Polynomial Texture Map (.ptm) File Format

Tom Malzbender
Dan Gelb

Hewlett-Packard Laboratories
11/01

Version 1.2

1.0 Background

Polynomial texture maps (PTM's) are an extension to conventional texture maps that allow enhanced image quality. As opposed to storing a color per pixel, these PTM's store second-order bi-quadratic polynomial coefficients per pixel. These polynomials model the changes that appear to a pixel color based on either light source position or viewing direction. For example, if Lu,Lv are the parameterized light source directions[1] and a_{0-5} the scaled and biased polynomial coefficients, a color channel intensity C are arrived at via:

$$C = a_0 L_u^2 + a_1 L_v^2 + a_2 L_u L_v + a_3 L_u + a_4 L_v + a_5$$

This document describes the file format used for representing PTM's. PTM files are designated by the suffix .ptm.

IMPORTANT NOTE: **The file assumes *little endian* format, meaning that the least significant bit is stored first. In order to run on a *big endian* machine the file reader has to be modified to assemble the integers in the correct order.**

2.0 File Format

A PTM file consists of the following 5 sections, each separated by a newline character (with an optional space before the newline):

2.1 **Header String**. The ASCII string 'PTM_1.2' appears on the first line of the file. This identifies the file as a PTM file and provides the PTM version number being supported.

2.2 **Format String**. One of the following ASCII strings appears on the next line identifying the format of the file:

[1] Parameterized lighting directions are arrived at by projecting the normalized light vector into the 2 dimensional texture space (u,v) to yield L_u,L_v.

PTM_FORMAT_RGB
PTM_FORMAT_LUM
PTM_FORMAT_LRGB
PTM_FORMAT_PTM_LUT
PTM_FORMAT_PTM_C_LUT
PTM_FORMAT_JPEG_RGB
PTM_FORMAT_JPEG_LRGB
PTM_FORMAT_JPEGLS_RGB
PTM_FORMAT_JPEGLS_LRGB

2.3 **Image Size**. The next line consists of an ASCII string containing the width and height of the PTM map in pixels. It is legal to have a newline separating the width and the height.

2.4 **Scale and Bias**. The PTM coefficients are stored in the file as a single byte per coefficient. Since this limits the coefficients to be in the range of [0,1], scale and bias values are provided that can map these values to arbitrary values. A total of 6 bias and 6 scale values are provided, one for each of the 6 polynomial coefficients. The six ASCII floating point scale values appear in the file first, follow by the six ASCII integer bias values, all separated by spaces. A newline can be used between the scale values and the bias values. It is legal to have a space character before the newlines. The final coefficient values are arrived at by:

$$\mathbf{C_{final} = (C_{raw} - bias) * scale}$$

2.5 **Lookup Table.** This is only present for formats that use a lookup table, namely PTM_FORMAT_PTM_LUT and PTM_FORMAT_PTM_C_LUT. For PTM_FORMAT_PTM_LUT. First a specification of the number of entries in the lookup table is given, where nentries is an integer:

nentries N_LUT_ENTRIES

This is followed by the lookup table itself which is format dependent. Each of the nentries of the lookup table are specified as a sequence of unsigned chars with no characters between entries. For the PTM_FORMAT_PTM_LUT each entry contains six PTM coefficients for a Luminance channel:

$$\mathbf{a_0^I, a_1^I, a_2^I, a_3^I, a_4^I, a_5^I}$$

In this case the color values which are modulated by the evaluated luminance polynomial are passed for each pixel later in the file.

For PTM_FORMAT_PTM_C_LUT both PTM coefficients and color reside in the lookup table and are specified for each entry as

$$a_0^I, a_1^I, a_2^I, a_3^I, a_4^I, a_5^I, R, G, B$$

2.6 **Color Matrix.** This is only present for PTM_FORMAT_LUM. The matrix is stored as16 ASCII floating point values in the order such that it can be loaded directly into OpenGL functions- this means in column-major order

$$\begin{matrix} a_{00} & a_{10} & a_{20} & T_x \\ a_{01} & a_{11} & a_{21} & T_y \\ a_{02} & a_{12} & a_{22} & T_z \\ 0 & 0 & 0 & 1 \end{matrix}$$

Note that this is the matrix by which the colors computed from the coefficients have to be multiplied. The T_x, T_y and T_z values can be used as translations in the color space.

2.7 **Uncompressed Per Textel Data**. When an uncompressed file format is used, the rest of the file consists of information for each textel, typically the raw polynomial coefficients themselves, specified as a single unsigned char per coefficient. The arrangement of the coefficients is format dependent, and taken from one of the following depending on the format string provided earlier in the file:

2.7.1 **PTM_FORMAT_RGB -** In this case, an array of coefficients are stored separately for each color channel. These are provided in the following order for a single texel channel:

$$a_0, a_1, a_2, a_3, a_4, a_5$$

All red coefficients are stored in a block, followed by the green channel coefficients, followed by the blue channel.

Coefficients are provided for all textels in this manner in simple reversed scanline order (meaning from bottom to top). Note that no separators such as <cr> are used between the individual coefficients nor between the pixels or blocks.

2.7.2 **PTM_FORMAT_LUM-** Here we store only six polynomial coefficients followed by two values for *Cr* and *Cb* which then make up the *CrYCb* color space, where *Y* is computed from the polynomial coefficients. Hence

$$a_0^Y, a_1^Y, a_2^Y, a_3^Y, a_4^Y, a_5^Y, C_r, C_b$$

Again, coefficients are provided for all textels in this manner in simple reversed scanline order (meaning from bottom to top). Note

that no separators such as <cr> are used between the individual coefficients nor between the pixels.

2.7.3 **PTM_FORMAT_LRGB -** Here we store only six polynomial coefficients representing Luminance (normalized usually) followed by three values for R,G,B which then make up LRGB for color calculation. The coefficients are stored per texel in a block as:

$$\mathbf{a_0^I, a_1^I, a_2^I, a_3^I, a_4^I, a_5^I}$$

Followed by the RGB color values in a block ordered as:

$$\mathbf{R, G, B}$$

Again, coefficients are provided for all textels in this manner in simple reversed scanline order (meaning from bottom to top). Note that no separators such as <cr> are used between the individual coefficients nor between the pixels.

2.8 **Lookup Table Indicies.** For formats that use a lookup table, one of the following is present:

2.8.1 **PTM_FORMAT_PTM_LUT –** Here we store a lookup table index for each textel. For a lookup table with 256 entries or less this is an unsigned char per textel. For larger lookup tables this is two bytes in little endian format.

$$\mathbf{i_1, i_2, i_3, \ldots.}$$

After these indicies are specified for all textels, color information for each textel is provided as unsigned char's per element:

$$\mathbf{R_1, G_1, B_1, R_2, G_2, B_2, R_3, G_3, B_3, \ldots}$$

2.8.2 **PTM_FORMAT_PTM_C_LUT** - Here we store a lookup table index for each textel. For a lookup table with 256 entries or less this is an unsigned char per textel. For larger lookup tables this is two bytes in little endian format.

$$\mathbf{i_1, i_2, i_3, \ldots.}$$

For both of these cases, values are again provided for all textels in this manner in simple reversed scanline order (meaning from bottom to top).

2.9 **Compressed Data.** When the file format is one of:
PTM_FORMAT_JPEG_RGB,PTM_FORMAT_JPEG_LRGB,
PTM_FORMAT_JPEGLS_RGB,PTM_FORMAT_JPEGLS_LRGB

the file contains the following:

2.9.1 **Compression Parameter**. It consists of an ASCII string that contains the parameter being fed to the JPEG-LS or to the JPEG encoder. When the file is encoded with JPEG-LS, the parameter represents the lossless mode (if zero) or the maximum absolute value of the loss for each pixel (if greater than zero). For JPEG, the parameter represents an encoding quality factor ranging between 20 and 100 (best quality).

2.9.2 **Transforms**. It is a sequence of 18 (for RGB PTM's) or 9 (for LRGB PTM's) ASCII integers. The (*i*+1)-st integer represents the transforms that must be applied to the reference plane (see section 9) before prediction of the coefficient plane indexed by *i*. Each transform is represented by a constant value that is a power of 2 (see table below), so in order to specify multiple transforms, constants can simply be OR-ed together to form a single integer. The following transforms are currently implemented:

Transform	**Constant Name**	**Integer Value**
No transform	NOTHING	0
Plane Inversion	PLANE_INVERSION	1
Motion Comp.	MOTION_COMPENSATION	2

2.9.3 **Motion Vectors**. It is a sequence of 36 (for RGB PTM's) or 18 (for LRGB PTM's) ASCII signed integers. The first half represents the *x* coordinates and the second half the *y* coordinates of the 18 (or 9) motion vectors. Since integers are used to represent half pixel displacements, these values must be divided by 2 in order to obtain the final displacements along the *x* and *y* dimensions.

2.9.4 **Order**. It is a sequence of 18 (for RGB PTM's) or 9 (for LRGB PTM's) ASCII integers that represent the order in which the corresponding coefficient plane must be decoded. This order guarantees causality in the decoding process. If plane *i* is predicted from plane *j* (called *reference* for *i*), order of *j* will be smaller than order of *i* and decoding of *j* must precede decoding of *i*. To start the decoding process, there is always (at least) a plane that is not predicted from any other plane and whose order is 0.

2.9.5 **Reference Planes**. A sequence of 18 (for RGB PTM's) or 9 (for LRGB PTM's) ASCII integers that represent the index of the reference plane used for encoding the coefficient plane *i*. Coefficient planes are indexed starting from zero: If plane *i* is predicted from plane *j*, then the (*i*+1)-st integer in the sequence is *j*. A special reference index "–1" is used to indicate a plane that is intra coded (i.e., that it is not predicted from any other plane).

2.9.6 **Compressed Size**. A sequence of 18 (for RGB PTM's) or 9 (for LRGB PTM's) ASCII integers in which the (*i*+1)-st integer is the size, in bytes, of the *i*-th compressed coefficient plane. Since coefficients are not interleaved and compressed planes may have different sizes, this information (and the side information size, see below) must be combined to extract properly the compressed planes from the "Compressed Coefficient Planes" section.

2.9.7 **Side Information Size**. A sequence of 18 (for RGB PTM's) or 9 (for LRGB PTM's) ASCII integers in which the (*i*+1)-st integer represents the size (in bytes) of the side information used to correct possible overflows occurring during the (lossy) encoding of the *i*-th coefficient plane (see below). If no overflow occurred during the encoding of the *i*-th plane, the corresponding side information size will be zero.

2.9.8 **Compressed Coefficient Planes**. Compressed coefficient planes are stored plane by plane, in sequence, following the original plane ordering. For RGB format the plane ordering is:

$$\mathbf{a_0^r, a_1^r, a_2^r, a_3^r, a_4^r, a_5^r,}$$
$$\mathbf{a_0^g, a_1^g, a_2^g, a_3^g, a_4^g, a_5^g,}$$
$$\mathbf{a_0^b, a_1^b, a_2^b, a_3^b, a_4^b, a_5^b}$$

For LRGB format the plane ordering is:

$$\mathbf{a_0^I, a_1^I, a_2^I, a_3^I, a_4^I, a_5^I, r,g,b}$$

Coefficients are not interleaved and each plane must be extracted by using the information provided in the sections "Compressed Size" and "Side Information Size." Each compressed plane is stored according to the bit-stream format corresponding to the compression algorithm used (JPEG or JPEG-LS). When a lossy mode is used, each compressed plane is followed by a sequence of zero or more bytes representing the side information necessary to correct overflows resulting from modular arithmetic. This "**Side Information**" section consists of a sequence of pairs (Pixel Position, Pixel Value), represented with five consecutive bytes as follows:

(Pixel Position, Pixel Value) = P_3, P_2, P_1, P_0, V.

Pixel position is a 4-byte integer (with the highest order byte stored first), which represents a pixel position when the image is linearized in row order scan, top to bottom, left to right. The pixel value V is the original pixel value that must be substituted in the decoded plane in that position in order to fix the overflow. Overflows must be corrected before using the decoded plane as a prediction reference.