A Packed Planar RGB and YUV Format for Uncompressed Storage of High Dynamic Range Still Images and Videos

Christian R. Helmrich, Christian Lehmann, and Benjamin Bross

Video Coding and Analytics (VCA) Department
Fraunhofer Heinrich-Hertz-Institut (HHI)
Einsteinufer 37, 10587 Berlin, Germany

***Keywords*—10 bit; 12 bit; 4:2:0; 4:4:4; high dynamic range; planar; packing; RGB; still image coding; video coding; YCbCr; YUV.**

I. Introduction

DURING the development, evaluation, and maintenance of coding methods for the compression of still or moving pictures, it is frequently necessary to write, store, read, and archive uncompressed planar picture data as illustrated in Fig. 1, where each of the coded image components—typically red-green-blue (RGB) or luminance-chromatic (YUV)—are conveyed in non-interleaved picture-by-picture fashion. In case of high dynamic range (HDR) content, each picture component, called *plane* in the following, consumes more than 8 bit of capacity per image sample. For such word-lengths, only few format specifications have been published [1], and none of these cover the common use case of 4:2:0 YUV data with 10 or 12 bit per image sample applied in most image and video coding applications. The only common approach to convey uncompressed planar 4:2:0 YUV HDR image data is to write each 10 or 12-bit sample into a 16-bit word, which increases the required capacity by 33–60 %.

Therefore, we propose a *packed planar* 4:2:0 compatible uncompressed storage and transmission format, which we refer to as PRGB (for red-green-blue data) and PYUV (for luminance-chromatic content). This packed format can be specified for

- 10 bit per sample (PRGB10 and PYUV10) as in Section II,
- 12 bit per sample (PRGB12 and PYUV12) as in Section III,
- 14 bit per sample (PRGB14 and PYUV14) as in Section IV.

Note that, for 8-bit and 16-bit sample data, our format definition reduces to the conventional planar YUV specifications [2], [3]. It is also worth mentioning that the PYUV format supports 4:4:4 chromatic image content which is not spatially downsampled.

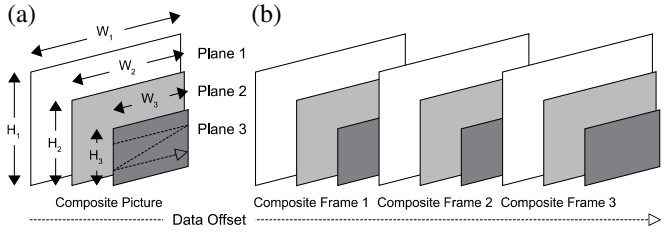

Fig. 1. Principle of planar picture data representation: (a) Still image, (b) video.

E-mail: christian.helmrich@hhi.fraunhofer.de. Doc. version 1, Aug. 17, 2018

II. Packed RGB or YUV Data With 10 Bit Per Sample

For a sample word-length of 10 bit, we propose to pack, for each image plane, four successive component samples (starting left-to-right with the first horizontal picture line and continuing vertically downward with the next picture line) into five bytes, i. e., into a 40-bit word as in Tab. 1. Thus, the number N of image samples in each plane must equal an integer multiple of four.

```
out[5*n]   = ( ( buf[4*n]           ) & 0xff  );
out[5*n+1] = ( ( buf[4*n+1] << 2 ) & 0xfc ) + ( ( buf[4*n]   >> 8 ) & 0x03 );
out[5*n+2] = ( ( buf[4*n+2] << 4 ) & 0xf0 ) + ( ( buf[4*n+1] >> 6 ) & 0x0f );
out[5*n+3] = ( ( buf[4*n+3] << 6 ) & 0xc0 ) + ( ( buf[4*n+2] >> 4 ) & 0x3f );
out[5*n+4] = ( ( buf[4*n+3] >> 2 ) & 0xff  );
```

List 1. Write-out of PRGB10 and PYUV10 data for each plane and $0 \le n < N/4$.

III. Packed RGB or YUV Data With 12 Bit Per Sample

For a sample word-length of 12 bit, we propose to pack, for each image plane, two successive component samples into three bytes, i. e., a 24-bit word as in Tab. 2. Hence, the number N of image samples per plane must equal an integer multiple of two.

```
out[3*n]   = ( ( buf[2*n]           ) & 0xff  );
out[3*n+1] = ( ( buf[2*n+1] << 4 ) & 0xf0 ) + ( ( buf[2*n] >> 8 ) & 0x0f );
out[3*n+2] = ( ( buf[2*n+1] >> 4 ) & 0xff  );
```

List 2. Write-out of PRGB12 and PYUV12 data for each plane and $0 \le n < N/2$.

IV. Packed RGB or YUV Data With 14 Bit Per Sample

For a sample word-length of 14 bit, we propose to pack, for each image plane, four successive samples into seven bytes, i.e., a 56-bit word. Thus, N must equal an integer multiple of four.

```
out[7*n]   = ( ( buf[4*n]           ) & 0xff  );
out[7*n+1] = ( ( buf[4*n+1] << 6 ) & 0xc0 ) + ( ( buf[4*n]   >> 8 ) & 0x3f );
out[7*n+2] = ( ( buf[4*n+1] >> 2 ) & 0xff  );
out[7*n+3] = ( ( buf[4*n+2] << 4 ) & 0xf0 ) + ( ( buf[4*n+1] >> 10 ) & 0x0f );
out[7*n+4] = ( ( buf[4*n+2] >> 4 ) & 0xff  );
out[7*n+5] = ( ( buf[4*n+3] << 2 ) & 0xfc ) + ( ( buf[4*n+2] >> 12 ) & 0x03 );
out[7*n+6] = ( ( buf[4*n+3] >> 6 ) & 0xff  );
```

List 3. Write-out of PRGB14 and PYUV14 data for each plane and $0 \le n < N/4$.

Sample	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
4*n	76543210	98			
4*n+1		543210	9876		
4*n+2			3210	987654	
4*n+3				10	98765432

Tab. 1. Layout of sample bits (0–9) in a 40-bit PRGB10/PYUV10 code-word.

Sample	Byte 1	Byte 2	Byte 3
2*n	76543210	ba98	
2*n+1		3210	ba987654

Tab. 2. Layout of sample bits (0–b) in a 24-bit PRGB12/PYUV12 code-word.

```
buf[2*n]   = ( ( in[3*n]   & 0xff ) ) + ( ( in[3*n+1] & 0x0f ) << 8 );
buf[2*n+1] = ( ( in[3*n+1] & 0xf0 ) >> 4 ) + ( ( in[3*n+2] & 0xff ) << 4 );
```

List 4. Read-in of PRGB12/PYUV12 data. Similar code for other bit-depths.

References

[1] FOURCC.org, "YUV formats – I420 YUV pixel format," 2011, online: https://www.fourcc.org/pixel-format/yuv-i420/

[2] G. Sullivan and S. Estrop, "Recommended 8-bit YUV Formats for Video Rendering," 2018, online: https://docs.microsoft.com/en-us/windows/desktop/medfound/recommended-8-bit-yuv-formats-for-video-rendering

[3] Microsoft Corp., "10-bit and 16-bit YUV Video Formats," 2018, online: https://docs.microsoft.com/en-us/windows/desktop/medfound/10-bit-and-16-bit-yuv-video-formats