IBM VIDEO TECHNOLOGY

Ultimotion Digital Video
Data Stream Specification

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page iii.

Notices

IBM authorizes you to use, under copyright, the Specification in creating your hardware or software product as long as the data bit stream implemented in your product is identical to the data bit stream defined in the Specification. You may only use the Specification in accordance with the condition above.

In addition, you must include the following notice with each sent copy of your hardware and/or software product: **Copyright International Business Machines Corporation 1994. All rights reserved. This product uses Ultimotion(tm) IBM video technology.**

If you are using Ultimotion in a hardware and/or software product packaged for retail distribution, contact IBM Intellectual Property Law Department at 951 NW 51st Street, Internal Zip 4318, Boca Raton, FL 33431 to inquire about an Ultimotion(tm) logo license agreement.

THE LICENSED WORK IS PROVIDED "AS IS." IBM MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESSED OR IMPLIED, WITH RESPECT TO THE LICENSED WORK INCLUDING THE IMPLIED WARRANTY OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK ARISING OUT OF USE OR PERFORMANCE OF THE LICENSED WORK REMAINS WITH YOU.

IN NO EVENT WILL IBM BE LIABLE FOR ANY LOST PROFITS, LOST SAVINGS, INCIDENTAL DAMAGES OR OTHER ECONOMIC OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ADDITION, IBM WILL NOT BE LIABLE FOR ANY DAMAGES CLAIMED BY YOU BASED ON ANY THIRD PARTY CLAIM.

Aside from the limited license granted above, you have no other rights, expressed or implied, to any other intellectual property of IBM, including patents, copyrights, trademarks, and/or trade secrets.

Trademarks and Service Marks

The following term, denoted by an asterisk (*) in this publication, is a trademark of the IBM Corporation in the United States or other countries:

Ultimotion

Introduction

Advances in microprocessor power, data storage, and compression technology have provided key technologies for creating and playing digital video data on personal computers. The high capacity disk drives and CD-ROMs satisfy the large storage needs of digital video data. Additionally, today's more powerful microprocessors provide sufficient power to handle digital video data in real time. When these advances are combined with image compression techniques, the result is a powerful integration of video and the personal computer.

Several compression algorithms are currently in use throughout the industry. Some of these algorithms are numerically intensive and use additional video hardware to compress and decompress the digitized video. Others are less numerically intensive and can be handled by software running on the main CPU and still maintain sufficient frame rates to provide motion. These are referred to as *software-only* algorithms. While software-only techniques are attractive due to their low cost, the video quality of these algorithms is typically lower than the hardware-based algorithms. Consequently, producers of digital video data struggle with trading off lower quality and cost of software-only techniques for the higher quality and cost of hardware-assisted video. Ultimotion* provides a motion video compression technique capable of producing a high level of quality in a software-only playback environment.

* Trademark of the IBM Corporation.

Overview of Ultimotion Video Compression

Computational Complexity

Decompression (playback) of Ultimotion video is possible with very low computational complexity to enable real-time playback in a software-only environment. The "nominal Ultimotion movie" is defined as 320 x 240, 15 frames per second at a 150KB per second date rate. This movie can be played on at least a 25 MHz 80386 microprocessor. Systems with faster microprocessors are capable of higher frame rates and larger output image sizes.

Compression (recording) of Ultimotion video can be performed with a range of computational complexity. In a software-only environment using frame-grabber hardware, Ultimotion video can be compressed either in real-time or asymmetrically in conjunction with frame-step recording.

Ultimotion real-time compression provides instant video turn-around at a very low cost. The most common quality type can be used to produce 160 x 120 videos at 15 frames per second with a 25 MHz 80486 microprocessor. A larger frame size can be used to produce 320 x 240 videos at 8 frames per second. The quality level is selectable to attain different data rates. Slower systems may also be used for real time capture or record, but the frame rates will be lower.

Asymmetric compression provides the most compression and quality, but must be used in conjunction with frame-step capture, or raw input source files. With a 25 MHz 80486 microprocessor, 320 x 240 frames can be compressed in one to four seconds. Specialized capture or compression hardware will enable real-time capture with the same quality level now attained through asymmetric compression.

Scalability

Ultimotion is a playback-time scalable video data stream. While the frame rate and output image size (resolution) are set when the video is created, the characteristics of a playback scalable video are modified during playback according to the capabilities of the playback platform. These playback-platform capabilities depend on the type of microprocessor, data bandwidth, display driver, and video adapter available during playback. Based on the characteristics of these components, the output quality of a single Ultimotion video stream can be "scaled" to the playback platform.

Processor Independence

In deriving the Ultimotion data stream, care was taken to balance the requirements of software-only and hardware-assisted decompression. The resulting data stream:

Uses byte-oriented data organization and simple integer arithmetic (that is, no multiplication or division) so that non-computationally intensive decompression algorithms can be used.

Organizes the data for efficient processing of unwanted data when being scaled during playback.

Data structures are organized for hardware-assisted techniques wherever feasible but do not utilize any particular processor instruction capability.

Playback Characteristics

While Ultimotion is a runtime scalable data stream, the magnitude that the data stream will "scale" is determined by the amount of information that is encoded in the data stream when it was created. That is, the amount of data placed in the stream at creation time determines the "maximum" playback characteristics that a particular stream can achieve. In turn, the processing capabilities of the playback system determine how much of the data can be processed and presented during playback. Therefore, the playback characteristics of a given video are a function of the data put into the video by the author *and* the processing capabilities of the playback system.

The factors affecting the data stream at creation time are:

Resolution - width and height of video
Frame duration - frequency at which frames are to be displayed
I-frame rate - frequency at which reference frames are to occur
Data rate - average amount of data allowed for an interval of video

Factors affecting the playback of a video are:

Processing power of the CPU
Throughput of data storage (for example, CD-ROM, hard disk, and LAN)
Efficiency of the display subsystem

Resolution Scalability

Resolution of a video determines how much spatial information is in a video image, as well as the "normal" playback size. Ultimotion video compression enables the output image size to be efficiently scaled up or down as it is decompressed, based

on user preference or application requirements. Scaling down the output image size enables more efficient playback on slower systems.

Frame Rate Scalability

Frame duration for the frames in a video is usually referred to as frame rate. A frame compressed without regard to a prior frame uses intraframe compression and is called an *I-frame.* A frame compressed by considering only what has changed from a prior frame uses temporal compression and is called a *delta frame.* I-frames are a snapshot in time of the video. Delta frames show only what changed. The I-frame rate of a video is the frequency that I-frames occur in the data stream.

The I-frame rate affects several characteristics of a video stream:

- Data rate
- Quality of motion during scanning
- Performance of seek operations
- Ability to maintain smooth motion when frames are dropped to maintain audio synchronization
- Image re-paint time in a windowing system where no backup saving is used during clipping

Since I-frames contain the compressed data of the entire image, they are typically larger than delta frames. Ultimotion data streams that mix I-frames and delta frames:

- Reduce throughput requirements with smaller delta frames
- Provide points in the stream at which the entire image exists (good for seeking and scanning)
- Provide points in the video stream at which artifacts are repaired (with I-frames)

Generally, an I-frame rate of one I-frame every one or two seconds provides the best results.

Color Scalability

Ultimotion compression algorithms store images in a YUV direct color space. This color space is mapped at playback time to the number of colors available on the playback system. This avoids the need to have different versions of video content to fully exploit varying hardware capabilities.

Playback Platform Considerations

Direct Output

When decompressed, the Ultimotion data stream can produce either partial or complete video frames. The technique chosen to display these frames varies from platform to platform and largely depends on the display subsystem of the platform. Some systems allow a decompressor to access the display buffer directly where the decompressor may only update changed information. Others require changed information in full frames. The resulting performance characteristics of playback on various platforms is dependent on the efficiency of the display subsystem.

Window Clipping

In support of direct output in a windowing environment, Ultimotion decompression is well-suited for enabling clipping in the decompressor.

How Ultimotion Video Compression Works

The data stream combines several techniques to provide both temporal and spatial compression. These techniques include the following, which will be discussed in more detail in this section:

Variable chrominance subsampling
Luminance transition coding
Statistical luminance image coding

Interframe versus Intraframe Compression

While the data stream supports both interframe and intraframe compression, no specific distinction is made between frame types. An intraframe compressed frame is one that does not contain any unchanged data. This does not preclude the file format from making such a distinction, as in the Audio/Video Interleaved (AVI) file format. Capture or compression software generally enables periodic insertion of intraframe compressed video frames.

Format of Encoded Video Frame Data

During video compression, each video frame is decomposed into non-overlapping square *blocks*, nominally of 8 x 8 pixels. Blocks in turn are decomposed into *quadrants*, nominally of 4 x 4 pixels. The quadrants in a block are considered to be ordered in a counter-clockwise direction, beginning with the upper-left quadrant.

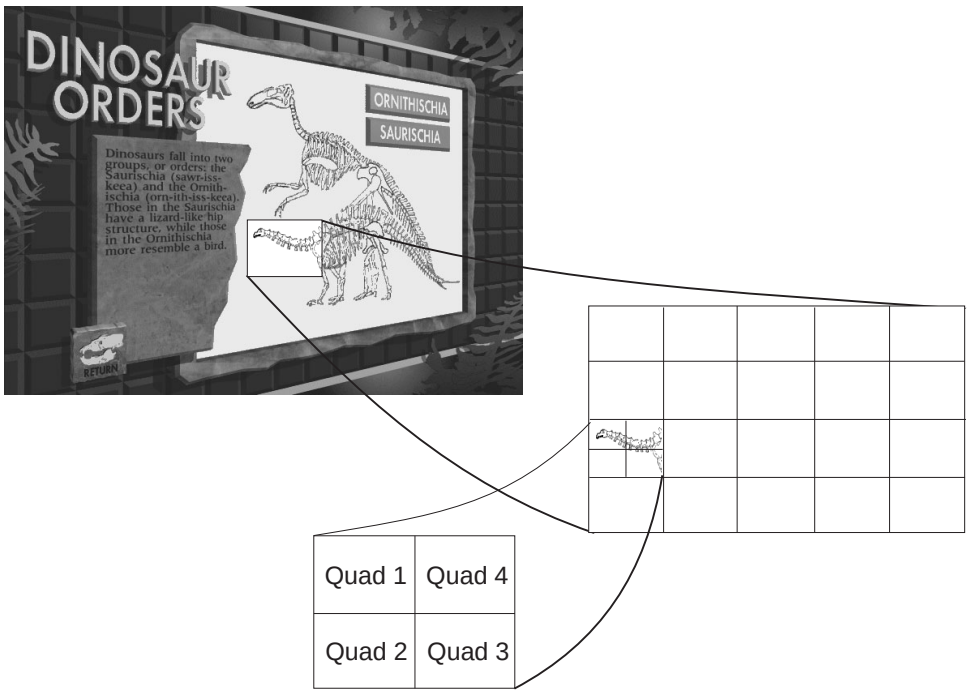

Figure 1. Decomposition of Blocks into Quadrants

Blocks

The format of the encoded video frame data is the sequence of all blocks in the frame, each encoded as shown in Figure 2 on page 3, and ordered in a raster scan of the grid of blocks in the video frame. The data for a block begins with a one-byte *block header* that defines the interpretation of the data that follows for the block and the quadrants contained in the block.

While the block size is variable on output, the preferred output block size is 8 x 8 pixels, which results in a direct mapping of some encoding types (statistical). Figure 2 on page 3 shows the format for each encoded block used to represent the decomposed frame.

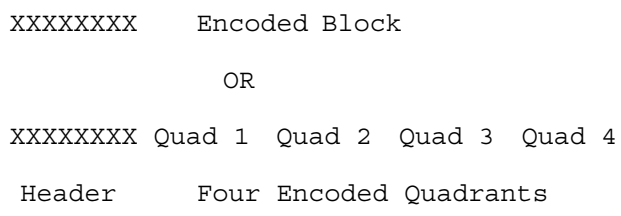

Figure 2. Format of an Encoded Block Decomposed into Quadrants

Color Space

The data stream uses a direct color space; that is, no in-stream palettes or lookup tables are used. The color space used provides 6 bits of luminance and 8 bits of chrominance (6Y 4U 4V). The encoding of chrominance is performed by quantizing a portion of the dynamic range of the U and V components. The RGB conversion is performed using the following equations, which assume an RGB 5-6-5 representation:

```
Y  = (( .299R +  .2935G +  .114B) x 2. 37) +  .5
U  = (((- .169R -  .163G +  .5  B) x 8.226)/6.375) + 5
V  = ((( .5  R -  .2 6G -  . 81B) x 8.226)/7.5  ) + 6
```

The value of Y is restricted to 0 through 63 and the values of U and V are restricted to 0 through 15.

Variable Chrominance Subsampling

The chrominance value for each quadrant is determined by taking a weighted average of the chrominance values of the pixels in the quadrant. The chrominance values for each quadrant in the block are then compared with the chrominance values for other quadrants in the block, and if the values are within a threshold *t* of each other, the chrominance of the entire block is considered to be perceptually similar, and a single chrominance value is used for the entire block. This results in this block of the image being sampled at a low frequency. If, on the other hand, the chrominance values for the quadrants in the block differ by more than *t*, the block is considered to contain perceptually dissimilar colors, and a unique chrominance value is used for each quadrant in the block. This results in this block of the image being sampled at a high frequency.

Chrominance Modes

The data stream uses a variable chrominance subsampling mechanism. Chrominance data appears in the data stream with a varying frequency depending on the *chrominance mode*:

Normal chrominance mode

This is the default mode. A single chrominance value (one byte) immediately follows the block header (except in the case of a completely unchanged block) and that chrominance value is used for the entire block.

Unique chrominance mode

This mode is entered whenever chrominance subsampling fails; that is, using a single chrominance value for an entire block results in "chrominance bleeding" artifacts. In this mode, the block header is not followed by a chrominance value. Instead, each quadrant encoding includes a unique chrominance value.

Single unique chrominance

Single unique chrominance is signaled by an escape value and indicates that a unique chrominance mode is used for the next block only. See “Escape Values” on page 13.

Luminance Transition Coding

Luminance transition coding (LTC) is based on the reduction of variation in luminance of a region of an image to a single direction. Each region of the image is analyzed to correlate individual luminance values and determine the direction in which the maximum luminance change is occurring. Sample values are then determined by taking the average number of pixels perpendicular to this direction. The luminance values within the region can then be mapped to one of a predetermined set of luminance values based on various functions of increasing luminance. These functions approximate commonly occurring luminance variations in natural images.

The resultant encoding of the region is then compared with the uncompressed image data for the region. The error (in luminance space) is calculated for use in comparing with other encoding types. The selection of an encoding type is based on the following two considerations: (1) comparison of the errors of the different types (where the error as defined above is inversely correlated to the quality of the representation) and (2) the amount of data each type consumes as allowed by the amount of compression to be obtained.

One-Byte Luminance Transition Region Encoding

The data for a one-byte encoding consists of a single luminance value and 2 bits that indicate a direction of increasing luminance of one step, if any. The one-byte encoding is useful because regions that have the same luminance value (known as homogeneous regions), or only a very small change in luminance, are common and should be encoded with as little data as possible.

Two-Byte Luminance Transition Region Encoding

The data for a two-byte encoding consists of the direction of *increasing* luminance and a transition index value that specifies a base Y value, a delta Y value, and a transition function. The transition index is the index into a table which defines the common set of four byte-transitions. The derivation of this index value is described later.

Four-Byte Luminance Transition Region Encoding

The data for a four-byte encoding consists of a direction of luminance change and four luminance sample values along the specified direction. The four-byte encoding is useful for regions in which the changes in luminance cannot be accurately mapped to the set of predetermined luminance values used in the two-byte encoding. Note that the specified direction (angle) does not include the opposite directions (± 180 degrees) because the luminance values need not be increasing (a requirement of the transition index). Thus the opposite directions are implied by the use of the four luminance values.

Format of Block Header

The encoding for a block begins with a *block header* specifying how the four quadrants it represents have been encoded. In addition to specifying these encoding types, certain values of the block header are reserved as escape values that contain control information about the interpretation of the data stream. The possibilities for processing a block as specified by the block header fall into two basic categories:

Quadrant encodings

Quadrant encodings specify that the block is comprised of four quadrants, NW, SW, SE, and NE. Each quadrant in turn is separately encoded with a quadrant encoding type. Quadrants in a block are encoded independently, and may use different quadrant encoding types, with some restrictions which are described later.

Block header escape

Rather than indicate the encoding type for a block and its respective quadrants, certain values are reserved as escape values to enable special processing of the data stream.

A block encoding is encoded using a one-byte block header followed by an encoding type. Figure 3 shows the format for a block encoding.

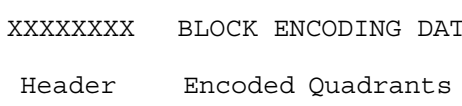

Figure 3. Format of a block encoding

A quadrant encoding block uses the two left-most bits of the block header to specify how the first quadrant in the stream was encoded. Similarly, the next two left-most bits of the block header specify how the next quadrant was encoded, and so on. Figure 4 shows the format for a quadrant block.

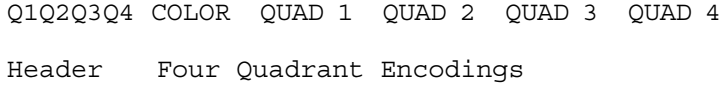

Figure 4. Format of Quadrant Decomposition Block

The specification of the block header bits depends on the current stream interpretation mode. This mode determines which set of four encoding types is used when a block is decomposed into quadrants. The stream interpretation mode can be switched anywhere within a frame. The stream interpretation mode defaults to mode 0 at the start of each frame.

The following combination of quadrant types are defined for mode 0 stream interpretation:

00: Unchanged quadrant
01: Homogeneous/shallow LTC quadrant
10: LTC quadrant
11: Statistical/extended LTC quadrant

The following combination of quadrant types are defined for mode 1 stream interpretation:

00: Unchanged quadrant
01: Homogenous quadrant
10: Subsampled 4-luminance quadrant
11: 16-luminance quadrant

Quadrant Encoding Types

Encoding types used when a block is comprised into quadrants are described here. All the quadrants of a block share a single 8-bit chrominance value unless unique chrominance mode is being used, in which case each of the quadrant encodings in a block are preceded by a unique chrominance value.

Unchanged Quadrant

The data for the quadrant is unchanged from the previous frame and no further data is encoded. Note that a block with four unchanged quadrants is the same as an unchanged block which contains no chrominance value. Figure 5 shows how the encoded data is organized for a block containing one changed quadrant and three unchanged quadrants.

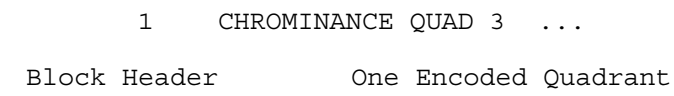

Figure 5. Format of block with one changed quadrant and three unchanged quadrants

Decoding Tips

To decode an unchanged quadrant:

1. Move pointers in the output data to the next quadrant of the block.
2. Continue processing the next two bits of the block header.

Compression attained with this quadrant encoding is 0.25 bits per pixel, assuming normal chrominance mode, including 2 bits chrominance and 2 bits for the block header, representing a 1/4 share of each for a quadrant. A completely unchanged block (4 unchanged quadrants) will attain a compression level of 0.125 bits per pixel, as no chrominance information is encoded for the block.

Luminance Transition Coding Quadrant

The data for the quadrant is described as an angle of direction of increasing luminance and an index indicating the luminance sample values along that direction. The delta Y values are distributed to provide lower quantization in shallow gradients. Figure 6 on page 8 shows how the encoded data is organized for an LTC quadrant.

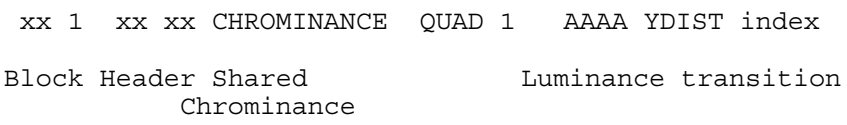

Figure 6. Format of an LTC quadrant

The quadrant is encoded as follows:

Four bits representing angle of direction of increasing luminance (0 indicates increasing luminance left to right — 0 degrees — with subsequent angles increasing in a counter-clockwise direction at increments of 22.5 degrees).

4 bits	**Angle of Direction**
0000	0 degrees
0001	22.5 degrees
0010	45 degrees
⋮	
1111	337.5 (-22.5) degrees

12 bits luminance transition index

Decoding Tips

To decode this quadrant:

1. Determine the luminance values to use for the quadrant by performing a table lookup.
2. Interpret the direction bits and reverse luminance values if angle value is greater than 7.
3. Using the shared chrominance for the block plus the 6 bits of the first luminance, convert the resulting color to the target color space.
4. Using the direction bits as a vector, store the color in the appropriate positions of the quadrant. This can be done with eight output routines that map the luminance values into patterns for the first eight angles. The remaining eight angles are the opposite directions from the first eight, and are covered by reversing the luminance values.
5. Convert the remaining Y values and store in appropriate positions in the quadrant.

Compression attained with this quadrant encoding is 1.25 bits per pixel, assuming normal chrominance mode, including 2 bits chrominance and 2 bits for the block header, representing a 1/4 share of each for a quadrant.

Statistical Pattern/Extended LTC Quadrant

The first bit of the encoding determines whether the data is a statistical pattern or an extended LTC encoding. If the first bit is 0, the data for the quadrant is described as two luminance values Y1 and Y2, and a pattern specifying the distribution of the two luminance values among the pixels in the quadrant. Figure 7 on page 9 shows how the encoded data is organized for a statistical quadrant.

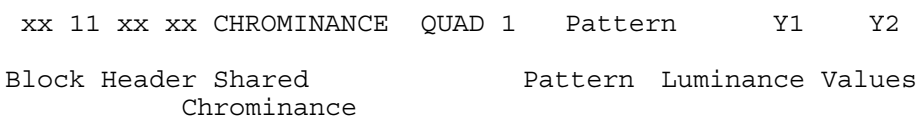

Figure 7. Format of a Statistical quadrant

The quadrant is encoded as follows:

16 bits of pattern in raster scan order (MSB = 0)
2 bits undefined (decoder must mask)
6 bits Y1
2 bits undefined (decoder must mask)
6 bits Y2

Decoding Tips

To decode this quadrant:

1. Convert the two luminance values to the output color space using the chrominance for the block or the quadrant, as defined by the current chrominance mode.
2. Output the two colors into pixels in the quadrant as defined by the pattern. The color obtained by converting Y1 is output wherever the corresponding pattern bit is 0, and the color obtained by converting Y2 is output wherever the corresponding pattern bit is 1.

Figure 8 shows how the encoded data is organized for an Extended LTC quadrant.

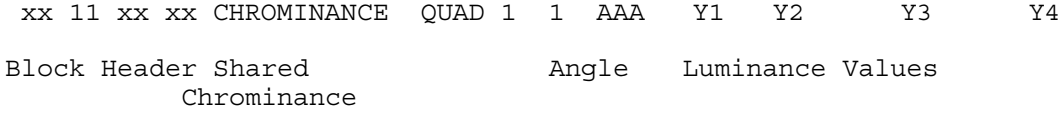

Figure 8. Format of an Extended LTC Quadrant

The quadrant is encoded as follows:

1 bit = 1

3 bits angle 0-157.5 degrees (000 = 0 degrees, 22.5 degree increments)

6 bits Y1

6 bits Y2

2 bits undefined (decoder must mask)

6 bits Y3

2 bits undefined (decoder must mask)

6 bits Y4

Decoding Tips

To decode this quadrant:

1. Shift the luminance values into the same positions that are obtained by the luminance transition index lookup.
2. Enter the appropriate output routine in the LTC quadrant processing based on the angle specified. (Note that only eight angles are encoded. Because individual Y values are uniquely specified, each angle + 180 degrees is implied by reversing the ordering.)

Format of Homogeneous Quadrant/Shallow Gradient Quadrant

An encoded *homogeneous/shallow gradient quadrant* is specified by its corresponding two bits in the block header and a luminance value. Figure 9 shows how the encoded data is organized for a decomposed block whose second quadrant is a homogenous/shallow gradient.

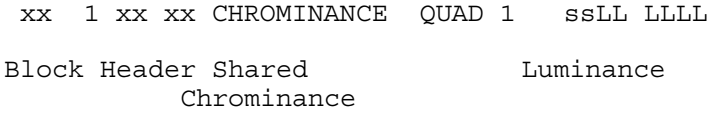

Figure 9. Format of a block with a homogenous/shallow gradient quadrant

The quadrant is encoded as follows:

2 bits representing Y delta and angle of direction of increasing luminance (00 indicates constant luminance, and values 01, 10, and 11 specify luminance increasing one level across the quadrant in the specified direction).

2 bits	Y delta, Angle of Direction
00	0, N/A
01	1, 45 degrees
10	1, 135 degrees
11	1, 270 degrees

Six bits Y

Decoding Tips

To decode this quadrant:

1. Using the shared chrominance value for the block, use the luminance value Y and convert the YUV color to the target color space.
2. If angle/delta is 0, store the target color in the entire quadrant.
3. If angle/delta is not 0, perform a second color space conversion using Y+1 as the luminance value, and output the two colors with an error diffused density pattern in the direction specified.
4. Increment output pointers to the next quadrant in the block.
5. Move to the next quadrant in the encoded stream.

Compression attained with this quadrant encoding is 2.25 bits per pixel, assuming normal chrominance mode, including 2 bits chrominance and 2 bits for the block header, representing a 1/4 share of each for a quadrant.

Subsampled 4-Luminance Quadrant

An encoded *subsampled 4-luminance quadrant* is specified by its corresponding two bits in the block header and four luminance values. When decoded to an 8 x 8 block size, this quadrant encoding type results in one color value per 2 x 2 pixel region in the quadrant. Figure 10 shows how the encoded data is organized for a block whose second quadrant is a Subsampled 4-luminance quadrant.

```
 xx 1  xx xx CHROMINANCE  QUAD 1   1111 1122   2222 3333   3344 4444

Block Header Shared             Four Luminance Values
           Chrominance
```

Figure 10. Format of Block with Sub-4 Quadrant

The quadrant is encoded as follows:

6 bits Y1

6 bits Y2

6 bits Y3

6 bits Y4

Decoding Tips

To decode this quadrant:

1. Using the shared chrominance value in this quadrant and the first luminance value (Y1), convert the YUV color to the target color space.
2. Store the output color in the corresponding 2 x 2 area of the quadrant.
3. Move output pointers to the next 2 x 2 of the quadrant in the order NW, NE, SW, SE.
4. Repeat the last 3 steps for each luminance in the encoded stream.

Compression attained with this quadrant encoding is 1.75 bits per pixel, assuming normal chrominance mode, including 2 bits chrominance and 2 bits for the block header, representing a 1/4 share of each for a quadrant.

16-Luminance Quadrant

An encoded *16-luminance quadrant* is specified by its corresponding two bits in the block header and 16 luminance values. When decoded with an 8 x 8 block size, this quadrant encoding contains one color value per pixel in the quadrant. Figure 11 shows how the encoded data is organized for a block whose second quadrant is a 16-luminance quadrant.

```
 xx 11 xx xx CHROMINANCE  QUAD 1   1111 1122   2222 3333   3344 4444   ....

Block Header Shared             16 Luminance Values
           Chrominance
```

Figure 11. Format of Block containing a 16-luminance quadrant

The quadrant is encoded as follows:

6 bits Y1

6 bits Y2

6 bits Y3

6 bits Y4

⋮

6 bits Y16

Decoding Tips

To decode this quadrant:

1. Using the shared chrominance value in this quadrant and the first luminance value (Y1), convert the YUV color to the target color space.
2. Store the output color in the corresponding pixel of the quadrant.
3. Repeat the last two steps for each pixel in the quadrant.

Compression attained with this quadrant encoding is 6.25 bits per pixel, assuming normal chrominance mode, including 2 bits chrominance and 2 bits for the block header, representing a 1/4 share of each for a quadrant.

Escape Values

Block headers with a value of 01110XXX are used as escape values, used to specify both block encoding types and data stream interpretation escape values. The following data stream interpretation escape values are defined:

Normal/unique chrominance mode toggle

This escape toggles the chrominance mode between normal and unique.

Single unique chrominance mode

This escape causes the following block to be interpreted in unique chrominance mode. The current chrominance mode is unchanged.

Frame guard byte

For efficiency, decoder implementations may choose not to bounds check addressing into the source data buffer. If the source buffer contains invalid data, an access violation might occur. Additionally, invalid data might be decoded as valid data without any means of software detection. Both of these problems are addressed with the "guard byte" value.

At least one guard byte always appears at the end of a coded frame, and is encoded in the stream. If the byte immediately following the last block encoding for a frame is not equal to this value, it is considered to be an error. As it is highly improbable that a buffer containing invalid data will be decoded and the above condition still met, decoding of invalid source data can be detected in this way.

If a guard byte value is encountered as an escape before the end of the frame, it is also considered to be an error. It is possible to stop a decoder from over-reading with invalid data by appending 64 bytes (the size of the largest block encoding) equal to 73H (the frame guard byte value) to the end of the

buffer. A decoder interpreting a stream will then encounter a guard byte before the end of the buffer is overrun.

Unchanged block run

This escape indicates a number of blocks are completely unchanged. The byte following the escape indicates the total number of blocks to skip. While it is perfectly valid to have consecutive unchanged blocks encoded with block header values of zero, it should be noted that for compression, three or more consecutive unchanged blocks should be encoded as an unchanged block run. Also, since two consecutive unchanged blocks may be encoded as an unchanged block run in the same space as zero block header values, an encoder can establish an unchanged run in the output data stream whenever two consecutive unchanged blocks are encountered.

General stream control escape

This escape provides a generalized escape mechanism for more escapes than permitted by the 3-bit direct escape in the byte header. This escape is followed by a 1-byte control value. The following generalized escapes are defined:

- Enable stream interpretation mode 0
- Enable stream interpretation mode 1

Note that since block header values 01110xxx are reserved as escape values, quadrant encoding combinations that would produce a block header value with any of these eight values are not allowed and must be altered to other encodings by the compressor.

Appendix A. Predetermined Luminance Transitions

A static set of luminance transitions across a region are defined. Seven types of predetermined transitions are defined for increasing luminance:

1. Linear

   The luminance level increases at a constant rate across the region.

2. Low-contrast edge at 1/4 block width from low-luminance edge

   The luminance level changes at an edge that is perpendicular to the direction of increasing luminance at a point where the area of the region following the edge is three times the area of that preceding the edge. The change in luminance is not instantaneous, analogous to a blurry transition.

3. Low-contrast edge at 1/2 block width from low-luminance edge

   Same as above, but with the change occurring at the point where the areas of the region on either side of the edge are equal.

4. Low-contrast edge at 3/4 block width from low-luminance edge

   Same as above, but with the change occurring at the point where the area of the region preceding the edge is three times the area of that following the edge.

5. High-contrast edge at 1/4 block width from low-luminance edge

   The luminance level changes at an edge that is perpendicular to the direction of increasing luminance at a point where the area of the region following the edge is three times the area of that preceding the edge. The change in luminance is instantaneous, analogous to a sharp, well defined edge.

6. High-contrast edge at 1/2 block width from low-luminance edge

   Same as above, but with the change occurring at the point where the areas of the region on either side of the edge are equal.

7. High-contrast edge at 3/4 block width from low-luminance edge

   Same as above, but with the change occurring at the point where the area of the region preceding the edge is three times the area of that following the edge.

The pre-determined luminance transitions are comprised of a subset of the total possible luminance changes in the range 0 through 63 for each of the transition types as follows:

Linear

Total Y delta = 2 3 4 5 6 7 8 11 14 17 20

Low-contrast Edge

Total Y delta = 4 5 6 7 8 11 14 17 20 23 26 29 32 36

High-contrast Edge

Total Y delta = 6 8 11 14 17 20 23 26 29 32 35 40 46

These selections of allowable Y delta values in two-byte Luminance Transition region encodings accomplish several objectives:

When combined with the seven transition types at all 64 initial luminance levels, the total number of valid transitions (those in which the Y delta value added to the initial luminance level does not exceed the maximum luminance level of 63) is 4096, which can be encoded in 12 bits in the two-byte encoding.

Regions with small changes in luminance (for example, <= 8) can be encoded with single luminance level precision. Regions with small changes in luminance are more common than regions with extreme changes.

Regions with larger changes in luminance ($8 < x < 32$) can be represented with an edge-to-edge error not exceeding one luminance level.

Extreme changes in luminance ($32 < x < 50$) can be represented in two-byte encodings if required by data rate constraints, although with some potential loss in quality as the extreme Y delta values are more highly quantized.

The following additional characteristics are exploited in the selection of allowable Y delta values for each transition type:

Transitions with Y delta values less than two are encoded with the one-byte encoding (resulting in a quantized direction).

Edge transitions with Y delta values less than 4 are deemed perceptually similar to a linear transition.

Edge transitions with Y delta values less than 6 are deemed perceptually similar with regard to edge type, that is, low or high contrast.

Transitions with Y delta values greater than 20 tend to be non-linear, that is, usually include an edge, and are coded only as such

Luminance changes greater than 50 are extremely rare and can be quantized with little perceptible quality loss.

The luminance transition index is produced by enumerating all valid transitions that are combinations of initial Y values of 0 through 63 and transition types with Y delta values as listed above. The following code fragment illustrates how these combinations are enumerated:

```
for (Y1 =  , Y1 <= 63, Y1++) {
   for (Y2 = Y1, Y2 <= 63, Y2++) {
      Ydelta = Y2 - Y1;
      if (Ydelta == <valid linear transition Y delta value>)
         <enumerate linear transition>;
      if (Ydelta == <valid low-contrast edge transition Y delta value>)
      {
         <enumerate low-contrast edge at 1/4 transition>;
         <enumerate low-contrast edge at 1/2 transition>;
         <enumerate low-contrast edge at 3/4 transition>;
      }
      if (Ydelta == <valid high-contrast edge transition Y delta value>)
      {
         <enumerate high-contrast edge at 1/4 transition>;
         <enumerate high-contrast edge at 1/2 transition>;
         <enumerate high-contrast edge at 3/4 transition>;
      }
   }
}
```

Appendix B. Data Stream Escape Value Constants

Escape values defined in the data stream are one-byte values, in the range 70H - 77H (01110XXX).

General Stream Control Escape - 70H

This escape is used to imbed control information in the stream. It is followed by a single byte value containing the control information.

Stream control byte values

Set stream interpretation mode 0 - 00H

Stream interpretation mode 0 block header quadrant decomposition cases:

1. 00: Unchanged quadrant
2. 01: Homogenous/shallow LTC quadrant
3. 10: Luminance transition coding (LTC) quadrant
4. 11: Statistical/extended LTC quadrant

Set stream interpretation mode 1 - 01H

Stream interpretation mode 1 block header quadrant decomposition cases:

1. 00: Unchanged quadrant
2. 01: Homogenous/shallow LTC quadrant
3. 10: Subsampled 4-luminance quadrant
4. 11: 16-luminance quadrant

Single Unique Chrominance Block Escape - 71H

This escape signals that the next block contains unique chrominance values. The chrominance mode (normal or unique) is unchanged. If the current chrominance mode is unique, this escape has no effect and is ignored.

Normal/Unique Chrominance Mode Toggle Escape - 72H

This escape causes normal chrominance mode to be entered if the current mode is unique chrominance mode, and causes unique chrominance mode to be entered if the current mode is normal chrominance mode.

Frame Guard Byte Value - 73H

This escape assists the decoder in determining if invalid data has been encountered in the input stream. See “Escape Values” on page 13 for a more detailed explanation.

Unchanged Block Run Escape - 74H

This escape value indicates some number of completely unchanged blocks. The number of unchanged blocks follows the escape in a single byte, and as such, is limited to 255.

Reserved Escape Values - 75H, 76H, 77H

These escape values are reserved.