INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC 1/SC 29/WG 4
MPEG VIDEO CODING

ISO/IEC JTC 1/SC 29/WG 4 m64712
Hannover – October 2023

Title: [VCM] Results of PUT RoI preprocessing and retargeting based codec with VTM (CE1.2)

Source: Poznań University of Technology (PUT),
Institute of Multimedia Telecommunications, Poznań, Poland
Authors: Marek Domański, Olgierd Stankiewicz, Sławomir Maćkowiak, Sławomir Różek, Tomasz Grajek, Maciej Wawrzyniak, Jakub Stankowski, Dominik Cywiński, Jakub Szekiełda

Abstract

This document is related to core experiment on RoI-based coding methods (CE1) [1] for Video Coding for Machines.

The considered tools originally has been submitted by Poznań University of Technology as a part of response [2] to "Call for Proposals for Video Coding for Machines" [3]. A general description of the tools and the coding results on previous implementations are provided in [2, 4, 5, and 6].

The document comprises the version of the tools tested under CE1.2[1] with the VTM as the inner codec.

This document presents detailed description and the actual results, as the tools are now adapted to the VCMRS [7, 8] software version 0.6.1. The presented solution keeps default configuration of temporal subsampling module in VCMRS, which is turned on. Therefore in the encoder, temporal resampling plugin is executed first, which is followed by the proposed RoI-based tools.

1. Introduction

The generic structures of the proposed tool of the encoder and the decoder are depicted in Figs. 1 and 4, respectively.

Our approach employs Region of Interest (RoI) detection (on the encoder side). The results of RoI detection are used for control of the following steps:

- **Simplification** – an image preprocessing step.
  Portions of video frames outside of RoIs, are simplified (low-pass filtered). These portions which are not of interest (outside of RoIs), e.g. background, are intensively filtered. High importance RoIs are left intact (are not filtered). After filtration, the frame portions outside RoIs are simplified, i.e. very low bitrate is sufficient to encode them by the inner encoder, i.e. the bitrate is spared leaving higher bit budgets for RoIs that correspond to objects. Thanks to this simplification, for the same total bitrate, the RoIs are encoded with higher bitrate and their quality in the decoded video is improved. It mostly yields higher precision of the machine vision tasks executed on the decoded video. It should be noted that (in contrast to the currently adopted RoI-based preprocessing) simplification preprocessing step is not removing the background entirely, which constitutes usability of the proposed technology also for human viewing.

- **Retargeting** – performed in rectangular regions of a video frame.
  A portion of a video frame is decimated, and the decimation ratio depends on the belonging to a RoI. Obviously, the process includes the anti-aliasing lowpass filtering followed by downsampling. Decimation and downsampling are mostly executed on background whereas the RoIs of high importance mostly remain unchanged. The information about downsampling ratios (horizontal and vertical) and the locations of the removed rows and columns are transmitted to a decoder as the side information with the video bitstream. In the decoder the retargeting is inversed, with the use of information about RoIs (signalled in the bitstream).

The proposal is built on top of existing coding technologies. The technology proposed is **agnostic to the general image/video coding technology** although the benefits are demonstrated with respect to the VCMRS [8] configured with VVC as inner codec.

The simplification tool may be used alone as a preprocessing tool that facilitate improved precision of machine vision tasks for low bitrates of the compressed streams. The performance is increased when the tools are employed jointly (because of lower pixel-rate). Such joint employment needs transmission of the metadata as described in the document.

2. Proposed tools encoder structure

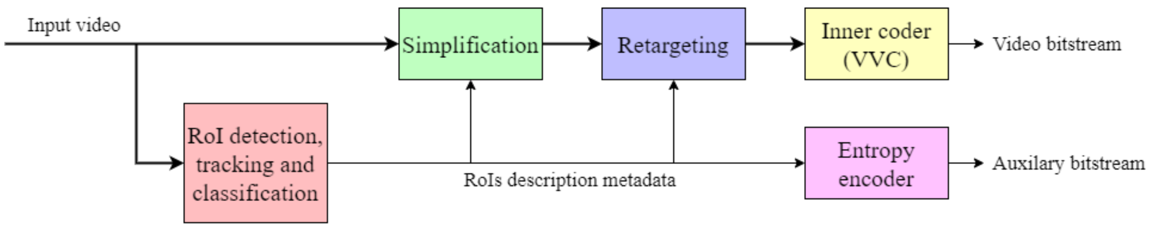

Fig. 1. The proposed tools in an encoder.

Description of the building blocks in a VCM encoder:

a) ***RoI detection, tracking and classification***: In this block the original image/video is analyzed and potential regions of interest (RoI) are determined in the areas where interesting objects may be located. It can be done e.g. using a special deep neural network. In general, any appropriate and efficient method may be used. Then the RoIs are classified as corresponding to important objects, small objects, etc.
In the submitted software implementing the tool, *Detectron2* package [9] is used for object detection and classification into 80 different classes.
For object tracking *Multi-Object-Tracker* Python package [10] has been used. It is used to reduce the effects of flickering of objects in successive frames, e.g. given object is alternately detected and not detected.

In our implementation, each RoI class has specified *optimal size for representation*, e.g. allowing high performance in machine vision tasks. This size differs per class, e.g. more pixels are required to detect a cat (complex fur) than a hippo (smooth skin). If the size of the detected RoI in greater than the *optimal size for representation* for given class, its resolution can be reduced seamlessly. This feature is used in the following Simplification and Retargeting steps.

b) ***Simplification***: The simplification is aimed at production of the content that may be represented with reduced number of bits after compression without any significant deterioration of the efficiency of the machine vision tasks executed on decoded (decompressed) image/video. In principle the simplification reduces details in the background whereas the objects and their neighborhoods remain untouched or only slightly simplified.
The results of RoI determination and classification are used to identifying objects, the bands around objects and the background. Here, we do consider the objects already replaced by simple content by the inpainting. The background is simplified, and possibly some bands around selected objects or even some objects (usually the objects of big size – relatively in an image or in a video frame). The goal of simplification is to produce such image/video that can be easily represented by a very small number of bits

(in compressed representation) whereas the object still can be well recognized (detected, tracked etc.). This may be optimized in context of particular class of object within given RoI.

The goal of simplification is to produce such image/video that can be easily represented by a very small number of bits (in compressed representation) whereas the object still can be well recognized (detected, tracked etc.).

Execution of simplification may be demonstrated using an example from Fig. 2.

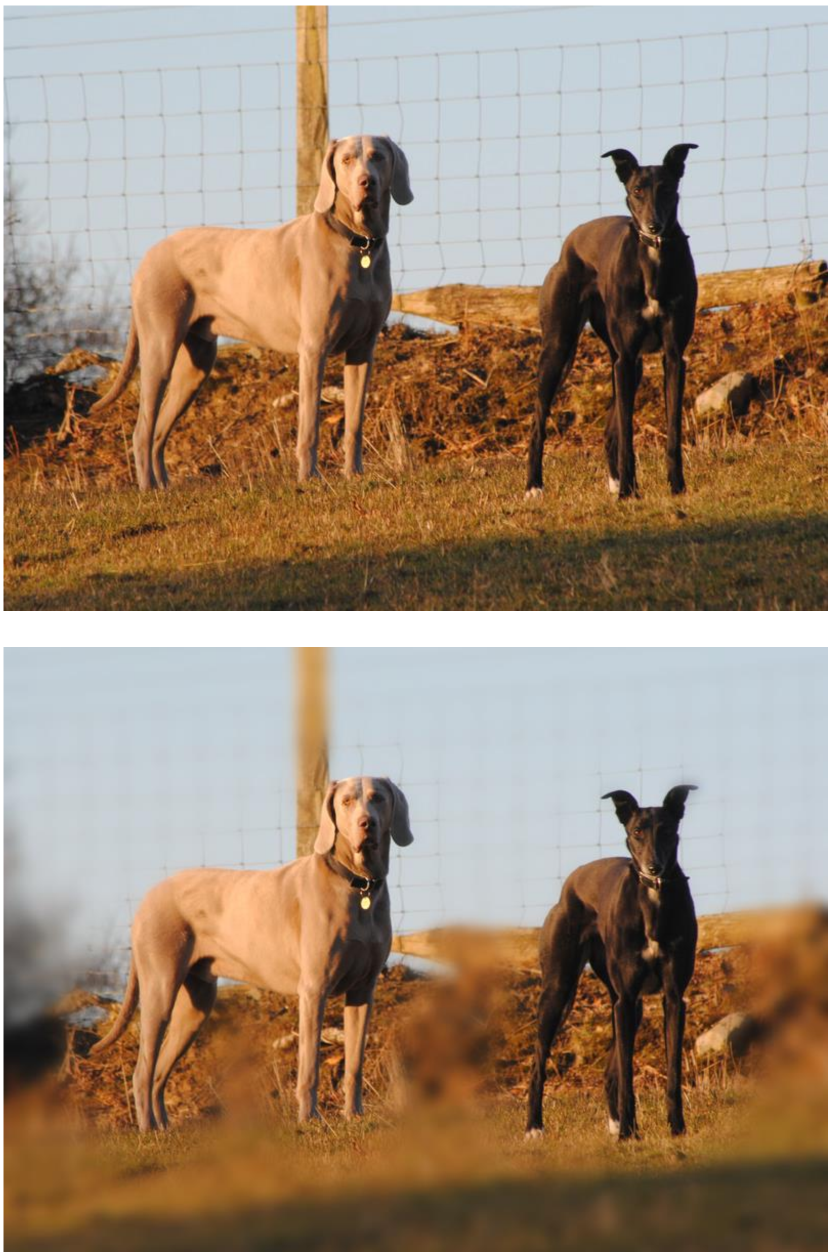

Fig. 2. An image with two objects: the original image (top), the image after simplification that has changed the background (bottom).

c) ***Retargeting***: The number of lines and columns of the input image or the video frame may be reduced according to the content identified in the process of RoI determination and classification. The lines and columns of samples are mostly removed when their do not comprise samples of objects. The lines and rows that comprise object samples may be reduced according to analysis of the objects. This reduction correspond to down-sampling the object that must not deteriorate the ability to recognize the objects, therefore it is possible if the object is oversampled with respect of its texture and other features. The sample (line / column) removal must be preceded by the respective local low-pass (anti-aliasing) filtering. In classical retargeting methods energy function has to be formulated which controls which rows and columns are primarily removed. Such would require signalization of the energy function (image) to the decoder. In our proposal a simple yet efficient retargeting technique is used which omits implicit energy image formulation, and instead rectangle-shaped RoI are used. Therefore, only coordinates of RoIs need to be transmitted.

The procedure is briefly demonstrated in Fig. 3.

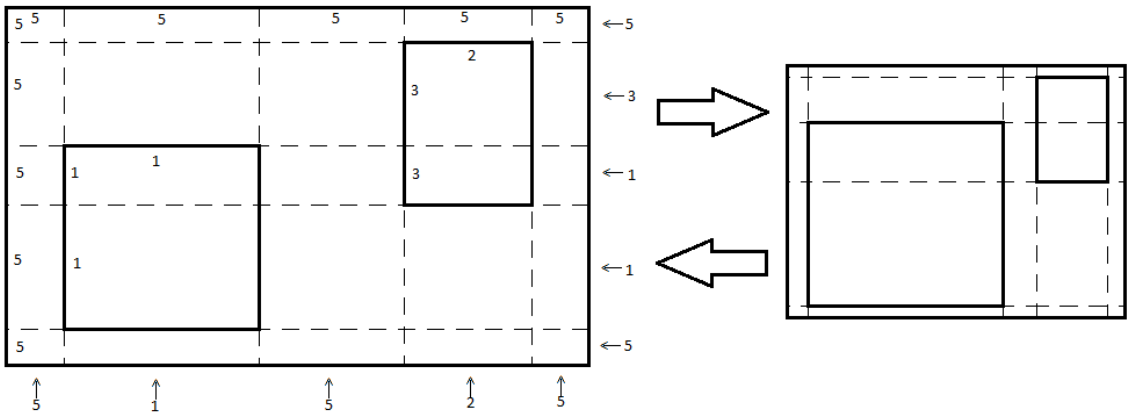

Fig. 3. Retargeting reduces the size of an image before basic compression using standard image/video technology thus reducing the total number of bits in the compressed representation. The two boxes are the bounding boxes of two objects.

d) ***Inner codec***: The proposal is blind to the image/video compression (coding) technology applied. Any existing or future the image/video compression (coding) technology may be used. Therefore, any valid image/video can be used. In CE1 VCMRS software is considered, configured with VVC as inner codec.

e) ***Entropy encoder*** : Formatting of RoI metadata for transmission as an auxiliary information in addition the bitstream produced by general image/video encoder. The format of RoI metadata must be standardized as this stream of data and must be unambiguously decodable. In our implementation we have used *bitstring* Python library [11].

3. VCM decoder structure

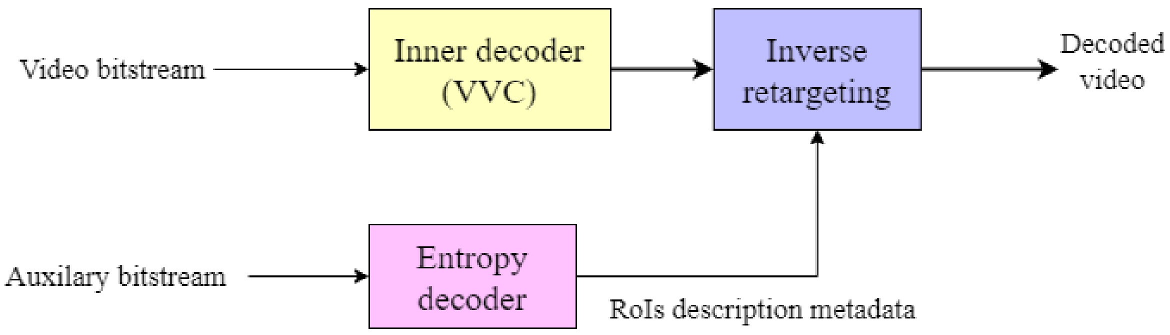

Fig. 4. The structure of the proposed tool in the decoder.

Description of the building blocks in a decoder:

a) ***Inner decoder*** : The proposal is blind to the image/video compression (coding) technology applied. Therefore, any valid image/video can be used. In CE1 VCMRS software is considered, configured with VVC as inner codec.

b) ***Entropy decoder*** : RoI metadata is decoded from the auxiliary bitstream. In our implementation we have used *bitstring* Python library [11].

c) ***Inverse retargeting*** : The inverse retargeting restores the original size (the numbers of rows and columns) of an image or video frames. Inverse retargeting exploits Metadata Class 2 that include retargeting parameters estimated by retargeting in an encoder. Moreover, Metadata Class 1 can be used to provide the information on RoIs. For a video sequence, the frame sizes after retargeting should be constant if a video encoder and decoder requires that. In fact, most of the video coding standards require that frame is size remains unchanged for a whole sequence starting with a IDR frame (Instantaneous Decoder Refresh).
Retargeting and inverse retargeting is demonstrated in Fig. 5.

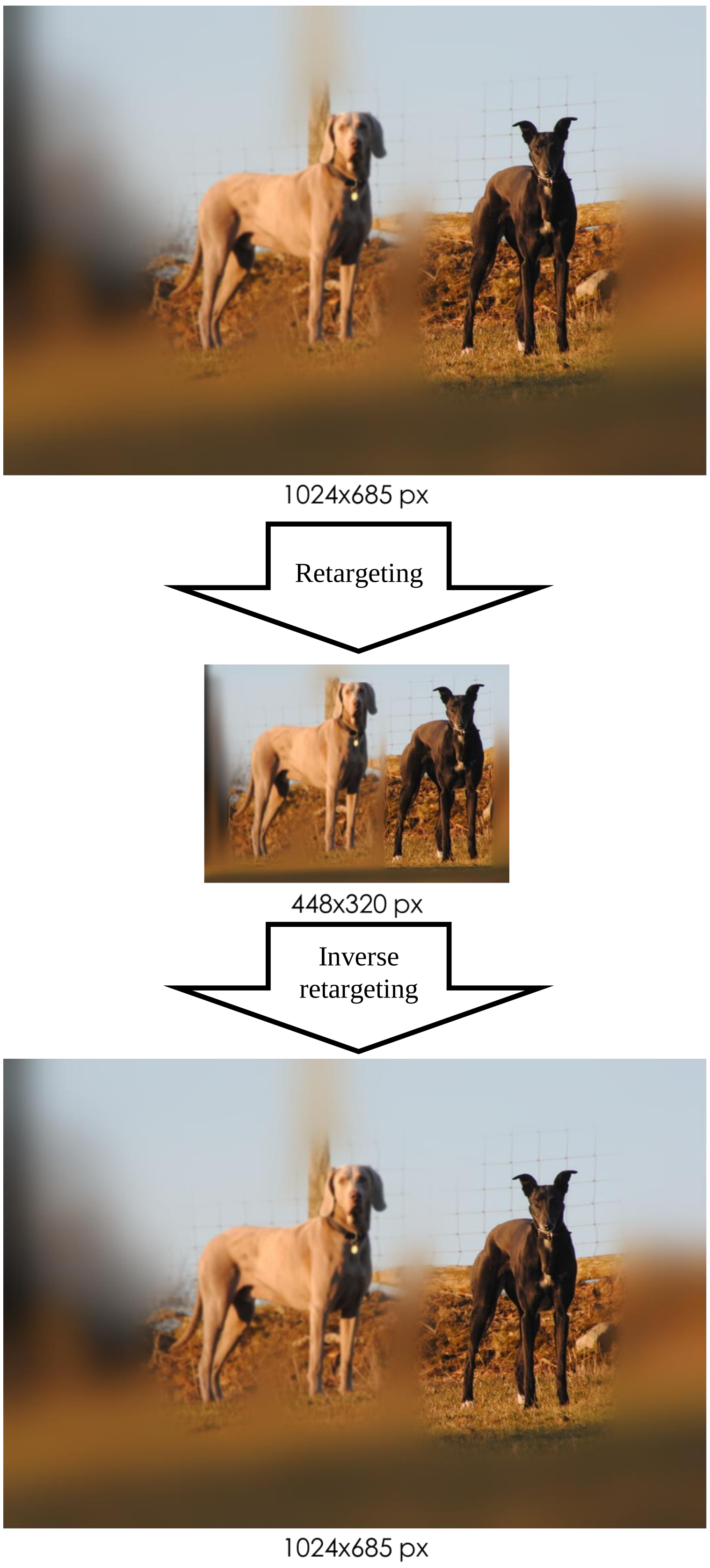

Fig. 5. An example simplified image which is retargeted for the sake of compression and then restored (inverse retargeted) to the original size and proportions.

4. Tracking

In order to avoid "blinking" of some objects in sequences, the tracking algorithm has been utilized. If the object disappears (mostly because of weak score) for few frames, the corresponding area is treated the same way, as if the object was present. Moreover, if the "object presence" in the area is extrapolated to the beginning of GOP. That reduces annoying "blinking" effect, and gives some gain in *mAP*-bitrate curves for Random Access configuration.

5. Implementation details

The tool has been implemented as ROI component in VCM-RS [8] software.

- encoder_poznanroi.py – main encoder component file
- decoder_poznanroi.py – main decoder component file
- poznanroi_config_codec.py – hard-coded parameters of the codec e.g. bluring parameters.
- poznanroi_config_classes.py – classes of objects used in the encoder
- poznanroi_detectron.py – interface to *Detectron2*
- poznanroi_blur.py – implementation of simplification (blurring) algorithm
- poznanroi_retarget.py – implementation of retargeting algorithm
- poznanroi_tracking.py – implementation of object tracking algorithm
- poznanroi_utils.py – utilities related to general processing.
- poznanroi_rois.py - utilities related to RoIs
- poznanroi_objs.py - utilities related to objects
- poznanroi_bitenc.py – bitstream encoding utilities
- poznanroi_bitdec.py – bitstream decoding utilities

The current implementation of the tools has additional requirements:

- *Detectron2* framework [9],
- *Multi-Object-Tracker* Python library [10],
- *bitstring* Python library [11] – bitstream encoding of RoIs.

It can be noted that the current implementation does NOT require GPU for operation (CPU only). It however employs GPU if available to speed-up processing. We have notices negligible differences in encoder performance when CPU is used vs GPU.

6. Syntax

Table 1. General RoI information syntax

Syntax element	C	Descriptor
VCM_info(payloadSize) {		
bits_random_access_period	5	u
random_access_period_minus_1	bits_random_access_period	u
bits_size_x	5	u
bits_size_y	5	u
output_image_size_x_minus_1	bits_size_x	u
output_image_size_y_minus_1	bits_size_y	u
resample_mul_int	7	u
bits_roi_gops_num	5	u
roi_gops_num	bits_roi_gops_num	u
if (codec_option_use_roi_retargeting) {		
for (i=0; i<roi_gops_num; i++) {		
roi_retargeting_info( )		
}		
}		

bits_random_access_period – number of bits on which random_access_period_minus_1 syntax element is signaled.

random_access_period_minus_1 – number of frames in group of pictures (gop) used for independent encoding/decoding. It specifies how often RoI metadata is retransmitted.

bits_size_x - number of bits on which output_image_size_x_minus_1 syntax element is signaled.

bits_size_y - number of bits on which output_image_size_y_minus_1 syntax element is signaled.

output_image_size_x_minus_1 - size of the original image to be outputted from the VCM decoder, minus 1.

output_image_size_y_minus_1 - size of the original image to be outputted from the VCM decoder, minus 1.

resample_mul_int – multiplier (0..127) in percents, allowing to impose additional shrinking of image during retargeting.

bits_roi_gops_num - number of bits on which roi_gops_num syntax element is signaled.

roi_gops_num – number of GOPs in sequence – this field is transmitted due to limitation of VCMRS software that allow only for one package of metadata per plugin and therefore data for all gops (required by random access period) is merged together.

codec_option_use_roi_retargeting - flag activaction

codec_option_use_dictionary_objects - flag activation

Table 2. RoI retargeting information syntax.

Syntax element	C	Descriptor
roi_retargeting_info() {		
hierarchy_power_base	8	u
if (hierarchy_power_base ! = 0) {		
bits_scale	3	u
bits_pos_x	5	u
bits_pos_y	5	u
bits_size_x	5	u
bits_size_y	5	u
bg_scale	bits_scale	u
bg_size_x_minus_1	bits_pos_x	u
bg_size_y_minus_1	bits_pos_y	u
bits_num_rois	5	5
num_rois	bits_num_rois	u
current_scale = 0		
for (i = 0; i < num_rois; i++) {		
if (current_scale!=0) {		
update_scale	1	u
if (update_scale):		
current_scale	bits_scale	u
}		
roi_scale[i] = current_scale		
roi_pos_x[i]	bits_pos_x	u
roi_pos_y[i]	bits_pos_y	u
roi_size_x_minus_1[i]	bits_size_x	u
roi_size_y_minus_1[i]	bits_size_y	u
}		
}		

hierarchy_power_base - base of power function basing on which retargeting scaling is calculated

bits_scale - number of bits on which bg_scale and roi_scale[i] syntax elements are signaled

bits_pos_x - number of bits on roi_pos_x[i] syntax element is signaled

bits_pos_y - number of bits on roi_pos_x[i] syntax element is signaled

bits_size_x - number of bits on roi_size_x_minus_1[i] syntax element is signaled

bits_size_y - number of bits on roi_size_y_minus_1[i] syntax element is signaled

bg_scale - scale level of the background (highest hierarchy RoI element)

bg_size_x_minus_1 - size (minus 1) of the background (highest hierarchy RoI element)

bg_size_y_minus_1 - size (minus 1) of the background (highest hierarchy RoI element)

bits_num_rois - number of bits on which num_rois syntax elements is signaled

num_rois - number of RoIs (excluding background)

current_scale - temporary decoding variale, used to update successive scales: roi_scale[i] roi_scale[i] - scale of RoI index i

roi_pos_x[i] - position of RoI index i

roi_pos_y[i] - position of RoI index i

roi_size_x_minus_1[i] - width (minus 1) of RoI index i

roi_size_y_minus_1[i] - height(minus 1) of RoI index i

7. Results

Table 3. Object detection performance summary for Random Access configuration

	Object Detection	End-to-End BD-Rate [%]			End-to-End BD-mAP	EncT	DecT
		mAP	mAP (low4)	Pareto mAP			
SFU-HW	Class A	####	6,48%	6,27%	0,79	208,50%	44,18%
	Class B	#NUM!	#NUM!	-17,60%	7,15	24,93%	43,30%
	Class C	####	-9,64%	-5,34%	0,74	17,23%	49,61%
	Class D	####	####	6,47%	-0,16	17,26%	50,24%
	Average	#NUM!	#NUM!	-4,58%	2,44	23,40%	47,34%
	Class O	####	-18,40%	-3,24%	1,64	25,57%	44,03%
	Class A+B+C+D	#ARG!	#ARG!	#ARG!	#ARG!	#DIV/0!	#DIV/0!

Table 4. Object tracking performance summary for Random Access configuration

	Object Tracking	End-to-End BD-Rate [%]			End-to-End BD-MOTA	EncT	DecT
		MOTA	MOTA (low4)	Pareto MOTA			
TVD	TVD-01-1	26,39%	10,51%	26,39%	-3,06	435,37%	94,59%
	TVD-01-2	####	54,20%	46,84%	-2,88	455,55%	97,13%
	TVD-01-3	8,15%	2,69%	8,15%	-0,64	375,75%	99,35%
	TVD-02-1	####	####	11,67%	-1,34	92,04%	100,45%
	TVD-03-1	3,13%	-4,52%	3,13%	-0,80	249,49%	101,25%
	TVD-03-2	44,03%	35,94%	44,03%	-4,33	324,85%	101,48%
	TVD-03-3	141,96%	142,49%	141,96%	-9,00	197,20%	102,50%
	Average	#NUM!	####	40,31%	-3,15	271,81%	99,50%

Table 5. Object detection performance summary for Low Delay configuration

	Object Detection	End-to-End BD-Rate [%]			End-to-End BD-mAP	EncT	DecT
		mAP	mAP (low4)	Pareto mAP			
SFU-HW	**Class A**	####	####	-50,85%	2,69	332,13%	101,86%
	Class B	#NUM!	-9,59%	-2,58%	1,85	36,22%	101,89%
	Class C	-0,33%	6,41%	-0,33%	-0,09	51,33%	91,45%
	Class D	0,76%	-5,01%	0,76%	0,38	32,35%	95,38%
	Average	#NUM!	####	-4,57%	0,87	46,18%	96,57%
	Class O	####	-18,04%	-20,38%	8,69	9,19%	101,34%
	Class A+B+C+D	#ARG!	#ARG!	#ARG!	#ARG!	#DIV/0!	#DIV/0!

Table 6. Object tracking performance summary for Low Delay configuration

	Object Tracking	End-to-End BD-Rate [%]			End-to-End BD-MOTA	EncT	DecT
		MOTA	MOTA (low4)	Pareto MOTA			
TVD	**TVD-01-1**	####	14,69%	24,63%	-3,20	724,79%	100,29%
	TVD-01-2	####	49,81%	46,07%	-3,40	688,38%	98,66%
	TVD-01-3	16,47%	0,89%	16,47%	-1,86	632,82%	98,91%
	TVD-02-1	-4,86%	-13,15%	-4,86%	-0,41	589,29%	100,99%
	TVD-03-1	-3,02%	-13,03%	-3,02%	0,44	464,12%	101,48%
	TVD-03-2	62,17%	34,97%	62,17%	-8,21	459,41%	102,66%
	TVD-03-3	106,26%	75,26%	106,26%	-7,07	451,08%	101,78%
	Average	#NUM!	21,35%	35,39%	-3,39	562,85%	100,67%

Table 7. Object detection performance summary for All Intra configuration

	Object Detection	End-to-End BD-Rate [%] mAP	mAP (low4)	Pareto mAP	End-to-End BD-mAP	EncT	DecT
SFU-HW	Class A	####	7,51%	14,90%	-2,61	431,37%	56,19%
	Class B	####	####	-3,53%	2,30	26,33%	53,63%
	Class C	20,18%	15,07%	20,18%	-2,48	53,39%	54,82%
	Class D	3,77%	-4,21%	3,77%	0,17	60,47%	60,15%
	Average	#NUM!	####	7,43%	-0,21	52,42%	56,14%
	Class O	####	-7,99%	-3,99%	0,14	11,62%	53,91%
	Class A+B+C+D	#ARG!	#ARG!	#ARG!	#ARG!	#DIV/0!	#DIV/0!
Image Datasets	OpenImages	-40,49%	-42,06%	-40,49%	3,18	42,34%	45,67%
	FLIR	####	-48,74%	-47,30%	3,81	64,15%	111,00%
	TVD	-50,55%	-48,92%	-50,55%	7,65	62,16%	131,25%
	Average	####	-46,57%	-46,11%	4,88	55,27%	87,30%

Table 8. Object tracking performance summary for All Intra configuration

	Object Tracking	End-to-End BD-Rate [%] MOTA	MOTA (low4)	Pareto MOTA	End-to-End BD-MOTA	EncT	DecT
TVD	TVD-01-1	69,95%	66,18%	69,95%	-5,36	166,10%	100,98%
	TVD-01-2	####	46,47%	33,29%	-1,68	138,87%	98,59%
	TVD-01-3	-45,29%	-44,83%	-45,29%	5,94	67,00%	94,27%
	TVD-02-1	-6,09%	-18,47%	-6,09%	-1,51	67,28%	100,56%
	TVD-03-1	22,88%	14,51%	22,88%	-3,72	59,28%	100,69%
	TVD-03-2	29,24%	12,13%	29,24%	-3,99	101,46%	100,28%
	TVD-03-3	-3,08%	-5,04%	-3,08%	0,55	107,79%	101,11%
	Average	####	10,13%	14,41%	-1,40	94,52%	99,47%

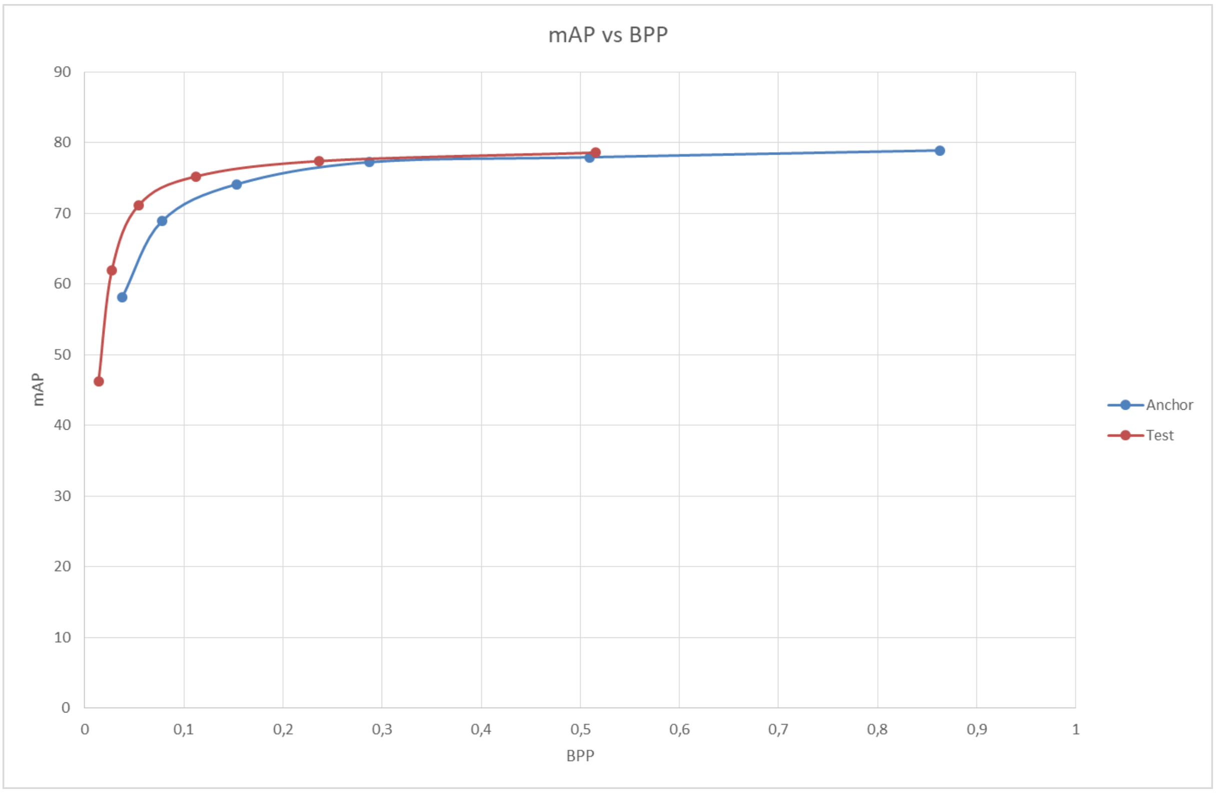

Fig. 6. Object detection performance figure (mAP – bitrate) on Openimages dataset (All Intra configuration)

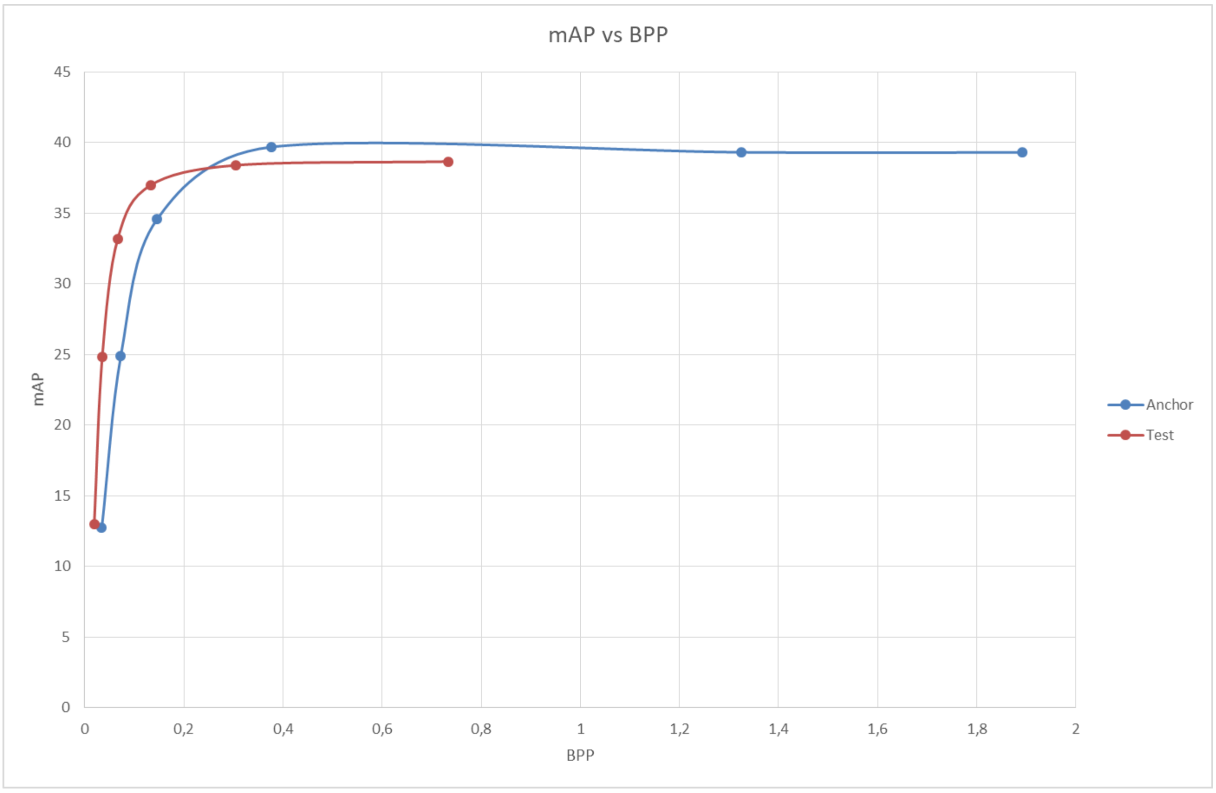

Fig. 7. Object detection performance figure (mAP – bitrate) on TVD images dataset (All Intra configuration)

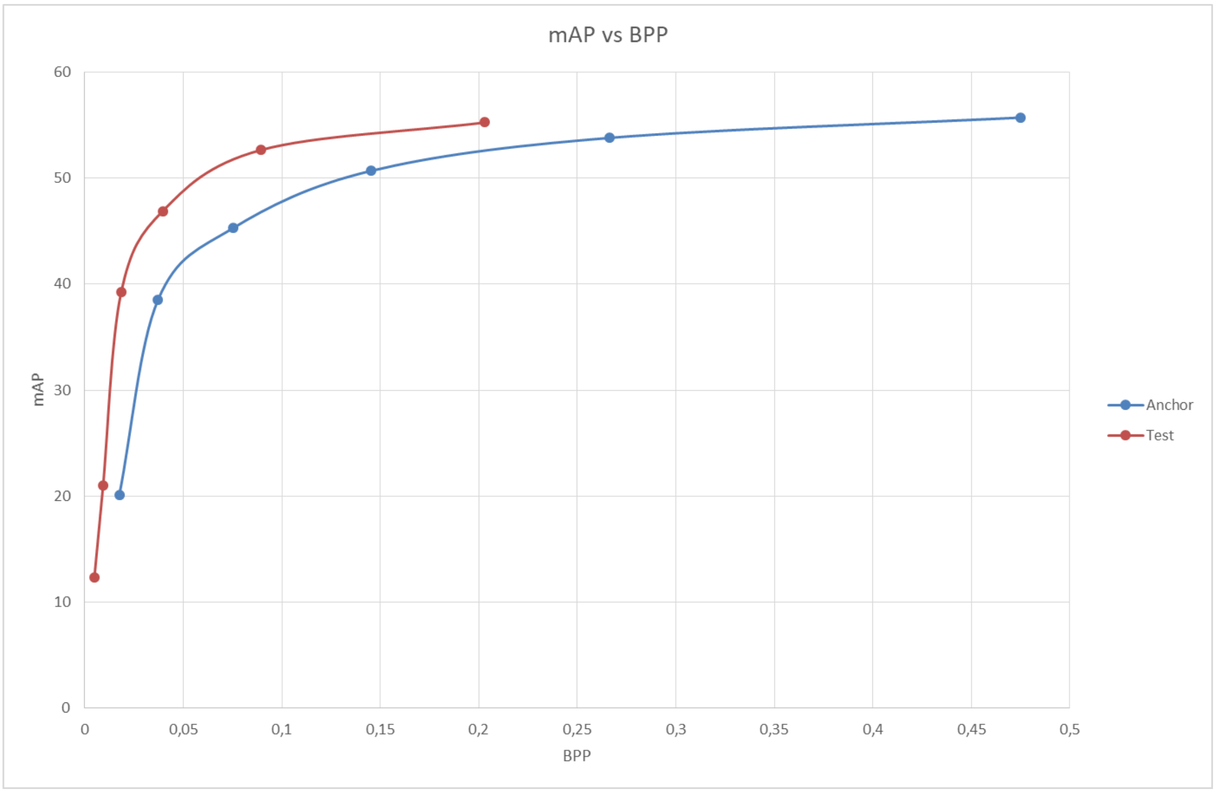

Fig. 7. Object detection performance figure (mAP – bitrate) on TVD images dataset (All Intra configuration)

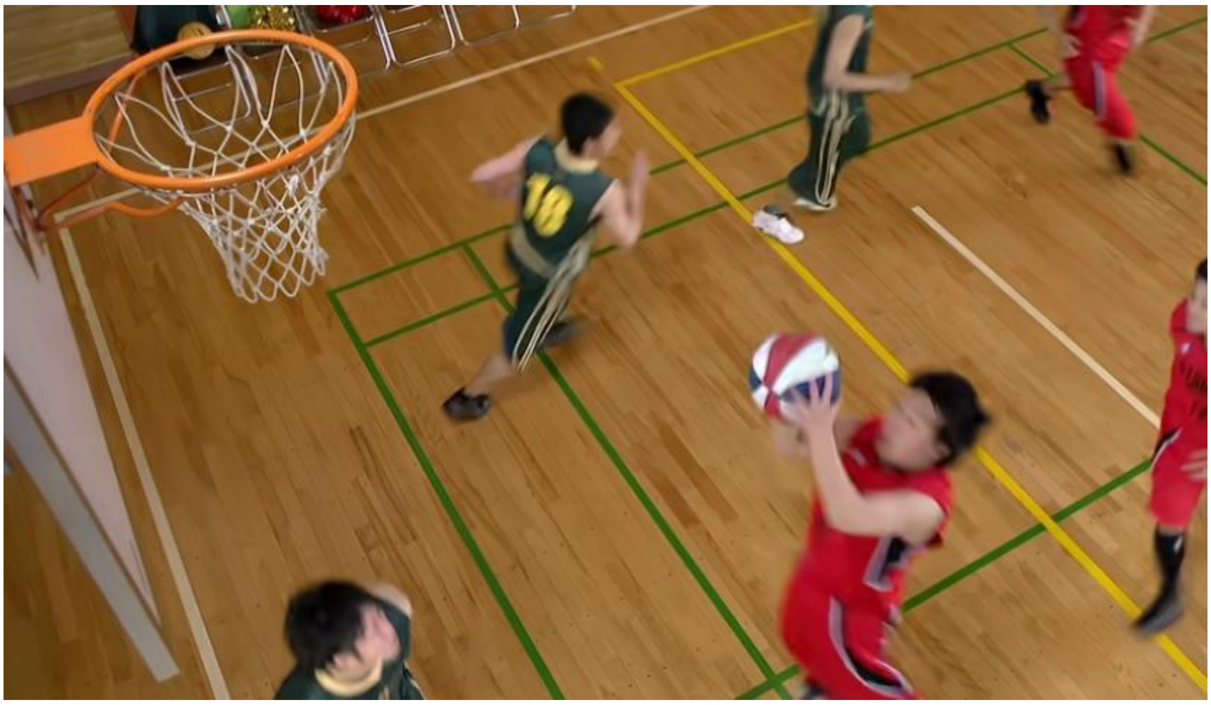

Fig. 9. Exemplary decoded frame from SFU dataset (BasketballDrill, qp=27, VTM Random Access)

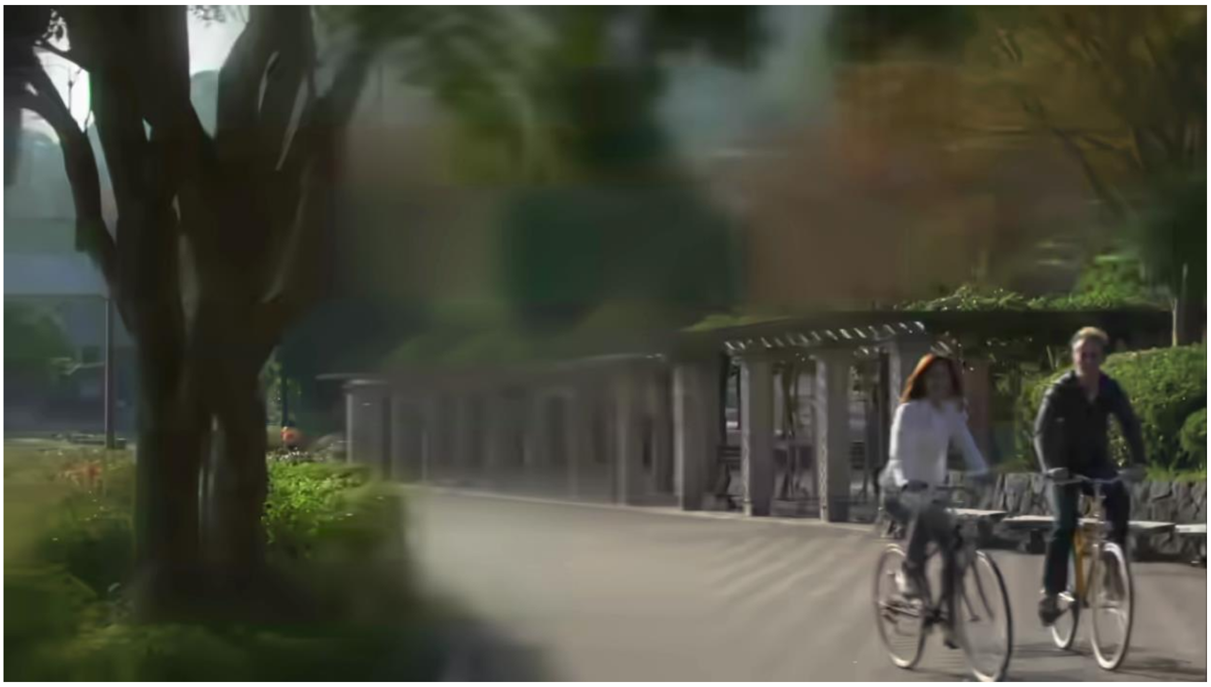
Fig. 10. Exemplary decoded frame from SFU dataset (ParkScene, qp=44, VTM Random Access)

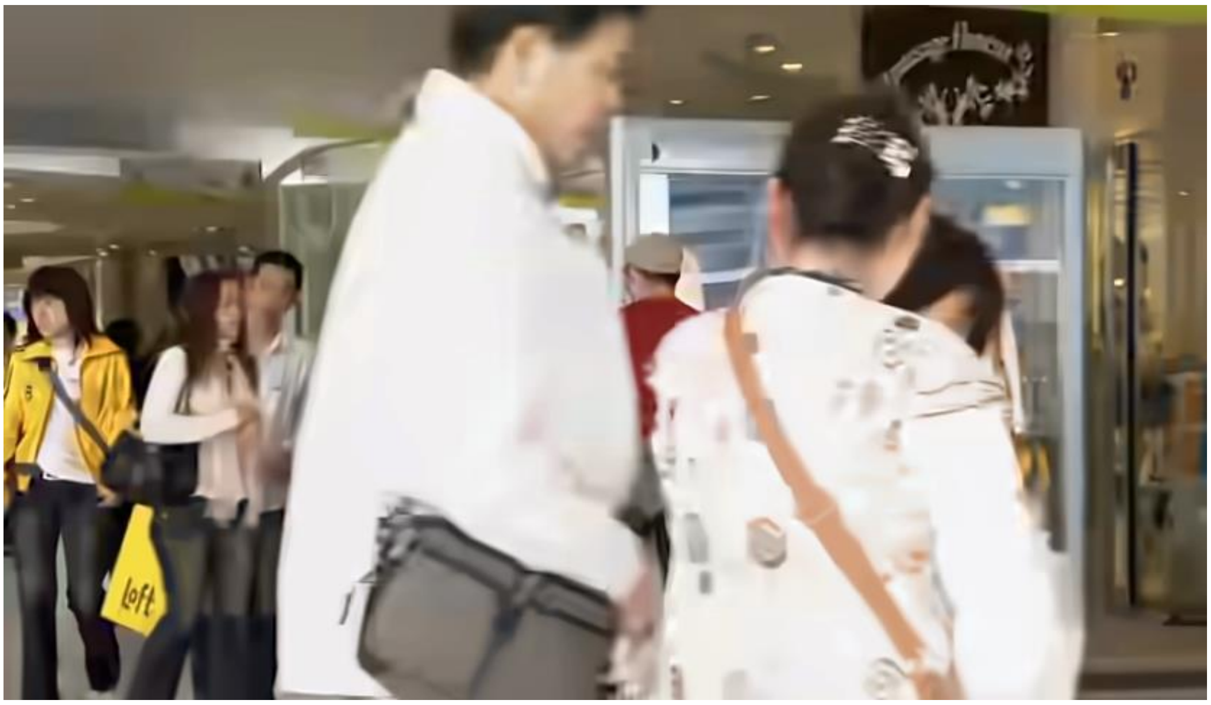

Fig. 11. Exemplary decoded frame from SFU dataset (BQMall, qp=47, VTM Random Access)

8. Conclusions

The conducted core experiment was related to the the RoI-based preprocessing and retargeting tools proposed originally to the CfP by Poznan University of Technology. These tools have been used as replacement for the currently adopted RoI-based preprocessing (which significantly remove background), The proposed simplification preprocessing step is not removing the background significantly, which enhances usability of the proposed technology also for human viewing.

The proposed tools have now been implemented in the newest VCMRS software version 0.6.1 and tested. The attained results demonstrate promising performance in the form of rate-distortion curves which however cannot be measured numerically due to problems related to non-monotonicity. Despite numerous attempts a set of QP values (that would guarantee monotonicity of the curves and allow reliable numerical evaluation) could not be found for video datasets.

The results also show good performance of the tools for image encoding, especially open-images, for low-bitrate applications. The attained gains are **around 40-50% of BD-Rate bitrate reduction.** Additionally it is shown that the encoding and decoding time can be vastly reduced (**even to 45% of the anchor processing time**). This is attained due to retargeting technique which vastly reduce resolution of the images encoded by the inner codec. It must be however highlighted that the time measurement methodology settled within VCM group is very vulnerable to computing environment, which disallowed to confirm such results for other smaller image datasets (too few samples to gather reliable statistics).

It can also be noted that the currently adopted RoI-based preprocessing tool is not working for images so the proposed tools are not overlapping and can be adopted noncompetitively.

9. Recommendation

- Adopt the proposed tools and enable them for image datasets (currently adopted RoI-based preprocessing tool is inactive for images).
- Consider improving evaluation methodology in order to avoid non-monotonicity problems, which disallow reliable assessment of the proposed tools for videos.
- Continue core experiment for video sequences.

References

[1] ISO/IEC JTC 1/SC 29/WG 04, "VCM-CE 1: RoI based coding methods", doc. N0380, July 2023, Geneva.

[2] Marek Domański, Olgierd Stankiewicz, Sławomir Maćkowiak, Sławomir Różek, Tomasz Grajek, Jakub Szekiełda, Dominik Cywiński, Jakub Siejak, "[VCM] Poznań University of Technology Proposals A and B in response to CfP on Video Coding for Machines", ISO/IEC JTC 1/SC29/WG4, doc M61519, October 2022, online.

[3] ISO/IEC JTC 1/SC29/WG2, "Call for Proposals for Video Coding for Machines" doc N191, April 2022, online.

[4] Marek Domański, Olgierd Stankiewicz, Sławomir Maćkowiak, Tomasz Grajek, Maciej Wawrzyniak, Jakub Stankowski, Sławomir Różek, Dominik Cywiński, Jakub Szekiełda, Jakub Siejak, "[VCM] CE1.6 - RoI-based simplification and retargetting for Video Coding for Machines", ISO/IEC JTC 1/SC29/WG4, doc. M62235, January 2023, online.

[5] Marek Domański, Olgierd Stankiewicz, Sławomir Maćkowiak, Sławomir Różek, Tomasz Grajek, Maciej Wawrzyniak, Jakub Szekiełda, Dominik Cywiński, Jakub Siejak, "[VCM CE1] Description and results of CE1.6 on PUT proposal on RoI preprocessing and retargeting ", ISO/IEC JTC 1/SC29/WG4, doc. M63119, April 2023, Antalya.

[6] Marek Domański, Olgierd Stankiewicz, Sławomir Maćkowiak, Sławomir Różek, Tomasz Grajek, Maciej Wawrzyniak, Jakub Szekiełda, Dominik Cywiński, Jakub Siejak, "[VCM CE1.6] Improvements and results to the codec based on CE1.6 ", ISO/IEC JTC 1/SC29/WG4, doc. M63672, July 2023, Geneva.

[7] Honglei Zhang , "Introduction to the VCM reference software (VCM-RS)", ISO/IEC JTC 1/SC 29/WG 4, doc. M62003, January 2023, online.

[8] https://mpeg.expert/software/MPEG/Video/VCM/VCM-RS

[9] https://github.com/facebookresearch/Detectron2

[10] https://github.com/adipandas/multi-object-tracker

[11] https://pypi.org/project/bitstring/