VCA: Video Complexity Analyzer

Vignesh V Menon
vignesh.menon@aau.at
Christian Doppler Laboratory ATHENA
Institute of Information Technology (ITEC)
Alpen-Adria-Universität Klagenfurt
Klagenfurt, Austria

Christian Feldmann
christian.feldmann@bitmovin.com
Bitmovin
Klagenfurt, Austria

Hadi Amirpour
hadi.amirpour@aau.at
Christian Doppler Laboratory ATHENA
Institute of Information Technology (ITEC)
Alpen-Adria-Universität Klagenfurt
Klagenfurt, Austria

Mohammad Ghanbari
ghan@essex.ac.uk
School of Computer Science and Electronic Engineering
University of Essex
Colchester, UK

Christian Timmerer
christian.timmerer@aau.at
Christian Doppler Laboratory ATHENA
Institute of Information Technology (ITEC)
Alpen-Adria-Universität Klagenfurt
Klagenfurt, Austria

ABSTRACT

For online analysis of the video content complexity in live streaming applications, selecting low-complexity features is critical to ensure low-latency video streaming without disruptions. To this light, for each video (segment), two features, *i.e.*, the average texture energy and the average gradient of the texture energy, are determined. A DCT-based energy function is introduced to determine the block-wise texture of each frame. The spatial and temporal features of the video (segment) are derived from this DCT-based energy function. The Video Complexity Analyzer (VCA) project aims to provide an efficient spatial and temporal complexity analysis of each video (segment) which can be used in various applications to find the optimal encoding decisions. VCA leverages some of the x86 Single Instruction Multiple Data (SIMD) optimizations for Intel CPUs and multi-threading optimizations to achieve increased performance. VCA is an open-source library published under the GNU GPLv3 license.
Github: https://github.com/cd-athena/VCA
Online documentation: https://cd-athena.github.io/VCA/
Website: https://vca.itec.aau.at/

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Software and its engineering** → *Software performance*.

KEYWORDS

Video Complexity, Texture, SIMD, Multi-threading, Video Streaming

ACM Reference Format:
Vignesh V Menon, Christian Feldmann, Hadi Amirpour, Mohammad Ghanbari, and Christian Timmerer. 2022. VCA: Video Complexity Analyzer. In *13th ACM Multimedia Systems Conference (MMSys '22), June 14–17, 2022, Athlone, Ireland.* , 6 pages. https://doi.org/10.1145/3524273.3532896

MMSys '22, June 14–17, 2022, Athlone, Ireland

ACM ISBN 978-1-4503-9283-9/22/06…$15.00
https://doi.org/10.1145/3524273.3532896

1 INTRODUCTION

Motivation: Video and its related applications are in enormous demand in today's homes. Moreover, there will be significant bandwidth demands in the future for applications like *Ultra High Definition* (UHD) [18] Virtual Reality (VR) streaming, 8K Wall TV [5]. Efficient video coding standards are inevitable to decrease the bandwidth requirements to stream high-quality videos, with a plethora of encoding parameters targeting various use cases and video content types. The optimal encoding parameters depend on the video content complexity, and available compute resources. Therefore, there is a need to extract features representing the video content complexity to predict the optimal encoding parameters for that video content. Video content complexity can be evaluated in spatial and temporal domains.

Video complexity analysis is shown to be a critical step for numerous applications. Haseeb *et al.* [7] used spatial and temporal complexity information in rate-distortion modeling. The video complexity evaluation is also essential in QoE evaluation metrics [3, 6, 16, 21]. Pinson *et al.* [16] measured the video quality objectively by utilizing the spatial content of the sequences. Barman *et al.* [21], Goring *et al.* [6], and Zadtootaghaj *et al.* [3] proposed machine learning-based QoE models, where spatial and temporal complexity values are used along with other influential factors for quality estimation of gaming videos. Fast video complexity analysis is used in online per-title encoding schemes, which determine optimized resolution, bitrate-ladder [1], framerate, and other relevant encoding parameters for live *HTTP Adaptive Streaming* (HAS) applications [12–14]. Optimized resolution for a given target bitrate can be modeled as a function of the spatial and temporal complexities and original framerate of the video [13]. Furthermore, the optimized framerate or a given target bitrate can be modeled as a function of the spatial and temporal complexities and the resolution of the video [10].

SITI[1] is the state-of-the-art open-source software to evaluate spatial and temporal complexity. Spatial Information (SI) indicates

[1] https://github.com/Telecommunication-Telemedia-Assessment/SITI

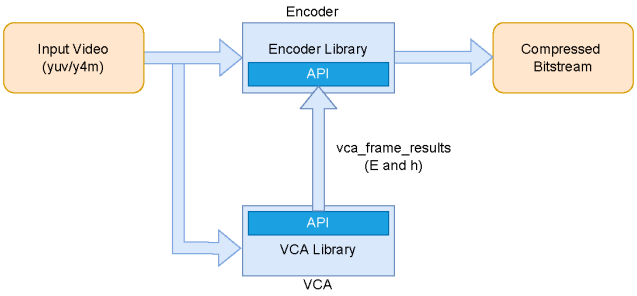


Figure 1: VCA for content-adaptive encoding applications.

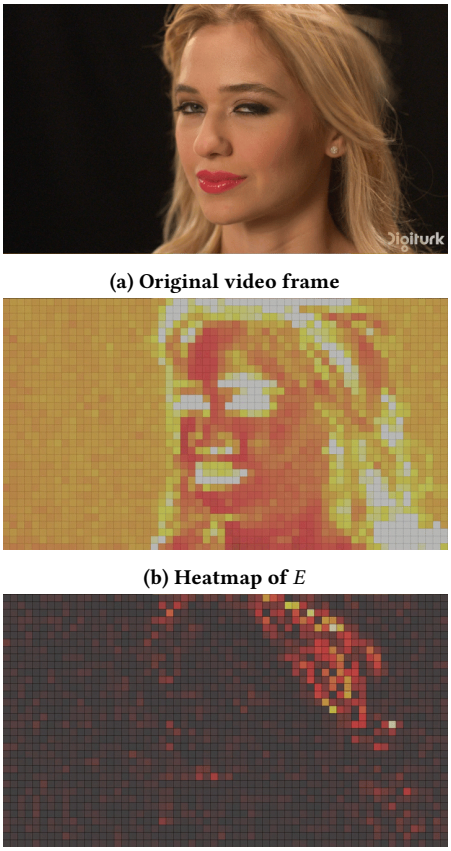
the maximum amount of spatial detail in a video. For a given N-frame sequence, SI is calculated by first filtering the luma samples (i, j) of each frame $F_n = F_0, F_1, \ldots, F_{N-1}$ with the Sobel filter, then computing the standard deviation over the Sobel filtered frame pixels, and finally taking its maximum [8]. Correspondingly, Temporal Information (TI) stands for the maximum amount of temporal variation between successive frames F_{n-1} and F_n. However, SITI implementation is not optimized for online analysis in applications like live streaming. Moreover, the correlation of the features to the video coding parameters like bitrate and encoding time is not very high. The primary objective of VCA is to overcome these limitations and provide an efficient spatial and temporal video complexity analyzer in terms of accuracy and speed for every video (segment). Motivated by the researchers' commitment to the open-source community, VCA is available as an open-source library, published under the GNU GPLv3 license.

Contributions: VCA leverages state-of-the-art hardware-level acceleration techniques like x86 SIMD optimizations and multi-threading optimizations for increased performance. While VCA is primarily designed as a video complexity analyzer library, a command-line executable is provided to facilitate testing and development. As shown in Figure 1, the raw video frames are input to VCA, which analyzes the spatial and temporal characteristics of the video. This analysis is transferred to the encoder via Application Programming Interface (API) to aid the encoding process. The combination of codec independence, platform independence, x86 SIMD implementation, and multi-threading optimizations makes this library unique compared to other available libraries for video complexity analysis like SITI [8]. This library is ideal for students, researchers, and professionals looking to expand their use of video complexity analysis.

Paper Organization: Section 2 describes the video complexity features analyzed by VCA, while Section 3 explains the relevance of these features in video coding applications. Section 4 introduces the API functions defined in the library. Section 5 describes the performance optimizations in VCA, and Section 6 shows an application of VCA. Finally, Section 7 concludes the paper.

2 VIDEO COMPLEXITY FEATURES

For the online analysis of video content complexity, selecting low-complexity features is critical to ensure low-latency video streaming



(a) Original video frame

(b) Heatmap of E

(c) Heatmap of h

Figure 2: Heatmap of E and h features of the 2^{nd} frame of *Beauty* test sequence of UVG dataset [15].

without disruptions. In a frame, two features, *i.e.*, the average texture energy and the average gradient of the texture energy, are calculated [11–13]. A DCT-based energy function is introduced to determine the block-wise texture of each frame, which is defined as:

$$H_{p,k} = \sum_{i=0}^{w-1} \sum_{j=0}^{w-1} e^{|(\frac{ij}{w^2})^2 - 1|} |DCT(i,j)| \tag{1}$$

where k is the block address in the p^{th} frame, $w \times w$ pixels is the size of the block, and $DCT(i, j)$ is the $(i, j)^{th}$ DCT component when $i + j > 0$, and 0 otherwise [9]. Exponentially higher costs are assigned to higher DCT frequencies since it is expected that a mixture of objects causes the higher frequencies. The texture is averaged to determine the *spatial energy* feature denoted as E as shown below:

$$E = \sum_{k=0}^{C-1} \frac{H_{p,k}}{C \cdot w^2} \tag{2}$$

Here, C represents the number of blocks per frame. Furthermore, the block-wise *SAD* of the texture energy of each frame compared to its previous frame is computed and then averaged for each frame of the segment to obtain the average *temporal energy* (h) as shown

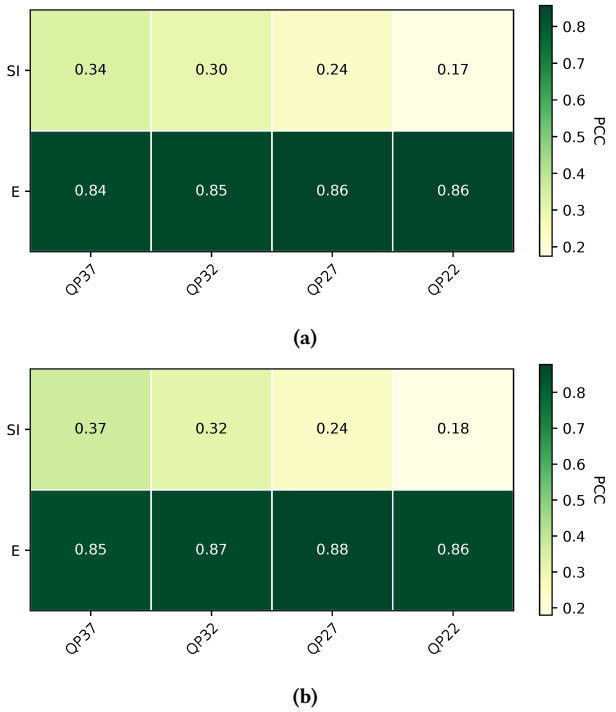


Figure 3: PCC between the spatial complexity features (SI and E) and bitrate in All Intra configuration with *medium* preset of (a) x264 encoder and (b) x265 encoder for the *VCD* dataset [2].

below:

$$h = \sum_{k=0}^{C-1} \frac{SAD(H_{p,k}, H_{p-1,k})}{C \cdot w^2} \tag{3}$$

An example block-wise heatmap of E and h features is shown in Figure 2. Furthermore, the gradient of h per frame, ϵ is defined as:

$$\epsilon = \frac{h_{p-1} - h_p}{h_{p-1}} \tag{4}$$

If $\epsilon = 0$, p^{th} frame is a duplicate of $(p-1)^{th}$ frame. The default value of block-width (w) is set as 32. It can be configured by the user as 8 or 16 or 32 using `-block-size` option.

3 ANALYSIS OF COMPLEXITY FEATURES

This section describes the relevance of the E and h video complexity features analyzed by `VCA` in video coding by analyzing the correlation of E and h with the ground truth encoding complexities as introduced in the following.

Firstly, the accuracy of the E feature is compared to the state-of-the-art SI feature. In this light, the correlation of SI and E features with the bitrate in All Intra (AI) configuration [4] is evaluated. AI configuration is characterized as an intra frame compression where each frame is encoded independently. Hence, each frame is compressed without referring to other frames. It implies that temporal complexity does not play any role in AI configuration, while the bitrate is directly proportional to the spatial complexity of the frames. Thus, bitrate in AI configuration is considered as the ground truth of the spatial complexity. Figure 3 shows the Pearson Correlation Coefficient (PCC) between the SI and E features with the bitrate for the *VCD* dataset [2]. As observed in Figure 3a that the average PCC of SI with bitrate is 0.26, while the average PCC of E with bitrate is 0.85 in x264[2] AVC [20] encoding. Similar results are observed in Figure 3b using x265[3] HEVC [17] encoding where the average PCC of SI with bitrate is 0.28, while the average PCC of E with bitrate is 0.86, respectively. Thus, the E feature correlates well with the spatial complexity than the state-of-the-art SI feature.

Secondly, the influence of E and h features in the rate-distortion (RD) complexity and encoding run-time complexity of the Low Delay P picture (LDP) configuration [4] is investigated. Figure 4 shows the PCC of SI, TI, E, and h features with the encoding bitrate of x265 encoder with $QP \in \{22, 27, 32, 37\}$ using *veryslow, medium, and ultrafast* presets. The average correlation of SI, TI, E, and h with bitrate is 0.03, 0.55, 0.51, and 0.67, respectively. Furthermore, the average correlation of SI, TI, E and h with encoding time is 0.25, 0.72, 0.35, and 0.68, respectively. Hence, E and h features strongly correlate with the RD complexity and encoding run-time complexity of the LDP configuration.

Finally, the correlation of SI and E features is analyzed for multiple resolutions of the same content. As observed in Figure 6a, the PCC of SI calculated in 2160p compared to 720p, and 1080p resolutions are 0.45 and 0.43, respectively. But, the PCC of E calculated in 2160p compared to that of 720p, and 1080p resolutions are 0.91 and 0.94, respectively. Thus, E exhibits better correlation across resolutions, facilitating optimizations including computations in lower resolutions.

4 APPLICATION PROGRAMMING INTERFACE DESIGN

The API of this software is defined in the following file: *vcaLib.h* in the *source/lib/* folder of the source tree. All of the functions and variables, and enumerations meant to be used by the user are present in this header.

(1) `vca_analyzer_open()`: this function creates a new analyzer instance with all parameters input by the user. The returned pointer is then passed to all of the functions of this analyzer instance.
(2) `vca_analyzer_push()`: this function pushes an input video frame to the analyzer and starts the analysis. Please note that only the pointers will be copied, but no ownership of the memory is transferred to the library. The caller application must ensure that the pointers are valid until the video frame is analyzed. Once the results for a frame are pulled, the library will not use pointers anymore. This may be blocked until there is a slot available to work on. The user can set the number of frames to be processed in parallel.
(3) `vca_result_available()`: this function checks if a result of any frame is available to pull.

[2]https://www.videolan.org/developers/x264.html
[3]https://www.videolan.org/developers/x265.html

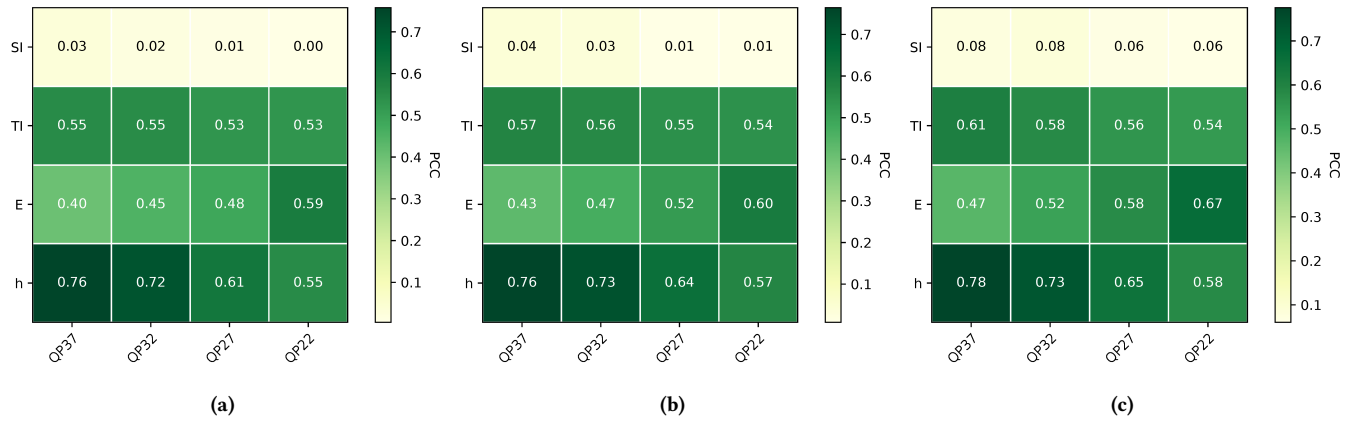


Figure 4: PCC between the spatial complexity features (SI and E) and temporal features (TI and h) with *bitrate* in the Low Delay P picture (LDP) configuration with (a) *veryslow* (b) *medium* and (c) *ultrafast* presets of x265 encoder for the *VCD* dataset [2].

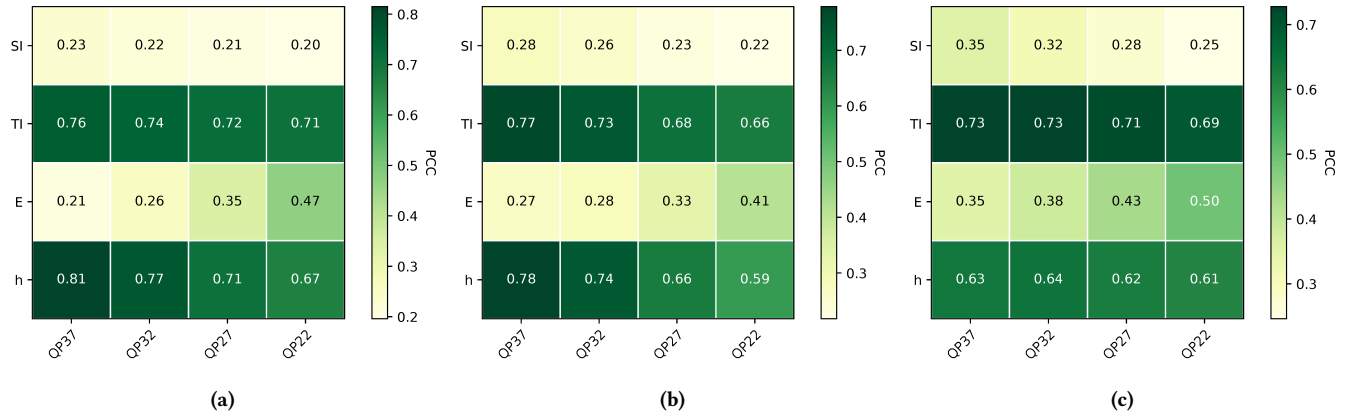


Figure 5: PCC between the spatial complexity features (SI and E) and temporal features (TI and h) with *encoding time* in low-delay inter-coding configuration with (a) *veryslow* (a) *medium* and (c) *ultrafast* presets of x265 encoder for VCD dataset [2].

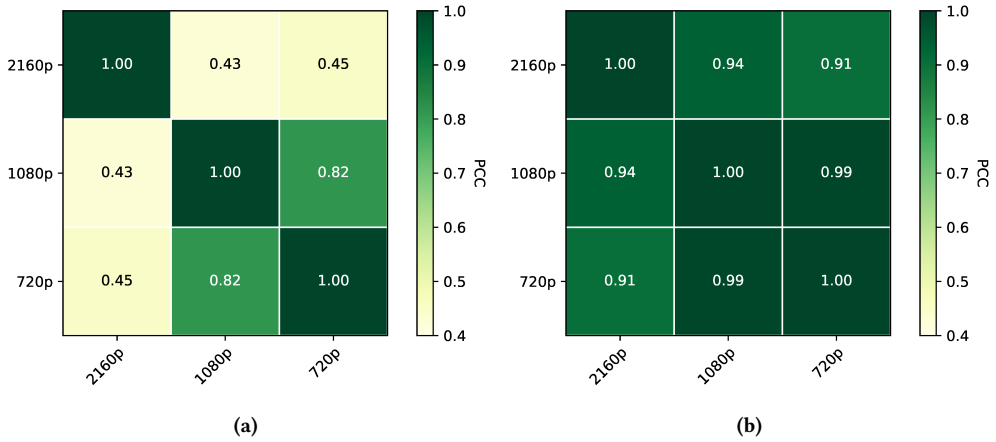


Figure 6: PCC between the spatial complexity features (a) SI and (b) E across multiple resolutions for the VCD dataset [2].

Algorithm 1: API implementation.

```
analyzer = vca_analyzer_open(vca_param) ;
while read each frame do
    vca_analyzer_push (analyzer, frame) ;
    while vca_result_available(analyzer) do
        vca_analyzer_pull_frame_result (analyzer, result) ;
        use the result in the application ;
vca_analyzer_close(analyzer) ;
```

(4) `vca_analyzer_pull_frame_result()`: this function pulls a result from the analyzer. This may be blocked until a result is available. Please use vca_result_available() to check whether a result is available.
(5) `vca_analyzer_close()`: this function closes the analyzer instance. The analyzer must be closed after its completion to free all its resources. An analyzer that has been closed cannot be restarted and reused. Once vca_analyzer_close() has been called, the analyzer handle must be discarded.

The manner in which the above-described API functions are expected to be called by the application is shown in Algorithm 1. The complexity analysis results are available via API as `vca_frame_results` structure which has the following data elements:

(1) `energyPerBlock` : stores $H_{p,k}$ (*cf.* Eq. 1)
(2) `averageEnergy` : stores E of the p^{th} frame (*cf.* Eq. 2)
(3) `sadPerBlock` : stores $SAD(H_{p,k}, H_{p-1,k})$
(4) `sad`: stores h of the p^{th} frame (*cf.* Eq. 3)
(5) `epsilon`: stores ϵ of the p^{th} frame (*cf.* Eq. 4)
(6) `poc`: frame number (p)
(7) `jobID`: an increasing counter that is incremented with each call to `vca_analyzer_push`

The calling application must make sure that the pointers to the block-wise data in the structure (`energyPerBlock` and `sadPerBlock`) point to a valid memory block.

5 PERFORMANCE OPTIMIZATIONS

VCA v1.0 offers performance optimizations to analyze the video complexity in real-time applications like content-adaptive live encoding. Section 5.1 explains the x86 SIMD optimizations, while Section 5.2 discusses the multi-threading optimization.

5.1 x86 SIMD Optimization

VCA leverages the SIMD optimization of DCT functions implemented as intrinsic and assembly codes for x86 architecture. The intrinsic code of DCT is executed for Intel SSSE3 architecture. Handwritten Intel SSSE3 and AVX2 instructions accelerate the DCT calculation per block. Though modern compilers support SIMD auto-vectorization, handwritten assembly outperforms auto-vectorization tools [19]. As shown in Figure 7, the C code implementation of VCA is approximately five times faster than the state-of-the-art SITI implementation (VCA-only C). With the x86 SIMD optimization of the DCT function, VCA is about ten times faster than the SITI implementation (VCA-with SIMD). The NASM assembler is required to run the software with assembly optimizations. A CMake flag, namely `ENABLE_NASM` is added to the project, which is set to `ON` by default if NASM is pre-installed on the system. If NASM is not installed in the system, the flag is set to `OFF` during compilation. The user can also manually set it to `OFF`.

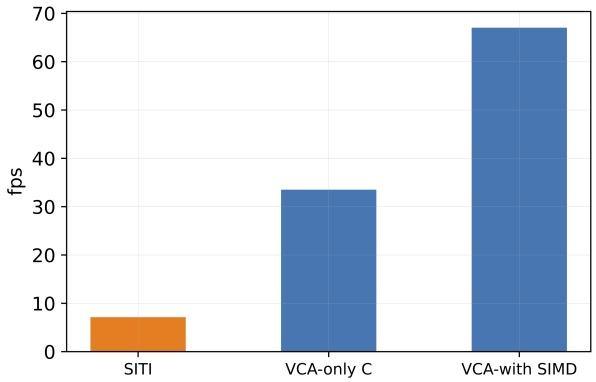


Figure 7: x86 SIMD optimization results.

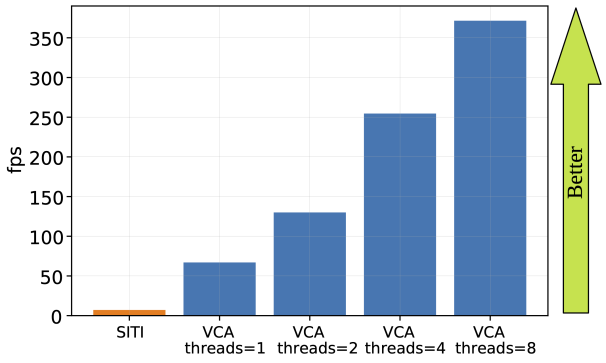


Figure 8: Multi-threading results.

5.2 Multi-Threading Optimization

To optimize the performance in multi-core CPUs, VCA leverage the multi-threading mechanism. With multi-threading optimization, multiple threads are created within a VCA execution instance, which execute independently but concurrently sharing process resources. Independent threads carry out DCT-energy computation per block (*cf.* Eq. 1). The number of threads to be utilized during the execution can be specified by `--threads` option. By default, the software uses the maximum number of available threads.

Figure 8 shows the impact of multi-threading optimization on the performance of VCA. With a single thread execution, VCA analyzed 67 frames per second, while with two threads execution, the speed accelerated to 130 frames per second. With eight threads execution, VCA could analyze 371 frames per second. Please note that the multi-threading results reported here are with x86 SIMD optimizations.

6 APPLICATIONS

In VCA v1.0, shot detection [11] is added as an application of the complexity features analyzed by VCA. Detecting shots is a critical step

Algorithm 2: Shot detection algorithm.

Inputs:
f: number of video frames per second
ϵ_k: feature value per frame k
T_1 , T_2 : maximum and minimum threshold for ϵ_k
Output: shot boundary detection
Step 1: **while** *Parsing all video frames* **do**
  if $\epsilon_k > T_1$ **then**
    k begins a new shot.
  else if $\epsilon \leq T_2$ **then**
    k does not begin a new shot.
Q : set of frames where $T_1 \geq \epsilon > T_2$
q_0: current frame number in the set Q
q_{-1}: previous frame number in the set Q
q_1: next frame number in the set Q
Step 2: **while** *Parsing* Q **do**
  if $q_0 - q_{-1} > f$ *and* $q_1 - q_0 > f$ **then**
    $q_0 \leftarrow$ IDR-frame, a new shot.
    Eliminate q_0 from Q.

in shot-based encoding schemes where each video is first divided into shots, and each shot is encoded independently. The encoding parameters are determined for each scene such that the overall quality of experience is improved. However, detecting gradual shot transitions is difficult because the criteria used to determine the significance of a change in the visual information between two frames are subjective and complex to describe quantitatively [11].

Once the ϵ feature (*cf.* Eq. 4) of all frames is determined, Algorithm 2 is applied to segment the video sequence into shots. Hard shot transitions characterized by high ϵ are detected in Step 1. Step 2 is designed to handle fade-in, fade-outs, and dissolves, where ϵ values will generally be high for a few consecutive frames. In these situations, frames after the gradual shot-cuts are IDR coded, as the subsequent frames shall have a better reference for encoding [11]. The default values of T_1 and T_2 values (*cf.* Algorithm 2) are set as 50 and 10 respectively. They can be also set manually using `–max-thresh` and `–min-thresh` options, respectively.

7 CONCLUSIONS

VCA determines spatial and temporal complexity for every video (segment), which can aid applications like online per-title encoding to predict encoding parameters. VCA is an open-source library published under the GNU GPLv3 license. VCA leverages x86 SIMD and multi-threading optimizations. While VCA is primarily designed as a video complexity analyzer library, a command-line executable is provided to facilitate testing and development.

This project development shall be continued and will be contributed open-source. The reader is encouraged to contribute to the project at https://github.com/cd-athena/VCA.

8 ACKNOWLEDGMENT

The financial support of the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, and the Christian Doppler Research Association is gratefully acknowledged. Christian Doppler Laboratory ATHENA: https://athena.itec.aau.at/.

REFERENCES

[1] Hadi Amirpour, Mohammad Ghanbari, and Christian Timmerer. 2021. PSTR: Per-Title Encoding Using Spatio-Temporal Resolutions. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*. 1–6. https://doi.org/10.1109/ICME51207.2021.9428247
[2] H. Amirpour, V. V. Menon, S. Afzal, M. Ghanbari, and C. Timmerer. 2022. VCD: Video Complexity Dataset. In *Proceedings of the 13th ACM Multimedia Systems Conference (MMSys '22)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3524273.3532892
[3] N. Barman, E. Jammeh, S. Ghorashi, and M. Martini. 2019. No-Reference Video Quality Estimation Based on Machine Learning for Passive Gaming Video Streaming Applications. *IEEE Access* 7 (2019), 74511–74527. https://doi.org/10.1109/ACCESS.2019.2920477
[4] Jill Boyce, Karsten Suehring, Xiang Li, and Vadim Seregin. 2018. *JVET-J1010: JVET common test conditions and software reference configurations.*
[5] Cisco. 2020. Cisco visual networking index: Forecast and methodology, 2018–2023 (White Paper). (2020).
[6] S. Göring, R. Rao, and A. Raake. 2019. nofu — A Lightweight No-Reference Pixel Based Video Quality Model for Gaming Content. In *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*. 1–6. https://doi.org/10.1109/QoMEX.2019.8743262
[7] A. Haseeb, M. Martini, S. Cicalò, and V. Tralli. 2012. Rate and distortion modeling for real-time MGS coding and adaptation. In *2012 Wireless Advanced (WiAd)*. 85–89. https://doi.org/10.1109/WiAd.2012.6296574
[8] ITU-T. 2021. P.910 : Subjective video quality assessment methods for multimedia applications. https://www.itu.int/rec/T-REC-P.910-202111-I/en
[9] Michael King, Zinovi Tauber, and Ze-Nian Li. 2007. A New Energy Function for Segmentation and Compression. In *2007 IEEE International Conference on Multimedia and Expo*. 1647–1650. https://doi.org/10.1109/ICME.2007.4284983
[10] V. V. Menon, H. Amirpour, C. Feldmann, A. Ilangovan, M. Smole, M. Ghanbari, and C. Timmerer. 2022. Live-PSTR: Live Per-title Encoding for Ultra HD Adaptive Streaming. In *2022 NAB Broadcast Engineering and Information Technology (BEIT) Conference.*
[11] Vignesh V Menon, Hadi Amirpour, Mohammad Ghanbari, and Christian Timmerer. 2021. Efficient Content-Adaptive Feature-Based Shot Detection for HTTP Adaptive Streaming. In *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2174–2178. https://doi.org/10.1109/ICIP42928.2021.9506092
[12] Vignesh V Menon, Hadi Amirpour, Mohammad Ghanbari, and Christian Timmerer. 2022. CODA: Content-aware Frame Dropping Algorithm for High Frame-rate Video Streaming. In *2022 Data Compression Conference (DCC)*. IEEE, 475–475.
[13] Vignesh V Menon, Hadi Amirpour, Mohammad Ghanbari, and Christian Timmerer. 2022. OPTE: Online Per-title Encoding For Live Video Streaming. In *2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.
[14] Vignesh V Menon, Hadi Amirpour, Christian Timmerer, and Mohammad Ghanbari. 2021. Efficient Multi-Encoding Algorithms for HTTP Adaptive Bitrate Streaming. In *2021 Picture Coding Symposium (PCS)*. 1–5. https://doi.org/10.1109/PCS50896.2021.9477499
[15] A. Mercat, M. Viitanen, and J. Vanne. 2020. *UVG Dataset: 50/120fps 4K Sequences for Video Codec Analysis and Development.* Association for Computing Machinery, New York, NY, USA, 297–302. https://doi.org/10.1145/3339825.3394937
[16] M.H. Pinson and S. Wolf. 2004. A new standardized method for objectively measuring video quality. *IEEE Transactions on Broadcasting* 50, 3 (2004), 312–322. https://doi.org/10.1109/TBC.2004.834028
[17] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand. 2012. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on circuits and systems for video technology* 22, 12 (2012), 1649–1668.
[18] T. Fautier. 2018. UHD Worldwide Service Deployment Update. https://ultrahdforum.org/wp-content/uploads/UHD-Worldwide-Service-Deployment-Update-April-2018.pdf
[19] P. Tiwari, V. V. Menon, J. Murugan, J. Chandrasekaran, G. Akisetty, P. Ramachandran, S. Venkata, C. Bird, and K. Cone. 2018. Accelerating x265 with Intel® Advanced Vector Extensions 512. *White Paper on the Intel Developers Page* (2018). https://www.intel.com/content/dam/develop/external/us/en/documents/mcw-intel-x265-avx512.pdf
[20] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. 2003. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology* 13, 7 (2003), 560–576.
[21] S. Zadtootaghaj, N. Barman, S. Schmidt, M. Martini, and S. Möller. 2018. NR-GVQM: A No Reference Gaming Video Quality Metric. In *2018 IEEE International Symposium on Multimedia (ISM)*. IEEE, 131–134. https://doi.org/10.1109/ISM.2018.00031