Online Efficient Bio-Medical Video Transcoding on MPSoCs Through Content-Aware Workload Allocation

Arman Iranfar*, Ali Pahlevan*, Marina Zapater*, Martin Žagar†, Mario Kovač†, and David Atienza*
*Embedded Systems Laboratory (ESL), Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland
†Faculty of Electrical Engineering and Computing Univeristy of Zagreb, Croatia
*{arman.iranfar, ali.pahlevan, marina.zapater, david.atienza}@epfl.ch
†{martin.zager, mario.kovac}@fer.hr

***Abstract*—Bio-medical image processing in the field of telemedicine, and in particular the definition of systems that allow medical diagnostics in a collaborative and distributed way is experiencing an undeniable growth. Due to the high quality of bio-medical videos and the subsequent large volumes of data generated, to enable medical diagnosis on-the-go it is imperative to efficiently transcode and stream the stored videos on real time, without quality loss. However, online video transcoding is a high-demanding computationally-intensive task and its efficient management in Multiprocessor Systems-on-Chip (MPSoCs) poses an important challenge. In this work, we propose an efficient motion- and texture-aware frame-level parallelization approach to enable online medical imaging transcoding on MPSoCs for next generation video encoders. By exploiting the unique characteristics of bio-medical videos and the medical procedure that enable diagnosis, we split frames into tiles based on their motion and texture, deciding the most adequate level of parallelization. Then, we employ the available encoding parameters to satisfy the required video quality and compression. Moreover, we propose a new fast motion search algorithm for bio-medical videos that allows to drastically reduce the computational complexity of the encoder, thus achieving the frame rates required for online transcoding. Finally, we heuristically allocate the threads to the most appropriate available resources and set the operating frequency of each one. We evaluate our work on an enterprise multicore server achieving online medical imaging with 1.6x higher throughput and 44% less power consumption when compared to the state-of-the-art techniques.**

I. Introduction

Changes in world demographics together with the paradigm shift towards prevention in healthcare delivery and the improvements in medical imaging technology are leading to a growing demand for multimedia applications for medical diagnosis as a basis for computer (eHealth) [1] and mobile healthcare (mHealth). By the end of 2017, 3.4 billion people worldwide will own a smartphone and half of them will be using mHealth apps [2]. In 2017, if its potential had been fully unlocked, mHealth could have saved €99 billion in healthcare costs in the EU [2]. One of the overarching objectives of the third EU health program is to contribute to innovative, efficient and sustainable health systems where medical imaging can enable a cross-border system for allowing medical diagnostics to be performed in a collaborative, distributed and mobile environment [3].

To enable bio-medical diagnosis on-the-go, it is necessary to centralize the storage of bio-medical videos in highest quality possible, and send/share them online while considering the vast amount of data they represent and the strict quality needed for diagnostic procedures. In fact, exchanged data in form of studies consisting of different multimedia content should be available to users regardless of their current location by providing them with access to data from their mobile devices.

This work has been partially supported by the EC H2020 MANGO project (GA No. 671668), the ERC Consolidator Grant COMPUSAPIEN (GA No. 725657), and EuroLab-4-HPC project (GA. No. 671610).

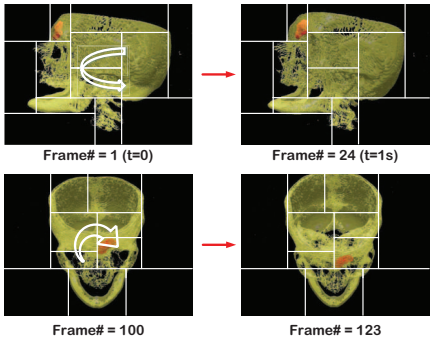


Fig. 1. A medical image frame sample, motion, and tiling

Enabling online bio-medical video transcoding (i.e., the real-time decoding and encoding of videos from the format in which they are stored to the format of the demanded by the mobile device) is the preliminary requirement of eHealth realization. This is, however, challenging due to the complexity of High Efficiency Video Coding (HEVC) [4], as the most promising standard that achieves the same video quality as its predecessors while doubling the video compression. In transcoding, the encoder is approximately 100 times more complex [5]. The main complexity comes from inter-prediction, and in particular motion estimation, which is vital for high compression and online transcoding.

Bio-medical videos have unique features, which make online encoding feasible, if taken into account. Fig. 1 shows 4 bio-medical images extracted from a sample video. As shown, the useful information concentrates on the center of the image. Hence, partitioning a frame (i.e., tiling) enables content-based parallel processing, and per-tile tuning of the encoding parameters to meet its specific requirements. Moreover, once tiling is done, the encoder can rely on the given structure for several next coming frames. This is because, in general, specialists want to rotate the videos along a certain axis to better observe an area of interest. In Fig. 1, this is shown in the two upper images and two lower ones where even after 24 frames (i.e., 1 second) the initial tiling is still valid. Moreover, the whole frame moves in the same direction (to the right for

the images at the top, and downward for those at the bottom).

In this work, we propose for the first time in literate, a methodology to exploit the above mentioned unique features of bio-medical videos, enabling online efficient bio-medical imaging on Multiprocessor Systems-on-Chip (MPSoCs). In particular, our main contributions are as follows:

- we propose a content-aware tiling strategy that tunes encoding complexity for efficient frame parallelization,
- we propose a new fast motion estimation algorithm that suits well for bio-medical videos resulting in 4x speedup,
- we present a high-throughput thread allocation policy that increases the number of users served for real-time eHealth and telemedicine applications achieving 1.6x higher throughput and 44% less power consumption without any compression degradation and video quality loss.

II. Related Work

A. Multimedia Transcoding

A recent framework for mHealth [6], enables a wide range of medical software and integrates video transcoding. However, it is not oriented towards efficient online coding. The authors in [7] present a patent that describes a new system for cost-efficient use of imaging devices by hospitals and other service providers. Based on the fact that it is not necessary to encode all multimedia data, [8] presents a Big Data based multimedia transcoding method for telemedicine. However, none of these works present a content-aware solution.

A multimedia slice transcoding that removes the boundary fuzzy area is proposed in [9], while a robust dynamic resource provisioning scheme for transcoding with heterogeneous Quality of Service (QoS) criteria is proposed in [10]. However, none of them provides a complete and comprehensive solution that considers both video contents and diagnostic procedures.

B. Motion Estimation

Research about new motion search algorithms is quite rich in the literature [11], [12], [13], [14], [15]. Although the classical full search algorithm [4] provides the best motion estimation, it is not applicable for real-time and online applications due to its intolerable runtime overhead. Many other motion search algorithms with reduced computational complexity at the cost of less encoding efficiency (video quality in Peak Signal-to-Noise Ratio (PSNR), and compression in bitrate) have been proposed such as three step search [11], diamond search [12], cross search [13] one-at-a-time search [14], and hexagonal-based search [15]. However, none of them are application-specific. Hence, they are not efficient for bio-medical video transcoding.

C. Workload Parallelization for Multimedia Applications

Several works have targeted the parallelization of transcoding for multimedia applications. Indeed, video frames can be clustered as group of pictures (GOPs) and can be independently processed providing workload parallelization (e.g., [16]). At frame-level, workload parallelization is enabled through two different schemes by HEVC standard: Wavefront Parallel Processing (WPP) [17] and tiling. While wavefront dependencies prevent all partitions from being processed concurrently, tiles can be regarded as independent threads providing more parallelization and are leveraged by a few works [18], [19]. However, none of these works are specifically targeting bio-medical applications.

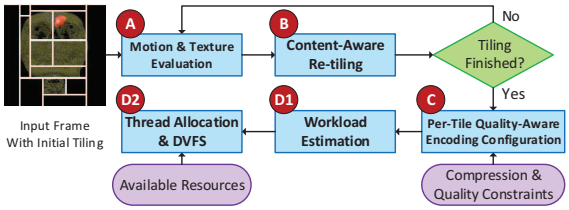


Fig. 2. Our proposed framework

III. Proposed Framework

In this work, we address efficient bio-medical video encoding where several users request videos that need to be online transcoded in a limited number of resources (i.e., available cores on a MPSoC). Although significant performance and power improvements can be reached by introducing hardware acceleration for parts of the algorithm (e.g., motion estimation), our proposed approach, regardless of the architecture or application-level optimization, is able to achieve online bio-medical video transcoding with maximized throughput and power savings. In this context, once the request of a user for online transcoding is admitted, the required framerate as well as the video quality and compression must be also satisfied.

Fig. 2 illustrates the proposed approach for enabling online bio-medical imaging under constraints of framerate, PSNR, and bitrate. The goal is to serve a number of users (i.e., doctors), each transcoding one video. For each frame in a video the motion and texture are evaluated (Subsec. III-A) for every initial tile (from the last processed frame, or simply a uniform tiling if no frame has been processed yet). Based on the knowledge of the motion and texture of different areas of the frame, we perform the re-tiling of the frame (Subsec. III-B). Once all tiles are specified, appropriate encoding parameters are set separately for each tile based on its texture and motion as well as PSNR and bitrate (Subsec. III-C). The predefined minimum tile size and the maximum number of tiles within a frame ensure fast ending of this phase. The workload corresponding to each tile and its encoding parameters is estimated via a look-up table (LUT) which is dynamically updated throughout the encoding process (Subsec. III-D1). Finally, based on the estimated workload and availability of the resources (i.e., available processors and operating frequencies of the platform) we allocate each tile to an available resource and set the operating frequency to enable real-time energy efficient video encoding (Subsec. III-D2).

A. Motion and Texture Evaluation

The diversity in luma samples (i.e., achromatic portion of the image) in a tile as well as the amount of motion play a major role in the encoding time. Therefore, efficiently tiling

(i.e., splitting a frame in tiles) based on its contents helps achieving more efficient thread allocation and scheduling. Moreover, other encoding configuration knobs such as quantization parameter (QP), which influence encoding efficiency and performance the most, can be tuned more effectively to save power once the texture and motion of the tiles are known.

The first step for efficient tiling is to quantify the motion and texture of different partitions of the frame. Hence, starting from the latest tiling, we can evaluate the diversity of the texture and the presence of motion. Since this evaluation must be fast enough to avoid any computational overhead, we use the coefficient of variation (CV), which is defined as the ratio of the standard deviation to the mean. Then, we classify the tile based on a predefined threshold of the texture, as follows:

$$T = \begin{cases} low & CV \leq T_{th,l} \\ medium & T_{th,l} < CV \leq T_{th,h} \\ high & CV > T_{th,h} \end{cases} \quad (1)$$

In order to obtain a low-overhead measure of the motion in a tile, we propose a pixel-to-pixel comparison of a limited number of pixels including four corners, the center, and the one with the maximum value, as follows:

$$M = \alpha \sum_{i=1}^{4} x_i + \beta c + \gamma m \quad (2)$$

where x_i, c, and m are booleans for pixel comparisons respectively at the 4 corners, center and maximum point. When pixels are equal, booleans are zero. In this formulation, α, β, and γ denote the importance of the comparison for different coordinates of the tile. Medical images require larger coefficients for the center and the maximum point. Hence we choose 1, 3, and 3 for α, β, and γ, respectively. Finally, we define the motion threshold ($M_{th} = 3$) based on which a tile is regarded as high- or low-motion as follows:

$$Motion = \begin{cases} low & M < M_{th} \\ high & M \geq M_{th} \end{cases} \quad (3)$$

In bio-medical imaging, those parts of the frame that contain useful data move or rotate in the same direction. This fact implies that only evaluating one initial tile for the motion can be sufficient to quantify the motion of all remaining tiles. This also explains why, for all real-life case studies considered, we can only take into account two levels for the motion.

B. Content-Aware Re-tiling

Based on the existing texture and motion in different parts of a frame, we split it into different number of tiles. Generally, most of the frames have the least amount of motion and texture in the corners and boarders. This is especially true for medical videos. Therefore, we start re-tiling from the initial tiles in the corners. If the motion and texture of the tile is *low* (cf. Section III-C), we increase the tile size by 25% more pixels first in the width and then in the height. This value was experimentally found and represents a trade-off between optimal tile size and the time it takes to find it. This procedure continues until the texture or the motion is not low anymore, and we keep the latest tile's coordinates.

After having finished the same procedure for all the corners and from boarders, we start seeking for the remaining tiles on the center, which more likely contain high motion and high texture. Since as described in Section III-A the motion in the center of the medical images is consistent, we only consider the texture for re-tiling. Here, the size of a tile plays a major role for encoding time. Therefore, we first extract a tile with the minimum size allowed and then re-partition the rest of the pixels such that all tiles are of similar size.

Finally, we limit the minimum number of tiles used for the high-texture and high-motion area of the frame to 4 based on our observations on actual bio-medical videos to keep the parallelization as high as possible, while achieving the desirable encoding efficiency.

C. Per-Tile Quality-Aware Encoding Configuration

1) Quantization Parameter (QP): QP plays a major role in encoding efficiency and performance [20]. Thus, selecting the most appropriate QP can result in lower computational complexity and yet efficient encoding. In other words, while smaller QPs are necessary for high-texture tiles, larger QPs can satisfy the required video quality and compression for the low texture tiles.

Therefore, we utilize QP equal to 37, 32, and 27 for the low, medium, and high texture tiles, respectively, as default values. However, we keep evaluating the outcome video quality (in PSNR) and compression (in bitrate) and consider two other extreme QP values (42 and 22). We observe that for very low-texture tiles $QP = 42$ can be used to further reduce encoding time and bitrate without PSNR degradation. Also, for extreme cases of high-texture tiles $QP = 22$ should be used to meet the PSNR constraint. Starting from the default QP values for different textures, we update the QP selection based on the measures of the corresponding tile of the previous frame. Algorithm 1 shows the pseudo code of per-tile quality-aware QP selection where $BR_{t-\Delta t}$, and $PSNR_{t-\Delta t}$ are the bitrate and PSNR of the tile of previous frame, $PSNR_{const}$, and $PSNR_{margin}$ show the constraint of video quality and the margin by which we can guarantee that further increasing of QP value would not result in dissatisfaction of PSNR constraint. Finally, $\boldsymbol{M}$ and $\boldsymbol{T}$ are arrays containing the motion and texture of the tiles in the frame, respectively.

2) Motion Estimation Search Window Algorithm: In bio-medical imaging the same motion can be applied to all the tiles. For instance, in Fig. 1 it is shown that all the tiles

Algorithm 1. Quantization parameter adaptation

Input: $\boldsymbol{BR_{t-\Delta t}}$, $\boldsymbol{PSNR_{t-\Delta t}}$, $\boldsymbol{M}$, $\boldsymbol{T}$
Output: $\boldsymbol{QP}$

```
for each tile with T ∈ T and M ∈ M
    if PSNR_{t−Δt} > PSNR_const + PSNR_margin
        QP ← QP + ΔQP
    else if PSNR_{t−Δt} < PSNR_const
        QP ← QP − ΔQP
    else
        Use default QPs w.r.t T ∈ T and M ∈ M
    end if
end for
```

move downward or to the right. Moreover, although the motion vectors inside different tiles may have different angles, the motion can still be considered in the same direction to start the motion prediction and estimation.

Thanks to the unique features of bio-medical images, we are able to apply efficient motion search algorithms with less computational overhead. In particular, when the motion is known to be *low*, simpler algorithms such as one-at-a-time search [14] or cross-search [13] can be applied. In particular, we leverage the cross-search algorithm for the low-motion tiles of the first frame in a GOP. Then, we use the one-at-a-time search algorithm for the remaining frames in the GOP in the direction of the motion vector obtained from the corresponding tiles of the first frame. This combined search approach further reduces the motion estimation time without encoding efficiency degradation. In this work, we consider search windows of size 64x64, 32x32, 16x16, and 8x8. However, for low-motion tiles, a search window size of 16x16 is sufficient for the first frame in the GOP and it can be further decreased to 8x8 for the next frames to reduce the computational complexity of the motion estimation.

On the other hand, for high-motion tiles, more accurate search algorithms, such as, the Test Zone (TZ) search used in the HEVC reference software [21], Diamond Search (DS) [12], and Hexagonal-based search [15] can be used. In particular, we rely on the Hexagonal-based search algorithm, which can be applied in two different ways: vertical and horizontal [15]. While both have the same complexity, the former outperforms the latter when the motion is more horizontal. Since at the beginning of a GOP the direction of the motion is not determined yet, we use the rotating Hexagonal-based search algorithm [15] for the first frame of the GOP. From the second frame until the last frame of a GOP, however, we either use the horizontal or vertical Hexagonal-based search algorithms based on the decision taken for the corresponding tile of the first frame. Moreover, for each high-motion tile of the first frame the maximum allowable search window is considered, while from the second frame smaller values are used to reduce the computational complexity.

D. Workload Estimation, Thread Allocation and Dynamic Voltage and Frequency Scaling (DVFS)

1) Workload Estimation: In order to perform the best thread allocation and DVFS to maximize the number of users that can be sustained by the target multicore server, while meeting the required framerate and encoding efficiency, we use an LUT-based approach. In fact, the LUT-based approach is very suitable for this context due to the nature of the proposed re-tiling approach, which includes a limited number of different attainable tile structures and numbers within a frame. Moreover, the number of different combinations of the encoding configurations are limited.

In our proposed framework, we store the histogram of the CPU time in the LUT and keep updating it throughout the whole video encoding. We use the stored histograms to estimate the workload for robust thread allocation and DVFS.

Another important feature of medical imaging application that makes robust workload estimation possible is the fact that medical images are classifiable in very limited categories based on part of the body that is under the study (such as bones, lung and chest, brain, spinal cord, ligament and tendon, etc). This feature allows us to use the obtained LUT of one MRI or CT data to the rest of images in the same class. As a result, CPU time over/under-estimation below 100 μs is observed when enough frames have been processed.

2) Thread Allocation and DVFS: In this work, we consider a Random Access (RA) encoding configuration as the baseline where B slices allow both intra- and inter-picture predictions for high PSNR and low bitrate (both of them are requisites for online bio-medical imaging). We consider GOP of size 8. We apply the re-tiling and thread allocation strategies once for a GOP. However, we keep dynamically adjusting the QP for every frame.

The thread (tile) allocation is performed once at the beginning of each GOP since re-tiling and setting the encoder parameters are done for the first frame (except for QP under the conditions discussed in Section III-C1). However, the resulted encoding time of the performed allocation is readout once a frame is released and, if it does not equal 1/FPS seconds, an alternative (and less) complex encoding configuration is applied to the next frame (only if the operating frequency is maximum). This alternative encoding configuration includes using a smaller search window and higher QP for the tiles recognized as the bottleneck of achieving the 1/FPS seconds for the previous frame. According to this strategy, the over-utilization of some of the threads (if any) can be compensated by under-utilization of the next frame(s), such that the required framerate (checked every second) can ultimately be satisfied.

As a result, in our allocation strategy, as described in Algorithm 2, we first determine the minimum number of cores

Algorithm 1. Thread allocation and DVFS

Input: $N_u, N_c, \boldsymbol{N_{thr}} = \{\boldsymbol{N^1_{thr}}, \boldsymbol{N^2_{thr}}, \dots \boldsymbol{N^{N_u}_{thr}}\}, \boldsymbol{T^i_f} = \left\{\boldsymbol{T^i_{f,1}}, \boldsymbol{T^i_{f,2}}, \dots \boldsymbol{T^i_{f,N^i_{thr}}}\right\}, \boldsymbol{F} = \{\boldsymbol{f_1}, \boldsymbol{f_2}, \dots \boldsymbol{f_{max}}\}, FPS$

Output: Serving maximum number of users and their thread allocation

$N^i_{core} \leftarrow \left(\sum_{j=1}^{N^i_{thr}} T^i_{f_{max},j}\right).FPS \qquad i \in \{1, 2, \dots N_u\}$

$U \leftarrow$ *Find the maximum u users w.r.t* $\sum_{k=1}^{u} \boldsymbol{Sort}^{ASC}_k(N^k_{core}) \leq N_c$

for $j = 1 : N^U_{thr}$ // Thread Allocation

  for $k = 1 : N^U_{core}$

    if $max_k(Load_k) > \frac{1}{FPS}$

      $Cap \leftarrow 1/FPS$

    else

      $Cap \leftarrow max_k(Load_k)$

    end if

    $Dist^j_k \leftarrow \left|Cap - \left(Load_k + T^{u_j}_{f_{max},j}\right)\right|$

  end for

  $ID_c \leftarrow$ *Find Core with* $min_k(Dist^j)$

  Allocate thread j *to core* ID_c

  $Load_{ID_c} \leftarrow Load_{ID_c} + T^{u_j}_{f_{max},j}$

end for

for $k = 1 : N_c$ // DVFS for Energy Efficiency

  if $Load_k \leq 1/FPS$

    Set $min(F)$ *to core* k *for slack time* $(\frac{1}{FPS} - Load_k)$

    $Load_k \leftarrow 0$

  else

    Set f_{max} *for the whole* $\frac{1}{FPS}$

    $Load_k \leftarrow Load_k - \frac{1}{FPS}$

  end if

end for

needed for each user (N^i_{core}) based on its threads CPU time (line 1) obtained from maximum frequency level ($T^i_{f_{max}}$) at each time slot (i.e., $1/FPS$). N_u and N^i_{thr} show the number of users and the number of threads (tiles) for the i^{th} user, respectively. According to this period, we can guarantee that after 1 second, the required frames are retired. To maximize the number of users (line 2), we select users with the lower N^i_{core} until we reach the total number of cores (N_c). Then, we try to allocate all the threads of the selected users (U) to the cores, while the cores CPU time used by the threads does not exceed the time slot period ($1/FPS$).

In this case, we first select one thread from the pool of selected users' threads and try to find the best core for it (lines 3-15). N^U_{thr} denotes the total number of threads available for U. Then, we determine a dynamic cap (Cap) finding the maximum load over all cores (i.e, CPU time already allocated to threads, which cannot be above $1/FPS$; lines 5-9). Based on this cap, we compute the euclidean distance (line 10) between the cap and the load of the k^{th} core ($Load_k$) with the consideration of allocating the thread to this core ($Dist^j_k$). Finally, we allocate the j^{th} thread to the core whose CPU time nearly reaches the cap (i.e., $min(Dist^j)$), and update its load (lines 12-14). This shows that this thread can make the CPU time of the target core more utilized than the other cores.

After allocation, for energy efficiency (lines 16-24), we check the load of each core whether it has the slack time (idle time). If it does, we set the minimum frequency ($min(F)$) to the core for the rest of the time. Otherwise, we keep the maximum frequency and shift the remaining load (CPU time) to the next interval.

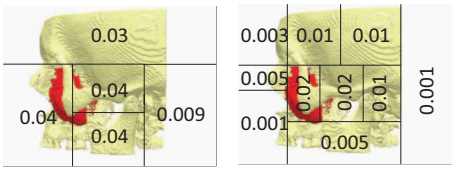
IV. Experimental Setup and Results

A. Experimental Setup

In this work, we perform the experiments on an Intel Xeon server running CentOS 7. The server includes four 8-core E5-2667 processors [22]. Each physical core contains two threads, and we consider 3.6GHz, 3.2GHz, and 2.9GHz as the available operating frequencies supported by the platform that enable reaching the framerate required for online transcoding. The maximum transition latency of the DVFS is 10.0μs, which is fast enough for our purpose. Each processor comes with 32KB instruction and data L1, 256KB private L2, and a 25.6MB shared L3. We implement our framework as well as the closest (and most competitive one) to us in the literature [19] on top of the Kvazaar [23], an open-source implementation of the HEVC encoder. We use 10 different anonymized bio-medical videos provided by our medical partners that represent a wide set typical videos used in diagnostic procedures, and all include a resolution of 640×480 and a frame rate of 24Hz (i.e., $FPS = 24$). However, our proposed methodology is valid for any arbitrary resolution and frame rate.

B. Experimental Results

1) Motion Estimation: Table I shows the average speedup, PSNR loss, and compression reduction resulted from simple hexagonal search (the default in the Kvazaar encoder) and our proposed method, for a 400-frame medical video compared to TZ search. In each column, $n \times m$ denotes uniform tiling where the frame width and height are divided by n and m, respectively. Our Results show how we improve the speed of the Hexagonal [15] algorithm (as explained in Sec. III-C2)., without any compression and PSNR loss. We use this speedup to benefit from slack time in the thread allocation and DVFS.



(a) Work of [19] (b) Our proposed approach

Fig. 3. Tile structure and the required CPU time (s) for each tile

2) Tiling, thread allocation, and DVFS: Fig. 3 shows the tiling structure of a frame and the CPU time needed to process each tile obtained from our proposed approach and that of [19], one of the latest approaches in literature. In [19], knowing the total capacity of each core, a limited number of predefined tile sizes and encoding configurations are created based on the capacity of each core, so that the workload of each one can completely utilize a core's capacity. Therefore, only one tile is assigned to each core. However, the existing motion and texture of each tile make a considerable difference in the processing time. Moreover, the re-tiling approach considered in the related work is only performed once the frequency of all cores is set to the minimum or maximum value. Therefore, the approach proposed in [19] cannot properly exploit the dynamic spatial and temporal changes in the contents of a bio-medical video. Indeed, considering the required framerate of 24Hz, the obtained tile structure based on the allocation strategy of [19] requires the use of 5 cores, where these cores should run with the maximum frequency. In contrast, our content-aware re-tiling strategy, including the use of the proposed fast motion search algorithm and thread allocation policy, results in utilizing 3 cores, where only two of them operate at the maximum frequency.

Then, TABLE II shows the average PSNR, bitrate, and the number of users served by our proposed approach and [19] when the number of users and their resource demands are much larger than the available resources (i.e., the queue of users is always full). Therefore, our goal here is to serve as many users as possible. As this table shows, the workload balancing strategy of [19] ensures that the total number of tiles (threads) are less than the available cores. In contrast, our tiling allows more tiles than the available cores with more diverse estimated CPU time. This diversity in execution time is due to different existing texture and motion inside a frame. In addition, the proposed allocation strategy exploits the size and applied encoding parameters to each particular frame in order to use the available resources more efficiently.

Moreover, thanks to our fast motion estimation algorithm, which on average resulted in **4x speedup**, our allocation strategy benefits from more slack time to either serve more users

TABLE I
SPEEDUP, PSNR LOSS, AND BITRATE DEGRADATION FOR THE PROPOSED MOTION ESTIMATION APPROACH AND HEXAGONAL-BASED ALGORITHM COMPARED TO TZ SEARCH FOR DIFFERENT NUMBER OF TILES FOR *uniform* TILING

		1X1	2X1	2X2	2X3	2X4	5X2	4X3	5X3	5X4	4X6	5X6
Proposed	*Speedup (x)*	1.3	2.0	2.5	4.1	4.4	5.1	4.5	5.6	4.8	4.9	5.2
	PSNR loss (dB)	0.00	0.00	0.00	0.01	0.06	0.18	0.31	0.17	0.12	0.31	0.07
	Compression loss (%)	0.0	0.1	0.2	0.2	0.3	0.2	0.3	0.2	0.3	0.4	0.4
Hexagonal [15]	*Speedup (x)*	1.2	1.9	2.3	3.5	4.0	4.4	3.9	4.5	4.0	4.0	4.2
	PSNR loss (dB)	0.00	0.00	0.00	0.02	0.06	0.18	0.32	0.17	0.12	0.32	0.07
	Compression loss (%)	0.0	0.2	0.2	0.2	0.4	0.2	0.3	0.2	0.3	0.5	0.4

TABLE II
PSNR, BITRATE, AND NUMBER OF SERVED USERS

		PSNR (dB)	Bitrate (Mbps)	# of Users
Proposed	*Max*	46.5	2.45	26
	Min	39.9	2.10	20
	Avg	40.5	2.23	23
Work [19]	*Max*	46.5	2.46	16
	Min	39.7	2.11	12
	Avg	40.6	2.23	15

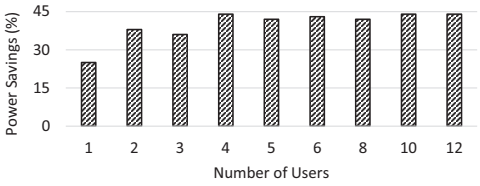


Fig. 4. Average power savings obtained from the proposed approach compared to [19] for different numbers of users

or further reduce power consumption. As TABLE II indicates, although both [19] and our approach used up the available resources to meet the required framerate, the new approach served more users without encoding efficiency degradation.

Finally, in Fig. 4 we further illustrate the achievable power savings of our new approach for bio-medical video transcoding for different numbers of users. As shown, our proposed approach can increase the total power savings by 44% compared to [19]. Although this figure shows the power savings when the same throughput is achieved by both approaches, our proposed approach also saves power from 40% to 7% even when the number of users is larger than 16, which is the maximum number of users served by [19].

V. CONCLUSION

In this work, we proposed a new approach for efficient content-aware tile-based parallelization of frames within bio-medical videos to enable online medical imaging and telemedicine. In particular, we capitalize on the different encoding configuration parameters and apply the most appropriate ones for each tile inside a frame based on its texture and the existing motion. Moreover, in our approach we include a new motion search algorithm specific to bio-medical image features, which attains a reduced computational complexity of motion estimation without encoding efficiency degradation. Overall, the results with real-life bio-medical videos indicate that our approach increased the number of users served for online bio-medical video encoding by 1.6 times and reached 44% more power reduction on average for the same number of users than latest state-of-the-art techniques.

REFERENCES

[1] M. Kovac, "E-health demystified: An e-government showcase," *Computer*, vol. 47, no. 10, pp. 34–42, 2014.

[2] "Healthcare in your pocket: unlocking the potential of mhealth," in *European Commission, DG Communications Networks, Content & Technology*. European Commission, 2017.

[3] "Third health programme (2014-2020)," in *European Commission, Directorate-General for Health and Food Safety*. European Commission, 2017.

[4] J. V. Team, "Advanced video coding for generic audiovisual services," *ITU-T Rec. H*, vol. 264, pp. 14 496–10, 2003.

[5] F. Bossen *et al.*, "HEVC complexity and implementation analysis," *IEEE TCSVT*, vol. 22, no. 12, pp. 1685–1696, 2012.

[6] L. Constantinescu *et al.*, "Sparkmed: a framework for dynamic integration of multimedia medical data into distributed m-health systems," *IEEE TITB*, vol. 16, no. 1, pp. 40–52, 2012.

[7] M. Poon *et al.*, "Method and system for transfer of cardiac medical image data files," Oct. 25 2016, US Patent 9,474,500.

[8] D. Zhu, "Big data-based multimedia transcoding method and its application in multimedia data mining-based smart transportation and telemedicine," *Multimedia Tools and Applications*, vol. 75, no. 24, pp. 17 647–17 668, Dec 2016.

[9] ——, "Overlapping boundary based multimedia slice transcoding method and its system for medical video and traffic video," *Multimedia Tools and Applications*, vol. 75, no. 22, pp. 14 233–14 246, 2016.

[10] G. Gao *et al.*, "Dynamic resource provisioning with qos guarantee for video transcoding in online video sharing service," in *ACM Multimedia Conference*, 2016, pp. 868–877.

[11] R. Li *et al.*, "A new three-step search algorithm for block motion estimation," *IEEE TCSVT*, vol. 4, no. 4, pp. 438–442, 1994.

[12] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block matching motion estimation," in *Information, Communications and Signal Processing*, vol. 1, 1997, pp. 292–296.

[13] M. Ghanbari, "The cross-search algorithm for motion estimation (image coding)," *IEEE TCOM*, vol. 38, no. 7, pp. 950–953, 1990.

[14] R. Srinivasan and K. Rao, "Predictive coding based on efficient motion estimation," *IEEE TCOM*, vol. 33, no. 8, pp. 888–896, 1985.

[15] C. Zhu *et al.*, "Hexagon-based search pattern for fast block motion estimation," *IEEE TCSVT*, vol. 12, no. 5, pp. 349–355, 2002.

[16] S. Sankaraiah *et al.*, "Performance optimization of video coding process on multi-core platform using gop level parallelism," *International Journal of Parallel Programming*, vol. 42, no. 6, pp. 931–947, 2014.

[17] F. Henry and S. Pateux, "Wavefront parallel processing," *Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-E196, Geneva*, 2011.

[18] M. Shafique *et al.*, "Power efficient and workload balanced tiling for parallelized high efficiency video coding," in *IEEE Int. Conf. on ICIP*, 2014, pp. 1253–1257.

[19] M. U. K. Khan *et al.*, "Power-efficient workload balancing for video applications," *IEEE TVLSI*, vol. 24, no. 6, pp. 2089–2102, 2016.

[20] A. Iranfar *et al.*, "Work-in-progress: a machine learning-based approach for power and thermal management of next-generation video coding on mpsocs," in *CODES+ISSS*, 2017, pp. 1–2.

[21] P. Bordes *et al.*, "Joint collaborative team on video coding (JCT-VC) of itu-t sg 16 wp 3 and iso/iec jtc 1/sc 29/wg 11," 2016. [Online]. Available: https://HEVC.hhi.fraunhofer.de

[22] Intel xeon processor e5-2667. [Online]. Available: http://ark.intel.com/products/92979/Intel-Xeon-Processor-E5-2667-v4-25M-Cache-3_20-GHz

[23] M. Viitanen *et al.*, "Kvazaar: Open-source HEVC/h. 265 encoder," in *ACM Multimedia Conference*, 2016, pp. 1179–1182.