## Temperature Colour Mapping

Based on the chosen RGB colour palette, it will first expand to the YUV colour table with self-defined number of elements in *Create\_color\_table()*. Then, it can map the temperature to corresponding YUV value by entry index. There are 2 methods for the colour mapping.

Now we have colour table with fixed 1200 elements.

**[0 ≤ Index < 1200]**

For adaptive colour mapping, , where 1199 is the index range of the table start from element 0, it is varied depends on the max & min value on each frame.

The Index is equal to the temperature value.

If maxT = 20°C, minT = 30°C in particular frame, and temp of the point of interest[i][j] = 25°C with RGB value {R0,G0,B0}, . Thus, the 599th colour code on the RGB colour table shown on the screen to represent 25°C. In alternative, adaptive mode is evenly distributing the whole colour palette to valid temperature on frame. Pixel[i][j] will be mapped to:

* RGB\_ColorTable[599] = {R0,G0,B0} <-> {Y0,U0,V0}

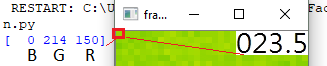
For non-adaptive colour mapping, the Index is equal to the temperature value right shifting 1 digit plus arbitrary colour offset (600). The shifting is applied in order to support 1 decimal value without using float data type. The offset is used for adjusting the pixel colour falling to certain range of palette.

If temperature of first pixel is 25.2°C, the index becomes 25.2\*10 + 600 = 852

* RGB\_ColorTable[852] = {R1,G1,B1} <-> {Y1,U1,V1}

According to the relationship of temperature data and corresponding YUV value, we can do reverse calculation on application software to get back temperature data.

Example:



We have a default non-adaptive colour palette with of 61 colours as follow:

RGB\_COLOR\_INFO\_T YUV\_ColorTable[1200] = {0};

RGB\_COLOR\_INFO\_T RGB\_ColorPalette[ColorPaletteSize]={

{176,0,240},

{142,0,240},

{108,0,240},

{65,0,240},

{0,0,235},

{0,0,218},

{0,0,201},g

{0,0,184},

{0,0,163},

{0,19,133},

{0,48,110},

{0,74,104},

{0,100,110},

{0,97,119},

{0,104,151},

{0,119,160},

{0,136,160},

{0,153,168},

{0,170,170},

{0,189,189},

{0,206,206},

{0,219,219},

{0,228,228},

{0,236,236},

{0,240,225},

{0,235,204},

{0,232,155},

{0,225,117},

{0,217,119},

{0,200,119},

{0,184,110},

{0,174,80},

{0,157,80},

{0,140,80},

{0,133,72},

{0,140,34},

{25,142,0},

{55,160,0},

{65,177,0},

{82,194,0},

{104,206,0},

{131,214,0},

{157,223,0},

{189,231,0},

{231,232,0},

{224,223,0},

{224,206,0},

{224,180,0},

{224,153,0},

{224,128,0},

{224,102,0},

{224,76,0},

{224,51,0},

{219,17,0},

{206,0,0},

{183,0,0},

{157,0,0},

{131,0,0},

{104,0,0},

{80,0,0},

{80,0,0},

};

The RGB\_ColorPalette is interpolated into to the YUV\_ColorTable domain to fit in 1200 elements in linear scale for UVC format, which means each entry there are 1200/60=20 more elements inserted, the interpolated R0 value = ( R[i+1]-R[i] ) / 20. It is batter to get raw YUV data and change the colour space to RGB by formula on Reference, then mapping to the temperature since YUV is what UVC transmitting.

\*Using open source library for conversion may have precision issue.

The interpolation size depends on the size of palette;

Where [0 <= i < sizeof(colorPalette) - 1] && [0 <= j < Interpolation size]

In this example, the first pixel temperature is mapping to the index 835:

{150, 214, 0} -> 835 (index) => 235 (temperature), which means 23.5 °C

## Reference

void Create\_color\_table(RGB\_COLOR\_INFO\_T RGB\_ColorPalette[],YUV\_COLOR\_INFO\_T YUV\_ColorTable[])

{

int i,j,interpolateSize;

unsigned char RVal,GVal,BVal, Color\_Y, Color\_U, Color\_V;

interpolateSize = COLORTABLESIZE/(COLORPALETTESIZE-1); //1200/60

for(i=0; i< (COLORPALETTESIZE-1); i++)

{

for(j=0;j<interpolateSize;j++)

{

// Here is expanded RGB Color Table

RVal = (unsigned char)((float)RGB\_ColorPalette[i].R + ((float)RGB\_ColorPalette[i+1].R - (float)RGB\_ColorPalette[i].R) /interpolateSize\*j);

GVal = (unsigned char)((float)RGB\_ColorPalette[i].G + ((float)RGB\_ColorPalette[i+1].G - (float)RGB\_ColorPalette[i].G) /interpolateSize\*j);

BVal = (unsigned char)((float)RGB\_ColorPalette[i].B + ((float)RGB\_ColorPalette[i+1].B - (float)RGB\_ColorPalette[i].B) /interpolateSize\*j);

// RGB2YUV

Color\_Y = (unsigned char)(RVal \* 0.299 + GVal \* 0.587 + BVal \* 0.114);

Color\_U = (unsigned char)(RVal \* (-0.169) - GVal \* 0.332 + BVal \* 0.500 + 128);

Color\_V = (unsigned char)(RVal \* 0.5 - GVal \* 0.419 - BVal \* 0.0813 + 128);

YUV\_ColorTable[i\*interpolateSize+j].YUVData = (Color\_V << 24) | ((Color\_Y << 16)) | (Color\_U << 8) | (Color\_Y);

}

}

}

Example on Python: <https://github.com/MeridianInno/MI_XCAM_rgb2temp_example>

XCAM Firmware SDK: <https://github.com/MeridianInnovation/MI_XCAM_formalRelease>