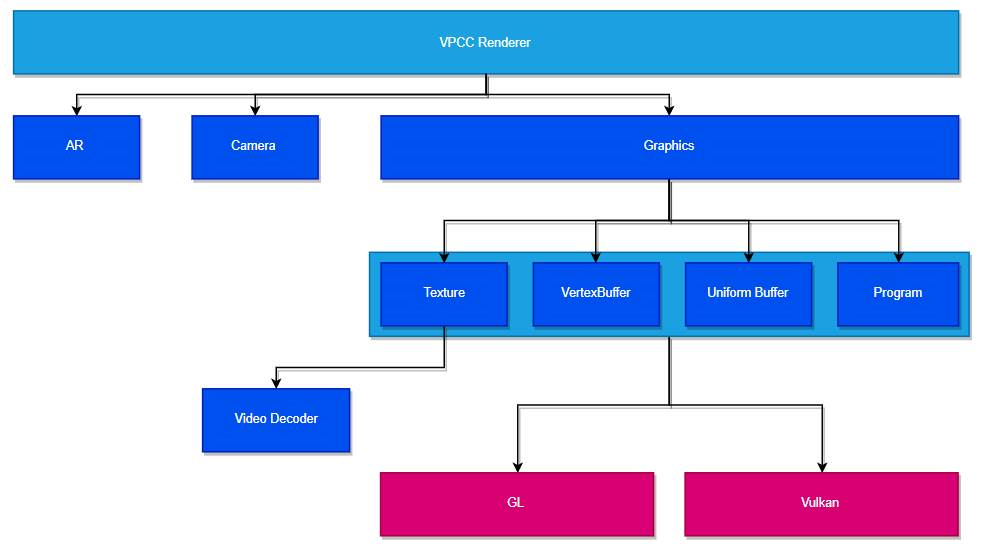
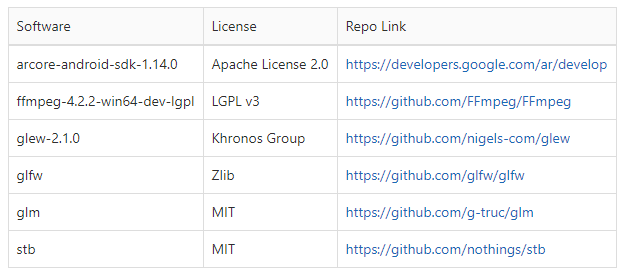


fig.4. v-pcc renderer architecture.

On each frame, the point cloud is rendered with the following process

* Generate a block to patch the map.  
  This defines a mapping of blocks in atlas space to corresponding patches.
* Bind the GPU shader program.  
  The geometry shader program is binded. The geometry shader program emits points directly on GPU, reducing CPU overhead and improving performance. Alternatively, this concept can be achieved using a compute shader approach.
* Update GPU textures  
  The decoded images from the occupancy, geometry, and attribute video streams are binded directly as GPU YUV (NV12) textures, with no copy in either CPU or GPU space
* Update GPU buffers  
  The uniform buffer: ubo, are updated with the following data:   
   frame width and height, occupancy resolution, and modelViewProj matrix. The uniform buffer is stored in the cache memory and contains data that is constant for that frame.
* Render the patches using single draw call
  + The uniform buffer is updated. The uniform buffer contains the patch data as a global array, including canvas (atlas) to patch matrices.
  + The vertex buffer is updated.  
    Where each entry defines a block in atlas space and corresponding patch index
  + The draw call is issued to render the points.  
    The primitive count is set to be equal to the number of blocks. Primitives (points) are emitted on GPU directly using geometry shader or compute shader
* GPU Rendering
  + Vertex Shader:  
    Copy the per-vertex data to the geometry shader
  + Geometry Shader:  
    Enables instancing, where each instance emits multiple vertices. This reduces CPU overhead. Max # vertices emitted per geometry shader invocation is equal to occupancy\_resolution \* occupancy\_resolution \* num\_layers.   
    Each invocation processes a sub-block in atlas space.

# OpenV3C External Build Dependencies



This product includes software developed at the Apache Software Foundation (http://www.apache.org/).

This software uses libraries from the FFmpeg project under the LGPLv2.1

# License

Copyright (c) 2020, Futurewei Technologies., Inc.

Licensed under the Apache License, Version 2.0 (the License);

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org//licenses//LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an AS IS BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License.

# Acknowledgement

The authors of the open-source framework would like to express their gratitude to Christian Tulvain and Marius Preda for providing the technical platform for the OpenV3C project. Your efforts support the goal of popularizing the immersive media delivery based on ISO/IEC 23090-5 standard in the international community.