Computer Vision Progress Report

Group members: Jiajun Li, Hanliang Jiang, Haoxiang Li
Project title: Reproduction and Optimization of Context-based Trit-Plane Coding Algorithm

Introduction:

According to the former progress plan, we decided to write code to measure average overhead, average compression rate, average peak signal-noise ratio of the context-base trit plane coding algorithm. We also experimented with image compression over different scalability. Based on the result and the fine granular scalability feature, we decided to further expand the algorithm to practical application. We designed and implemented a fully functioned coding-transformation-decoding-rendering application to simulate and measure its performance under real time environment. Aimed to build an application with features assemble FaceTime and whatsapp video call, we used programming tools like socket, multi-threading and video codec to support the scheme. After testing the application in different conditions, we generated output videos just like facetime does, either under the condition of facetime network congestion, or under a good network connection.

Details of the approach:

External code: janghl/CTC: Context-Based Trit-Plane Coding for Progressive Image Compression, CVPR2023 (github.com)

To test the average overhead, average compression rate, average peak signal-noise ratio of the context-base trit plane coding algorithm, we had written code for batch processing and harvested plentiful experiment data before the progress report was written.

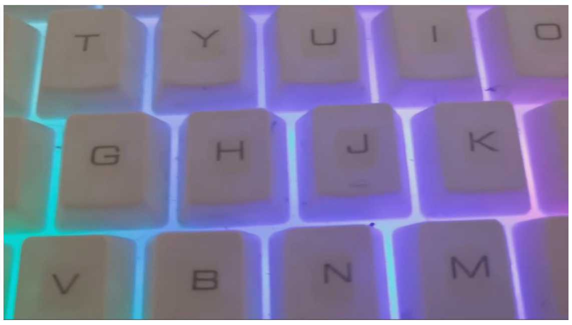
To implement the real time video transforming feature based on the algorithm's fine granular scalability, we used multi-thread technique to simulate the network connecting environment. The sender thread first chop the input video to frames, and each time after the following frame is generated, the sender cast a forward propagation within the preloaded image coding model. The model turns the frame into a fixed bunch of binary files. A time interval is preset properly to better simulate the network congestion condition. Each time before a frame is generated, a timer is started, and after the forward propagation process, the binary files are transformed through socket as many as possible by prefix increasing order before the alarm goes off. This protocol use the preset time interval to simulate the real-world physical runtime stream transportation condition.

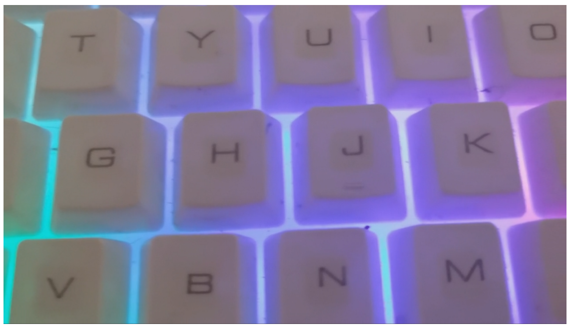
The receiver thread, on the other hand, works as a socket server and keeps its eyes and ears open for incoming data streams. Upon receiving meta data of the incoming binary file stream, the receiver store the files locally and decode the random number of binary files to get the image with a random resolution according to how many files in this round are received. If the receiver receive a header which indicates a new round of file streams are coming, it flush the whole local directory and begin to receive and store files for the new round. In the end, the receiver assemble the image generated to an intact video file.

This implementation functions well as expected. However, there are a few aspects to improve. First of all, the performance of real time simulation strictly depends on cuda and nvidia gpu, yet famous video transportation applications in the world are mostly deployed on top of embedded systems like mobile phones. Besides, if we want to fully test the performance of real-world conditions, we need internet connect rather than inter-thread connections, and we also need runtime frame generating interfaces rather than pre-recorded videos.

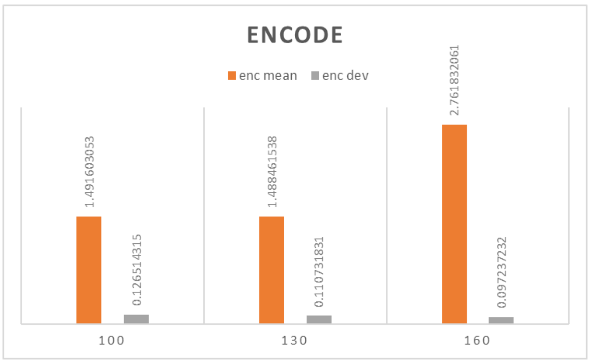
Results:

Below are screenshots of original video as an input and the generated video under the reconstruction level of 130 as the output.

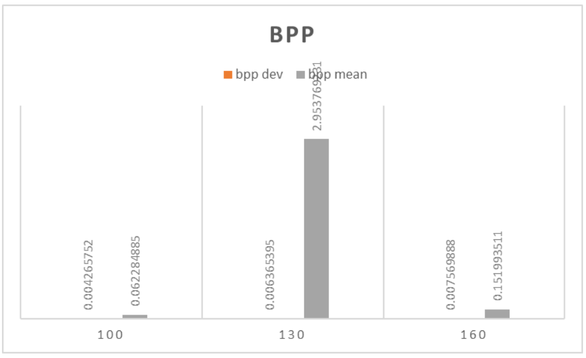

original video


generated video

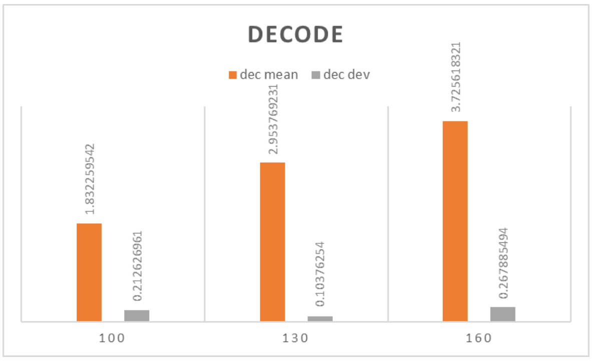
To test the performance of video transportation under different reconstruction levels, we test on identical video with recon-levels of 100, 130 and 160, to see which one maintains high quality while supports lower latency.



mean and deviation for Encoding



mean and deviation for BPP



mean and deviation for Decoding

Above are mean and deviation from the original experiment data.

Discussion and conclusions:

Comparing the screenshots of original video input and output video above, we can notice that the generated video is slightly blurred. With this small tradeoff, we achieve lower encoding decoding latency and transportation time, which benefits a lot in runtime video stream transportation.

From the histogram of experiment data above, among the three construction levels, we can conclude that recon level 100 satisfies both low encoding and decoding time and low compression rate(BPP).

Statement of individual contribution:

Three members in the group share the work evenly across different work, including runtime environment configuration, paper reading and programming. We contact and discuss with each other closely through online methods, and we talk about arrangement and progress very often after class.Through working together, we learned a lot of new skills on paper reading, debugging and project planning. Collaboration greatly accelerated our learning steps.

References:

"Seungmin Jeon, Kwang Pyo Choi, Youngo Park, and Chang-Su Kim. Context-Based Trit-Plane Coding for Progressive Image Compression. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2023."