Framebuffer Compression Using Dynamic Color Palettes

Ayub Gubran and Tor M. Aamodt *
University of British Columbia

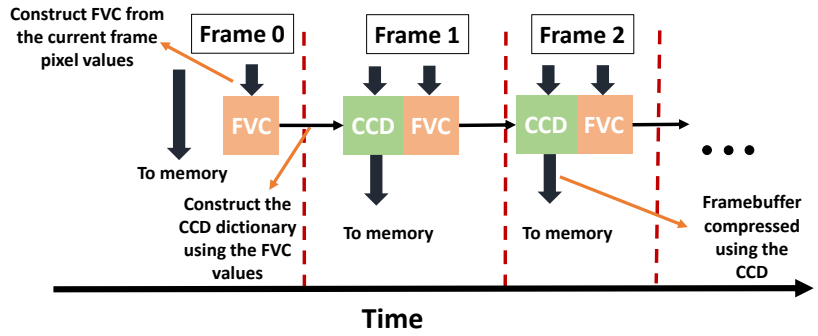

Figure 1: *DCP uses the color data in each frame to construct a compression palette for the next frame*

1 Introduction

Framebuffer traffic is a major source of memory bandwidth consumption. Each frame is pushed to memory and then accessed by the display controller to refresh the screen. These operations consume significant bandwidth. Reducing the cost of such bandwidth is especially important in the power and energy constraint mobile devices.

Hardware compression is utilized to reduce the cost of framebuffer operations. However, current framebuffer compression schemes are spatial schemes that use only local information in each frame–or part of it–to compress the framebuffer [Rasmusson et al. 2007; NVIDIA 2015].

In this work, we propose dynamic color palettes compression (DCP), an inter-frame compression scheme that exploits temporal coherence to predict compression parameters for each frame before rendering it. We found that our compression scheme performs particularly well with User Interface (UI) applications, which occupies about 70% of mobile devices usage time [Flurry Analytics 2013].

The key observation is that since consecutive frames are very similar in appearance, then we can obtain clues from one frame to compress the next frame. DCP compression uses the color data in each frame to build a dictionary structure (palette) that is used to compress the next frame.

2 Our Approach

As shown in Figure 1, starting from **Frame 0** we use the FVC (frequent values collector) to gather information on the most common colors in **Frame 0**. Then we use these most common color values in the FVC to construct the compression palette or CCD (Common Colors Dictionary) that is used to compress the next frame (**Frame 1**). While we use the CCD to compress **Frame 1**, we concurrently gather common color values using the FVC in order to construct a CCD for **Frame 2** and so on.

The FVC can be implemented as a small fully associative structure (e.g., 16-128 entries) that uses an eviction policy (similar to caches) that evicts the least common color frequencies. The CCD can be implemented as a lookup table that is similar in size to the FVC. The CCD holds translations from color values to their corresponding encoding.

Similar to other compression schemes, our scheme operates on frames in tiles of pixels. For each tile, each pixel is checked to see if it has an entry in the compression palette (CCD). If all the pixels in a tile have an entry in the CCD then the tile is compressed and sent to memory. Otherwise, the tile is sent to memory uncompressed. We use an auxiliary buffer to mark compressed and uncompressed tiles.

We also further optimize the encoding of the compressed color values by dynamically changing the palette size–and consequently the encoding size–in each frame based on the feedback from the previous frame.

We evaluated DCP compression using traces of framebuffer images from 13 mobile UI and 3D applications. Using a palette size of 64 we found that we achieve 57% and 65% higher compression rates in UI applications than other state-of-the-art compression schemes [NVIDIA 2015; Rasmusson et al. 2007]. However, DCP compression performs poorly with 3D applications, as color palettes are ineffective with frames that have a large variation of colors.

DCP compression can be implemented along other compression schemes that can handle the cases where the DCP compression performs poorly. In future work, we plan to evaluate using the DCP compression with G-buffers in 3D applications that use deferred shading (which is a popular technique in contemporary 3D games).

References

FLURRY ANALYTICS, 2013. Flurry Five-Year Report: Its an App World. The Web Just Lives in It. http://www.flurry.com/bid/95723/Flurry-Five-Year-Report-It-s-an-App-World-The-Web-Just-Lives-in-It.

NVIDIA, 2015. NVIDIA Tegra X1 Whitepaper.

RASMUSSON, J., HASSELGREN, J., AND AKENINE-MOLLER, T. 2007. Exact and error-bounded approximate color buffer compression and decompression. In *SIGGRAPH/EUROGRAPHICS Conference On Graphics Hardware: Proceedings of the 22 nd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*, vol. 4, 41–48.

* {ayoubg,aamodt}@ece.ubc.ca