Class BayerDemosaicOp

Table of contents

- Defined in File bayer_demosaic.hpp

Inheritance Relationships

Base Type

- `public holoscan::Operator` (Class Operator)

Class Documentation

class BayerDemosaicOp : public holoscan::Operator

Operator class to demosaic the input video stream.

==Named Inputs==

- **receiver** : `nvidia::gxf::Tensor` or `nvidia::gxf::VideoBuffer`
    - The input video frame to process. If the input is a VideoBuffer it must be an 8-bit unsigned grayscale video (`nvidia::gxf::VideoFormat::GXF_VIDEO_FORMAT_GRAY`). If a video buffer is not found, the input port message is searched for a device tensor with the name specified by `in_tensor_name`. The tensor must have either 8-bit or 16-bit unsigned integer format. The tensor or video buffer may be in either host or device memory (a host->device copy is performed if needed).

==Named Outputs==

- **transmitter** : `nvidia::gxf::Tensor`
    - The output video frame after demosaicing. This will be a 3-channel RGB image if `alpha_value` is true, otherwise it will be a 4-channel RGBA image. The data type will be either 8-bit or 16-bit unsigned integer (matching the bit depth of the input). The name of the tensor that is output is controlled by `out_tensor_name`. The output will be in device memory.

==Parameters==

- **pool**: Memory pool allocator (holoscan::Allocator) used by the operator.
- **cuda_stream_pool**: `holoscan::CudaStreamPool` instance (`CudaStreamPool`) to allocate CUDA streams. Optional (default: `nullptr`).
- **in_tensor_name**: The name of the input tensor. Optional (default: `""`).
- **out_tensor_name**: The name of the output tensor. Optional (default: `""`).
- **interpolation_mode**: The interpolation model to be used for demosaicing. Values available at: https://docs.nvidia.com/cuda/npp/nppdefs.html?highlight=Two%20parameter%20cubic%20filter#c.NppiInterpolationMode
  - NPPI_INTER_UNDEFINED (`0`): Undefined filtering interpolation mode.
  - NPPI_INTER_NN (`1`): Nearest neighbor filtering.
  - NPPI_INTER_LINEAR (`2`): Linear interpolation.
  - NPPI_INTER_CUBIC (`4`): Cubic interpolation.
  - NPPI_INTER_CUBIC2P_BSPLINE (`5`): Two-parameter cubic filter (B=1, C=0)
  - NPPI_INTER_CUBIC2P_CATMULLROM (`6`): Two-parameter cubic filter (B=0, C=1/2)
  - NPPI_INTER_CUBIC2P_B05C03 (`7`): Two-parameter cubic filter (B=1/2, C=3/10)
  - NPPI_INTER_SUPER (`8`): Super sampling.
  - NPPI_INTER_LANCZOS (`16`): Lanczos filtering.
  - NPPI_INTER_LANCZOS3_ADVANCED (`17`): Generic Lanczos filtering with order 3.
  - NPPI_SMOOTH_EDGE (`0x8000000`): Smooth edge filtering.

  Optional (default: `0`).

- **bayer_grid_pos**: The Bayer grid position. Values available at: https://docs.nvidia.com/cuda/npp/nppdefs.html?highlight=Two%20parameter%20cubic%20filter#c.NppiBayerGridPosition
    - NPPI_BAYER_BGGR (`0`): Default registration position BGGR.
    - NPPI_BAYER_RGGB (`1`): Registration position RGGB.
    - NPPI_BAYER_GBRG (`2`): Registration position GBRG.
    - NPPI_BAYER_GRBG (`3`): Registration position GRBG.

  Optional (default: `2`).
- **generate_alpha**: Generate alpha channel. Optional (default: `false`).
- **alpha_value**: Alpha value to be generated if `generate_alpha` is set to `true`. Optional (default: `255`).

==Device Memory Requirements==

When using this operator with a `BlockMemoryPool`, the minimum `block_size` is `(rows * columns * output_channels * element_size_bytes)` where `output_channels` is 4 when `generate_alpha` is true and 3 otherwise. If the input tensor or video buffer is already on the device, only a single memory block is needed. However, if the input is on the host, a second memory block will also be needed in order to make an internal copy of the input to the device. The memory buffer must be on device (`storage_type` = 1).

Public Functions

HOLOSCAN_OPERATOR_FORWARD_ARGS (BayerDemosaicOp)
BayerDemosaicOp()=default

virtual void setup(OperatorSpec &spec) override

Define the operator specification.

Parameters

spec – The reference to the operator specification.

virtual void initialize() override

Initialize the operator.

This function is called when the fragment is initialized by Executor::initialize_fragment().

virtual void compute(InputContext &op_input, OutputContext &op_output, ExecutionContext &context) override

Implement the compute method.

This method is called by the runtime multiple times. The runtime calls this method until the operator is stopped.

Parameters

- **op_input** – The input context of the operator.
- **op_output** – The output context of the operator.
- **context** – The execution context of the operator.

virtual void stop() override

Implement the shutdown logic of the operator.

This method is called multiple times over the lifecycle of the operator according to the order defined in the lifecycle, and used for heavy deinitialization tasks such as deallocation of all resources previously assigned in start.