ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-THMBCM036

VIDEO COMPRESSION FOR areaDetector

B. A. Sobhani, Oak Ridge National Laboratory, Oak Ridge, TN, USA

Abstract

At neutron sources such as SNS and HFIR, neutrons collide with neutron detectors at a much lower rate than light would for an optical detector. Additionally, the image typically does not pan or otherwise move. This means that the incremental element-by-element differences between frames will be small. This makes neutron imaging data an ideal candidate for video-level compression where the incremental differences between frames are compressed and sent, as opposed to image-level compression where the entire frame is compressed and sent. This paper describes an EPICS video compression plugin for areaDetector that was developed at SNS.

INTRODUCTION

Plugins exist to compress/decompress frames in areaDector. This kind of compression is useful and helps to reduce size of transmitted data, but is limited by the fact it only compresses in space (the x/y coordinates of the image), but not in time. So a video consisting of 100 frames of the same image would compress to 100 multiplied by the compressed size of a single frame. Video codecs, such as H.264 and others, employ differential frame compression [1] so that a series of 100 identical images would compress to something much smaller than 100 multiplied by the size of a single compressed.

LIBAVCODEC

In this section some terminology is briefly discussed.

Packets

One can think of a packet in libavcodec as being a compressed frame, with the exception that a packet does not by itself hold enough information to reconstruct a frame. A packet can only reconstruct a frame in conjunction with preceding frames. In the event of a missing packet, the video stream will experience some distortion in the subsequent frames.

Frames

The frame, in libavcodec, is the actual image that is displayed to the user. Note that this definition of frame is different from that used when discussing certain codecs, where I-frame, and P-frame, B-frame are more analogous to “packets” described in the previous section.

COMPRESSION

Compression was done in areaDetector by making additions to the ADSupport module. Frame data from the NDArray is sent through to an AVContext object. Packets were then extracted and broadcast via PVAccess.

DECOMPRESSION

Decompression code was developed as additions to the same ADSupport module. Then an ImageJ plugin was developed which is just a java wrapper to those functions. Doing it this way will allow for video decompression to be easily expanded to a variety of clients.

CURRENT STATUS AND TESTING

At the time of writing this paper, the proof of concept was very recently demonstrated to work. The test consisted of taking images (at the NDArray level in an areaDetector plugin), producing a packet (based on the current image and the previous images that were sent), storing the packet inside a PVA stream, and decoding this PVA stream in ImageJ to get the original image.

This particular test was done with the mpeg-1 codec – this was an arbitrary choice, and since libavcodec is a general interface to many codecs it will be easy to test other codecs.

Since the proof of concept was only recently demonstrated, more thorough performance testing has not yet taken place. Controlled tests should be done to compare the size of individually compressed frames vs video compression. Additionally, there should be testing done to measure degradation of compression over time. For neutron scattering data, compression typically gets much worse over time, as noise from the neutron events get accumulates. Video compression is expected to be much more robust in this aspect – testing must be conducted to confirm it.

ACKNOWLEDGEMENTS

Acknowledgements to Kaz Gofron and supervisor Bogdan Vacaliuc for preliminary discussions on how to implement this. Also to VULCAN beamline for allowing testing of compression using their beamline.

REFERENCES

[1] https://ottverse.com/i-p-b-frames-idr-keyframes-differences-usecases/