Machine-Learning-Based Method for Finding Optimal Video-Codec Configurations Using Physical Input-Video Features

Roman Kazantsev*, Sergey Zvezdakov*, and Dmitriy Vatolin*

*Lomonosov Moscow State University
Moscow, 119234, Russia
{roman.kazantsev,sergey.zvezdakov,dmitriy.vatolin}@graphics.cs.msu.ru

Abstract

Modern video codecs have many compression-tuning parameters from which numerous configurations (presets) can be constructed. The large number of presets complicates the search for one that delivers optimal encoding time, quality, and compressed-video size. This paper presents a machine-learning-based method that helps to solve this problem. We applied the method to the x264 video codec: it searches for optimal presets that demonstrate 9-20% bitrate savings relative to standard x264 presets with comparable compressed-video quality and encoding time. Our method is faster upto 10 times than existing solutions.

Introduction

The popularity of video-hosting services (e.g., YouTube, Netflix, and Twitch) and the broad distribution of mobile video cameras have contributed to the recent exponential growth of video data. This massive amount of data stimulates demand for more-efficient video-compression methods to better enable storage and transmission. Therefore, this domain is advancing on all fronts: new compression standards are emerging periodically (e.g., HEVC, VVC, and AV1), dozens of companies are developing video codecs, and researchers are constantly producing new video compression approaches, including ones based on machine learning. In this paper we propose a machine-learning-based method for finding video-codec settings that provide optimal encoding time, quality, and compressed-video size.

Modern video codecs have a wide range of encoding and decoding parameters. Because of this wide range and the many different parameter values, numerous video-codec configurations (presets) are possible. As a result, the main encoding challenge lies in configuring the codec to meet the user's requirements for encoding time and video-quality loss during compression. The problem of finding a suitable configuration can be formulated as a multicriterion-optimization problem, where these indicators are minimized in parallel on some set of codec configurations. Note that the compressed-video size is known and is determined by the bitrate parameter, so it is absent in the list of indicators.

Problem Definition

In this work we solve the abovementioned problem for the x264 video codec on P, a subset of all configurations. Because x264 has more than 40 parameters, it has too

many possible configurations to address in their entirety. We therefore consider only major ones that see frequent use in practice. For instance, the following parameters affect the compressed-video quality and the encoding time: `--`*partitions* (block sizes for motion estimation), `--`$bframes$ (the number of B-frames between I-frames and P-frames), `--`ref (the number of reference frames used for motion estimation), `--`me (the motion estimation algorithm). Varying these parameter values creates the set P which includes 1306 configurations. Note that only the encoder parameters are configured; the default decoder parameter values are unchanged.

Let us introduce auxiliary mathematical objects for the problem definition. $g_{p,v}(c)$ is a bitrate function that depends on the compression quality when encoding video v using configuration p. The function $f_{p,v}(b)$ is the time required to encode video v with bitrate b using configuration p. We use structural similarity index (SSIM) [1] to measure compressed-video quality v' with respect to the original video v. Let $p_{def} \in P$ be a default configuration that is `--`*preset medium* in x264. The objective optimization functions are the relative average bitrate required to encode a quality unit and the relative average time needed to encode a bitrate unit: $Q_v(p) = \frac{\int_{c_1}^{c_2} g_{p,v}(c)dc}{\int_{c_1}^{c_2} g_{p_{def},v}(c)dc}$ and $T_v(p) = \frac{\int_{b_1}^{b_2} f_{p,v}(b)db}{\int_{b_1}^{b_2} f_{p_{def},v}(b)db}$. Therefore the following optimization problem is considered for input video sequence v:

$$(Q_v(p), T_v(p)) \to min, p \in P. \tag{1}$$

The optimization problem (1) would be more complete if decoding time appeared in the criteria list, but we chose to skip it for this work. The known solution to the multicriterion-optimization problem is the Pareto set $P^*(v) = \{p \in P | \forall p' \in P \wedge p' \neq p : Q_v(p) < Q_v(p') \vee T_v(p) < T_v(p')\}$.

In fact, both criteria in the problem have disadvantages because they guarantee an optimal result on average across all bitrates. In the meantime, hovewer, it is helpful to avoid unstable presets that fail to provide monotonic compression quality with increasing bitrate.

We normalized the objectives in problem (1) using values that correspond to the default preset p_{def}, that is `--`*preset medium*, to avoid hardware dependence and to enable better interpretation of the results. For example, $Q_v(p) = 0.7$ means that changing the configuration from p_{def} to p will decrease bitrate by 30%. Or in case of $T_v(p) = 0.8$, the configuration p will accelerate p_{def} by 20%.

Our work have established that the Pareto sets can vary for different input video sequences. For that reason, it makes sense to solve the problem individually for each video sequence as given in the problem definition (1). For example, as Fig. 1 shows, the Pareto set for the "Sea Cost" video (109321182_2428_2691[1]) performs poorly on the "Samsung NX500 4K Test Video: Focus Tests" video (128948492_7246_7501).

Problem (1) is relevant because frequently used standard configurations of x264 (*ultrafast*, *superfast*, *veryfast*, *faster*, *fast*, *medium*, *slow*, *slower*, *veryslow*, *placebo*) can perform poorly and be far from optimal. Fig. 2 illustrates a breach between standard configurations and the Pareto set for "Sea Coast".

[1]the video name has a format *id_start_count* where *id* is `vimeo.com` video identifier, *start* is a start-frame index, and *count* is a number of frames. The video appears at the URL `vimeo.com/id`.

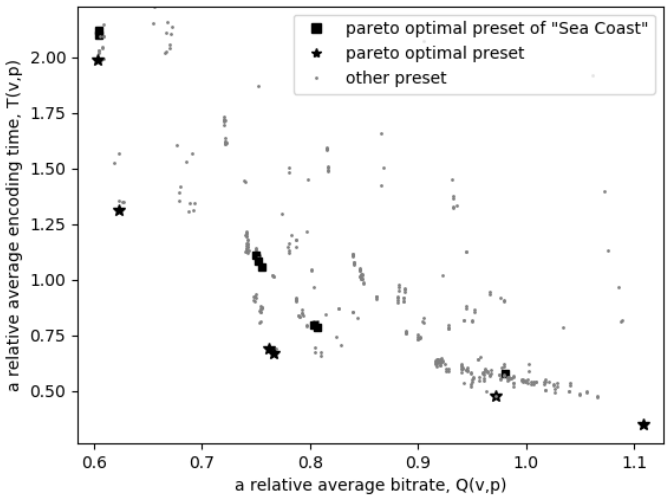

Figure 1: The performance of the Pareto set of "Sea Coast" video (and other presets) on "Samsung NX500 4K Test Video: Focus Tests" video

We now propose our machine-learning-based method for searching the Pareto set of x264 configurations for a given video sequence.

Related Works

A dozen algorithms that solve problem (1) breaking down into two groups: classical algorithms and machine-learning based algorithms.

Note that among the classical algorithms, Popov's [2] and NSGA-II [3] are based on a gradient descent and a genetic approach respectively. The genetic approach generates candidates for the Pareto set by mutating some subset of configuration-parameter values in accordance with a rule specific to the algorithm. These algorithms can also solve more-general multiobjective optimization problems. Evaluating candidate's performance, however, requires calculating objective-function values for that candidate. This process is time-consuming in our case because we must launch a codec. Thus, multiple runs are necessary to solve the problem for each video sequence. These classical algorithms are unable to apply to the current problem knowledge gained by solving earlier problems involving other videos.

For the past 10 years, machine-learning-based approaches (Zvezdakov's [4] and Murashko's [5]) have become increasingly popular. They employ knowledge accumulated while solving a problem using video sequences in a training dataset. Depending on the algorithm, this accumulated knowledge can represent various statistics, for example, the average objective-function change when shifting from one configuration to other. These statistics describe some regularity in the data and summarize results for arbitrary videos, helping to reduce problem-solving time and to avoid launching the codec. In Murashko's algorithm [5], for example, a codec launch is unneces-

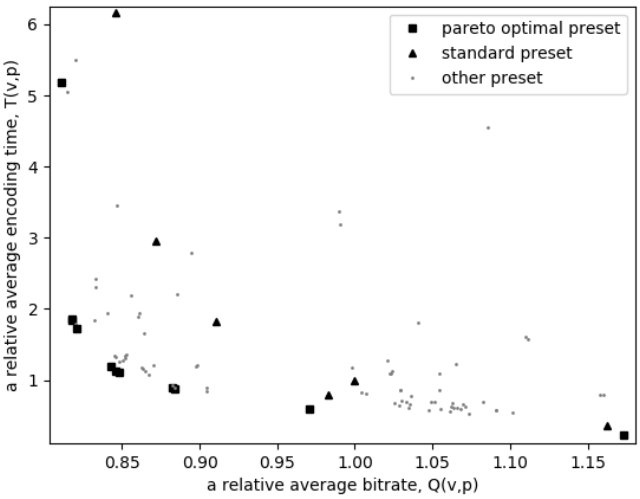

Figure 2: The performance of the Pareto optimal and standard presets for "Sea Coast" video

sary. Instead the algorithm calculates input-video features (descriptor) in a less time-consuming manner and predicts the results. But it solves a more particular problem: namely, how to find the optimal video resolution and predict compressed-video size for a given preset.

Computing physical video features is cheaper than running a codec multiple times. Therefore, we propose a machine-learning-based algorithm that predicts the Pareto-optimal set of configurations using specifically designed physical video features while avoiding the need to launch a codec.

Dataset Creation

No public datasets are available to solve the optimization problem using machine learning. For that reason we created our own. Our first step was to select 351 video sequences and 1,306 x264 configurations. The video sequences come from `vimeo.com` in UltraHD resolution with high bitrates. The codec configurations are partly constructed on the basis of a preliminary x264-options analysis [6]. For each pair of sequence v and configuration p, we have computed the relative average bitrate required to encode a quality unit $Q_v(p)$ and the relative average time needed to encode a bitrate unit $T_v(p)$. This computation took three weeks on a cluster of 145 computers running Intel Xeon E3-1125v3 CPU with 16GB RAM.

Different types of videos have their own peculiarities. For example, cartoons characteristically exhibit a sparse color distribution in each frame, simple object shapes, and contrast transitions on boundaries. Video games also have a sparse color distribution, but they may be more realistic in the complexity of object shapes and smooth boundary transitions. Videos captured by drones display a specific type of movement

in a certain direction. The main distinguishing characteristic of sports content is the numerous moving objects of different sizes. On the basis of such characteristics, we manually crafted nine physical features constituting a video descriptor. Computing these features for all 351 video sequences took two days on two computers with Intel Core i7-4770R CPU running at 3.20GHz with 16GB RAM.

Video Physical Features

The following is a description of the physical video features we employed in our investigation.

SI (Spatial perceptual Information) [7] measures the amount of spatial detail in a frame; it is the standard deviation over the pixels of Sobel-filtered frame. It is usually higher for more-spatially-complex scenes. The formula $SI = std_{space}[Sobel(F_n(i,j))]$ yields the spatial complexity of the n-th frame, where $F_n(i,j)$ is the luminance of pixel (i,j).

TI (Temporal perceptual Information) [7] indicates the amount of temporal changes in a frame; it is the standard deviation over differences between pixel values (in the luminance plane) at the same location for successive frames. It is usually higher for high-motion frames. The formula $TI = std_{space}[F_n(i,j) - F_{n-1}(i,j)]$ yields the temporal complexity of the n-th frame.

TI_ME (Temporal perceptual Information with Motion Estimation) [8] estimates the temporal complexity of a frame after motion compensation. Its formula is $TI\_ME = std_{space}[min_{k,l}(F_n(i,j) - F_{n-1}(k,l))]$. For motion estimation and compensation, we use the algorithm from [9] with the parameters $min\_block\_size = 8$, $max\_block\_size = 8$, $precision = QuaterPixel$, and $loss\_metric = SAD$.

Blur estimates frame's acutance of a frame and is based on the blur effect [10]. The feature of the effect is that the luminance of a blurred image exhibits little change after repeated appliations. The metric is higher for sharper frames.

Blur_Lap accesses the acutance of a frame using the Laplace operator [11]. It is the standard deviation over frame's responses to the Laplace operator: $Blur\_Lap = std_{space}[L(m,n) - L]$, where $L(m,n)$ is the frame's response of at location (m,n) and L is an absolute mean of the responses.

Mean_Y, ***Mean_U***, and ***Mean_V*** are the average values for each color component of the n-th frame in YUV space and are calculated as follows:
$Mean\_Y = \frac{1}{width \times height} \sum_{i=1}^{width} \sum_{j=1}^{height} F_n(i,j)$

Var_Y, ***Var_U***, and ***Var_V*** are the standard deviations for each color component of the n-th frame in YUV space.

Color3D_Hist [12] is the three-dimensional color histogram for a frame; it models the color distribution of pixel values in RGB space. Each color channel is evenly divided into four parts. Thus, the size of the histogram is $64 = 4 \times 4 \times 4$.

Motion_Hist is the two-dimensional motion histogram that estimates the length and direction (angle) distributions of frame's motion vectors. It contains five bins for length and five bins for direction. Moreover, it uses a logarithm of the length value to better estimate short motion vectors. To find the motion vectors, we use a block-based motion estimation algorithm [9] with the following parameters: $min\_block\_size = 8$, $max\_block\_size = 8$, $precision = QuaterPixel$, $loss\_metric = SAD$.

We have already applied these features effectively when solving the video-genre-classification problem in [13].

Our dataset is split into two parts, yielding a 3 : 1 ratio of training to validation data. In the next section we describe a method that predicts Pareto front of x264 configurations for a given video sequence using its video physical features.

Proposed Method

Before we describe the method, let us introduce the concept of the Pareto frontier structure $S_P(v)$ built from presets P for video sequence v. We designate the Pareto frontier $P_i(v)$ with rank i as the solution to the problem: $(Q_v(p), T_v(p)) \to min, p \in P \setminus \cup_{k=0}^{i-1} P_k(v)$, where $i = 0, 1, ..., l$ and l is the rank of the farthest Pareto frontier. Note that $P_0(v)$ is simply the solution of problem (1). The Pareto frontiers appear in the calculation of the Pareto-frontier structure: $S_P(v) = (x_1, x_2, ..., x_{|P|})$, where x_k is the rank of the Pareto frontier to which the k-th preset from ordered set P belongs. We calculated the Pareto-frontier structure for each video sequence in our dataset. Using an agglomerative clustering method, we clustered all videos from the dataset according to similarity of their Pareto-frontier structures. For two videos v_1 and v_2 with the structures $S_P(v_1) = (x_1, x_2, ..., x_{|P|})$ and $S_P(v_2) = (y_1, y_2, ..., y_{|P|})$, respectively, we define their similarity $\rho(v_1, v_2)$ on the basis of the Spearman's rank-correlation coefficient: $\rho(v_1, v_2) = 1 - \frac{6}{|P|(|P|-1)(|P|+1)} \sum_{i=1}^{|P|} (x_i - y_i)^2$. The pair of v_1 and v_2 map to the same cluster if $\rho(v_1, v_2) \geq \theta$, where the optimal value of the threshold θ is experimentally determined to be 0.9. Therefore, we obtained four clusters (classes):

- C_1, a scene with slight foreground and background movement.
  Examples: 157797521_0_250, 150984258_474_724, 138590264_4244_4494.

- C_2, a scene with camera movement.
  Examples: 160359830_3282_3532, 160474708_6703_6953, 143092229_4316_4567.

- C_3, a (nearly) static scene.
  Examples: 136832821_4099_4349, 127865290_0_250, 146523627_10195_10445.

- C_4, a (nearly) static scene with a blurred background.
  Examples: 137292905_1543_1793, 136972283_515_765, 130202285_7062_7312.

Since videos from the same cluster have similar Pareto-frontier structures, we assume that solutions of problem (1) for these videos will be similar; that is, $P^*(v_1) \approx P^*(v_2)$ for videos v_1 and v_2 from a given cluster. We assign to a cluster C the Pareto-optimal set $P^*(v_C)$ of some arbitrarily selected video v_C from this cluster and call it the cluster's Pareto set P_C^*. Moreover, we assign each cluster C a configuration-enhancement function (CEF) $E_C(p)$ that outputs a preset $p^* \in P_C^*$. This preset outperforms the input preset from $P \setminus P_C^*$; that is, $E_C(p) = p^*$ so that $Q_{v_C}(p^*) \leq Q_{v_C}(p) \wedge T_{v_C}(p^*) < T_{v_C}(p) \vee Q_{v_C}(p^*) < Q_{v_C}(p) \wedge T_{v_C}(p^*) \leq T_{v_C}(p)$. Therefore, we can reduce the original problem (1) to a classification problem where the model should predict a cluster index using the physical features of input video v. The cluster

Table 1: Class error metrics

C, a class	$err(C), \%$	$err^{\text{standard}}(C), \%$
C_1	22.69	22.76
C_2	24.70	24.76
C_3	26.37	26.45
C_4	27.99	27.95

Table 2: Average cluster-preset bitrate and encoding time gains relative to standard configurations, along with average bitrate loss relative to the optimal preset

p_{std}	bitrate gain against p_{std}, %	bitrate loss against optimal preset, %	encoding time gain against p_{std}, %
ultrafast	0.0	0.0	0.0
superfast	0.0	0.0	0.0
veryfast	-2.3	4.7	9.7
fast	16.5	2.9	5.0
faster	6.4	2.5	-0.2
medium	21.1	2.8	9.1
slower	19.3	1.4	15.0
slow	22.3	1.3	2.8
veryslow	13.0	1.9	26.4
placebo	15.9	2.0	80.1

Pareto set and CEF assigned to the predicted cluster will be a solution of the original problem (1) for video v.

During inference the method returns two outputs: the cluster Pareto set and the cluster CEF. In practice, if someone uses a preset from P, the CEF can produce a configuration that outperforms the input configuration.

To validate the clustering and to estimate the cluster Pareto sets, we compute for each class C the maximum ratio of configurations that outperform one configuration from the cluster Pareto set P_C^* averaged over all videos from that cluster as follows: $err(C) = \frac{1}{|C|}\sum_{v\in C}\max_{p\in P_C^*}\frac{|F(p,v)|}{|P|}$, where $F(p,v) = \{p' \in P : Q_v(p') < Q_v(p) \wedge T_v(p') < T_v(p)\}$. Also, the maximum ratio of standard configurations that outperform a given configuration from the cluster Pareto set P_C^* averaged over all videos is computed by the following formula: $err^{standard}(C) = \frac{1}{|C|}\sum_{v\in C}\max_{p\in P_C^*}\frac{|F^{standard}(p,v)|}{|P|}$, where $F^{standard}(p,v) = \{p' \in P^{standard} : Q_v(p') < Q_v(p) \wedge T_v(p') < T_v(p)\}$ and $P^{standard}$ includes all the standard configurations of x264 codec such as $ultrafast$, $superfast$, $veryfast$, $faster$, $fast$, $medium$, $slow$, $slower$, $veryslow$, $placebo$. Table 1 shows these error metrics for each class.

Additionally, we tested the cluster Pareto sets and the corresponding CEFs in a more practical way. We calculated the average percentage of bitrate and time-encoding savings for configurations that the CEF returned for the standard configurations versus the standard and optimal configurations themselves. Table 2 shows these results.

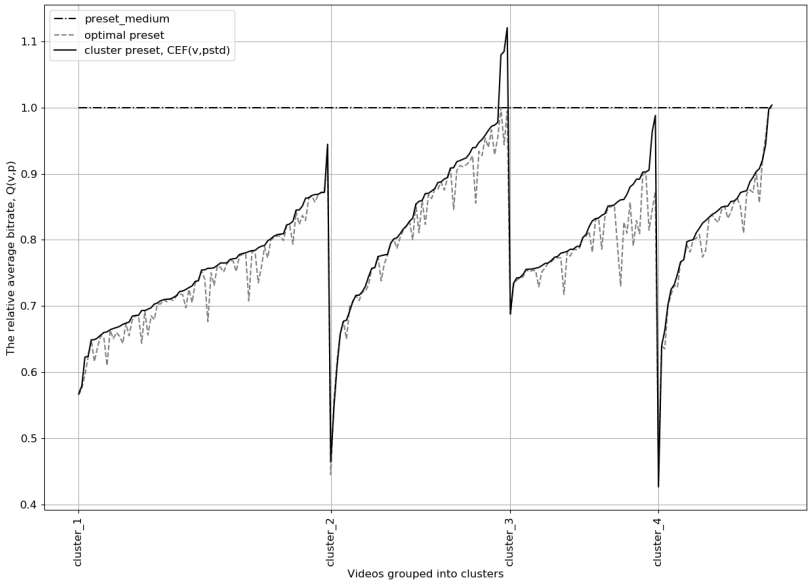

Figure 3: Comparison of relative average bitrates obtained using the optimal preset, cluster preset, and *medium* preset for each video in the clusters

In addition, we analyzed the distributions of these metrics over all videos in each class. Fig. 3 shows the distribution of bitrate saving for configurations that the CEF returned: specifically, a *medium* standard configuration and the CEF returned configuration for *medium* fails for, at most, only 1% of video per class.

To solve the classification problem, we trained an ensemble of decision trees using gradient boosting from XGBoost library. The training uses the following optimal gradient-boosting parameters: $booster = gbtree, max\_depth = 3, reg\_alpha = 1, learning\_rate = 0.001, gamma = 1, subsample = 0.6, num\_round = 222$. We employed cross-validation to evaluate the model accuracy; yielding a result of 75.2%.

The decision-tree ensemble allows to easily estimate feature importance, that is, the average gain (or tree purity) that a feature brings when using it to prune the tree. We estimated the most relevant features to be ***Blur_Lap*** and ***TI***. They have also proven useful in the video-genre-classification model from [13].

Experimental Results and Comparisons

We have tested the proposed method on some JVET and Xiph.org video sequences that are not a part of our dataset. The selected sequences are in the FullHD format and have different motion, color, and content characteristics. The results about bitrate savings and time-encoding savings for each sequence are presented in Table 3.

We did performance comparison for our method with alternative methods (Popov's, NSGA-II, Zvezdakov's). Table 4 shows the average bitrate savings obtained with the Pareto optimal presets versus the standard presets and execution time on average over

Table 3: Bitrate savings [%] obtained using the CEF returned presets versus the standard presets on JVET and Xiph.org sequences

Sequence	Faster	Fast	Medium	Slow	Slower	Veryslow	Placebo
Cactus	6.0	16.5	17.5	15.0	13.3	8.0	15.7
DugsAndLegs	17.2	27.9	26.5	21.0	15.4	1.3	11.5
KristenAndSara	4.5	12.3	16.0	24.7	25.1	0.0	9.0
ParkScene	12.6	17.5	29.0	22.8	19.0	14.1	16.3
PeopleOnStreet	16.8	21.9	27.6	33.7	31.4	20.2	29.8
Average	**11.4**	**19.2**	**23.3**	**23.4**	**20.8**	**8.7**	**16.4**

Table 4: Average bitrate savings [%] of the Pareto optimal presets versus standard presets and execution time obtained using each method on the validation set

Method	Faster	Fast	Medium	Slow	Slower	Veryslow	Placebo	Time, sec.
Popov's	8.0	29.0	28.4	34.9	32.2	28.3	29.0	10039.7
NSGA-II	15.9	30.2	29.7	34.9	32.2	29.0	28.3	13686.4
Zvezdakov's	11.0	30.2	29.8	34.9	32.2	28.7	29.4	7705.2
Ours	15.8	21.3	21.8	27.7	24.9	9.7	10.5	**735.5**

all the validation set for each method. As we can see our method underperforms the alternative solutions in terms of bitrate savings. However, the alternative methods are slower upto 10 times.

Conclusions

In the paper we proposed the method for finding optimal presets that demonstrate 9-20% bitrate savings against the x264 standard presets with comparable compressed video quality and encoding time. The method is faster upto 10 times than alternative solutions. This method can be applied to other video codecs but it will require to create new dataset.

As a drawback we have not managed to enhance a couple of fast standard presets ($ultrafast$, $superfast$) due to the created dataset that poorly covers these cases.

Acknowledgements

We would like to thank to the Faculty of Computational Mathematics and Cybernetics of Lomonosov Moscow State University for providing the computing resources that enabled our investigation. We would also like to acknowledge our colleagues from the Graphics and Media Laboratory, in particular, Alexander Yakovenko and Maxim Velikanov for their valuable input. This work was partially supported by Russian Foundation for Basic Research under Grant 19-01-00785 a.

References

[1] Z. Wang, A.C. Bovik, H.R. Sheik, and E.P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*,

vol. 13, no. 4, pp. 600–612, April 2004.

[2] V. Popov, "Automatic method of choosing pareto optimal video codec's parameters," M.S. thesis, Lomonosov Moscow State University, 2009.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, August 2002.

[4] S. Zvezdakov and D. Vatolin, "Building a x264 video codec model," in *Innovative technologies in cinema and education: IV International Symposium*, Moscow, Russia, 2017, pp. 56–65, VGIK Moscow.

[5] O. Murashko, *Using machine learning to select and optimise multiple objectives in media compression*, Ph.D. thesis, The University of St Andrews, St Andrews KY16 9AJ, UK, 2018.

[6] D. Vatolin, A. Parshin, V. Popov, and K. Ragulina, "x264 codec strong and weak points," Tech. Rep., CS MSU Graphics and Media Lab of Lomonosov Moscow State University, Moscow, Russia, 2008.

[7] Telecommunication Standardization Sector of ITU, "Subjective video quality assessment methods for multimedia applications," in *Series P. Telephone Transmission Quality, Telephone Installations, Local Line Networks*, pp. 4–6. April 2008.

[8] I. Brailovskiy and N. Solomeshch, "Quality modelling for videocoding," *Information Technologies*, vol. 1, no. 1, pp. 42–48, 2012.

[9] K. Simonyan, S. Grishin, D. Vatolin, and D. Popov, "Fast video super-resolution via classification," in *Proceedings of 15th IEEE International Conference on Image Processing*, San Diego, CA, USA, October 2008, pp. 349–352.

[10] F. Crete-Roffet, T. Dolmiere, P. Ladret, and M. Nicolas., "The blur effect: Perception and estimation with a new no-reference perceptual blur metric," in *SPIE Electronic Imaging Symposium Conf Human Vision and Electronic Imaging*, San Jose, CA, USA, January 2007, pp. 6492–6516.

[11] R. Bansal, G. Raj, and T. Choudhury, "Blur image detection using laplacian operator and opencv," in *Proceedings of the SMART-2016*, Moradabad, India, November 2016.

[12] N. Lakshmi, Y. Latha, A. Damodharam, and K. Prasanna, "Implementation of content based video classification using hidden markov model," in *Proceedings of 7th International Advance Computing Conference (IACC)*, Hyderabad, India, January 2017.

[13] R. Kazantsev, S. Zvezdakov, and D. Vatolin, "Application of physical video features in classification problem," *International Journal of Open Information Technologies*, vol. 7, no. 5, May 2019.