https://doi.org/10.2352/ISSN.2470-1173.2018.13.IPAS-328

Real-time 3DRS motion estimation for frame-rate conversion

Petr Pohl, Valery Anisimovskiy, Igor Kovliga, Alexey Gruzdev and Roman Arzumanyan
Samsung R&D Institute Russia Moscow, Russia

Abstract

In this article we present an updated version of a block-wise video motion estimation (ME) algorithm that is intended to be part of real-time frame rate conversion (FRC) software system. As a baseline, we choose a variant of a 3D recursive search (3DRS) algorithm, due to its relative simplicity and known real-time performance potential. We introduce updating of this algorithm, which provides effective multithreaded parallelisation in multicore systems and decreases the number of computations without a noticeable decrease in quality.

Introduction

A motion estimation (ME) algorithm is a crucial part of many algorithms and systems, for example video encoders, frame rate conversion (FRC), and structure from motion. The performance of the ME algorithm typically provides an overwhelming contribution to the performance of the ME-based algorithm in terms of both computational complexity and visual quality, and it is therefore critical for many ME applications to have a low-complexity ME algorithm that provides a good-quality motion field. However, the ME algorithm is highly task-specific, and there is no "universal" ME that is easily and efficiently applicable to any task. Since we focus our efforts on FRC applications, we choose the 3D Recursive Search (3DRS) algorithm [1] as a baseline ME algorithm, as it is well suited for real-time FRC software applications.

The 3DRS algorithm has several important advantages that allow a reasonable quality of the motion field used for FRC to be obtained at low computational cost. Firstly, it is a block matching algorithm (BMA); secondly, it checks a very limited set of candidates for each block; and thirdly, many techniques developed for other BMAs can be applied to 3DRS to improve the quality, computational cost, or both. Many 3DRS-based ME algorithms are well known.

One considerable drawback of 3DRS-based algorithms with a meandering scanning order is the impossibility of parallel processing when a spatial candidate lies in the same row as the current block being propagated. **Figure 1** shows the processing dependency. The green blocks are those which need to be processed before processing the current block (depicted by a white colour in a red border). Blocks marked in red cannot be processed until the processing of the current block is finished. A darker colour shows a direct dependency.

This drawback limits processing speed, since only one processing core of a multicore processor (MCP) can be used for 3DRS computation. This can also increase the power consumption of the MCP since power consumption rises super-linearly with increasing clock frequency. A time-limited task can be solved more power-efficiently on two cores with a lower frequency than on one core with a higher frequency.

In this work, we introduce several updates to our variant of the 3DRS algorithm that allow multithreaded processing to obtain a motion field and also improve the computational cost without any noticeable degradation in the quality of the resulting motion field.

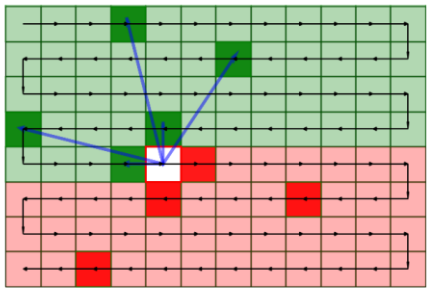

Figure 1. Trees of direct dependencies (green and red) and areas of indirect dependencies (light green and light red) for a meandering order

Baseline 3DRS-based algorithm

The 3DRS algorithm is based on block matching, using a frame divided into blocks of pixels $B(\vec{X})$, where $\vec{X} = \binom{x}{y}$ are the coordinates of the centre of the block. Our FRC algorithm requires two motion fields for each pair of consecutive frames $F(t-1), F(t)$. The forward motion field D_{FW} is the set of displacement vectors $D_{FW}(\vec{X}, t-1)$ assigned to the blocks of $F(t-1)$. These displacement vectors point to frame $F(t)$. The backward motion field D_{BW} is the set of displacement vectors $D_{BW}(\vec{X}, t)$ assigned to the blocks of $F(t)$. These displacement vectors point to frame $F(t-1)$.

We used following candidate set $CS(\vec{X})$ to search the current motion field D_{cur}:

$$CS(\vec{X}) = \{\, CS_{spatial}(\vec{X}), CS_{temporal}(\vec{X}), CS_{random}(\vec{X}), \},$$

$$CS_{spatial}(\vec{X}) = \{\, c\vec{sD} \,|\, \vec{D}_{cur}(\vec{X} + US_{cur})\},$$

$$US_{cur} = \begin{Bmatrix} -1W & 0 & -4W & -1W & 2W \\ 0 & -1H & -1H & -4H & -3H \end{Bmatrix},$$

$$CS_{temporal}(\vec{X}) = \{\, \overrightarrow{csD} \,|\, \vec{D}_{pred}(\vec{X} + US_{pred})\},$$

$$US_{pred} = \begin{Bmatrix} 0 & 0 & 1W & 4W \\ 0 & 1H & 0 & 2H \end{Bmatrix},$$

$$CS_{random}(\vec{X}) = \left\{ \overrightarrow{csD} \middle| \overrightarrow{csD}_{best}(\vec{X}) + \begin{matrix} rnd(2) & rnd(2) & rnd(9) \\ rnd(2) & rnd(2) & rnd(9) \end{matrix} \right\},$$

$$\overrightarrow{csD}_{best}(\vec{X}) = \underset{\overrightarrow{csD} \in \{CS_{spatial}(\vec{X}), CS_{temporal}(\vec{X})\}}{\operatorname{argmin}} MAD(\vec{X}, \overrightarrow{csD}),$$

where W and H are the width and height of a block (we used 8x8 blocks); $rnd(k)$ is a function whose result is a random value from the range $[-k, -k+1, \ldots, k-1, k]$; MAD is the mean absolute difference between the window over the current block $B(\vec{X})$ of one frame and the window over the block pointed to by displacement

vector $\vec{D}$ in another frame; and the size of the windows is 16x12. $\vec{D}_{cur}$ is the motion vector from the current motion field, and $\vec{D}_{pred}$ is a predictor obtained from the previously found motion field. If the forward motion field $D_{FW}(t-1)$ is searched, then the predictor will be inverted $(-D_{BW}(t-1))$; if the backward motion field $D_{BW}(\vec{X},t)$ is searched, then the predictor will be formed from $D_{FW}(t-1)$ by projecting one to block-grid of frame $F(t)$ with the subsequent inversion $(-P(D_{FW}(t-1)))$.

In fact, two ME passes are used for each pair of frames: the first pass is an estimation of the forward motion field, and the second pass is an estimation of the backward motion field. We used different scanning orders for the first and second passes. The top-to-bottom meandering scanning order (from top to bottom, and from left to right for odd rows, and from right to left for even rows) is used for the first pass (left-hand image of **Figure 2**). A bottom-to-top meandering scanning order (from bottom to top, and from right to left for odd rows and from left to right for even rows; rows are numbered from bottom to top in this case) is used for the second pass (right-hand image of **Figure 2**).

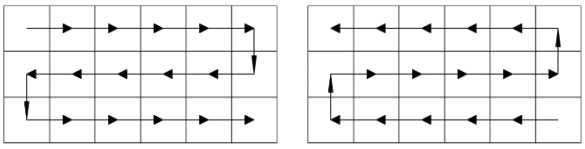

Figure 2. Left: Top-to-bottom meandering order used for forward ME, Right: Bottom-to-top meandering order used for backward ME

The relative positions US_{cur} of the spatial candidate set $CS_{spatial}(\vec{X})$ and the relative positions US_{pred} of the temporal candidate set $CS_{temporal}$ in the above description are valid only for top-to-bottom and left-to-right directions. If the direction is inverted for some coordinates, then the corresponding coordinates in US_{cur} and US_{pred} should be inverted accordingly. Thus, the direction of recursion changes in a meandering scanning order from row to row. In **Figure 3**, we show the sources of spatial candidates ($CS_{spatial}$, green blocks) and temporal candidates ($CS_{temporal}$, orange blocks) for two scan orders. On the left-hand side this is the top-to-bottom, left-to-right direction, and on the right-hand side, it is the top-to-bottom, right-to-left direction.

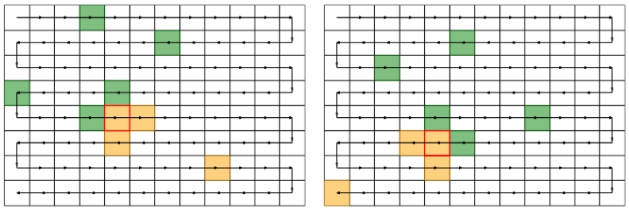

Figure 3. Left: sources of spatial and temporal candidates for a block (marked by red box) in an even row during forward ME; Right: sources of a block in an odd row during forward ME

The block erosion process [1] was skipped in our modification of 3DRS. For additional smoothing of the backward motion field, we applied additional regularisation after ME. For each block, we compared $MAD_{narrow}(\vec{X},\vec{D})$ for the current motion vector $\vec{D}$ of the block and $MAD_{narrow}(\vec{X},\vec{D}_{median})$, where $\vec{D}_{median}$ is a vector obtained by per-coordinate combining of the median values of a set consisting of the nine motion vectors from an 8-connected neighbourhood and from the current block itself. Here, MAD_{narrow} is different from MAD, which is used for 3DRS matching, and the size of the window for MAD_{narrow} is decreased to 8x8 (to give equal block sizes). The original motion vector is overwritten by $\vec{D}_{median}$ if MAD_{narrow} is better or worse by a small margin.

Wave-front scanning order

The use of a meandering scanning order in combination with the candidate set described above prevents the possibility of parallel processing several blocks of a given motion field. This is illustrated in **Figure 1**. The blocks marked in green should be processed before starting the processing of the current block (marked in white in a red box) due to their direct dependency via the spatial candidate set $CS_{spatial}$ and the blocks marked in red, which will be directly affected by the estimated motion vector in the current block. The light green and light red blocks mark indirect dependencies. Thus there are no blocks that can be processed simultaneously with the current block, since all blocks need to be processed either before or after the current block.

In [2], the authors propose a parallel processing which preserves the direction switching of a meandering order. The main drawback is that the spatial candidate set is not optimal, since the upper blocks for some threads are not processed at the beginning of row processing. If we change the meandering order to a simple "raster scan" order (always from left to right in each row), then the dependencies become smaller (see **Figure 4**).

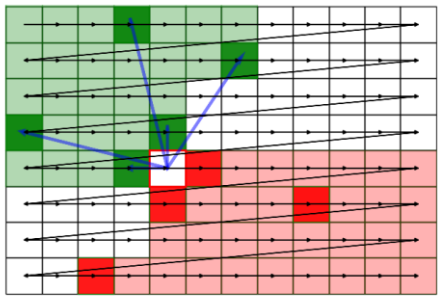

Figure 4. Trees of dependencies for a raster scan order

We propose changing the scanning order to achieve wave-front parallel processing, as proposed in the HEVC standard [4], or a staggered approach as shown in [3]. A wave-front scanning order is depicted in **Figure 5**, together with the dependencies and sets of blocks which can be processed in parallel (highlighted in blue). The set of blue blocks is called a wave-front.

In the traditional method of using wave-front processing, each thread works on one row of blocks. When a block is processed, the thread should be synchronised with a thread that works on an upper row, in order to preserve the dependency (the upper thread needs to stay ahead of the lower thread). This often produces stalls due to the different times needed to process different blocks. Our approach is different; working threads process all blocks belonging to a wave-front independently. Thus, synchronisation that produces a stall is performed only when the processing of the next wave-front starts, and even this stall can be eliminated since the starting point of the next wave-front is usually ready for processing, provided that the number of tasks is at least several times greater than the number of

cores. The proposed approach therefore eliminates the majority of stalls.

The wave-front scanning order changes the resulting motion field, since unlike the meandering scanning order, it uses "left to right" only relative positions US_{cur} and US_{pred} during the estimation of forward motion and "right to left" for backward motion.

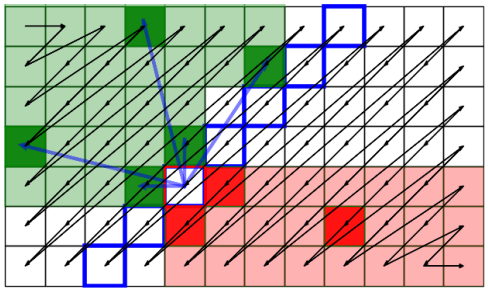

Figure 5. Wave-front scanning order

Slanted wave-front scanning order

The proposed wave-front scanning order has an inconvenient memory access pattern and hence uses the cache of the processor ineffectively. For a meandering scanning order with smooth motion, the memory accesses are serial, and frame data stored in the cache is reused effectively. The main direction of the wave-front scanning order is diagonal, which nullifies the advantage of a long cache line and degrades the reuse of the data in the cache. As a result, the number of memory accesses (cache misses) increases.

To solve this problem, we propose to use several blocks placed in raster order as one task for parallel processing (see **Figure 6**, where the task consists of two blocks). We call this modified order a slanted wave-front scanning order. This solution changes only the scanning order, and not the directions of recursion, so only $rnd(k)$ influences the resulting motion field. If $rnd(k)$ is a function of $\vec{X}$ (the spatial position in the frame) and the number of calls in $\vec{X}$, then the results will be exactly the same as for the initial wave-front scanning order.

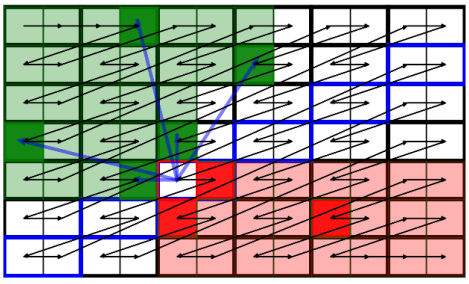

Figure 6. Slanted wave-front scanning order (two blocks in one task)

The quantity of blocks in one task can vary; a greater number is better for the cache, but can limit the number of parallel tasks. Reducing the quantity of tasks limits the maximum number of MCP cores used effectively, but also reduces the overhead for thread management.

Double-block processing

The computational cost for motion estimation can be represented as a sum of the costs for a calculation of the MAD and the cost of the control program code (managing a scanning order, construction of a candidate set, optimisations related to skipping the calculation of the MAD for the same candidates, and so on). During our experiments, we recognised that a significant number of calculations were spent on the control code.

To decrease this overhead, we introduce double-block processing. This means that one processing unit consists of a horizontal pair of neighbouring blocks (called a double block) instead of a single block. The use of double-block processing allows us to reduce almost all control cycles by half. One candidate set is considered for both blocks of this double block. For example, in forward ME, a candidate set $CS(\vec{X})$ from the left block of a pair is also used for the right block of the pair. However, calculation of the MAD is performed individually for each block of the pair, and the best candidate is considered separately for each block of the pair. This point distinguishes double-block processing from a horizontal enlargement of the block.

A slanted wave-front scanning order where one task consists of two double-block units is shown in **Figure 7**. The centre of each double-block unit is marked by a red point, and the left and right blocks of the double-block unit are separated by a red line. The current double-block unit is highlighted by a cyan rectangle. For this unit, the sources are shown for the spatial candidate set (green blocks) and for the temporal candidate set (orange blocks) related to block A. The same sources are used for block B that belong to the same double block as A.

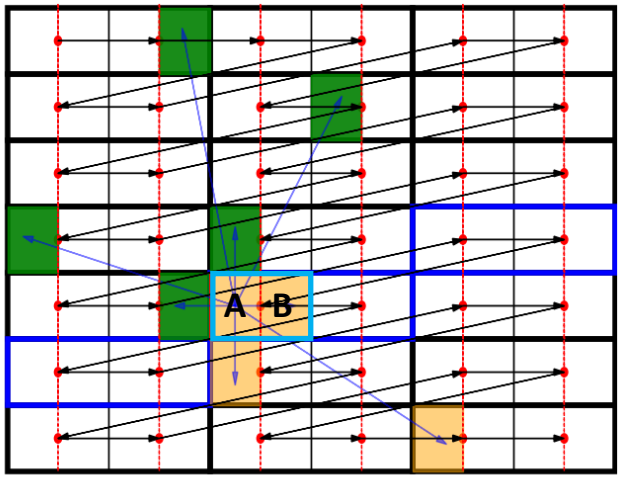

Figure 7. Candidate set for a double-block unit. The current double block consists of a block A and a block B. The candidate set constructed for block A is also used for block B

Thus, a candidate set is not used for block B. However, it may be that a true candidate set for block B is useful when the candidate set of block A gets results for blocks A and B that are too variable (the best MAD values); this is possible on some edges of a moving object or when a non-linear object is used. We therefore propose an additional step for the analysis of MAD values, which are related to the best motion vectors of blocks A and B of the double-block unit. Depending on results of this step, we either accept the previously

estimated motion vectors or carry out a motion estimation procedure for block B.

$\vec{X}_A = \begin{pmatrix} x \\ y \end{pmatrix}$ and $\vec{X}_B = \begin{pmatrix} x+W \\ y \end{pmatrix}$

are the coordinates of the centres of blocks A and B.

$$\vec{D}_{best}\left(\vec{X}, CS(\vec{X}_A)\right) = \underset{\overrightarrow{csD} \in \{CS(\vec{X}_A)\}}{\operatorname{argmin}} \; MAD(\vec{X}, \overrightarrow{csD})$$

$$Ma = MAD\left(\vec{X}_A, \vec{D}_{best}\left(\vec{X}_A, CS(\vec{X}_A)\right)\right)$$

$$Mb = MAD\left(\vec{X}_B, \vec{D}_{best}\left(\vec{X}_B, CS(\vec{X}_A)\right)\right)$$

$$if\ Mb > T1\ or\ (Mb - Ma) > T2\ then$$

{

$$do\ ME\ for\ single\ block\ B$$

}

$T1$ and $T2$ above are threshold values. Reasonable values of the threshold for the usual 8-bit frames are $T1 = 16$ and $T2 = 5$.

Evaluation and Results

The quality of the proposed updates was checked for various FullHD video streams (1920x1080) with the help of the FRC algorithm. To get ground truth, we down-sampled the video streams from 30fps to 15fps and then up-converted them back to 30fps with the help of motion fields obtained by the tested ME algorithms.

The initial version of our 3DRS-based algorithm was described above in the section entitled "Baseline 3DRS-based algorithm". This algorithm was modified with the proposed updates. Input frames for ME were down-sampled twice per coordinate using an 8-tap filter, and the resulting motion vectors had a precision of two pixels. The up-conversion algorithm worked with the initial frames in FullHD resolution, and was based on motion compensated interpolation (MCI). To calculate the MAD luminance component of frame was used for forward ME and backward ME both. Additionally, one chrominance component was used for forward ME and another for backward ME. In backward ME, the wave-front scanning order was also switched to bottom-to-top and right-to-left.

Table 1 presents the results of the quality of the FRC algorithm based on: (a) an initial version of the 3DRS-based algorithm; (b) a version updated using the wave-front scanning order; and (c) a version updated using both the wave-front scanning order and double-block units.

The proposed updates retain the quality of the FRC output, except for a small drop when double block processing is enabled. Here we note again that changing the number of units in the task and the number of threads does not change the results of ME (when $rnd(k)$ is controlled appropriately), and hence the quality of FRC is fully preserved. To check the quality, it is sufficient to measure only these three modifications of ME.

A speed performance of the proposed updates for the 3DRS-based algorithm was evaluated using a Samsung Galaxy S8 mobile phone based on the MSM8998 chipset. The clock frequency was fixed within a narrow range for the stability of the results. Threads performing ME algorithms were assigned to one cluster of cores. The MSM8998 chipset uses a big.LITTLE configuration with 4+4 cores in clusters with different performance.

Table 1: Comparison of initial quality and proposed updates: baseline (A), wave-front scan (B), wave-front + double block (C)

Video stream	A PSNR [dB]	B PSNR [dB]	C PSNR [dB]
Climb	42.81	42.81	42.77
Dvintsev12	27.56	27.55	27.44
Turn2	31.98	31.98	31.97
Bosphor	43.21	43.21	43.20
Jockey	34.44	34.44	34.34
Average:	**36.00**	**36.00**	**35.94**

Table 2 presents the overall results for performance in terms of speed. Column 8 of the table contains the mean execution time for the sum of forward and backward ME for a pair of frames using five test video streams which were also used for quality testing. Experiment E1 shows the parameters and speed of the initial 3DRS-based algorithm as described in the section entitled "Baseline 3DRS-based algorithm". Experiment E2 shows a 19.7% drop (E2 vs. E1) when the diagonal wave-front scanning order is applied instead of the meandering order used in E1. The proposed slanted wave-front used in E3 and E4 (eight and 16 blocks in each task) minimises the drop to 4% (E4 vs. E1). The proposed double-block processing increases the speed by 12.6% (E5 vs. E4) relatively version without using this processing.

Table 2: Comparison of execution times for proposed updates

1) Name of experiment	2) Scanning order	3) Multithread control code used	4) Number of threads	5) Double block used	6) Number of units in a task	7) Number of blocks in a task	8) Mean execution time (ms)
E1	meandering	no	1	no	1	1	24.91
E2	**wave-front**	no	1	no	1	1	29.81
E3	wave-front	no	1	no	**8**	**8**	26.44
E4	wave-front	no	1	no	**16**	**16**	25.90
E5	wave-front	no	1	**yes**	**8**	16	22.63
E6	wave-front	**yes**	1	**no**	**1**	**1**	33.29
E7	wave-front	yes	1	no	**8**	**8**	28.67
E8	wave-front	yes	1	no	**16**	**16**	27.94
E9	wave-front	yes	1	**yes**	**8**	16	24.38
E10	wave-front	yes	**2**	yes	8	16	14.77
E11	wave-front	yes	**4**	yes	8	16	14.14

If we preserve conditions of experiments E9–E11 and only pin ME threads to a small cluster, we obtain different results (see E12–E15 in Table 3). The parallelisation of three threads is closer to the ideal (E14 vs. E12). The attempt to use four threads did not give a good improvement (E15 vs. E14). Our explanation of this fact is that one core is at least partially occupied by OS work.

Table 3: Execution times of ME on small cluster

Name of experiment	Number of threads	Mean execution time (ms)	Comparison with E12
E12	1	55.03	100%
E13	2	27.38	49.7%
E14	3	20.12	36.6%
E15	4	18.23	33.1%

Conclusion

The main contribution of this paper is a set of updates for a 3DRS-based ME algorithm which allow for efficient multithreaded multicore implementations on recent mobile devices. The wave-front scanning order allows multithreaded implementation, and gives a speedup of ~2.7x when three threads are used. Using a slanted wave-front scanning order improves the memory access pattern and almost eliminates the reduction caused by applying the wave-front scanning order instead of raster-scan-based scanning. Double-block processing also gives a contribution to computation speedup of ~12%.

Our updated 3DRS-based algorithm was used in a real-time FRC system performing FullHD content conversion by doubling the frame rate (15 to 30 fps) at an overall power consumption of less than 500 mA on a Samsung Galaxy S8. No updates of SoC hardware (MSM8998) such as the addition of special instructions or a dedicated ASIC accelerator were used.

To improve our FRC system, we plan to improve the ME algorithm in several ways. The first of these is to increase the spatial resolution of the motion field with a low computational overhead. The second is a decrease of algorithm sensitivity to periodic patterns, flat areas and changes in scene brightness. The third is to improve the precision of occlusion detection and handling, which is partly related to ME and partly to the MCI stage of the FRC system.

References

[1] G. deHaan, P. Biezen, H. Huijgen, O.A. Ojo "True-Motion Estimation with 3-D Recursive Search Block Matching," in IEEE Trans. on Circuits and Syst. for Video Technol., vol. 3, no. 5, 1993

[2] G. Al-Kadi, J. Hoogerbrugge, S. Guntur, A. Terechko, M. Duranton, O. Eerenberg "Meandering Based Parallel 3DRS Algorithm for the Multicore Era," in Proc. of the IEEE International Conf. on Consumer Electronics, Las Vegas, NV, USA, 2010

[3] S. Fluegel, H. Klussmann, P. Pirsch, M. Schulz, M. Cisse, W. Gehrke "A Highly Parallel Sub-Pel Accurate Motion Estimator for H.264," in IEEE 8th Workshop on Multimedia Signal Processing, Victoria, BC, Canada, pp. 387-390, 2006.

[4] C. Chi, M. Alvarez-Mesa, B. Juurlink "Parallel Scalability and Efficiency of HEVC Parallelization Approaches," in IEEE Trans. on Circuits and Syst. for Video Technol., vol.22, no. 12, pp. 1827-1838, Dec. 2012.

Author Biographies

Petr Pohl received an MS (2002) in Technical Cybernetics from the Faculty of Electrical Engineering (FEE) of Czech Technical University (CTU). Since then he has worked at Neovision Ltd. (2002–2011) and Samsung R&D Institute Russia (2011–present). His research interests include computer vision and image and video processing. His latest works are related to motion estimation, temporal interpolation and dense tracking.

Valery Anisimovskiy received his BS (1998) and MS (2000) in Applied Mathematics and Physics from the Moscow Institute of Physics and Technology (MIPT). He then worked at the Institute for Nuclear Research of RAS (2000–2004), SPIRIT Corp (2004–2009), Luxoft (2009–2011), and Huawei Russian Research Centre (2011–2014). Since 2014, he has worked at Samsung R&D Institute Russia, Moscow. His research interests include machine learning, pattern recognition, computer vision, video coding and audio coding.

Igor Kovliga received an MS degree in Computer Science from Moscow Institute of Electronic Engineering, Russia (2001) and a PhD in Technical Science from the Institute of Physical Problems named Lukin, Russia (2005). He currently works in the Samsung R&D Institute Russia in Moscow. His research interests include video/image compression algorithms and the optimisation of image/video processing algorithms for target platforms.

Alexey Gruzdev received his MS (2014) in Applied Mathematics and Computer Science from the Lomonosov Moscow State University (MSU). He then worked at Samsung R&D Institute Russia (2013–2017). Since 2017 he has worked at ARM Ltd., UK. His research interests include computer graphics, image and video processing, computer vision and high-performance parallel computing.

Roman Arzumanyan received his BS (2010) and MS (2012) in Applied Mathematics from Southern Federal University, Rostov-on-Don. Since then he has worked at Samsung R&D Institute Russia, Moscow (2012–2015), Intel corp., Moscow (2015–2017). He is currently with Nvidia Corp., Moscow as Developer Technology Engineer. His research interests include video coding, high performance and GPGPU.