TECHNISCHE UNIVERSITÄT
ILMENAU

Faculty of Electrical Engineering and Information Technology
Institute for Media Technology
Audio Visual Technology

MASTER THESIS

Netflix Like Encoding Optimization

Submitted by: Vijaykumar Singh Rana

Major: Media Technology

Advisor: Prof. Dr.-Ing. Alexander Raake
Co-Advisor: M.Sc. Steve Göring

Ilmenau, October 23, 2019

Acknowledgments

optional

Zusammenfassung

maximum of 2400 chars; one paragraph

Abstract

maximum of 2400 chars; one paragraph

Contents

Chapter 1

Introduction

In today's world, media is being offered to the users on a large scale. That means users around the world are indulged in the streaming services for their favourite shows. These show are delivered to them based on the available bandwidth. From the production point of view, these shows must be encoded using schemes that would allow the end-users to have a good quality of experience by the content of the show. YouTube[Yt], Netflix[Neta], Prime Video[Pri] etc. are the most popular video hosting websites on internet that provides VOD (video on demand). They have evolved their schemes for transcoding videos in a way such that the end-user would receive reasonable quality of videos corresponding to the bandwidth available to them. The service providers that can deliver these video streaming services have to use various schemes for transcoding videos based on the subjective quality for end-users to have a good quality of experience. These video stream providers mainly aim to reduce the bandwidth, consequently reducing the costs and to satisfy the end user so that they can perceive the best possible quality. One of such schemes that provides streams based on the bandwidth is Dynamic Adaptive Streaming over HTTP (DASH). As the name suggests, it uses a streaming technique that provide media content with quality that can be delivered with available bandwidth.

Dynamic Adaptive Streaming over HTTP (DASH), also known as MPEG-DASH, is an adaptive bitrate streaming technique that enables high quality streaming of media content over the Internet delivered from conventional HTTP web servers. The content is made available at a variety of different bit rates, i.e., alternative segments encoded at different bit rates covering aligned short intervals of playback time. While the content is being played back by an MPEG-DASH client, the client automatically selects from the alternatives the next segment to download and play based on current network conditions. The client automatically selects the segment with the highest bit rate possible that can be downloaded in time for playback without causing stalls or re-buffering events in the playback. Considering video streaming the main goal of a video stream provider is to reduce bandwidth consequently the cost and to satisfy the end user, so that they are able to perceive the best possible quality. [Sto11]

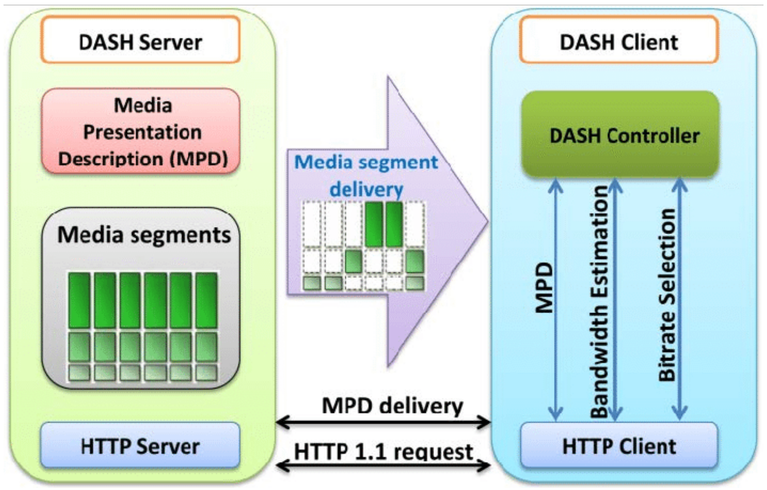


Figure 1.1: The overview of Dynamic Adaptive Streaming over HTTP technique. [VMC17]

Currently, YouTube and Netflix are some of the most popular video hosting websites in the world. [Moo+19]They provide the end-users with streaming service which is affordable and provide the best possible quality content based on the fluctuating network conditions. To provide the quality content, YouTube uses chunked video playback. The video is split into smaller parts called chunk. Each of the chunks are encoded from low quality to high quality. The suitable quality chunk is selected based on network bandwidth. Because of chunking, the web browsers are able to manage the smaller pieces of the videos, in their dedicated cache for playback. For fluctuating network speeds and optimal user viewing experience, YouTube uses Adaptive Bitrate Streaming (ABS). Figure 1.2 demonstrates how a video stream is selected based on the network condition. When the video starts on YouTube it starts with low quality and then based on the network condition the quality keeps switching. As it can be seen in the Figure 1.2 when the available bandwidth gets higher, the high bitrate stream is then selected. in the next timestamp there appears a network congestion and eventually the quality drops i.e., high bit rate stream is not streamed and based on lower bandwidth low bit rate stream is then selected. Further as the network bandwidth changes, the suitable stream is then selected. YouTube also acquires the screen dimensions and when enough network bandwidth is available then the YouTube would send the stream with the resolution adaptive to screen dimensions.

Netflix also uses the similar adaptive streaming approach to provide streaming service to the users. Netflix also uses granular approach of chunking the video stream based on the shots. A sequence of frames shot by a camera until a point is called a shot. A video sequence is sequence of shots alternated based on the actors in the video sequence. A scene on the other hand is a collection of shots focused on an actor(s) or or an object(s). [BR96] In this thesis,

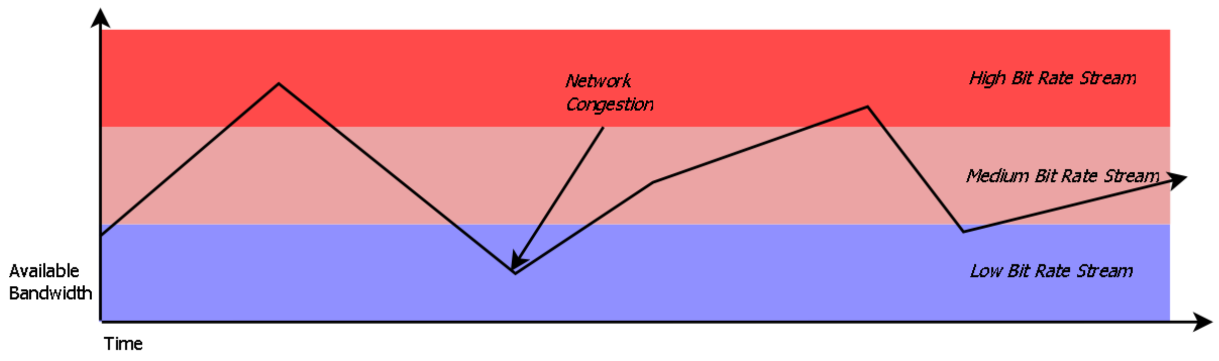
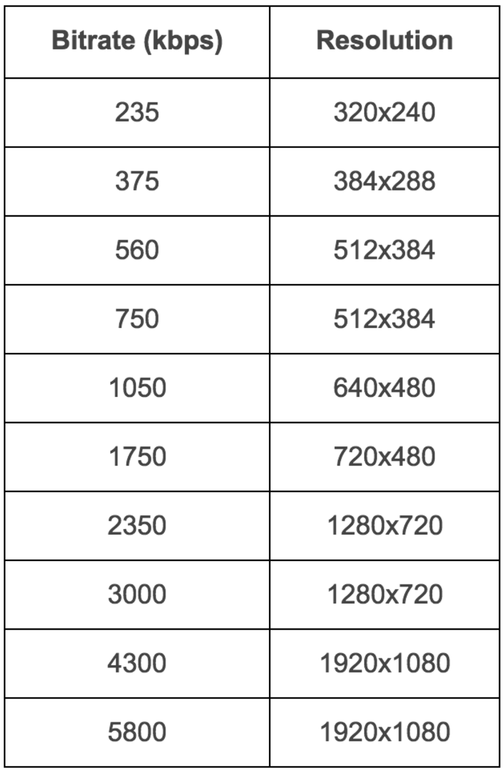


Figure 1.2: Adaptive Streaming [OJ]

scene and shot are referred to be same. These terminologies are very important from the point of view of this thesis. Netflix uses techniques of granular optimizations, i.e., Netflix analyzes the scenes for a video sequence individually and optimizes each of them to produce the best quality of content for given set of bitrates. On the top level Netflix uses Per-Title Encode Optimization where a scheme to encode the video optimally is prepared. For every individual video sequence the scheme is generated because the content of the video sequences varies and hence Netflix analyzes and then encodes the video individually. The objective is to deliver better quality of experience for users with bandwidth available to them. The reason for using per title encoding is because the approach of "one-size-fits-all" fixed bitrate ladder is not a good solution. In many cases the video content differs from video to video. So setting a fixed bitrate for certain resolution is not a good solution and it is inefficient. Figure 1.3 show a table with pairs for bitrate and resolution. Now based on this table 5800 kbps should be good enough for generating a stream of FULL HD resolution(1080p). But certain scenes might contain fluctuating luminance for a region in the frames which leads to camera noise. The camera noise distorts the quality of the the scene that results into blockiness artifacts in the noisy region of the frames. On the other hand for generating Full HD stream for a video containing cartoon content 1750 Kbps is more than enough, so utilizing 5800 Kbps is not an optimal solution and would lead to bandwidth wastage. [Aar+15].

In case of Netflix, the videos are not just encoded using conventional constant bitrate video encoders. Netflix encodes the videos by using multiple coded representations (with multiple bitrates) of a scene for different resolutions and the encoded segments are delivered, based on the request to the end-user. [Kat18]It allows the end-users to stream high quality video with low bitrate and provides the end-users seamless playback of the content. The reason behind achieving this is subjective. From the point of view of an end-user who can afford good network bandwidth would not like to have the blurriness or blockiness in the content. This requires the development of algorithms that could deal with the issues of artifacts in videos encoded with low bitrates. The content of a given video differs with other videos more or less, as it can have scenes with more of constant background covering large area of frames or it can



Bitrate (kbps)	Resolution
235	320x240
375	384x288
560	512x384
750	512x384
1050	640x480
1750	720x480
2350	1280x720
3000	1280x720
4300	1920x1080
5800	1920x1080

Figure 1.3: Bitrate-Resolution pairs or Bitrate Ladder [Aar+15]

be a dynamic scenes based on lot of motion involved i.e., rapid change in the frame content over time ot it can have fluctuating intensity values for pixels over the frames in the scene. When using constant bitrate technique, the dynamic scenes may utilize the bits efficiently but constant scenes where the region of the frame is not changing too much, the bits allocated for those frames would be more than required, thus leading to over utilization of bits .The worst case scenario would be a scene where it is completely dark which does not require as many number of bits as a scene with frames containing varying pixel intensity. The figure 1.4 demonstrates how the quality varies for different video sources. Certain sources never reach high quality even though the bitrate is high enough as 25000 kbps.

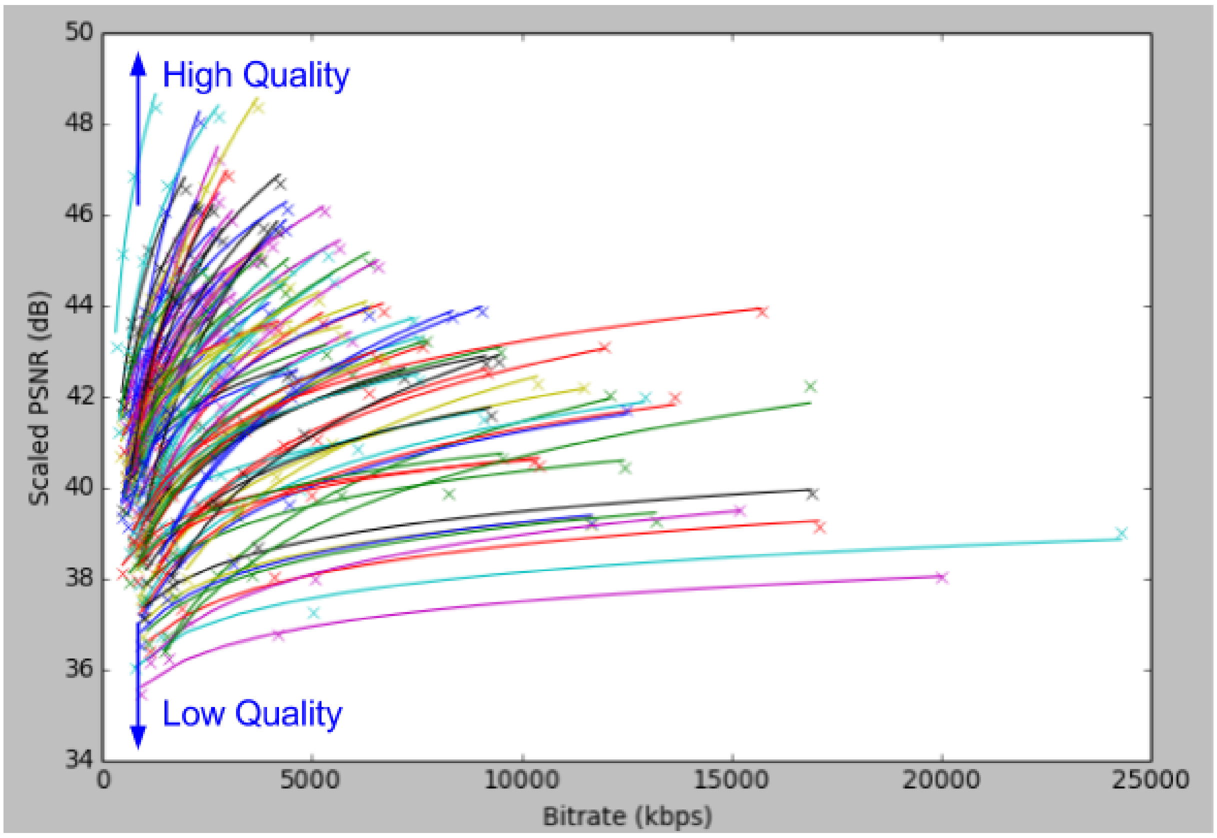


Figure 1.4: 100 randomly sampled sources at 1080p resolution using x264 constant QP (Quantization Parameter) rate control. [Aar+15]

Also the process of achieving best quality goes further into encoding each of chunks based on the scenes. In a given video sequence, the set of frames where the presentation is continuous while remaining only in one place is called a scene.[CLG07] Every video sequence can be subdivided onto multiple scenes. As each of the scenes have different characteristics, so the idea is to encode them individually with required bitrates for different resolutions based on the subjective quality.

The metrics like PSNR (Peak Signal to Noise Ratio) [Psn], SSIM (Structural Similarity Metric)[Wan+04], VMAF (Video Multi-Method Assessment Fusion) [Blo18] are used for video quality evaluation. The videos encoded for each quality with all possible bitrates are evaluated. Finally those bitrates are chosen which provides high quality amongst the different resolutions on a rate-distortion or rate-quality curve.

1.1 Motivation & Goals

Section 1 describes about how Netflix approaches to encoding a video by subdividing it into segments ans then encode them individually to provide a good quality of experience from subjective point of view. This pipeline includes segmenting a video first. Next each of these segments are assessed and divided into scenes. These scenes are based on the type of content. These scenes have relatively similar kind of frames which makes it easier to encode them with low bitrate compared to encoding a video with heterogeneous content. For this Netflix applies encoding with all possible bitrates for each resolution [Blo17]. Then based on the quality parameters obtained from a rate distortion plot the appropriate bitrate is chosen for each resolution. Netflix applies a brute force technique in order to compute quality parameter for each bitrate for all the resolutions. This requires intensive computational resources. This leads to the research question if there is a way to reduce the overall computation costs and approximate this whole process using optimization techniques.

To have such comparison both the approaches will be implemented; a) brute force technique of computing quality parameters by computing quality for each bitrate for different resolutions to generate a rate-distortion curve and b) selecting fewer points from the generated rate-distortion curve and then trying to approximate using mathematical equations e.g., a logarithmic curve.

The motivation here is to do the computation for less number of bitrates and try to obtain other points by the means of any approximation method. This will include interpolation and extrapolating the remaining bitrate points to generate a rate-distortion curve that would resemble closely to the crate-distortion curve genrated using brite-force technique. Then the overall number of encoding steps would be reduced. The next step would be to compare the resulting points after approximation with points from the brute-force approach. Further analysis would provide error between points obtained using approximation approach and points from the brute-force approach. Based on the error the plausibility of the approach to minimize the overall encoding steps can be analyzed. The smaller steps for resolution will be considered and corresponding to each resolution, the computation of the quality parameters(QP) for all possible bitrate will be done.

1.2 Thesis Structure

In order to provide the information regarding the work done and the contributions in this master thesis, the documentation is structured in the following manner:

- ▷ Chapter 1 addresses about current overview of encoding pipeline used by Netflix and the challenges about setting it up. In addition to that how the challenging aspects of Netflix pipeline could be replaced using approximation methods in order to achieve reasonably similar results which is the main motivation of this thesis topic.
- ▷ Chapter 2 describes about principles of encoding scheme, video quality metrics and tools already available to set up the pipeline for different approaches for transcoding video

sequences. Additionally it includes theories about previous work done with regards to video encoding optimization.

- ▷ Chapter 3 focuses on the various encoding pipelines used to generate the rate-quality curves for the video sequences by transcoding them with multiple bitrate points for various resolutions. The approaches being the pipeline for brute-force technique used by Netflix, Bisection approximation technique and finally the approximation technique based on the mathematical equation of logarithmic curve. Apart from that it also addresses the challenges that were encountered in each of the approaches including the pros and cons with respect to the approaches.
- ▷ Chapter 4 is about analysis and evaluation of the results obtained after the implementation of the approximation method of mathematical equation, where the resulting curves describes how plausible is the approximation approach of mathematical equation with respect to brute-force technique.
- ▷ Finally Chapter 5 concludes about the approximation approach and comments about overall work done and finally providing some future work recommendations.

Chapter 2

Fundamentals

2.1 Related Work

As described in the previous chapter 1 about Netflix encoding the videos, firstly using the different encoding strategy for each video and then further using granular approach to optimize each of the scenes for a the given video. From the point of view of adaptive streaming, the problem was not about encoding a single scene for each video but rather how to do it for collection of all the scenes with all the possible bitrate/qualities range that the user would desire the service to receive. This led to development of a framework which considered all the above points from Netflix Technology called Dynamic Optimizer [Kat18]. This framework analyzes an entire video over multiple quality and resolution pairs, finds the best trade off to generate optimal scheme to encode a scene. [Man+18] The idea behind finding the optimal scheme to encode a video sequence is that a video consists mainly of numerous shots, so each of shots should be encoded independent of other shots. These shots have little correlation with each other which can be treated an element that can be encoded independently of each other. Each of the shots to be encoded can be considered as visual content which is uniform and homogeneous in sense of the object or the actor the scene is focused on. This means applying constant or fixed quality for an individual shot fits well. From the perception point of view, viewers are not bothered by the coding difference that would occur at the boundaries of adjacent shots i.e., whenever a there is a scene or a shot change it does not affect the subjective quality of experience. It provides a good deal to adjust the coding resolution and quantization parameter at such points of shot boundaries. The decision of choosing optimal resolution and quantization parameter to be used for each of the shots to be encoded, is done on the basis of perceptual video quality metrics like SSIM (Structural Similarity) and VMAF (Video Multimethod Assessment Fusion). Dynamic Optimizer uses cloud computing to generate multiple encodings for a given set of frames from a shot for various resolutions and quantization parameters.[KG18] In this thesis topic the idea is similar to setup a pipeline

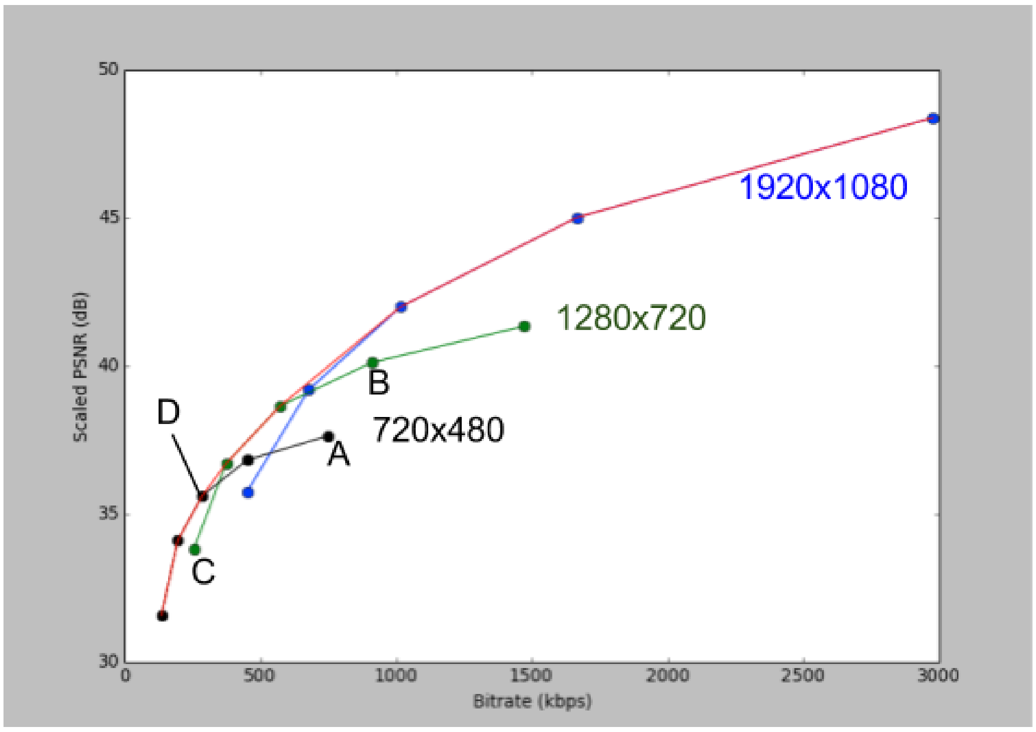


Figure 2.1: Example encodes showing individual R-D curves and convex hull.[Aar+15]

for video encoding. So in the similar way the videos will be split into shots. Each of the shots will be reproduced for various resolutions.Each of those resolutions of the scene will be coded with multiple bitrate values producing encodes for each bitrate value. Each of the encodes will be then tested with VMAF for VMAF score (quality parameters) for each bitrate for a given resolution. For each bitrate and its corresponding VMAF score a plot will be generated. The plot would be similar to the Figure 2.1. The labels in the plots are the curves represented in the plot. Curve **A** (in black) is the plot generated using scaled PSNR values as points (R_i, Q_i - set of bitrate points and its corresponding quality score/ parameter for a given resolution) for various bitrates for the resolution 720x480. Similarly curve **B** (in green) is generated using scaled PSNR values for various bitrates for the resolution 1280x720 and finally curve **C** is generated using scaled PSNR values. The red curve **D** is the convex hull obtained over all the curves from the resoltions and its R_i, Q_i points. This curve is used as the basis to select the points for a given bitrate. Although the quality parameters is generated using traditional metric of scaled PSNR in the figure, this thesis topic will include quality parameters generated using VMAF. As discussed in the previous section, the next step will include approximation technique where a fewer points for a curve will be selected and then the rest of the points will be approximated.

The transcoding procedure used by Netflix requires to encode all video sequences with all possible bitrate settings. Netflix uses transcoding pipeline in the cloud based services to transcode the videos. Transcoding each video with unique set of settings where each of its segments are transcoded with various bitrates settings requires large amount of computational and storage resources. The authors from [KZS15] discussed about how large number of transcoded video segments are never used which leads to wasteful management of storage and computation. They have described about a possible solution that when a user requests a segment of a video, the bitrates with which the segment is transcoded can be predicted using a Markov model predictor. [OR02] They have also described that based on the Netflix study of codec and bit-rate combinations for a single segment can result up to 120 transcoding operations before it is finally delivered to the client platforms. For a given video of a regular television show of about one hour, assuming having scenes on an average duration of 10 secomds will have 360 video segments. Applying 120 transcoding operations for each of those scene or a video segment is not feasible approach. They have shown that based on the the next video segment requested by the user, the corresponding bitrate with which the segment is to be encoded, can be predicted. The main objective of the authors being to transcode video segment with only select settings of bitrate and quality. In this thesis the motivation is quite similar where instead of transcoding a given video segment with multiple bitrates the idea is to transcode for fewer bitrate points for a given resolution and then try to approximate remaining bitrate points based on the mathematical approximation of curve generated using the few select bitrate points. Therefore reducing the computational complexity of trancoding pipeline.

The authors from [Sat+19] presented an alternative to per-scene video encoding done by Netflix. They used the CRF (Constant rate factor) encoding recipe for 3 resolutions and for each of those resolutions they computed 4 different (R_i, Q_i) points. For generating quality points for each of the encodes, they used a full reference metric of PEVQ (Perceptual Evaluation of Video Quality) from OPTICOM [Pev]. The authors tried to explain how the function of (R_i, Q_i) can be used to model the prediction of PEVQ quality scores by using the combination of logistic and inverse exponential functions. Using this approach they try to compute all the quality points for all bitrates for a given resolution and finally getting all computations for a given video. Their approach also takes into consideration the tradeoffs between bitrate and quality. In this thesis topic the approach is quite similar. The mathematical approximation would be done using a logarithmic curve instead which is same as an inverse exponential function. The logarithmic curve will be fitted with respect to fewer points selected from (R_i, Q_i) pairs for every given resolution.

2.2 Two-Pass Encoding

The pipeline used for the brute force technique by Netflix requires to transcode video at variable bitrates for every resolution of a shot (or a scene) from a given video. The transcoding scheme

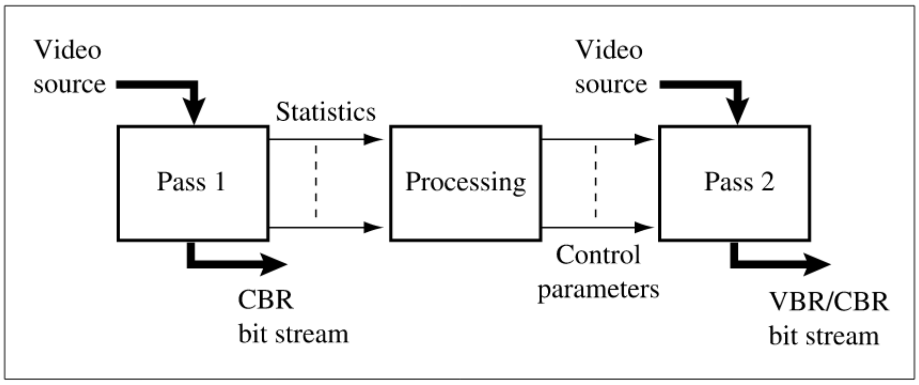


Figure 2.2: Schematic for Two-Pass Video Encoding System.[WRG99]

that is used in this thesis topic is Two-Pass Video Encoding. The two-pass encoding scheme is a for non-real time applications that enables processing a video in two passes. The first pass usually analyzes the given video sequence and generates the information about the statistics regarding statistics about the each element of the video. The results obtained from the first pass is then used to optimally encode the video in the second pass. This kind of video encoding uses variable bitrate (VBR) to encode the different elements of the video unlike the CBR encoding where the whole video is encoded and compressed using a constant bitrate (CBR). In the CBR encoding the rate remains constant but the quality of the the video sequence keeps varying based on the varying content. During the first pass the video sequence is encoded using the conventional CBR and at the same time the statistics data about the coding the elements of the video is are obtained. The first pass statistics are processed to produce the control parameters for the second pass.The second pass is then used to generate VBR compressed stream of the the given video sequence using the first pass statistics data. The control parameters which are generated between the first pass and the second pass are used distribute the bits appropriately to the different elements of the video such that at the end the video is encoded with variable rate and a constant quality.[WRG99]

Figure 2.2 shows the overview of a two pas encoding system. As shown in the figure, the first pass is responsible for applying constant rate CBR encoding to the given video sequence. The statistics obtained from this pass are then processed after the video sequence. The processing of the statistics ids done after gathering the data from the complete video sequence in the first pass. So the processing for statistical data cannot be done before. Based on the statistical data processed after the first pass, the control parameters are computed. In the second the video is encoded using the computed control parameters for each of the elemental parts of the video. Thus the final video encode is optimized CBR sequence or VBR stream. [WRG99]

So to transcode a given video sequence requires a tool called FFMPEG [Ffmb]. The following section describes the overview about the FFMPEG tool.

Command Tool	Usage
ffmpeg	is used for converting audio or video formats
ffplay	is a command line media player that uses Simple Directmedia Layer(SDL) and FFMPEG libraries
ffprobe	displays text based media information in formats like XML, CSV etc

Table 2.1: FFMPEG Command Line Tools

2.3 FFMPEG

FFMPEG is an open source cross platform tool which allows for handling audio and video data by the means on encoding, decoding, transcoding, multiplexing, demultiplexing etc. This tool allows for many options for customizing video data by the means of its capabilities. [Ffmb]

FFMPEG provides various command line tools, which is described in the table 2.1.

FFMPEG also provides libraries for various applications described in the table 2.2.

Libraries	Usage
libswresample	provides functions to resample audio
libavresample	provides functions to resample audio from Libav project [tea]
libavcodec	provides native FFmpeg audio/video encoders and decoders
libavformat	provides demuxers and muxers for audio/video container formats
libavutil	includes hash functions like SHA-1, LZO decompressor and Base64 encoder/decoder.
libpostproc	provides functions for old H.263 video postprocessing
libswscale	provides functions for video image scaling and colorspace/pixelformat conversion
libavfilter	is the substitute for vhook which allows the video/audio to be modified or examined between the decoder and the encoder

Table 2.2: FFMPEG Libraries

After the video is processed and encoded, the quality estimation has to be done. There are numerous schemes and metrics available to compute the quality of a video based on the quality of a video. The focus of this thesis for the video quality estimation is based on a full-reference metric introduced by Netflix and later made open-source framework. The metric is called Video Multi-method Assessment Fusion or VMAF. The following section describes it in details.

2.4 Video Quality Metric: Video Multi-Method Assessment Fusion (VMAF)

VMAF was developed by Netflix along with the external researchers which uses machine-learning approach. The idea is to fuse together the elementary. The elementary metrics are the highly efficient objective video quality metrics based on the human perception. The first set of elementary metrics are chosen based on the image quality metric called Visual Information Fidelity (VIF). This metric provides five different quality scores for each resolution of a given image. The second set of elementary metrics are chosen based on quantification of structural loss which occurs due to blurriness and encoding compression called Detail Loss Measure[LMN12] (DLM). VIF and DLM are both image based quality metric where it computes the quality based on the spatial features. Finally the feature based on temporal information is obtained based on the temporal change in the information i.e., between the consecutive frames, the difference in the pixel is calculated. For this, the luminance component for each of the corresponding pixel is taken into consideration. This is same as the TI (Temporal Indicator)[Win12].[KG18].

All the features based on quality metrics computed above are then fused through a Support Vector Machine (SVM) regressor [CV95]. The regressor produces a video quality score in the range [1.100], where 0 means the lowest quality to 100 which means nearly perfect quality. The regressor uses the parameters based on the collection of subjective testing based on full reference technique i.e., the users rated the quality of reference and the distorted videos respectively. [KG18]

VMAF is a full reference quality metric. As compression and scaling several artifacts result in the video so VMAF uses the existing perceptual quality scores from multiple quality assessment algorithms to estimate the quality of a given video. This is fused with the support vector machine. As it is a full reference metric, the original source and encoded/compressed source both are taken in to consideration. For multiple encoded versions of a source video the comparison is done using a scaled VMAF score. [Ras17][KG18].

The figure 2.3 shows VMAF system diagram. The studies show that VMAF quality score comes really close to human perception for video quality in terms of adaptive streaming where artifacts are caused mainly by scaling and compression. Netflix has made the VMAF project as

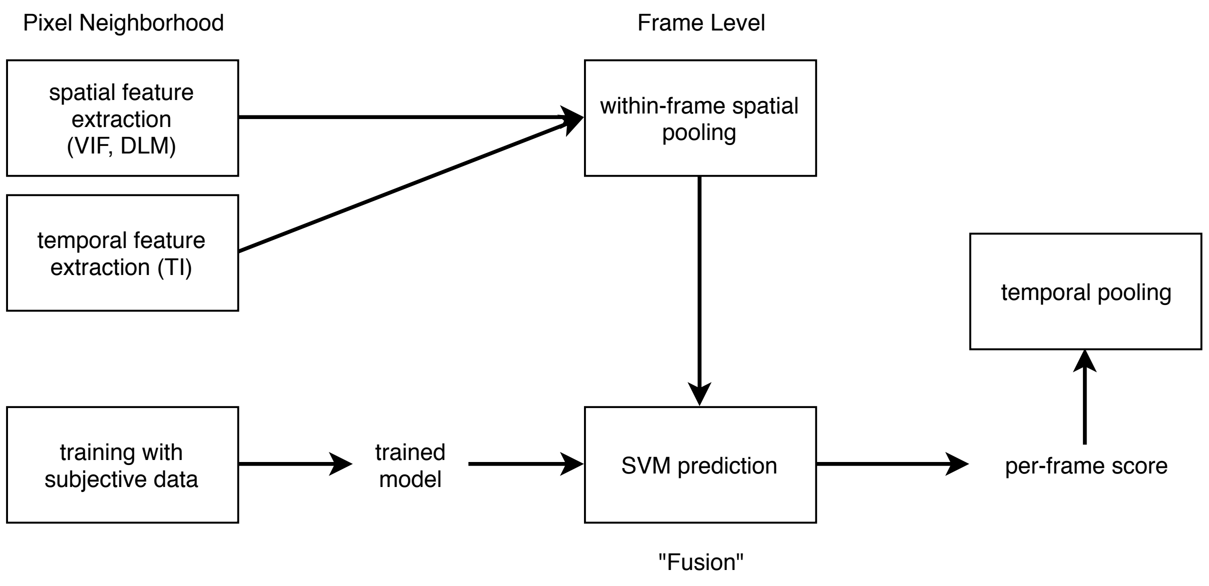

Pixel Neighborhood
Frame Level
spatial feature extraction (VIF, DLM)
temporal feature extraction (TI)
within-frame spatial pooling
temporal pooling
training with subjective data
trained model
SVM prediction
per-frame score
"Fusion"

Figure 2.3: High Level VMAF System Diagram [KG18]

an open-source projects which means researchers and developers can contribute to this project for further improvement and new use-cases. [KG18]

Chapter 3

Design & Implementation

3.1 Theoretical System Design

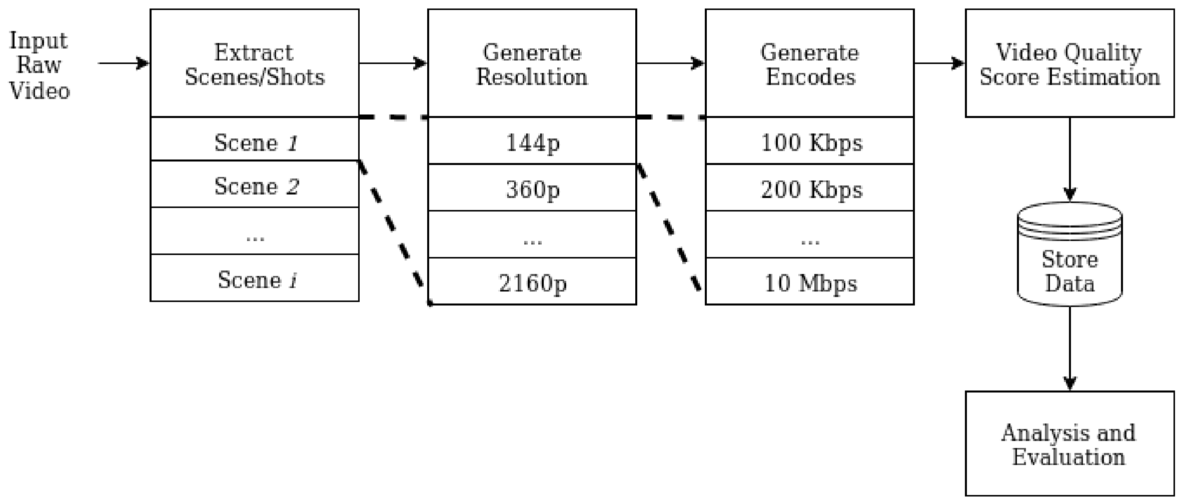


Figure 3.1: Overview of Theoretical Video Encoding Pipeline

The initial idea was to implement set of algorithms for various operations to transcode a given video sequence. Figure 3.1 provides the general overview of the encoding pipeline. As described in previous chapters about how Netflix uses various transcoding operations about how a given video sequence is transcoded with all possible bitrate settings for each resolutions. Similarly in the figure ?? each of blocks represent the operations that takes place for transcoding a video.

Following subsections will describe about the pipeline and how each of the blocks are responsible and for what operations.

3.1.1 Extract Scenes/Shots

A raw video is input to the pipeline and at first operation of extracting the scenes is performed. In this phase a given video sequence is analyzed to find the scenes. The scenes are analyzed by the means of change in correlation of adjacent frames. Using a threshold value to find the difference in the correlation, the scene or shot boundaries can be identified. There are various approaches available which goes from simple fade in fade out changes to content based detection to find the scene boundaries in a given video sequence.

3.1.2 Generate Resolutions

In the previous phase, a given video sequence is subdivided into multiple scenes or shots. Now in this phase for each of the scenes, different resolutions are generated. For every scene Scene-*i*, its different resolutions are generated. The resolution settings could range from the lowest resolution of 144p upto 1440p (QHD). Please note that for the highest quality of 2160p is not produced by down-scaling but instead the original scene will be taken into consideration.

3.1.3 Generate Encodes

This is in general overview about transcoding phase describing how each video is dealt with in the pipeline. After down-scaling the scenes from previous phase to various resolutions, each of those resolutions are then subjected to transcode with bitrate settings ranging from 100kbps to 10Mbps. For each of the resolutions, several bitrate transcoded encodes are generated. So to summarize, a given video sequence is subdivided into *'X'* number of chunks called the scenes or shots, then for each of the scenes, *'Y'* number of resolutions are generated and then finally for each of the resolution, *'Z'* number of encodes are generated. So, at the the end, for a given video sequence there will be a total of *X*Y*Z* number of encodes are generated.

3.1.4 Video Quality Score Estimation

After the encodes are generated, each of them have to be analyzed using a video quality metric. The metric to be used in this thesis is VMAF from Netflix which is a full reference metric. The original split scene will be used as reference for all its resolutions with multiple encode generated using various bitrate settings. For every encode, a score value is generated. The resultant scores are segregated based on encodes corresponding to each resolution. Finally the results are stored for further evaluation.

3.1.5 Analysis and Evaluation

The scores which were generated from the encodes are kept separately for each of the resolutions. This is done to obtain a rate-distortion curve. Each rate-distortion curve corresponds to each resolution which would describe the change in quality as the bitrate increases. It is anticipated that as the bitrate increases the quality score would initially increment quickly and later the increment would away. The curve would resemble to the Figure 2.1 from section 2. In the same manner, the rate-quality curve will be generated for all the other resolutions.

In this thesis, the pipeline was assumed to take into consideration a single scene for each of the different video sequences. The video sequences were raw videos with resolution 2160p. The main reason for processing only a scene was to test the idea of shot based encodes used by Netflix [Man+18]. As each shot has its own background and foreground properties it can be encoded based on the complexity of its content optimally as described by Netflix Tech blog in [Man+18]. So the first objective was to extract scenes or shots as described in the figure 3.1.

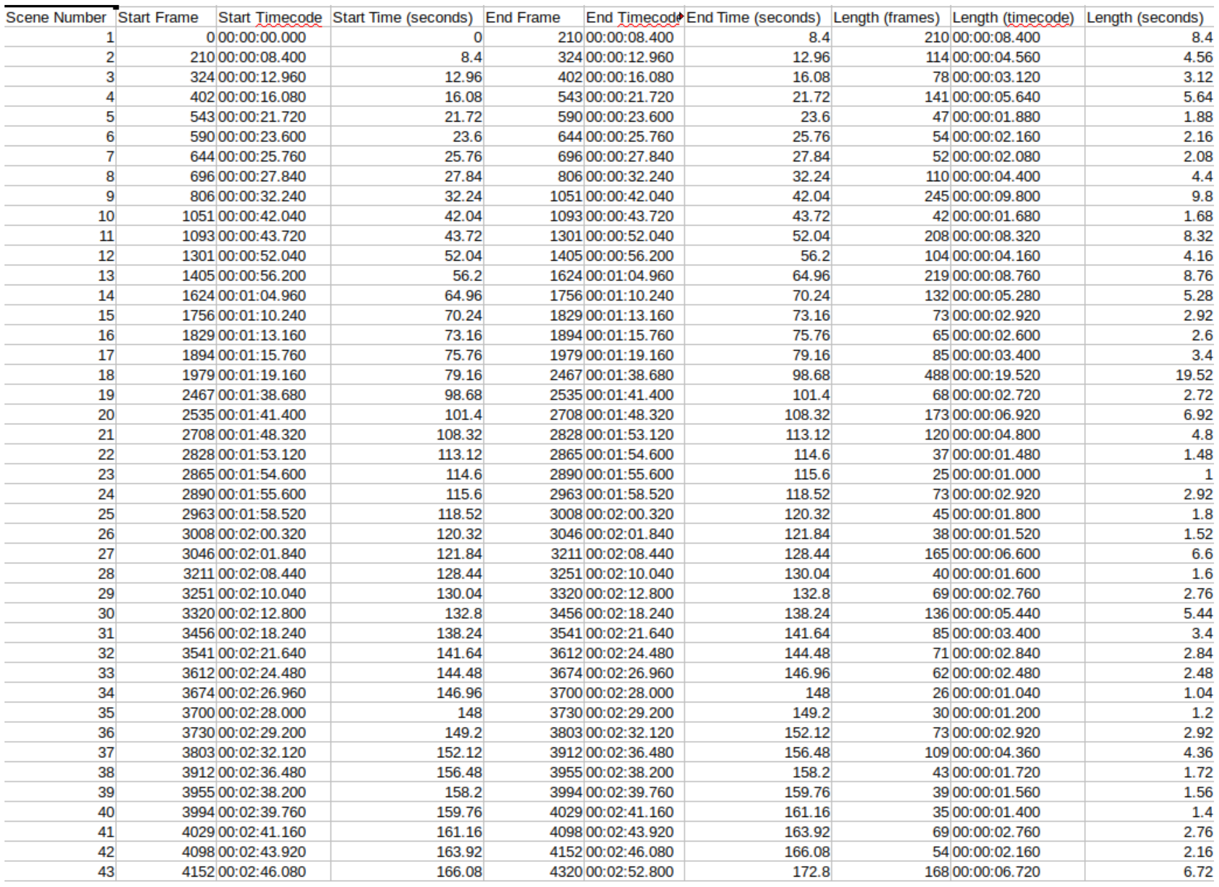
As described before the scene detection can be done based on threshold between adjacent frames or content based difference in adjacent frames. Some of the tools available for detecting scenes in a given video sequence are as follows:

- ▷ FFMPEG blackframe filter [Ffma]
- ▷ Shotdetect [Can]
- ▷ Matlab Scene Change Detetction [**matlabcsd**]
- ▷ PySceneDetect [**pyscenedetect**]

FFMPEG blackframe filter is video scene detection tool based on threshold technique where the intensities of adjacent frames are correlated to find the scene or shot boundaries. The drawback is that this tool is limited to threshold only. If there are consecutive scenes which are dark then this tool may fail.

On the other hand *Shotdetect* is another video scene detection tool which finds scene for a given video sequence based on content change only. This tool works on the basis of change in the motion in the consecutive frames. The change in attributes of RGB (Red, Green and Blue) and HSV (Hue, Saturation and Value) intensity is also taken into consideration for finding scene or shot boundaries. [Joh]

Matlab Scene Change Detection is yet another sophisticated video segmenting tool to find scenes or shots in a given video sequence. This tool is based on a model which uses edge detection techniques to obtain features for adjacent features. The frame is then processed block-wise to compare features in the corresponding blocks of adjacent frames. If the change



Scene Number	Start Frame	Start Timecode	Start Time (seconds)	End Frame	End Timecode	End Time (seconds)	Length (frames)	Length (timecode)	Length (seconds)
1	0	00:00:00.000	0	210	00:00:08.400	8.4	210	00:00:08.400	8.4
2	210	00:00:08.400	8.4	324	00:00:12.960	12.96	114	00:00:04.560	4.56
3	324	00:00:12.960	12.96	402	00:00:16.080	16.08	78	00:00:03.120	3.12
4	402	00:00:16.080	16.08	543	00:00:21.720	21.72	141	00:00:05.640	5.64
5	543	00:00:21.720	21.72	590	00:00:23.600	23.6	47	00:00:01.880	1.88
6	590	00:00:23.600	23.6	644	00:00:25.760	25.76	54	00:00:02.160	2.16
7	644	00:00:25.760	25.76	696	00:00:27.840	27.84	52	00:00:02.080	2.08
8	696	00:00:27.840	27.84	806	00:00:32.240	32.24	110	00:00:04.400	4.4
9	806	00:00:32.240	32.24	1051	00:00:42.040	42.04	245	00:00:09.800	9.8
10	1051	00:00:42.040	42.04	1093	00:00:43.720	43.72	42	00:00:01.680	1.68
11	1093	00:00:43.720	43.72	1301	00:00:52.040	52.04	208	00:00:08.320	8.32
12	1301	00:00:52.040	52.04	1405	00:00:56.200	56.2	104	00:00:04.160	4.16
13	1405	00:00:56.200	56.2	1624	00:01:04.960	64.96	219	00:00:08.760	8.76
14	1624	00:01:04.960	64.96	1756	00:01:10.240	70.24	132	00:00:05.280	5.28
15	1756	00:01:10.240	70.24	1829	00:01:13.160	73.16	73	00:00:02.920	2.92
16	1829	00:01:13.160	73.16	1894	00:01:15.760	75.76	65	00:00:02.600	2.6
17	1894	00:01:15.760	75.76	1979	00:01:19.160	79.16	85	00:00:03.400	3.4
18	1979	00:01:19.160	79.16	2467	00:01:38.680	98.68	488	00:00:19.520	19.52
19	2467	00:01:38.680	98.68	2535	00:01:41.400	101.4	68	00:00:02.720	2.72
20	2535	00:01:41.400	101.4	2708	00:01:48.320	108.32	173	00:00:06.920	6.92
21	2708	00:01:48.320	108.32	2828	00:01:53.120	113.12	120	00:00:04.800	4.8
22	2828	00:01:53.120	113.12	2865	00:01:54.600	114.6	37	00:00:01.480	1.48
23	2865	00:01:54.600	114.6	2890	00:01:55.600	115.6	25	00:00:01.000	1
24	2890	00:01:55.600	115.6	2963	00:01:58.520	118.52	73	00:00:02.920	2.92
25	2963	00:01:58.520	118.52	3008	00:02:00.320	120.32	45	00:00:01.800	1.8
26	3008	00:02:00.320	120.32	3046	00:02:01.840	121.84	38	00:00:01.520	1.52
27	3046	00:02:01.840	121.84	3211	00:02:08.440	128.44	165	00:00:06.600	6.6
28	3211	00:02:08.440	128.44	3251	00:02:10.040	130.04	40	00:00:01.600	1.6
29	3251	00:02:10.040	130.04	3320	00:02:12.800	132.8	69	00:00:02.760	2.76
30	3320	00:02:12.800	132.8	3456	00:02:18.240	138.24	136	00:00:05.440	5.44
31	3456	00:02:18.240	138.24	3541	00:02:21.640	141.64	85	00:00:03.400	3.4
32	3541	00:02:21.640	141.64	3612	00:02:24.480	144.48	71	00:00:02.840	2.84
33	3612	00:02:24.480	144.48	3674	00:02:26.960	146.96	62	00:00:02.480	2.48
34	3674	00:02:26.960	146.96	3700	00:02:28.000	148	26	00:00:01.040	1.04
35	3700	00:02:28.000	148	3730	00:02:29.200	149.2	30	00:00:01.200	1.2
36	3730	00:02:29.200	149.2	3803	00:02:32.120	152.12	73	00:00:02.920	2.92
37	3803	00:02:32.120	152.12	3912	00:02:36.480	156.48	109	00:00:04.360	4.36
38	3912	00:02:36.480	156.48	3955	00:02:38.200	158.2	43	00:00:01.720	1.72
39	3955	00:02:38.200	158.2	3994	00:02:39.760	159.76	39	00:00:01.560	1.56
40	3994	00:02:39.760	159.76	4029	00:02:41.160	161.16	35	00:00:01.400	1.4
41	4029	00:02:41.160	161.16	4098	00:02:43.920	163.92	69	00:00:02.760	2.76
42	4098	00:02:43.920	163.92	4152	00:02:46.080	166.08	54	00:00:02.160	2.16
43	4152	00:02:46.080	166.08	4320	00:02:52.800	172.8	168	00:00:06.720	6.72

Figure 3.2: Scene Information in CSV Format of a Random Video Sequence using PySceneDetect

in corresponding blocks for adjacent frame exceeds the specified threshold for the blocks, it is then able to determine the scene boundary.

Finally **PySceneDetect**, a Python [Pyt] based command line tool which provides various operations for scene detection in video sequences. It is an open source project being contributed by other users on Github [Bre19]. PySceneDetect uses various features based detection methods to find scenes in a given scene. The methods are based on various techniques of threshold only technique similar to ffmpeg blackfilter to advanced content aware detection of a scene or shot. The tool utilizes both open source computer vision libraries of OpenCV [Ope] and FFMPEG libraries.

In this thesis, *PySceneDetect* is used for finding scene or shot boundaries for any given video. The reason is that it allows for scene detection based content and threshold both. Secondly it is open source project whereas *Matlab Scene Change Detect tool* requires purchasing the Matlab and Simulink toolkits. Figure 3.2 shows the result in a tabular form when an input video sequence is used with PySceneDetect. After finalising the scene detection tool, initial setup was designed.

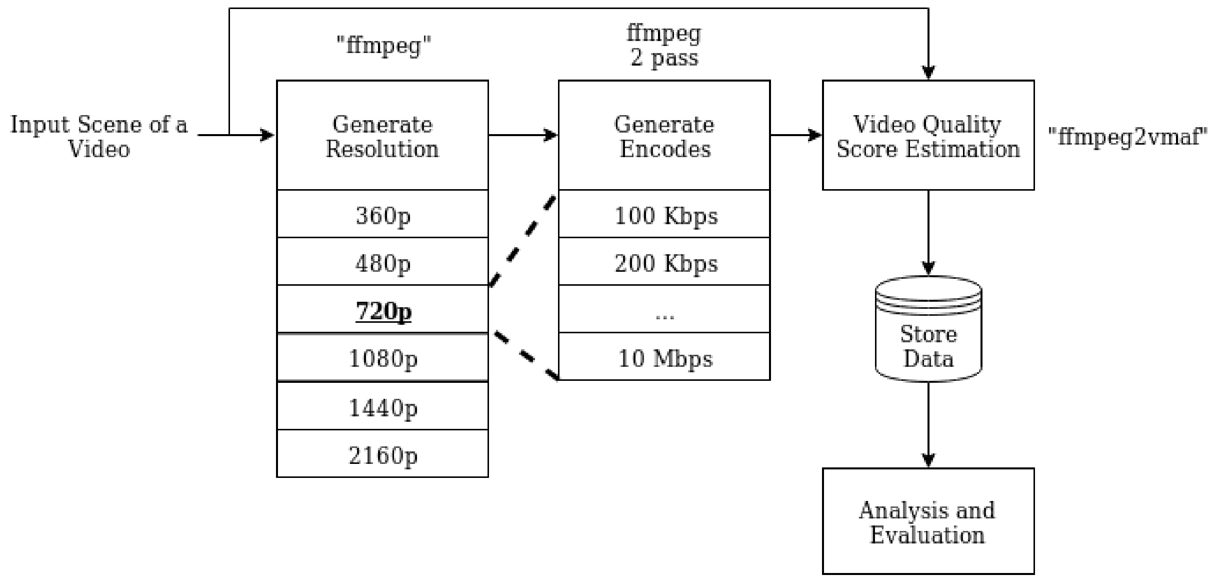


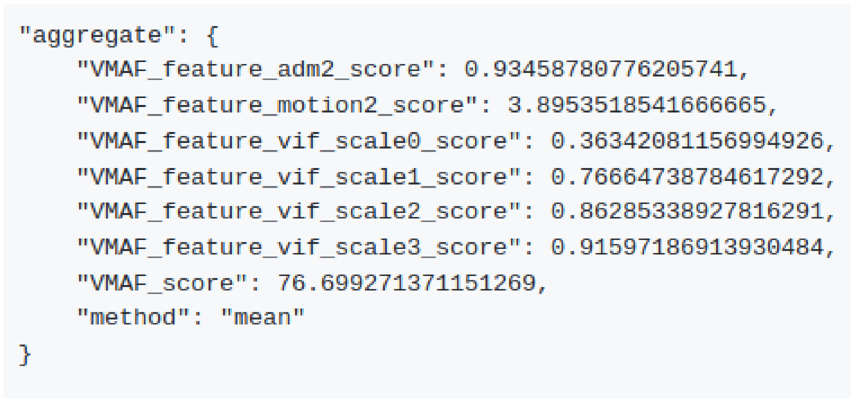
Figure 3.3: Video Encoding Pipeline for only 720p Resolution

3.2 Initial Design Setup

The motivation with this design was to test out if encoding a single resolution with different bitrate settings would yield out the quality scores similar to one of the rate-distortion curves from figure 2.1. The encodes were generated using bitrates ranging from 25 kbps till 10 Mbps. A total of 67 points in between 25 Kbps to 10 Mbps were chosen. The initial 29 points ranged between 25 Kbps to 750 Kbps with interval of 25 Kbps. The rest of the points ranging from 750 Kbps till 10 Mbps had an interval of 250 Kbps. The reason for having higher resolutions of bitrate points was that the Rate-Distortion curve tends to rise quickly in the lower bitrate region and then flattens in higher bitrate region.

The figure 3.3 shows the overview of the video encoding pipeline used for the approach for encoding a single resolution of 720p. First an scene is inputted to the pipeline and the first block generates its different resolutions from 360p, 480p, 720p, 1080p, 1440p till 2160p i.e., 6 resolution versions of the scene. Then the 720p resolution of the scene is forwarded to next block where the scene is transcoded using several bitrate points as described above for 67 different bitrate points.

The encodes generated are then passed to the next block Video Quality Score Estimation. Each of the encodes are analyzed with respect to original scene to estimate the quality. The metric as already described is VMAF from Netflix. This open source project provides various command line tools for video quality estimation. One of the command line tools which is used here is "ffmpeg2vmaf". This command line tool is provided with the original video and encoded video as arguments. The tool compares the encoded video with the original video and generates results in various format such as json, text and xml. One of the sample output



```
"aggregate": {
    "VMAF_feature_adm2_score": 0.93458780776205741,
    "VMAF_feature_motion2_score": 3.8953518541666665,
    "VMAF_feature_vif_scale0_score": 0.36342081156994926,
    "VMAF_feature_vif_scale1_score": 0.76664738784617292,
    "VMAF_feature_vif_scale2_score": 0.86285338927816291,
    "VMAF_feature_vif_scale3_score": 0.91597186913930484,
    "VMAF_score": 76.699271371151269,
    "method": "mean"
}
```

Figure 3.4: Sample VMAF Output[Netb]

can be seen in the figure 3.4. The only result that was used in this thesis is the VMAF_score, the second last entry in figure 3.4. The other generated values are the features extracted from the encode after comparing with the original scene.

The Figure 3.5 shows the schematic of the plot generated with quality scores generated using VMAF metric against the bitrates used for encoding the scene. The curve rises quickly in the region of bitrates from 25 Kbps to 2 Mbps, then it starts to flatten. It shows that quality does not increase as much as it increases in lower bitrate points.

The rate vs quality curve in figure 3.5 turned out to be as expected. Next step was to setup the pipeline for all the resolutions of the video.The figure 3.6 shows the pipeline used in determining rate quality curve for all resolutions. Each of resolutions generated are subjected to 67 different bit rate points. Each of the encodes are then compared to the original scene using VMAF tool called "ffmpeg2vamf". The results for each of the encodes are separated based on the resolutions. Encodes and their scores were grouped together while analyzing the quality of the encodes. After the evaluation of the scores computed for the corresponding encodes, the plot was generated.

The figure 3.7 shows rate vs quality curve for all the resolutions. The observation at curves of different resolution shows the characteristics about the intersection points between curves that can be used to generate convex hull above all the curves for individual resolutions. The curve generated using VMAF scores for each bitrate points corresponding to each of the resolutions are way too much detailed. In order to have clear picture, some of the extra points can be ignored and plot only the remaining points of rate-quality pair corresponding to each resolution. Consecutively this would provide much clear curves. For a given bitrate, if there exists quality points for more than one resolutions then the low resolution quality points must be ignored because higher resolution with high quality for a given bitrate would offer more quality of experience in the subjective sense for a viewer. After trimming down the irrelevant

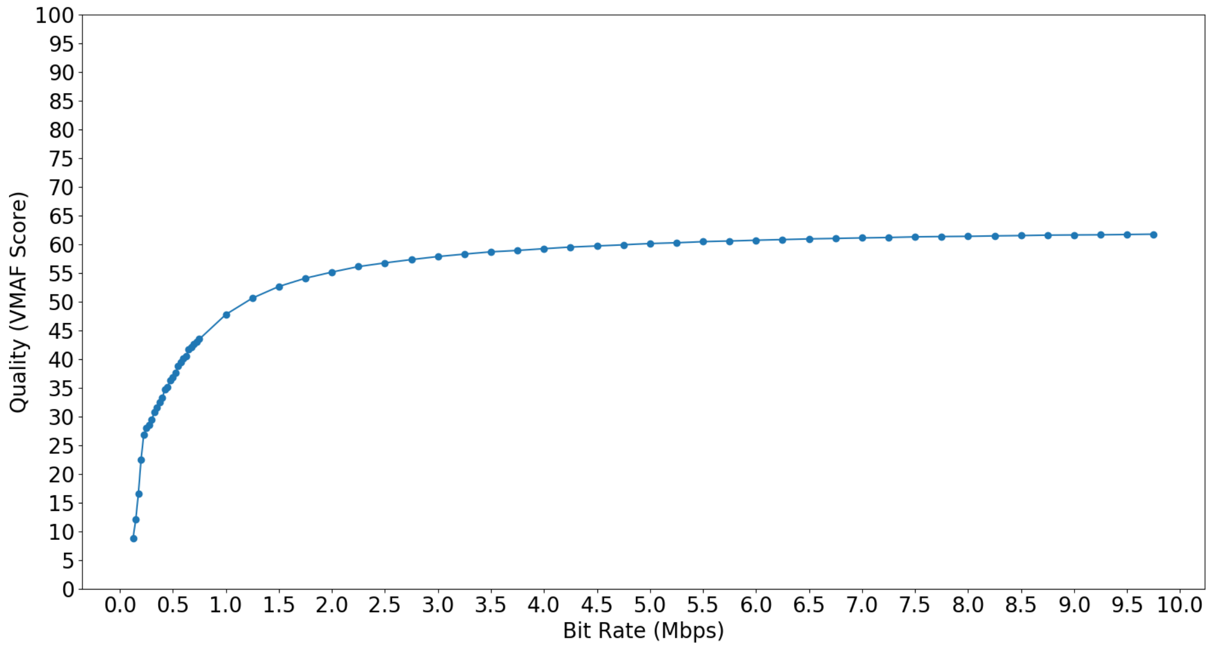


Figure 3.5: Rate vs Quality Curve at 720p resolution over different Bitrates

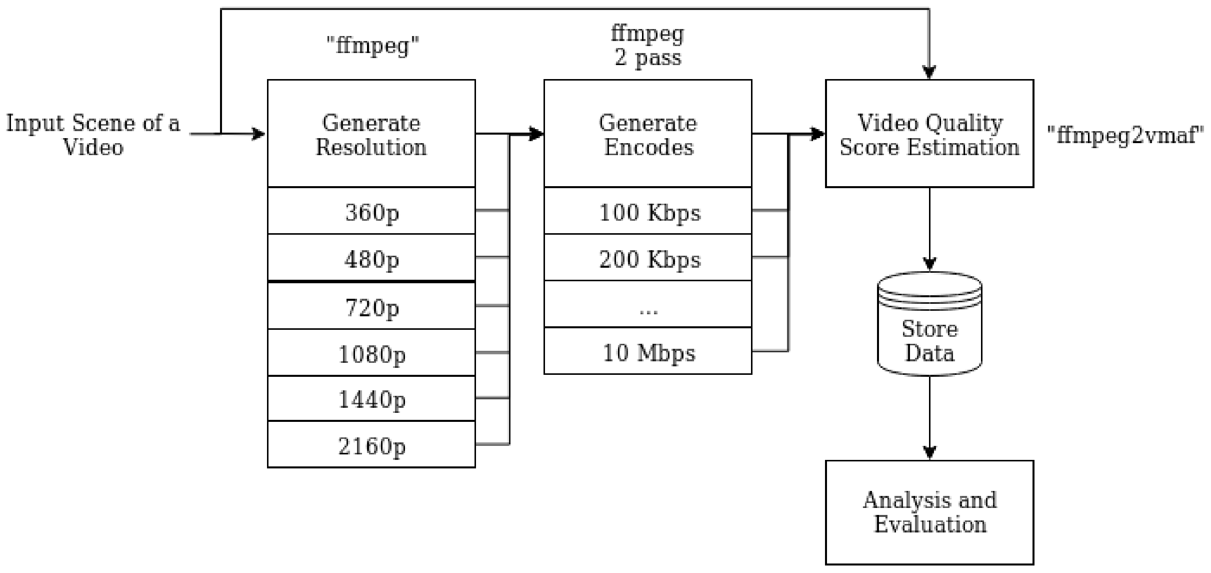


Figure 3.6: Video Encoding Pipeline for all Resolution

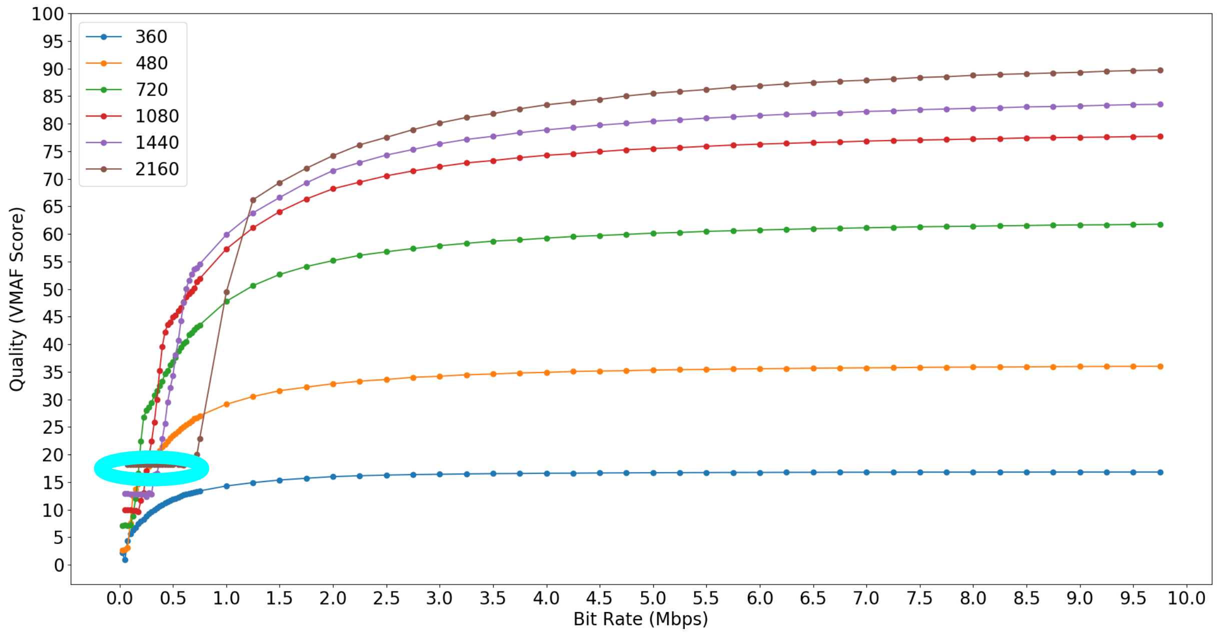


Figure 3.7: Rate vs Quality Curve at all resolutions over different Bitrates

points of rate-quality pairs and then setting up the plot again would allow to understand it better. After trimming down the rate-quality points the plot looks like figure 3.8.

Analyzing the plot in the figure 3.8 shows that the curves for low resolutions i.e., 360p, 480p and 720p the frequency rate-quality pairs is too high when it comes to change in quality. The change in quality for those curve is not as rapid as it is for the curves of higher resolutions i.e., 1080p, 1440p and 2160p. This analysis shows that the curves for the higher resolutions were able to achieve quality higher VMAF Score than 60 at the 1.5 Mbps. The reason behind it is the scene that is being analyzed here had very smooth and slow movements with most of the uniform regions across the frames. Such video scenes where the movement is least and background remains more or less constant they reach higher quality at lower bitrates as well. For dynamic videos the curves might take higher bitrates ranging in 5-10 Mbps to reach satisfactory quality for the viewers. The rate-quality curves tend to have much higher slope as the bitrate increases initially for videos with low amount of motion than the rate-quality curve of dynamic scenes where there is sudden motions and content change. In this thesis there were several other videos used with unique attributes. Some of the videos had dynamic motion in it and after the analysis when the rate-quality curve was plotted. The curves did not flatten as much as the curves for smooth video scenes. The further sections of this thesis will show the proof for the claim being made about relationship of the characteristics of rate-quality curve with the different scenes. Take the figure 3.5 into consideration. For the bitrate points below 2.5 Mbps, there was reasonable quality difference after each bitrate points from the start but after 2.5 Mbps the curve tends to flatten, which signifies that the quality will not improve as

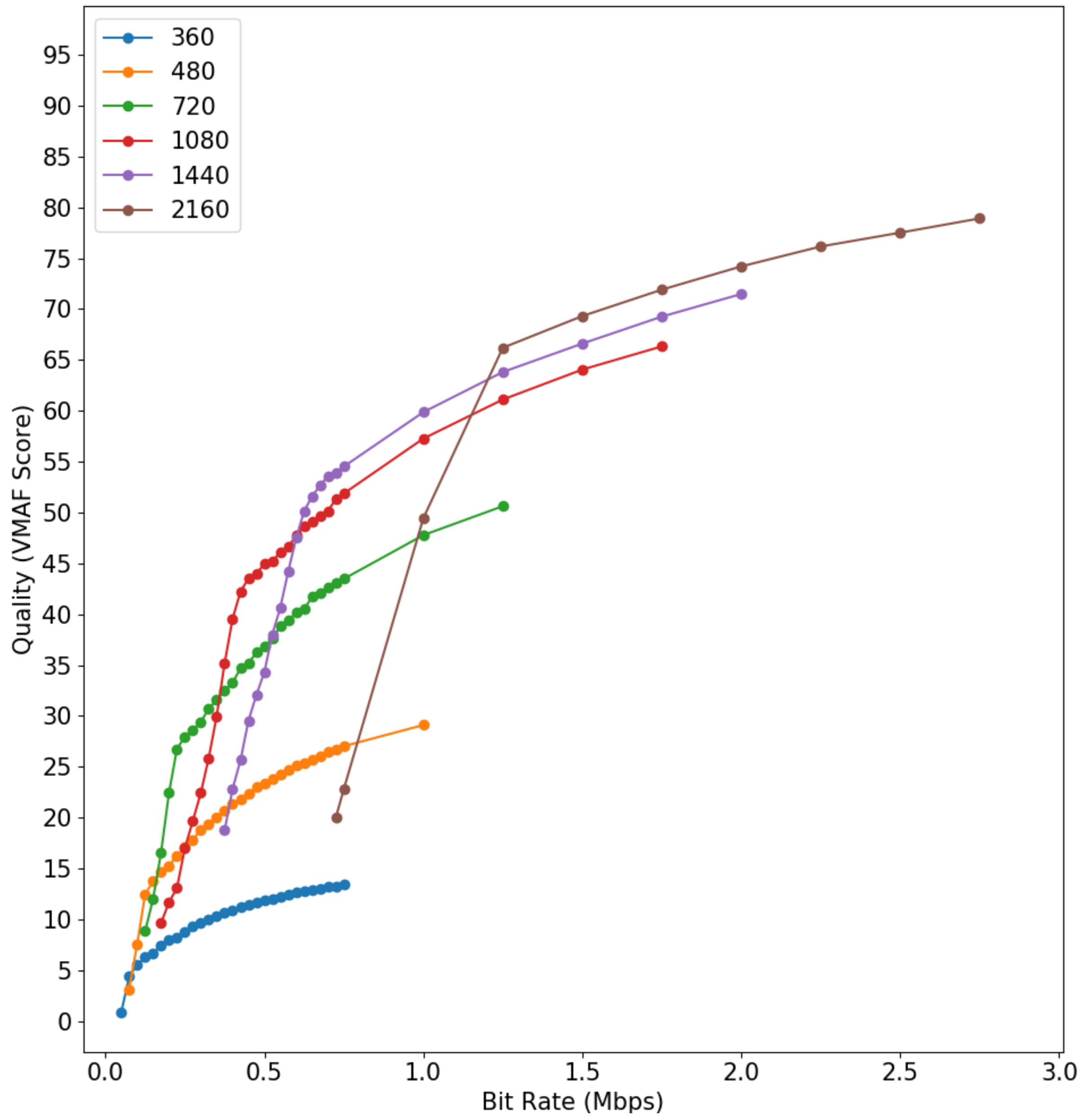


Figure 3.8: Rate vs Quality Curve at all resolutions over Selected Bitrates

much after 2.5 Mbps bitrate. Now this point may vary from video to video depending upon the content of the video.

About the initial design setup and its results, it yielded smooth plots for individual resolutions and looked promising. This approach could be used to analyze other scenes or shots as well. But this approach has a downside. The number of bitrate points required for each of the resolutions for computing VMAF score quality can not be anticipated. The bitrate points for the resolutions would also vary in case of different kind of video scenes. Another reason is the computational resource limitation. Netflix uses pipeline in the cloud based on parallel computing. Netflix uses the Linux EC2 instances which provides powerful computing resources. [Blo17] For this thesis, the computational resources were limited. So transcoding a scene for 6 resolutions and then computing 67 encodes for each of the resolutions reaches high computational complexity. The problem of computational cost had to be reduced in order to find a feasible solution hat can provide enough bitrate points to do the transcoding. Secondly the bitrate points must be recurring where the quality for those bitrate points really have reasonable difference. This required to develop a new approach which could divide and conquer the bitrates points which are relevant. So the next section would describe about the new approach and how it performs for different video scenes.

3.3 Pipeline based on Bisection Method Approximation

- ▷ start with a theoretical approach
- ▷ describe the developed system/algorithm/method from a high-level point of view
- ▷ go ahead in presenting your developments in more detail

Chapter 4

Analysis/ Evaluation

Measurement results / analysis / discussion: 1/3

- ▷ whatever you have done, you must comment it, compare it to other systems, evaluate it, using e.g. subjective tests
- ▷ usually, adequate graphs help to show the benefits of your approach
- ▷ caution: each result/graph must be discussed! what's the reason for this peak or why have you observed this effect

Chapter 5

Conclusion

Conclusion: 1 page

- ▷ summarize again what your thesis did, but now emphasize more the results, and comparisons
- ▷ write conclusions that can be drawn from the results found and the discussion presented in the paper
- ▷ future work (be very brief, explain what, but not much how)

Bibliography

[Aar+15] Anne Aaron et al. "Per-title encode optimization". In: *The Netflix Techblog* (2015).

[Blo17] Netflix Technology Blog. *High Quality Video Encoding at Scale*. 2017. URL: https://medium.com/netflix-techblog/high-quality-video-encoding-at-scale-d159db052746.

[Blo18] Netflix Technology Blog. *VMAF: The Journey Continues*. 2018. URL: https://medium.com/netflix-techblog/vmaf-the-journey-continues-44b51ee9ed12.

[BR96] John S Boreczky and Lawrence A Rowe. "Comparison of video shot boundary detection techniques". In: *Journal of Electronic Imaging* 5.2 (1996), pp. 122–129.

[Bre19] Breakthrough. *Breakthrough/PySceneDetect*. 2019. URL: https://github.com/Breakthrough/PySceneDetect.

[Can] Canonical. URL: http://manpages.ubuntu.com/manpages/bionic/man1/shotdetect.1.html.

[CLG07] Vasileios Chasanis, Aristidis Likas, and Nikolaos Galatsanos. "Scene detection in videos using shot clustering and symbolic sequence segmentation". In: *2007 IEEE 9th Workshop on Multimedia Signal Processing*. IEEE. 2007, pp. 187–190.

[CV95] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[Ffma] URL: https://ffmpeg.org/ffmpeg-filters.html#blackframe.

[Ffmb] *FFmpeg*. URL: https://ffmpeg.org/.

[Joh] Johmathe. *johmathe/shotdetect*. URL: https://github.com/johmathe/shotdetect/blob/master/description-pak.

[Kat18] Ioannis Katsavounidis. *Dynamic optimizer - a perceptual video encoding optimization framework*. 2018. URL: https://medium.com/netflix-techblog/dynamic-optimizer-a-perceptual-video-encoding-optimization-framework-e19f1e3a277f.

[KG18] Ioannis Katsavounidis and Liwei Guo. "Video codec comparison using the dynamic optimizer framework". In: *Applications of Digital Image Processing XLI*. Vol. 10752. International Society for Optics and Photonics. 2018, 107520Q.

[KZS15] Dilip Kumar Krishnappa, Michael Zink, and Ramesh K Sitaraman. "Optimizing the video transcoding workflow in content delivery networks". In: *Proceedings of the 6th ACM Multimedia Systems Conference*. ACM. 2015, pp. 37–48.

[LMN12] Songnan Li, Lin Ma, and King Ngi Ngan. "Full-reference video quality assessment by decoupling detail losses and additive impairments". In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.7 (2012), pp. 1100–1112.

[Man+18] Megha Manohara et al. *Optimized shot-based encodes: Now Streaming!* 2018. URL: https://medium.com/netflix-techblog/optimized-shot-based-encodes-now-streaming-4b9464204830.

[Moo+19] Ben Moore et al. *The Best Video Streaming Services for 2019*. 2019. URL: https://www.pcmag.com/roundup/336650/the-best-video-streaming-services.

[Neta] *NETFLIX*. https://www.netflix.com/.

[Netb] Netflix. *Netflix/vmaf*. URL: https://github.com/Netflix/vmaf/blob/master/resource/doc/VMAF_Python_library.md.

[OJ] Martin Ombura Jr. *How YouTube handles streaming 4,000,000,000+ daily videos without a hitch.* https://medium.com/@martinomburajr/how-youtube-handles-streaming-4-000-000-000-daily-videos-without-a-hitch-8542741e957a.

[Ope] *OpenCV*. 2019. URL: https://opencv.org/.

[OR02] James Oly and Daniel A Reed. "Markov model prediction of I/O requests for scientific applications". In: *Proceedings of the 16th international conference on Supercomputing*. ACM. 2002, pp. 147–155.

[Pev] *the Standard for Perceptual Evaluation of Video Quality*. URL: http://www.pevq.com/.

[Pri] *PrimeVideo*. https://www.primevideo.com/.

[Psn] *Peak signal-to-noise ratio*. 2019. URL: https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio.

[Pyt] *Welcome to Python.org*. URL: https://www.python.org/.

[Ras17] Reza Rassool. "VMAF reproducibility: Validating a perceptual practical video quality metric". In: *2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. IEEE. 2017, pp. 1–2.

[Sat+19] Shahid Mahmood Satti et al. “Low Complexity" Smart" Per-Scene Video Encoding”. In: *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE. 2019, pp. 1–3.

[Sto11] Thomas Stockhammer. “Dynamic adaptive streaming over HTTP–: standards and design principles”. In: *Proceedings of the second annual ACM conference on Multimedia systems*. ACM. 2011, pp. 133–144.

[tea] Libav team. *Libav - Open source audio and video processing tools*. `https://www.libav.org/`.

[VMC17] Van-Huy Vu, Ibrahim Mashal, and Tein-Yaw Chung. “A Novel Bandwidth Estimation Method Based on MACD for DASH.” In: *KSII Transactions on Internet & Information Systems* 11.3 (2017).

[Wan+04] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[Win12] Stefan Winkler. “Analysis of public image and video databases for quality assessment”. In: *IEEE Journal of Selected Topics in Signal Processing* 6.6 (2012), pp. 616–625.

[WRG99] Peter H Westerink, Rajesh Rajagopalan, and Cesar A Gonzales. “Two-pass MPEG-2 variable-bit-rate encoding”. In: *IBM Journal of Research and Development* 43.4 (1999), pp. 471–488.

[Yt] *YouTube*. `https://www.youtube.com/`.

List of Figures

List of Tables

Appendix A

AppendixExample

Declaration

I declare that the work is entirely my own and was produced with no assistance from third parties.
I certify that the work has not been submitted in the same or any similar form for assessment to any other examining body and all references, direct and indirect, are indicated as such and have been cited accordingly.   Ilmenau,

Todo list