INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC 1/SC 29/WG 2
MPEG TECHNICAL REQUIREMENTS

ISO/IEC JTC 1/SC 29/WG 2 M56099
Online – January 2021

Title: **[VCM] Poznan University of Technology additional anchor experiment for object detection**
Source: WG 2 MPEG Technical requirements
Editor(s): **Marek Domański, Tomasz Grajek, Sławomir Maćkowiak, Sławomir Różek, Olgierd Stankiewicz, Jakub Stankowski**
Poznań University of Technology, Poznań, Poland
Status: Approved

Abstract
This contribution reports the additional anchor result for object detection. The results have been obtained using COCO val. 2017 dataset. The processing flow follows the methodology described in Evaluation Framework for VCM document. Especially the Versatile Video Coding for compression, and FFmpeg software for image processing have been used. The evaluation have been done using the neural networks from Facebook Research Detectron2 project.

1. Introduction
For this experiment, the COCO 2017 validation dataset [1] have been used, which contains 5000 images in JPEG format. The methodology of experiment follows the recommendations described in Evaluation Framework for VCM document [2]. After some preprocessing done onto dataset, the images are compressed and decompressed using VVC codec [3]. Four different scaling factors (25%, 50%, 75%, 100%), and seven different QPs (17,22,27,32,37,42,47) have been tested. The decoded images are fed into the convolutional neural network input. Finally, the impact of compression onto the detection performance is investigated.

2. Experiment description
To repeat the anchor generation process, the following steps on dataset have been performed:
a) Preprocessing using FFmpeg software [4]:
- Padding to obtain even size for 100% scale images:

```
ffmpeg –i coco_image.jpg –vf "pad=ceil(iw/2)*2:ceil(ih/2)*2" -f rawvideo -
pix_fmt yuv420p10le -dst_range 1 coco_image_100.yuv
```

- Scaling to obtain images with 25%, 50%, and 75%.

```
ffmpeg –i coco_image.jpg –vf "scale=ceil(iw*/8)*2:ceil(ih*/8)*2" -f rawvideo -
pix_fmt yuv420p10le -dst_range 1 coco_image_25.yuv
ffmpeg –i coco_image.jpg –vf "scale=ceil(iw*/4)*2:ceil(ih*/4)*22" -f rawvideo -
pix_fmt yuv420p10le -dst_range 1 coco_image_50.yuv
ffmpeg –i coco_image.jpg –vf "scale=ceil(iw*3/8)*2:ceil(ih*3/8)*2" -f rawvideo -
pix_fmt yuv420p10le -dst_range 1 coco_image_75.yuv
```

- Simultaneous conversion into YUV format

b) Encoding and decoding using VTM [3] software

```
EncoderApp.exe -c VVCSoftware_VTM-VTM-8.2/cfg/encoder_intra_vtm.cfg
-i coco_image_scale.yuv -o decoded_scale_qp.yuv -b bitstream.vvc -q QP
--ConformanceWindowMode=1 -wdt WDT -hgt HGT -f 1 -fr 1
--InternalBitDepth=10 --InputBitDepth=10
```

c) Postprocessing of the decoded images

- Conversion decoded images into PNG format:

```
ffmpeg -f rawvideo -pix_fmt yuv420p10le -s WDTxHGT -src_range 1 -i
decoded_scale_qp.yuv -frames 1 -pix_fmt rgb24 decoded_scale_qp.png
```

- Cropping into original size for 100% images

```
ffmpeg -i decoded_100_qp.png -vf "crop=ORG_WDT:ORG_HGT" decoded_100_qp_org.png
```

- Upscaling into original size for 25%, 50%, and 75% images

```
ffmpeg -i decoded_scale_qp.png -vf "scale= ORG_WDT:ORG_HGT"
decoded_scale_qp_org.png
```

d) Evaluation using Faster R-CNN X101-FPN and COCO evaluation framework:

```
# python meta code
# import necessary modules before
path = 'path/to/decoded/images'
cfg_file = 'detectron2/configs/COCO-
Detection/faster_rcnn_X_101_32x8d_FP_3x.yaml'
anno_file = 'COCO/annotations/instances_val2017.json'
opts = ['MODEL.WEIGHTS' 'detectron2://COCO-
Detection/faster_rcnn_X_101_32x8d_FPN_3x/139173657/model_final_68b088.pkl']
cfg = get_cfg()
cfg.merge_from_file(cfg_file)
cfg.merge_from_list(opts)
register_coco_instances('coco_2017_val_scaled_decoded',{}, anno_file, path)
dicts = load_coco_json(anno_file, path)
model = build_model(cfg)
DetectionCheckpointer(model).load(opts[1])
```

```
evaluator = COCOEvaluator('coco_2017_val_scaled_decoded', distributed=True,
output_dir=path)
loader = build_detection_test_loader(cfg,'coco_2017_val_scaled_decoded')
eval_result = inference_on_dataset(model, loader, evaluator)
with open('eval_result.json', "w") as result_file:
    json.dump(eval_result, result_file)
```

3. Object detection results

The evaluation results of object detection for different scales and QPs are listed in the table below. The BPP was calculated with reference to the original image size. For detection, the neural-network-based implementation Faster R-CNN X101-FPN [5, 6] was used according to [2].

Scale	QP	AP	AP50	AP75	APs	APm	APl	BPP
Uncompressed		**39.6461**	57.0530	43.8662	22.6243	42.8130	52.1682	-
100%	17	**38.5523**	55.7223	42.7728	21.4344	41.6722	50.9663	2.3478
	22	**38.1539**	55.1943	42.1362	21.4053	41.0176	50.3860	1.5943
	27	**37.0846**	53.8311	40.9848	20.2389	40.0794	49.3140	1.0057
	32	**34.5266**	50.7434	38.1235	18.0135	37.2643	46.9073	0.5806
	37	**28.8021**	43.2156	31.3173	13.7394	31.3168	40.4821	0.3082
	42	**20.9033**	32.0153	22.6167	7.8678	22.7452	31.9941	0.1468
	47	**11.5946**	18.2196	12.3310	2.5414	11.6938	20.8506	0.0619
75%	17	**36.7161**	53.4318	40.2360	18.5596	40.1880	49.8819	1.2507
	22	**36.1158**	52.8694	39.7235	17.8626	39.3464	49.3977	0.8284
	27	**34.5206**	50.6367	38.0363	16.5292	37.8136	47.8157	0.5241
	32	**30.8726**	45.8255	33.9413	14.1679	33.7485	43.9561	0.3080
	37	**24.8192**	37.7559	26.8895	10.2250	27.1970	37.0647	0.1649
	42	**16.2801**	25.4173	17.5249	4.7075	17.1409	27.4275	0.0811
	47	**7.8697**	12.7701	8.2531	1.0772	7.1023	15.2627	0.0373
50%	17	**32.3205**	47.6440	35.1893	13.7895	35.5593	46.6873	0.5751
	22	**31.3104**	46.3088	34.1350	12.6794	34.3326	45.5849	0.3915
	27	**29.0040**	43.2619	31.6855	11.6375	31.7160	42.9143	0.2528
	32	**24.4664**	37.2137	26.2792	8.0841	26.9893	37.7503	0.1507
	37	**17.6854**	27.5809	19.0166	4.5538	18.6932	29.9357	0.0823
	42	**10.2643**	16.4226	10.8337	1.6682	9.8530	20.0505	0.0423
	47	**4.1166**	6.7473	4.3531	0.2780	2.4777	9.6696	0.0207
25%	17	**18.4251**	28.3282	19.7304	3.2171	18.3384	33.1185	0.1701
	22	**17.3825**	26.7154	18.5204	2.8736	16.8663	31.7055	0.1199
	27	**15.0169**	23.3646	16.1857	2.2689	14.2474	28.0743	0.0801
	32	**11.3653**	17.9152	12.0985	1.4412	10.1584	22.4035	0.0496
	37	**6.8075**	11.1065	7.2486	0.4072	5.3196	14.6188	0.0287
	42	**3.0735**	5.1144	3.0758	0.0968	1.8461	7.2506	0.0160
	47	**1.1711**	1.8935	1.2622	0.0149	0.4861	2.8077	0.0087

Table 1. Anchor results for object detection on COCO 2017 val. dataset.

Some metrics from the Table 1. are also shown on the figures below. Each line represents one scaling factor and contains seven points for each QP.

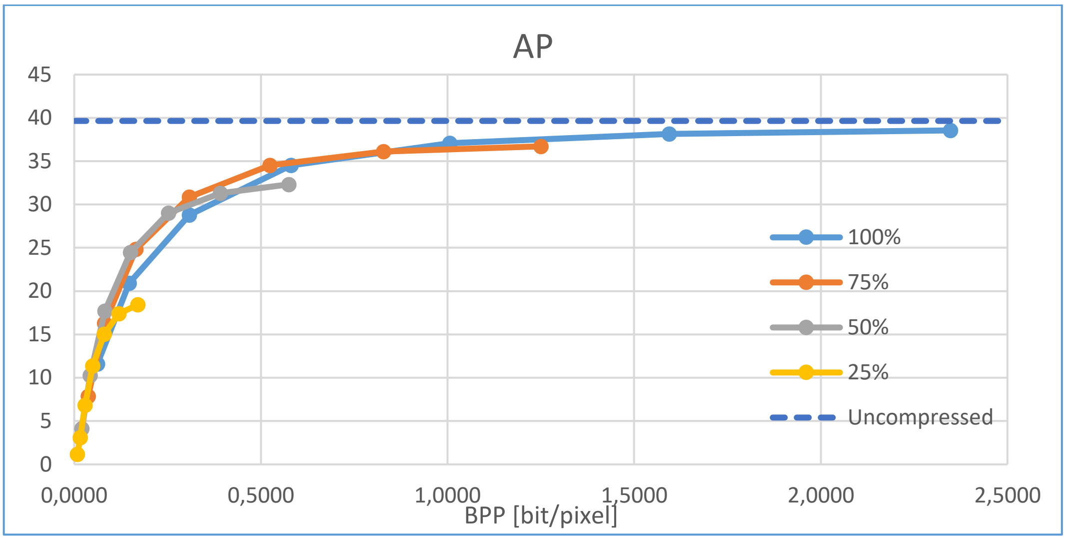

Fig. 1. Average precision for all objects.

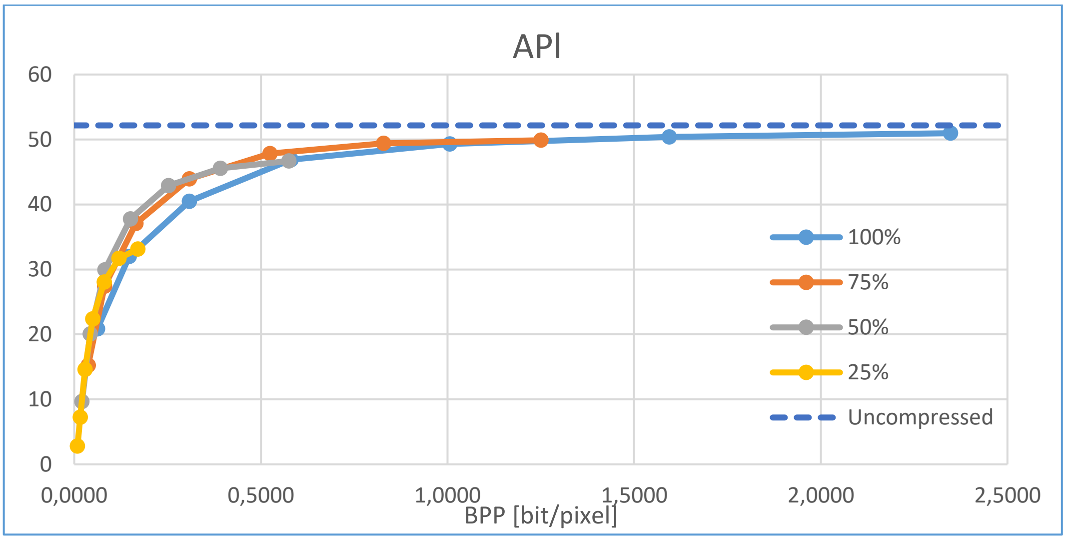

Fig. 2. Average precision for large targets.

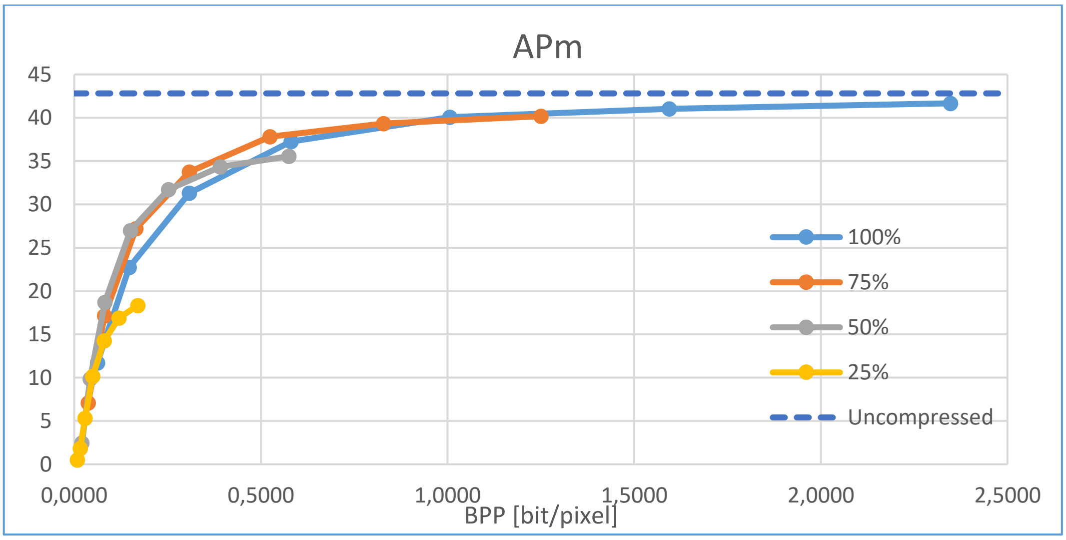

Fig. 3. Average precision for medium targets.

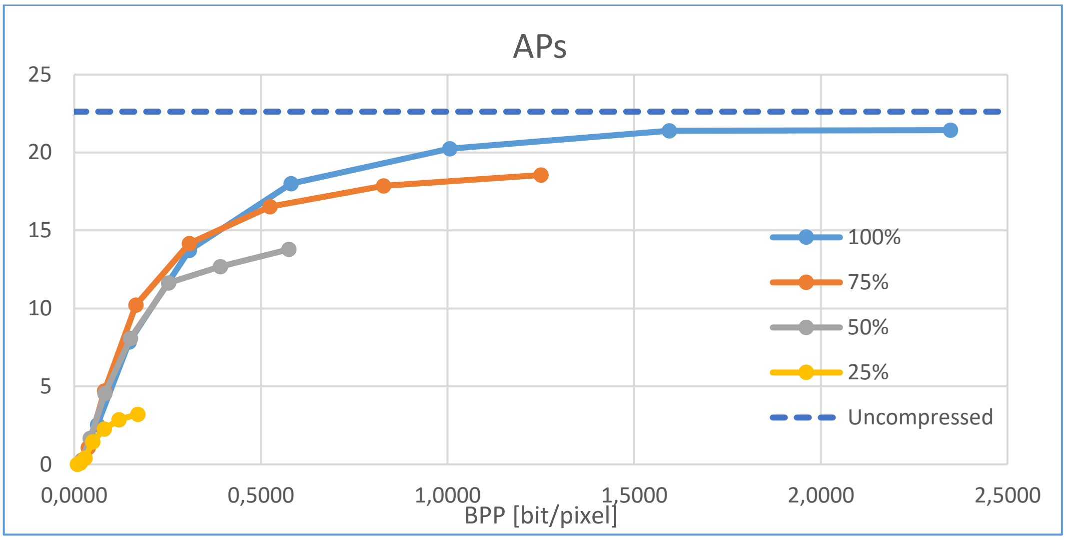

Fig. 4. Average precision for small targets.

4. Conclusions

The outcomes from experiment are similar to the anchors reported in [7, 8], but are not the same. For further experiments, to achieve consistency and uniqueness of the results, it is necessary to exact define the neural network weights and parameters set to use. It is also recommended to provide more precise experiment descriptions in the contributions, with more technical details, to simplify eventual test repetition by 3rd party.

5. Acknowledgement

This work was supported by Project 0314/050.

6. References

[1] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft COCO: Common objects in context," In ECCV, 2014.
[2] ISO/IEC JTC 1/SC 29/WG 2, "Evaluation Framework for Video Coding for Machines", Doc. N19, October 2020.
[3] VTM8.2, https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tree/VTM-8.2.
[4] FFmpeg, https://github.com/FFmpeg/FFmpeg .
[5] Y. Wu, A. Kirillov, F. Massa, W. Lo and R. Girshick, "Detectron 2", 2019, https://github.com/facebookresearch/detectron2.
[6] Faster R-CNN X101-FPN, https://github.com/facebookresearch/detectron2/blob/master/configs/COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml .
[7] Sheng-Po Wang, Erh-Chung Ke, Ching-Chieh Lin, Chun-Lung Lin (ITRI), "[VCM] Anchor generation results for object detection on COCO dataset", Doc. M54861, October 2020.
[8] Tsung-Hua Li, Yu-Chieh Nien, Deng-Rung Liu (Foxconn), "[VCM] Crosscheck of m54861 (Anchor generation results for object detection on COCO dataset)", Doc. M54862, October 2020.