Video Replayer

Table of contents

List of Figures

So far we have been working with simple operators to demonstrate Holoscan SDK concepts. In this example, we look at two built-in Holoscan operators that have many practical applications.

In this example we'll cover:

- how to load a video file from disk using **VideoStreamReplayerOp** operator
- how to display video using **HolovizOp** operator
- how to configure your operator's parameters using a YAML configuration file

> **Note**
>
> The example source code and run instructions can be found in the examples directory on GitHub, or under `/opt/nvidia/holoscan/examples` in the NGC container and the debian package, alongside their executables.

Operators and Workflow

Here is the diagram of the operators and workflow used in this example.

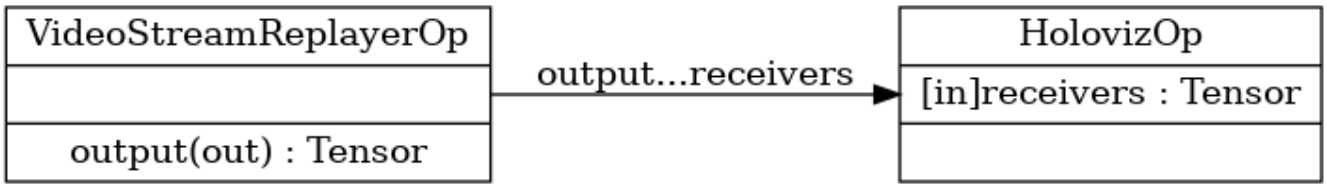

Fig. 8 *Workflow to load and display video from a file*

We connect the "output" port of the replayer operator to the "receivers" port of the Holoviz operator.

Video Stream Replayer Operator

The built-in video stream replayer operator can be used to replay a video stream that has been encoded as gxf entities. You can use the `convert_video_to_gxf_entities.py` script (installed in `/opt/nvidia/holoscan/bin` or available on GitHub) to encode a video file as gxf entities for use by this operator.

This operator processes the encoded file sequentially and supports realtime, faster than realtime, or slower than realtime playback of prerecorded data. The input data can optionally be repeated to loop forever or only for a specified count. For more details, see operators-video-stream-replayer.

We will use the replayer to read gxf entities from disk and send the frames downstream to the Holoviz operator.

Holoviz Operator

The built-in Holoviz operator provides the functionality to composite real time streams of frames with multiple different other layers like segmentation mask layers, geometry layers and GUI layers.

We will use Holoviz to display frames that have been sent by the replayer operator to it's "receivers" port which can receive any number of inputs. In more intricate workflows, this port can receive multiple streams of input data where, for example, one stream is the original video data while other streams detect objects in the video to create bounding boxes and/or text overlays.

Application Configuration File (YAML)

The SDK supports reading an optional YAML configuration file and can be used to customize the application's workflow and operators. For more complex workflows, it may be helpful to use the application configuration file to help separate operator parameter settings from your code. See Configuring an Application for additional details.

> **Tip**
>
> For C++ applications, the configuration file can be a nice way to set the behavior of the application at runtime without having to recompile the code.

This example uses the following configuration file to configure the parameters for the replayer and Holoviz operators. The full list of parameters can be found at operators-video-stream-replayer and operators-holoviz.

> %YAML 1.2 replayer: directory: "../data/racerx" # Path to gxf entity video data basename: "racerx" # Look for <basename>.gxf_{entities|index} frame_rate: 0 # Frame rate to replay. (default: 0 follow frame rate in timestamps) repeat: true # Loop video? (default: false) realtime: true # Play in realtime, based on frame_rate/timestamps (default: true) count: 0 # Number of frames to read (default: 0 for no frame count restriction) holoviz: width: 854 # width of window size height: 480 # height of window size tensors: - name: "" # name of tensor containing input data to display type: color # input type e.g., color, triangles, text, depth_map opacity: 1.0 # layer opacity priority: 0 # determines render order, higher priority layers are rendered on top

The code below shows our `video_replayer` example. Operator parameters are configured from a configuration file using `from_config()` (C++) and `self.**kwargs()` (Python).

Ingested Tab Module

Running the Application

Running the application should bring up video playback of the video referenced in the YAML file.