Motivation for h264 source coding

Purpose

The purpose of this document is:

1. Clarify the different parameters that affect the quality, size and nature of a h264 encoded video.
2. Determine how the video bitrate effect live video streaming.

Theory

H264 uses temporal redundancy, which refers to the fact, that frames next to each other are often very similar. Actually consecutive frames are almost identical and H264 exploits this redundancy by storing the error of difference in correlation to the number of shifts, from a frame to the next. This method is called motion compensation, and the number of shifts that is stored is called the motion vector.

H264 consist of 3 types of frames:

- I-frame (Intra-coded picture) which is the 'whole' image. When a video stream begins, the first frame will always be an I-frame. After the first I-frame, these frame will come at a given rate.
  From the standard: "I slice: A slice that is not an SI slice that is decoded using prediction* only from decoded samples within the same slice."

- P-frame (Predicted picture), is the change from the previous I-frame / P-frame. P-frames is significant smaller than I-frames, since they do not contain the whole picture.
  From the standard: "P slice: A slice that may be decoded using intra prediction from decoded samples within the same slice or inter prediction from previously-decoded reference pictures, using at most one motion vector and reference index to predict the sample values of each block."

- B-frame (Bi-predictive frame), is a computer generated frame, which is created from a future P/I frame and the previous P/I frame. B-frames is created to make the picture smoother.
  Note that B-frames cannot exist in a live stream, since future does not exist.
  From the standard: "A slice that may be decoded using intra prediction from decoded samples within the same slice or inter prediction from previously-decoded reference pictures, using at most two motion vectors and reference indices to predict the sample values of each block.

- "*prediction: An embodiment of the prediction process.
  prediction process: The use of a predictor to provide an estimate of the sample value or data element currently being decoded."

I frames is the least efficient frame in H264, but all playbacks must start with an I-frame, since they don't refer to any other frames during encoding. For a more responsive playback, the I-frames should come in a certain interval.
There are no magic number, and this should be chosen to the specific scenario!

There are primarily two strategies in controlling the bitrate when encoding h264 video, which can be seen in figure 1, [5]:

- **Constant bitrate(CBR)** sets a fixed amount of data that can be used to describe each frame, this gives a predictable data flow. The problem with Constant bitrate is, when the data rate of the not code video is lower than the fixed, the coding will force the video data to be equal to the fixed rate, without getting any graphically improvements and vice versa if the rate is higher.

- **Variable bitrate(VBR)** varies the amount of data that can be used by each frame, this enables more data to be allocated when needed. Furthermore, when not needed this technique saves bandwidth. The drawback of this type of encoding is that the data rate is nonpredictable, as seen in figure 1.

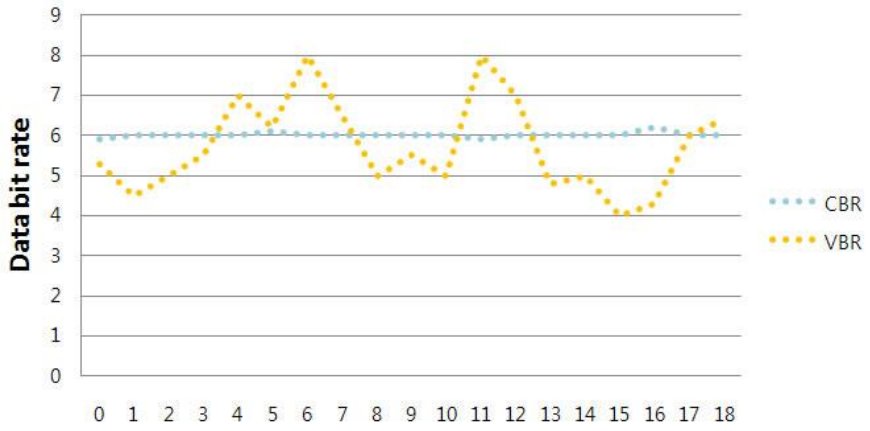


Figure 1 from
http://update.websamsung.net/OnlineTutorial/MPcamera/HTML_EN/contents/pcMenu03.html

Methods

Due to the large amount of parameter to consider when performing h264 and the limitations of the different implementations it is chosen not to examine things such as I-frame period, quantization and the effect of various frame rates. But instead to examine How the different bitrates controlling techniques effect the dataflow of the video stream since this among other things effects the system latency and predictable.

For reference configuration and how it affects the average bitrate see the table below

Name	Resolution	Video (kbps)
240p	424x240	576
360p	640x360	896
432p	768x432	1088
480p	848x480	1216
480p HQ	848x480	1536

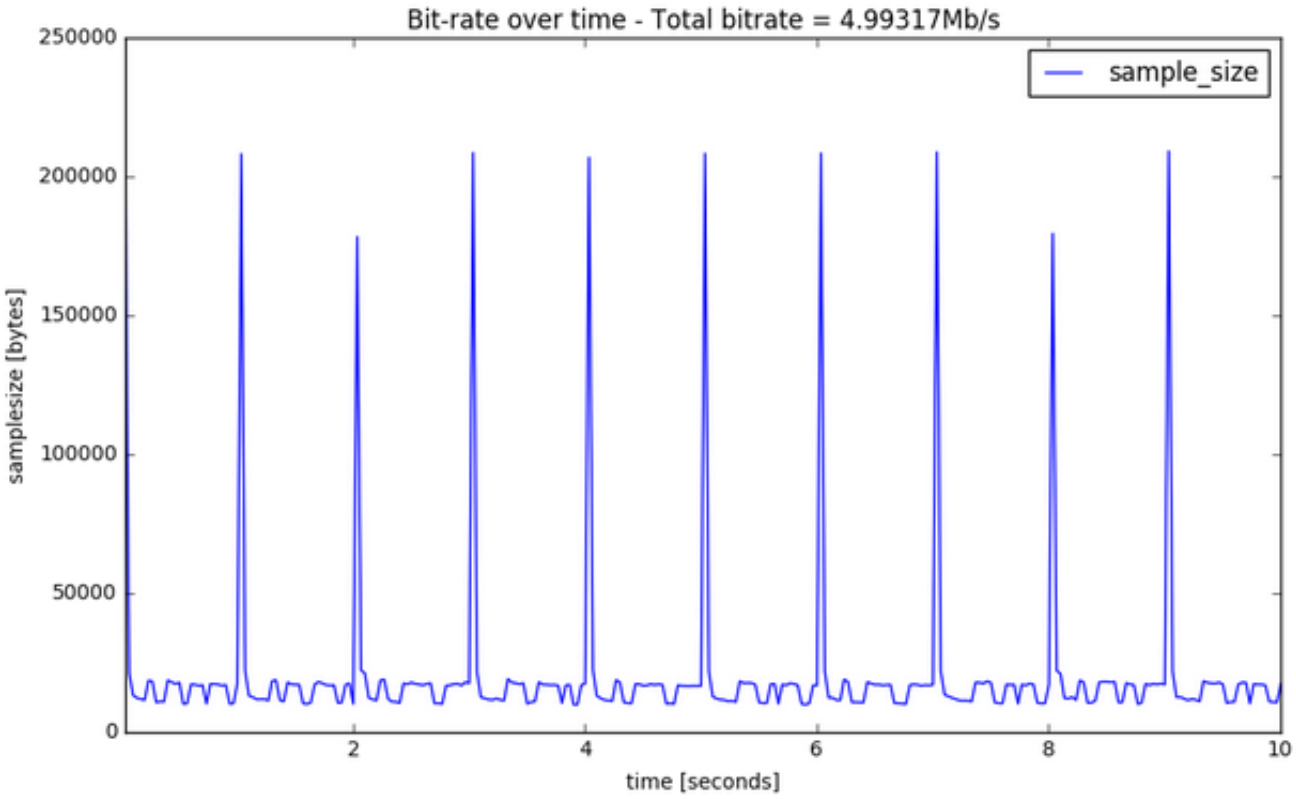
576p	1024x576	1856
576p HQ	1024x576	2176
720p	1280x720	2496
720p HQ	1280x720	3072
1080p	1920x1080	4992
1080p HQ	1920x1080	7552
1080p Superbit	1920x1080	20000

*http://www.lighterra.com/papers/videoencodingh264/

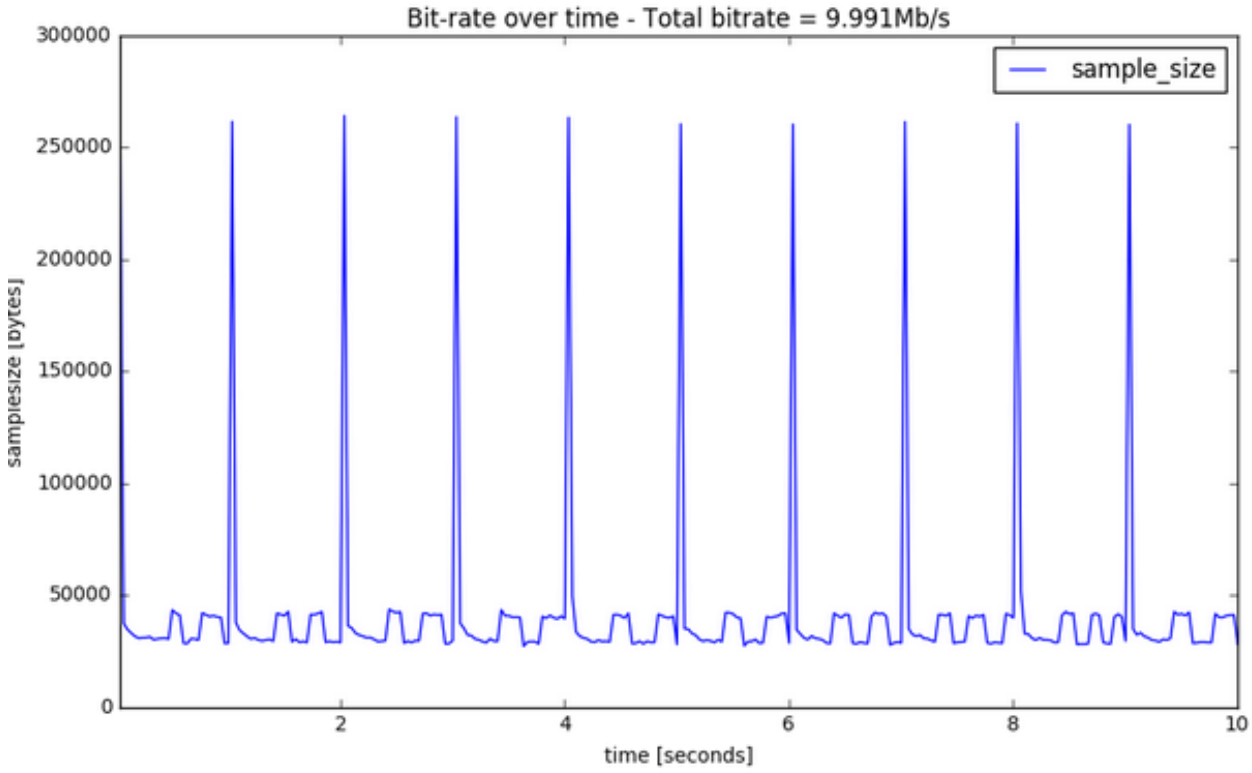
In order to test these two different encoding schemes videos have be captured using the Parrot Bebop 2 Drone at rates [30 20 10 5] Mbs at 30 fps 1080p. However, since the Parrot does not offer the option of a constant bit rate, we have chosen only to further investigate the variable bitrate.

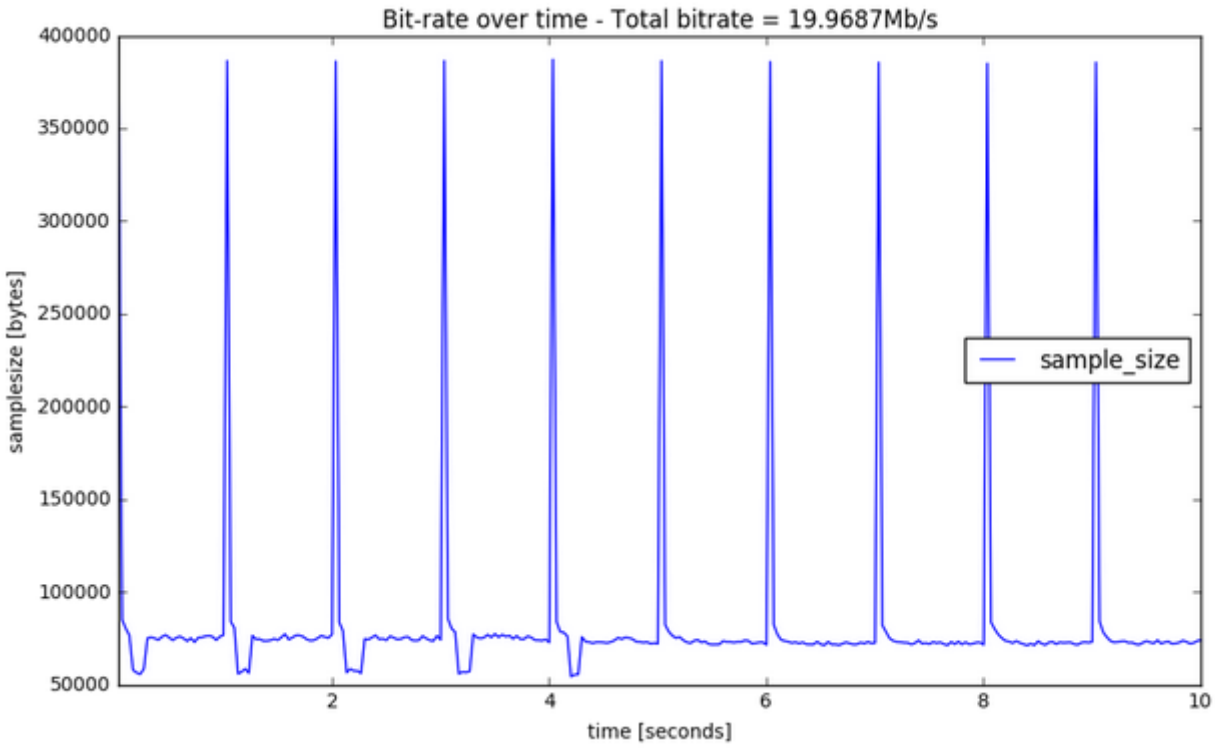
Test Parameters of live streaming:

In the test we have increased the quality of the video to see how this effect the bit rate. In the test, it is not of interest how the quality have been increased, only how it effect the bitrate.



The problem with variable bitrate is that if we can support 5mb/s, and we set the variable bitrate to 5mb, then in some periods the bitrate can be higher, and this will lead to extra **playback** delay.

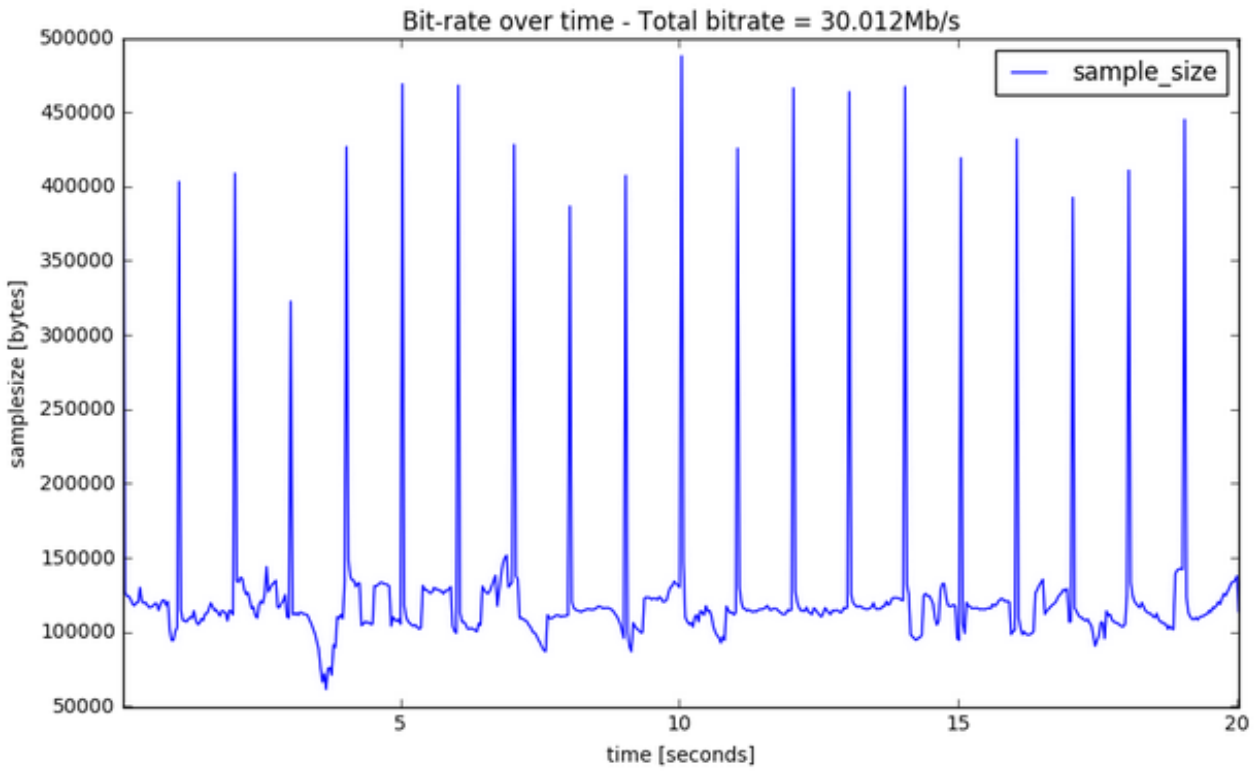




The magnitude changes on the spikes when we change the bitrate(VBR). Above is three examples of 20, 10 and 5 Mbit

So the p-frames will not take much space, but the I-frames in the other hand, will be very large.

In this worksheet we have not yet been able to look at all the parameters, but only varying the bitrate. If VBR is used, most of the bandwidth is used for supporting the I-frames, at least in the static case. This is also seen in the moving case, as we can see on the next image.



The conclusion is, that if we shall be able to support the VBR, we need a buffer system, which introduces a delay.

Reference1:http://www.streaminglearningcenter.com/articles/producing-h264-video-for-flash-an-overview.html?page=4

Reference 2:http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.264-200305-S!!PDF-E&type=items

Reference 3:www.springer.com/cda/content/document/cda_downloaddocument/9781461422297-c1.pdf?SGWID=0-0-45-1338306-p174267072

Reference 4:http://guvcview.sourceforge.net/Doc.html
https://sourceforge.net/p/guvcview/git-master/ci/master/tree/

Reference 5:
http://update.websamsung.net/OnlineTutorial/MPcamera/HTML_EN/contents/pcMenu03.html