Savio Alphonso
Michael Baart
Team 43

Colour Space Conversion Progress Report

Problem

Our Goal is convert a 640x480 image encoded with 10-bit RGB intensity values to Luminance/Chrominance (YCrCb) values with downsampling averaging the 4 pixels..

Approach

Our solution to the RGB to YCC colour space conversion problem we intend to take advantage of ARM's SIMD ISA Neon to accelerate the matrix and vector calculations. In addition we will also discuss alternative methods using software only methods. We will be testing our code and implementations on a Raspberry Pi 3B which contains a ARM Cortex-A53 (ARM v8-a) 64-bit processor

The first step in converting our image from a 10-bit RGB to YCrCb is to first translate our RGB values to YCC using the provided formula.

$$\begin{cases} Y' - 16 = +0.257R' + 0.504G' + 0.098B' \\ C_B - 128 = -0.148R' - 0.291G' + 0.439B' \\ C_R - 128 = +0.439R' - 0.368G' - 0.071B' \end{cases}$$

Floating-point coefficients will be replaced with their integer. Furthermore we will make use of the ARM's SIMD ISA NEON to accelerate the matrix calculations while incorporating some of the optimizations we have learned. NEON architecture has 128-bit registers. This allows us to process 8 RGB values at once (10-bit RGB stored as 16bits). Since we are processing 8 values simultaneously we can also consider optimizing with loop unrolling and software pipelining to ensure minimal/no true dependencies. Something else we have to consider is the industrial standard (ITU-R BT.601-4) specifies that Luminance and Chrominance are 8-bit values. So we must map our 10-bit RGB values to 8-bit. We also have to consider that we want to use saturated addition, which ARM seems to support with their SIMD ISA and the base ISA.

Savio Alphonso
Michael Baart
Team 43

The next step after our YCC conversion is to downsample our chrominance values. We will achieve this by taking a 2x2 area and calculate the average by adding the 4 values together and shifting the values by 4.

References:

1. https://developer.arm.com/architectures/instruction-sets/simd-isas/neon
2. https://developer.arm.com/architectures/instruction-sets/simd-isas/neon/neon-programmers-guide-for-armv8-a/optimizing-c-code-with-neon-intrinsics/optimizing-matrix-multiplication
3. https://static.docs.arm.com/ddi0596/a/DDI_0596_ARM_a64_instruction_set_architecture.pdf