Lab 2 Report

Tejas Joshi

ECEN 5653 RTDM

[5 pts]Read *Seeing Forbidden Colors*.
Please summarize the paper's main points (at least 3 or more) in a short paragraph.
The paper discusses that the opponent colors are not visible to the human eye.

The opponency begins in the retina and midbrain which follow Hering's law of color opponency. Crane and Piantanida found a way to bend these rules with some image stabilization. Though this was not too popular at that time for several reasons. The authors of the paper along with a Lt.Col Gleason eliminated the pitfalls in the Crane & Piantanida methods and were able to portray the Forbiddned Colors when equiluminant images were stabilized for viewing for observers. With these results they were able to develop a model which could prove how color opponency would arise in the brain, They also were able to explain the tiger stripes phenomenon when viewing opponent colors together which resmbled reacting chemicals which diffused assymetrically or at different rates. This was a result of excitation of striped patterns of neurons in the primary visual cortext located at the back of the head. Similarly viewing halucinations of fans were observed to have concentric circles and the reverse for small physical circles.

1.**[5 pts]Read The Netpbm sourceforge documentation and convert the sharpened sunset picture from the paper Using Intel Streaming SIMD Extensions and Intel Integrated Performance Primitives to Accelerate Algorithms in o a PGM. Insert the resulting grayscale image into your lab and your code used to convert it. Describe why PPM is not a great way to represent a color image (frame) compared to formats like JPEG 2000, summarized on Wikipedia. Can JPEG200 support lossless image compression? How does JPEG 2000 compare to PNG? (summarize each answer in a simple paragraph).**

The Netpbm library consists of functions to conv

2.ert images to PNM formats to any other formats like jpeg, bmp etc. and vice-versa.

I was unable to use the pgm function on the sharpened image as it kept giving me errors as shown below:

```
ppmtopgm: EOF / read error reading a row of pixels
S[illegible]rix@lynx:~/prog/sse$ 62;9;c
```

but i was able to convert the sunset.ppm file to a pgm and then perform the sharpen on it which would have the same effect as shown below:

The pgm image

sharpened pgm Image:

The command used:

```
drix@lynx:~/prog/sse$ ppmtopgm sunset.ppm
```

PPM is losless and is also very space consuming , the color image rperesented in PPM format woul occupy greater space than a JPEG 2000.

Yes, JPEG can also support lossless image compression .

JPEG 2000 is mostly used in it lossy form , but the lossless format does not provide enough compression as provided by the PNG and hence is less preffered when losless data is involved.

3.[5 pts]**Read** Using Intel VTune Performance Analyzer and Intel Integrated Performance Primitives for Real-Time Media Optimization**.**
Please summarize the paper's main points (at least 3 or more) in a short paragraph.

The VTune analyzer uses the PMU which performs hardware performance , This is helpul in media encoders as there is a lot of repetetive code. It can also track cache hits and misses down to specific lines of code which can be possibly optimized for faster performance. This analyzer can also be used with accelerated, parallel/threaded code for analysis to find hotspots without much optimization skills and experience. This is specially useful when dealing with large amounts of data as required in Real Time Digital Media.

4.**[10 pts]Debugging POSIX threading code (2 part question) using the example-2 code: Please instrument the example-2 code to prove the order of thread executions and joins using syslog**

1.[5 pts] Instrument the code in example-2 to show each thread create, when the thread starts, when it finishes, and each join in the main thread. Please provide a "tail -f /var/log/messages" to show your trace - is the order in syslog correct? Why or why not?

The order of the creation of threads is sequential, and the start, exit and join process are followed on individual thread basis. i.e thread 1 starts , exits and joins with the main thread.

Please view the messages_var_log.txt attached for the trace of syslog.

2.[5 pts] Now add the readTOD() function calls to each syslog so that you have a timestamp for each log entry. Are all entries in order? If not, does the timestamped order make sense?

The entries are in order and are timestamped monotonically sequential fashion.

5.[**25 pts]MPEG encode and decode with Linux ffmpeg - for these exercises please "yum install ffmpeg" and "yum install vlc" on your Fedora Core 12 Linux installation.**

1.**[5 pts] Download the Open Source HD Video 1080p MPEG-2 transport stream from our class website [535MB]. Using ffmpeg re-encode this transport stream into an MPEG-2 program stream with video only. Verify your result by playing it with VLC. What is the peak stream and input bit-rate you observe in the original MPEG-2 transport stream using the VLC Codec Information Statistics capability and how does this compare to the peak input and stream bit rate of your ffmpeg video-only program stream?**

Command used to re-encode the video without sound

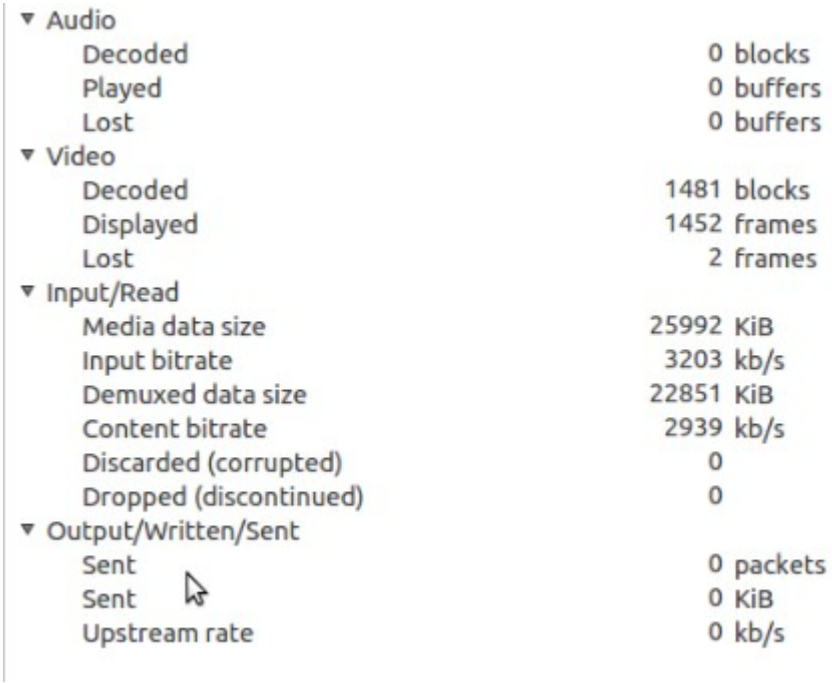
```
ffmpeg -i Open_Source_HD_Video_1080p.ts -an Open_Source_HD_Video_1080p_ffmpeg_.ts
```

The oputput of the ffmpeg command provided some bitrate information as follows:

```
    Last message repeated 1 times  49053kB time=126.72 bitrate=3171.1kbits/s
frame= 3179 fps= 50 q=24.8 Lsize=   49095kB time=127.12 bitrate=3163.8kbits/s
video:45200kB audio:0kB global headers:0kB muxing overhead 8.616902%
drix@lynx:~/prog/lab 3/example 3/test_results$
```

```
drix@lynx:~/prog/lab3/example3/test_results$ avprobe Open_Source_HD_Video_1080p_
ffmpeg_.ts
avprobe version 0.8.6-4:0.8.6-0ubuntu0.12.04.1, Copyright (c) 2007-2013 the Liba
v developers
  built on Apr  2 2013 17:02:36 with gcc 4.6.3
[mpegts @ 0x1c2f7a0] max_analyze_duration reached
Input #0, mpegts, from 'Open_Source_HD_Video_1080p_ffmpeg_.ts':
  Duration: 00:02:07.12, start: 1.400000, bitrate: 3163 kb/s
  Program 1
    Metadata:
      service_name    : Service01
      service_provider: Libav
    Stream #0.0[0x100]: Video: mpeg2video (Main), yuv420p, 1920x1080 [PAR 1:1 DA
R 16:9], 104857 kb/s, 25 fps, 25 tbr, 90k tbn, 50 tbc
drix@lynx:~/prog/lab3/example3/test_results$
```

Also the VLC player provides maximum bitrate information from the below screenshot



Rates for the original video:

```
drix@lynx:~/prog/lab3/example3/test_results$ avprobe Open_Source_HD_Video_1080p_
AVI.avi
avprobe version 0.8.6-4:0.8.6-0ubuntu0.12.04.1, Copyright (c) 2007-2013 the Liba
v developers
  built on Apr  2 2013 17:02:36 with gcc 4.6.3
[mpeg4 @ 0x1dc27a0] Invalid and inefficient vfw-avi packed B frames detected
Input #0, avi, from 'Open_Source_HD_Video_1080p_AVI.avi':
  Metadata:
    encoder         : Lavf52.32.0
  Duration: 00:02:07.02, start: 0.000000, bitrate: 1719 kb/s
    Stream #0.0: Video: mpeg4 (Advanced Simple Profile), yuv420p, 640x480 [PAR 1
:1 DAR 4:3], 29.97 tbr, 29.97 tbn, 29.97 tbc
    Stream #0.1: Audio: mp3, 48000 Hz, stereo, s16, 128 kb/s
drix@lynx:~/prog/lab3/example3/test_results$
```

2.[5 pts] Using the same Open Source HD Video MPEG-2 video-only program stream you created with ffmpeg, now use ffmpeg to extract one of the frames with the shallow pool of rippling water and stoney bottom. Take the single frame that you extracted and using your favorite graphics editing tool (e.g. Paint or GIMP), shrink the image down to a thumbnail size (80x60) and paste that image into your lab report to prove you completed this.

Command used:

```
ffmpeg -i Open_Source_HD_Video_1080p_ffmpeg_simple.ts -ss 00:00:34.000 -f image2 -vframes 1 out.p
```

the original output frame is attached.

The thumbnail created using gimp is :

the sharpened image:

3.**[5 pts] Take your full size rippling water single frame and apply the edge enhancing PSF algorithm from Using Intel Streaming SIMD Extensions and Intel Integrated Performance Primitives to Accelerate Algorithms to a 320x240 resolution version of this frame - paste in the original frame and the enhanced frame into your lab report.[5 pts] Measure the time it takes to enhance your single 320x240 rippling water frame. Would you be able to enhance all frames with the same implementation at 30 fps? Provide the time it takes to enhance your 320x240 frame WITHOUT vector processing instruction acceleration using the SSE compilation options described in the paper. Provide your CPU clk rate, number of cycles and approximate milliseconds to enhance the frame.**

The time take to enhance the 320x240 image without the ssse is a shown below :

```
drix@lynx:~/prog/sse$ ./sharpen rippling_water_320x240.ppm
Cycle Count=1000138
Based on usleep accuracy, CPU clk rate = 1000138 clks/sec,     1.0 Mhz
Convolution time in cycles=7990, rate=1, about 7 millisecs
```

4.**[5 pts] Transcode "Big Buck Bunny" 720p [128xx720] AVI container MP4 files as described inUsing Intel VTune Performance Analyzer and Intel Integrated Performance Primitives for Real-Time Media Optimization. You should wind up with an MPEG-2 program stream with no audio as well as an MPEG-4 program stream that can be played with VLC. What is the maximum, minimum, and average bit rate for the MPEG-2 version and the MPEG-4 version, how do they compare, and if there are differences, why?**

Commands used:

ffmpeg -i big_buck_bunny_720p_surround.avi -an -vcodec copy big_buck_no_audio.mp4

ffmpeg -i big_buck_no_audio.mp4 -vcodec mpeg2 big_buck_no_audio.mp2

Bitrate for the MPEG-2 video

```
drix@lynx:~/prog/lab3/example_mpeg2$ avprobe big_buck_no_audio.mp2
avprobe version 0.8.6-4:0.8.6-0ubuntu0.12.04.1, Copyright (c) 2007-2013
the Libav developers
  built on Apr  2 2013 17:02:36 with gcc 4.6.3
[mpegvideo @ 0x1b567a0] max_analyze_duration reached
[mpegvideo @ 0x1b567a0] Estimating duration from bitrate, this may be in
accurate
Input #0, mpegvideo, from 'big_buck_no_audio.mp2':
  Duration: 00:00:05.05, bitrate: 104857 kb/s
    Stream #0.0: Video: mpeg2video (Main), yuv420p, 1280x720 [PAR 1:1 DA
R 16:9], 104857 kb/s, 25 fps, 25 tbr, 1200k tbn, 48 tbc
drix@lynx:~/prog/lab3/example_mpeg2$
```

Bitrate for the MPEG-4 video:

```
drix@lynx:~/prog/lab3/example_mpeg2$ avprobe big_buck_no_audio.mp4
avprobe version 0.8.6-4:0.8.6-0ubuntu0.12.04.1, Copyright (c) 2007-2013
the Libav developers
  built on Apr  2 2013 17:02:36 with gcc 4.6.3
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'big_buck_no_audio.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2mp41
    encoder         : Lavf53.21.1
  Duration: 00:09:56.45, start: 0.000000, bitrate: 4003 kb/s
    Stream #0.0(und): Video: mpeg4 (Simple Profile), yuv420p, 1280x720 [
PAR 1:1 DAR 16:9], 4001 kb/s, 24 fps, 24 tbr, 5000k tbn, 24 tbc
drix@lynx:~/prog/lab3/example_mpeg2$
```

The MPEG4 video has a greater compression ration and is better suited for streaming compared to the MPEG2 video which has similar video quality but occupies greater storage space.

6.**[50 pts]Basic MPEG2 Transport Stream Analysis (5 part question) using MPEG2 example-mpeg2 code as a starting point and reference code along with the ISO MPEG2 13818-1 spec, 13818-2 spec, and documentation on Source Forge MPEG 2 Headers:**

This portion of the lab tests your basic understanding of transport for MPEG2 and timing along with fundamentals of elementary streams.

1.[10 pts] Build bitrate.c code and run bitrate on Open_Source_HD_Animation_1080p.ts and on Open_Source_HD_Video_1080p.ts MPEG2 transport files. Please report the maximum and minimum bit rates for each with PCR samples that are adjacent. Please also watch both MPEG transport streams using VLC on your Linux system and open streaming statistics to verify bit-rates are correct. Finally, describe exactly what the PCR is used for and what PTS/DTS are used for as well.

The output for the bitrates for the Animation is:

```
Found maxBitRate=39999903.3 bps, minBitRate=39999866.9 bps
Estimated estBitRate=39.999882 Mbps
```

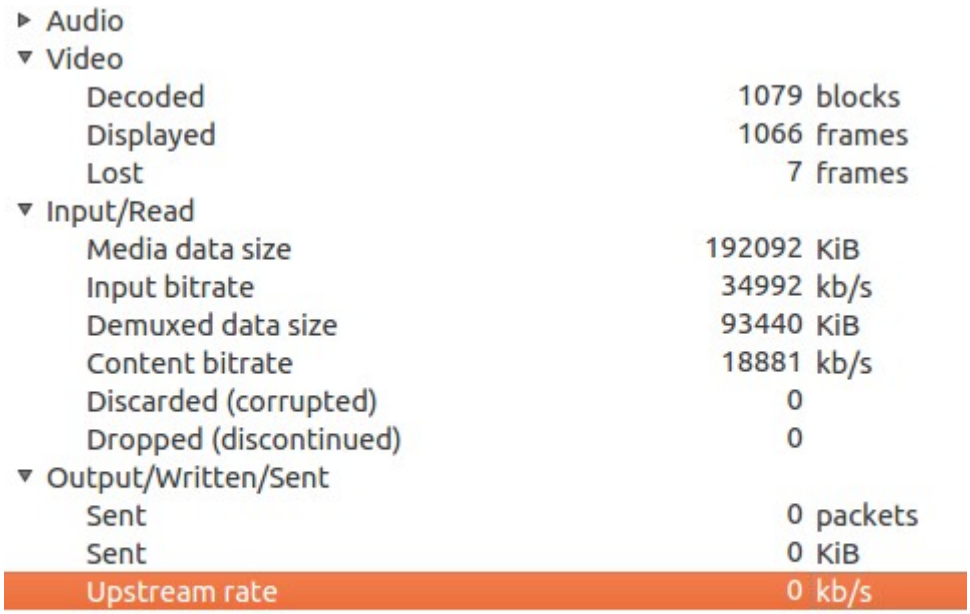
The bitrates from VLC stats:

▼ Audio		
Decoded	954	blocks
Played	954	buffers
Lost	0	buffers
▼ Video		
Decoded	919	blocks
Displayed	911	frames
Lost	1	frames
▼ Input/Read		
Media data size	154416	KiB
Input bitrate	40020	kb/s
Demuxed data size	97472	KiB
Content bitrate	22468	kb/s
Discarded (corrupted)	0	
Dropped (discontinued)	0	
▼ Output/Written/Sent		
Sent	0	packets
Sent	0	KiB
Upstream rate	0	kb/s

The output for the bitrates of Video 1080p is :

```
Found maxBitRate=34999968.3 bps, minBitRate=34999935.7 bps
Estimated estBitRate=34.999939 Mbps
drix@lynx:~/prog/lab 3/example_mpeg2$
```

Vlc stats:



PCR or (Program Clock reference) : the transmitter and reciever syncronization is possible with the refrence clock and is inserted at the encoder buffer and extracted before the reciever buffer. PCR are inserted at the interval of 100ms usually.

While transmission the stream/packets may be affected by delay or time jitter.PCRs are also affected by the multiplexing of multiple program streams.

PTS(Presentation Time Stamp): this value indicates the time to remove the previous buffer contents (containing video or audio in their respective buffers) be removed and instantaneously decoded and presented to the display. This is usually entered at intervals not exceeding 700ms

DTS(Decode Time Stamp):

This indicates the time for which the contents of the reciever buffer must be emptied and decoded. This is used when picture reordering is uded for B-images.

[10 pts] Build pids.c code and run on Open_Source_HD_Animation_1080p.ts and on Open_Source_HD_Video_1080p.ts MPEG2 transport files. Please report the video, audio, PAT, PMT, and PCR PIDs found and describe the use of each of these in the transport stream. Look at the PCR as well as PTS/DTS and verify that the example code is correct and describe how a decode would use this information for a single program transport stream and then how this would be useful for a multi-program transport stream. (Hint - you'll want to make sure the PAT is fully parsed correctly before parsing timing information).

Pid's for the Animation file:

```
numPIDs=4


SUMMARY OF INFO:

Playback count=1:
STREAM 1
devID=0, port=0
streamID=0, streamName=, transportBitRate=0
TRIPLET INFO: annex=0, freq=0, mode=0, port=0, progNum=0, tsid=0
numPrograms=1
PROGRAM 1
        videoPID[0]=49
        audioPID[0]=50
        pmtPID[0]=32
        pcrPID[0]=48
```

Pids for the Video:

```
numPIDs=4


SUMMARY OF INFO:

Playback count=1:
STREAM 1
devID=0, port=0
streamID=0, streamName=, transportBitRate=0
TRIPLET INFO: annex=0, freq=0, mode=0, port=0, progNum=0, tsid=0
numPrograms=1
PROGRAM 1
        videoPID[0]=49
        audioPID[0]=50
        pmtPID[0]=32
        pcrPID[0]=48
```

for the complete output : view pids_out_anim.txt and pits_out_vid.txt files which contians the PAT PMT and PCR and PIDs

PID

Each table or elementary stream in a transport stream is identified by a 13bit packet ID (PID) , A demultiplexer extracts the individual streams from the transport streams in part for packets identified by the same PID. In most applications, time-division multiplexing will be used to decide how often a particular PID appears in the transport stream.

PAT(Program Association Table): It lists all programs in the trasport stream identified by a 16bit value called a program_number. Every program listed in a PAT has a PID for its Program Map Table(PMT). The value 0x0000 program_number is reserved to specify the PID where to look for Network Information Table(NIT). If such a program is not present in PAT the default PID value(i.e 0x0010) shall be used for NIT.

PMT(Program Map Tables) contains information about programs. For each program there is one PMT where Program Map Tables (PMTs) contain information about programs. For each program, there is one PMT. While the MPEG-2 standard permits more than one PMT section to be transmitted on a single PID, most MPEG-2 "users" such as ATSC and SCTE require each PMT to be transmitted on a separate PID that is not used for any other packets. The PMTs provide information on each program present in the transport stream, including the program_number, and list the elementary streams that comprise the described MPEG-2 program. There are also locations for optional descriptors that describe the entire MPEG-2 program, as well as an optional descriptor for each elementary stream. Each elementary stream is labeled with a stream_type value.

2.**[10 pts] Build esparse.c code and run on Open_Source_HD_Animation_1080p.ts and on Open_Source_HD_Video_1080p.ts MPEG2 transport files. Please modify this code as needed to parse all the MPEG packet types described in the 13818-2 found in the video elementary stream data and describe the use of each of these in the transport stream. (Hint - the following link at sourceforge on MPEG headers has a nice overview of MPEG2 elementary stream start codes). Did you find all the headers and fields that were described in the 13818-2? If not, why do you think some are missing?**

3.Command used

```
drix@lynx:~/prog/lab3/example_mpeg2$ ./esparse Open_Source_HD_Animation_
1080p.ts 49 50 32 48
```

Output:

```
PES SLICE Hdr: ID=0x3E, length=18, byteCnt=544105497
PES SLICE Hdr: ID=0x3F, length=18, byteCnt=544105692
PES SLICE Hdr: ID=0x40, length=18, byteCnt=544105872
PES SLICE Hdr: ID=0x41, length=18, byteCnt=544106056
PES SLICE Hdr: ID=0x42, length=18, byteCnt=544106235
PES SLICE Hdr: ID=0x43, length=18, byteCnt=544106384
PES SLICE Hdr: ID=0x44, length=18, byteCnt=544106763

num PES=3251
```

Comman Used:

```
drix@lynx:~/prog/lab3/example_mpeg2$ ./esparse Open_Source_HD_Video_1080
p 49 50 32 48
```

Output

```
PES SLICE Hdr: ID=0x40, length=18, byteCnt=561850724
PES SLICE Hdr: ID=0x41, length=18, byteCnt=561850736
PES SLICE Hdr: ID=0x42, length=18, byteCnt=561850748
PES SLICE Hdr: ID=0x43, length=18, byteCnt=561850760
PES SLICE Hdr: ID=0x44, length=18, byteCnt=561850772

num PES=3179
```

It was possible to view the picture_start_code Header (0x00)

the slice_start_code Header (0x01 through 0xAF)

and the extension_start_code header (0xBF)

Header o/p screen:

```
PES SLICE Hdr: ID=0x44, length=18, byteCnt=560975632
******** PES VIDEO Hdr: ID=0xE0, length=0, byteCnt=561149736
PES PICTURE Hdr: ID=0x00, length=2, byteCnt=561149760
PES EXT Hdr: ID=0xB5, length=133, byteCnt=561149769
```

The rest have not been implemented in the video file.

4.[30 pts, 20 pts + 10 bonus points] Build gopidx.c code and verify that it correctly finds MPEG2 GOPs and I-frames and provides a file with indexes for each I-frame. Modify the code so that it generates a re-timed MPEG2 file with adjusted PCR/PTS/DTS so that you can get the new file to play in VLC, but with I-frames only, eliminating P and B frames. What is the speed-up (fast forward rate) of the resulting transport stream? What is the new bit-rate of the I-frame only transport stream? If you can't get the file to play correctly in VLC, please describe why. Either way, please describe how you would implement trick play operations including FF, REW, PAUSE, and RESTART using indexing into GOPs and I-frames and/or modification of PTS/DTS. Finally, please describe the fundamental differences between I, B, and P frames in an MPEG2 GOP. Be sure to read about the discontinuity bit in the 13818 standards and describe why it is important in your solution, if it is.

NOTE: 10 bonus points will be awarded if you can demonstrate clean FF and REW streams with VLC - this must be demonstrated including file generation to the professor before the end of the semester. Everyone is expected to implement the PAUSE and RESTART operations and to turn in their code for this.

Output:(Animation file)

```
220 I-frames, 40112 hdrs with PID=49
Maximum I-frame=676424 @ 455840216

*****Summary*****
Number of TS bytes =544125204
Number of TS packets =2894284
Number of PAT =0 (0x0) =2177
Number of PMT =32 (0x20) =2177
Number of VID =49 (0x31) =1678512
Number of AUD =50 (0x32) =30429
Number of NULL=8191 (0x1fff) =1178768
Number of UNK =0 (0x0) =2221

****Video PID****
Number of I-FR=220
Number of GOP=220
Number of PIC=3251
Number of PES=3251
Number of SEQ=220
```

Output :(Video file)

```
214 I-frames, 3788 hdrs with PID=49
Maximum I-frame=621716 @ 417825112

*****Summary*****
Number of TS bytes =561924104
Number of TS packets =2988959
Number of PAT =0 (0x0) =2569
Number of PMT =32 (0x20) =2569
Number of VID =49 (0x31) =1380044
Number of AUD =50 (0x32) =35685
Number of NULL=8191 (0x1fff) =1565470
Number of UNK =0 (0x0) =2622

****Video PID****
Number of I-FR=214
Number of GOP=214
Number of PIC=3179
Number of PES=3179
Number of SEQ=214
```

An I-frame is an 'Intra-coded picture', in effect a fully specified picture, like a conventional static image file. P-frames and B-frames hold only part of the image information, so they need less space to store than an I-frame and thus improve video compression rates.

A P-frame('Predicted picture') holds only the changes in the image from the previous frame. For example, in a scene where a car moves across a stationary background, only the car's movements need to be encoded. The encoder does not need to store the unchanging background pixels in the P-frame, thus saving space. P-frames are also known as delta-frames.

A B-frame ('Bi-predictive picture') saves even more space by using differences between the current frame and both the preceding and following frames to specify its content.

The Discontinuity bit if set to 1 if the TS packet is not in sync with the continuity counter or the program clock reference.