Computers, Materials & Continua CMC, vol.61, no.2, pp.439-463, 2019

Efficient Computation Offloading in Mobile Cloud Computing for Video Streaming Over 5G

Bokyun Jo[1], Md. Jalil Piran[2, *], Daeho Lee[3] and Doug Young Suh[4, *]

Abstract: In this paper, we investigate video quality enhancement using computation offloading to the mobile cloud computing (MCC) environment. Our objective is to reduce the computational complexity required to covert a low-resolution video to high-resolution video while minimizing computation at the mobile client and additional communication costs. To do so, we propose an energy-efficient computation offloading framework for video streaming services in a MCC over the fifth generation (5G) cellular networks. In the proposed framework, the mobile client offloads the computational burden for the video enhancement to the cloud, which renders the side information needed to enhance video without requiring much computation by the client. The cloud detects edges from the upsampled ultra-high-resolution video (UHD) and then compresses and transmits them as side information with the original low-resolution video (e.g., full HD). Finally, the mobile client decodes the received content and integrates the SI and original content, which produces a high-quality video. In our extensive simulation experiments, we observed that the amount of computation needed to construct a UHD video in the client is 50%-60% lower than that required to decode UHD video compressed by legacy video encoding algorithms. Moreover, the bandwidth required to transmit a full HD video and its side information is around 70% lower than that required for a normal UHD video. The subjective quality of the enhanced UHD is similar to that of the original UHD video even though the client pays lower communication costs with reduced computing power.

Keywords: 5G, video streaming, cloud, computation offloading, energy efficiency, upsampling, MOS.

1 Introduction

Immense demand for various video streaming services over wireless networks is exploding, especially for emerging, new bandwidth-hungry and delay-sensitive multimedia services. According to a report by CISCO, the global mobile data traffic is expected to reach to 77

[1] DEE, Kyung Hee University, Yongin, 17104, South Korea.

[2] DCSE, Sejong University, Seoul, 05006, South Korea.

[3] DSC, Kyung Hee University, Yongin, 17104, South Korea.

[4] DEE, Kyung Hee University, Yongin, 17104, South Korea.

[*] Corresponding Authors: Doug Young Suh. Email: suh@khu.ac.kr;
Md. Jalil Piran. Email: piran@sejong.ac.kr.

exabytes per month by 2022, at which point nearly four-fifths of all mobile traffic will be composed of video services [CISCO (2019)]. One reason for the exploding demand is that modern mobile devices such as smartphones and tablets are becoming more powerful and affordable. Such devices are being used for a multitude of activities, including online multimedia applications such as Internet protocol television (IPTV), video on demand (VoD), video conferencing, multimedia-based learning, and game streaming and so on. Multimedia applications require significant network bandwidth, computation, and power to stream, decode, and display the content [Yadav, Zhang, Kaiwartya et al. (2018)].

These tasks have a deep correlation with one another. For example, the computation required for decoding, displaying, and quality enhancement has a trade-off relationship with network bandwidth and power consumption. Therefore, the current trend is to provide seamless streaming services by adjusting those parameters according to network conditions using adaptive video streaming technology. In addition, research is underway to minimize the effects of time-varying parameters by developing hardware technology and continuously developing intelligent software [Eswara, Manasa, Kommineni et al. (2018)]. Although 5G technologies will increase capacity and enable new video codecs such as high-efficiency video coding (HEVC) and provide higher video compression efficiency, network providers will continue to struggle with congestion [Concolato, Feuvre, Denoual et al. (2018); Habiba and Hossain (2018); Piran, Islam and Suh (2018); Jalil, Tran, Doug et al. (2017)].

In addition to the network and video codec fields described above, the emerging multimedia services such as panorama, 360°, virtual reality (VR), and augmented reality (AR) have huge volume requirements. Therefore, for any purpose of processing or communications, those applications demand powerful processors and very high-capacity network links, i.e., high bandwidth. It is clear that traditional technologies cannot support those bandwidth-hungry and delay-sensitive applications [Zink, Sitaraman and Nahrstedt (2019); Fischbach, Wiebusch and Latoschik (2017); Yun, Jalil and Suh (2018)].

Although quad-HD (QHD) and ultra-HD (UHD) screen resolutions are increasingly used in mobile devices, previous research has shown that UHD 2160p video resolution has little benefit in terms of user-perceived video quality over full-HD 1080p video resolution. Considering the quality of video that human can actually perceives, it is possible to provide a high-quality video service using less information than would be needed to directly transmit a high-quality video generated with a large computation and network burden [Cheon and Lee (2018); Ghadiyaram, Pan and Bovik (2019)].

1.1 Problem statement

In the current video streaming frameworks that offer high-quality video to clients, a server first encodes a high-resolution (HR) video and then transmits it to a client. The client then decodes the received video bit stream to get the service of HR video. This process produces a large amount of network traffic [Fan, Yin, Min et al. (2017)]. As a solution, the upsampling algorithm has been used to improve the delivered video resolution. The upsampling algorithm for a video edge region requires a complicated computing process, which burdens the client and can cause mobile devices, which have limited electric power, to consume a lot of power.

Generally, upsampling can be implemented by image interpolation, which is achieved by curve

fitting or a regression analysis. Bilinear, Bicubic, and Lanczos are the most widely available interpolation methods for resizing an image. Although these methods are fast in terms of speed, there is a disadvantage in that the distribution of pixel values is not smooth [Lo, Wang and Hua (2018)]. Bilinear has a smoothing effect that produces blurred but jagged edges. Bicubic keeps the edges smooth and increases the perceived contrast to some extent. Compared with the other two algorithms, Lanczos has the best performance and the best compromise in reducing aliasing, increasing sharpness, and minimizing ringing [Oh, Yea, Vetro et al. (2009); Wang, Wang, Wang et al. (2019)]. To improve video degradation, the upsampling method uses direction-oriented interpolation (DOI) with edge information. That interpolation method shows higher performance in terms of objective and subjective evaluations than the existing interpolation method, but it requires a lot of complex computation.

Recently computation offloading to the mobile cloud has attracted plenty of attention, as researchers tackle difficulties with the upsampling algorithms used for video services. Cloud computing is a web-based software service, in which programs are put in utility data servers and accessed over the Internet whenever necessary by computers or mobile phones [Akherfia, Gerndta and Harroud (2018)]. Particularly for video streaming services, a feasible architecture should be able to achieve maximum benefit through cloud computing.

1.2 Our contributions

To tackle the abovementioned issues with conventional video streaming, in this paper, we propose a framework to improve the quality of video streaming services in 5G. The goal is to offload all the complicated and heavy operations from the mobile client to the cloud, thereby reducing the computational complexity on the client side and minimizing the transmission bit rate, which decreases network traffic and client power consumption. Then, the mobile client that received the video content will upsample the content using side information (SI) and serves a high-quality video with improved quality of experience (QoE).

In the proposed framework, we use cloud offloading and DOI with region matching as the upsampling algorithm to improve the quality of any low-resolution (LR) video with SI, i.e., edge information, without additional computation on the client side. Specifically, we modularize the proposed upsampling application and obtain an optimal offloading solution using a mathematical approach through optimization of the offloading decision.

To do so, the mobile client first uses our proposed offloading decision optimizer to offload the modules with heavy computational complexity to the cloud servers. Second, the cloud server performs the DOI required to generate SI about the original LR video content. The DOI (edge detection and edge upsampling) detects the edge regions of the original video and interpolates them to create HR content. Third, the interpolated edge content is compressed by the proposed SI compressor. The compressed SI is then sent to the client along with the original video content. Thus, energy efficient (EE) is improved because the data rate is much lower than that is required to stream HR content using conventional streaming methods. Finally, the mobile client receives, decodes, and integrates the SI and original content, which produces an interpolated, high-quality video using the Bilinear metric.

Our computation offloading to the mobile cloud computing (MCC) reduces computation on the client side compared with the existing methods, which require the clients to perform all processes. In other words, the cloud-offloading method can increase the availability of

the clients [You, Huang and Chae (2016)]. Moreover, the energy consumption of the mobile devices is reduced significantly. In terms of QoE, we found that the mean opinion score (MOS) differs little between the enhanced video and the original video, even though the amount of computation and bandwidth required for the cloud offloading are much less than those of the original. Consequently, our proposed framework enables users to enjoy high-quality video content while minimizing the bandwidth and power consumption [Kumcu, Bombeke, Platisa et al. (2017)].

The rest of this paper is organized as follows. We review the related works and discuss their drawbacks in Section 2. Our proposed framework for energy-efficient computation offloading in MCC for video streaming over 5G is presented in Section 3. In Section 4, we evaluate the performance of our proposed framework both objectively and subjectively. In Section 5, we draw conclusions.

2 Related work

In the literature, computation offloading to the cloud has been discussed by many researchers. In Guo et al. [Guo, Liu, Yang et al. (2019); Wang, Wu, Yuan, et al. (2019)], the authors used the concept of a collaborative mobile cloud (CMC) and proposed an energy-efficient offloading method to minimize mobile power consumption. In addition, as the concept of edge computing implies, they analyzed the collaboration between cloud computing and edge computing when distributing tasks among mobile devices at the edge nodes and the cloud server.

In Boukerche et al. [Boukerche, Guan and Grande (2018)] suggested a dynamic algorithm to offload computation to the cloud. In their proposed method, the offloading decision is made using parameters such as communication topology, device energy, and the cost of offloading the computation. Unlike Guo et al. [Guo, Liu, Yang et al. (2019); Wang, Wu, Yuan et al. (2019); Boukerche, Guan and Grande (2018)], we focus on modularized applications and QoE, as well as cost and EE. Our proposed framework offloads the highly complicated edge detection and upsampling processes required to improve LR video to the cloud, a virtual environment in which users can rent as many computational resources as needed, and thus it alleviates the client's burden [Raja, Chitra and Jonafark (2018)]. Because spatial edges, i.e., image boundaries, greatly influence user judgment about video quality [Panetta, Samani and Agaian (2018); Long, Li, Xie et al. (2018)], that edge information is sent to the client. In this paper, we call that edge information as SI.

In Guan et al. [Guan and Melodia (2017)] studied the factors affecting the power consumption in MCC and found that MCC can save energy consumption when mobile users offload heavy operations. However, they considered only the factors affecting power consumption and ignored the quality of the delivered video and QoE. In Zhou et al. [Zhou, Dastjerdi, Calheiros et al. (2017)], the authors considered the limitations of battery capacity and computing capability in mobile. They proposed a computation offloading algorithm that could be used in various topologies.

In Mahmoodi et al. [Mahmoodi, Uma and Subbalakshmi (2019)] optimized the problem for independent applications supported by a mobile device. They modularized a single application (interpolated video streaming) and performed optimization for a set of modules. However, they did not consider specific modules, e.g., decoder and renderer that cannot be

offloaded. In Lyu et al. [Lyu, Tian, Sengul et al. (2017)], the authors optimized for the number of independent applications supported by a mobile device. However, they limited their discussion to energy and cost efficiency, ignoring QoE.

In this paper, we consider all the above issues that have not been addressed yet. More specifically, we investigate the issue of computation offloading for video services in 5G networks. Our goal is to obtain a high-quality video from information derived from a LR video with edge detection. The advantages of the proposed framework are multifold. First, it improves the quality of LR video content and thus enhances the QoE significantly. Second, it also reduces network traffic through a decrease in the transmission bit rate. Third, it minimizes client power consumption through a cloud-offloading-based computation reduction.

3 Our proposed energy-efficient framework for computation offloading

The framework proposed in this paper is intended to deliver high-quality video using information derived from a LR video and the edge detection of that video while reducing network traffic by decreasing the transmission bit rate and minimizing client power consumption by offloading computation to the cloud. Mobile devices necessarily have limited computation resources, power, and bandwidth [Xie, Zeng, Li et al. (2017)]. We introduce a framework for exploiting the abundant computation resources of the cloud for mobile video services. A considerable amount of processing for video enhancement could be offloaded from the client to the cloud, which would reduce the energy consumption in the mobile device. Consequently, users can enjoy high-quality video while minimizing their bandwidth and power consumption.

Before we explain the proposed framework in detail, we will discuss an interesting video streaming technology developed by the Moving Picture Experts Group (MPEG), an international organization intended to establish global standards for various technologies, including video codec and transport protocols. Typically, the international standard for H.264/AVC (Advanced Video Coding) and H.265/HEVC (High Efficiency Video Coding) was published in the video codec field and the standards for dynamic adaptive streaming on HTTP (DASH) and MPEG media transport (MMT) were published in the transport protocol field. In recent years, standardization of network-based media processing (NBMP) has been actively under-way. NBMP is a standard technology that improves QoE by re-creating or transcoding media content through the intelligent middle of a network such as the cloud [ISO/IEC (2017); ISO/IEC (2019)]. The system architecture of NBMP is shown in Fig. 1.

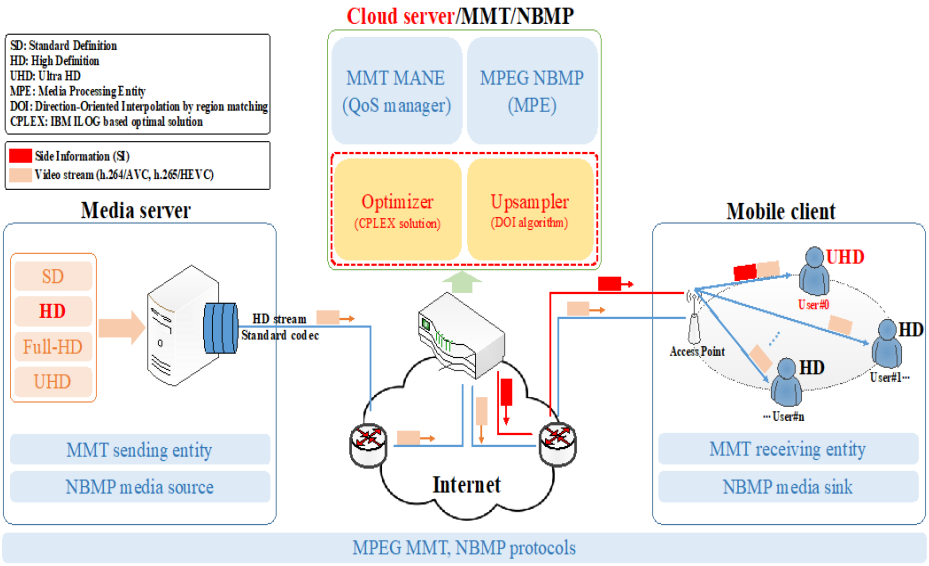


Figure 1: The proposed method is applicable to the MPEG protocols e.g., NBMP, MMT

However, NBMP is required to be based on the MPEG video codec and transport protocol standards named above, which is a disadvantage because it cannot be applied to new types of service models (such as our proposed model, which includes SI metadata) that can be used for improving the QoS/QoE of users.

3.1 Problem formulation

We use computation offloading in MCC to provide high-quality video with low energy and cost. Therefore, it is necessary to decide whether a module should be offloaded by using a mathematical optimization solution rather than a simple insight or empirical recognition.

We consider a mobile device that is processing tasks $j = \{1, \dots, n\}$ on a single application. Let the binary variable $\rho_j \in \{0,1\}$ be the offloading index of modules; $\rho_j = 1$ denotes that task j is executed on the mobile device, and $\rho_j = 0$ denotes that task j is offloaded to the cloud server. Let $\xi \in \{0,1\}$ be the service index of a mobile device that can be derived as:

$$\xi = 1 - \prod_{j=1}^{n} \rho_j \quad (1)$$

where $\xi = 0$ denotes that all modules are executed locally on the mobile device, and $\xi = 1$ denotes that at least one module is offloaded to the cloud server.

We define the constraint functions for energy and cost efficiency (CE) as follows.

- **Total energy consumption constraint on the mobile device:** We assume that E_j is the energy required by a module handled on the mobile device, and r_j^t is the transmission energy for the SI generated by cloud offloading. The constraint function for the total energy consumed by the mobile device E is:

$$\min_{\rho_j} E = \sum_{j=1}^{n} \{\rho_j E_j + (1-\rho_j) r_j^t\} \quad (2)$$

- **Total cost consumption constraint:** The constraint function for the total cost, C, which comprises E_j, transmission energy, and the time using the cloud server, is as follows:

$$\min_{\rho_j} C = \sum_{j=1}^{n} \{\rho_j (\lambda_1 E_j) + (1-\rho_j)(\lambda_2 S_{decrease})\} - \lambda_3 T_c \quad (3)$$

Details of the parameters in the functions are described in Subsection 4-2.

- **Complexity analysis:** Based on those constraints, the optimization problem (linear programming formation) can be solved efficiently using IBM ILOG CPLEX. In other words, the optimum ρ_j value for each module of the proposed system is derived using CPLEX [Borodin, Bourtembourg, Hnaien et al. (2018)].

Algorithm 1: The iterative approximation algorithm

Input: n, λ_1, λ_2, λ_3, E_j, r_j^t, $S_{decrease}$, T_c

Output: The offloading index of modules $\{\rho_j\}$

1: Initialization all $\rho_j = 0$;

2: **while** ξ remains changing in the iteration

3: **do**

4: Sort $\{\rho_j\}$ in ascending order;

5: Reset as many prior $\{\rho_j\}$ as possible to 0 until constraints (2) and (3) are satisfied using CPLEX;

6: Update ξ as (1);

7: **end while**

As illustrated in Algorithm 1, we calculate the offloading decision ρ_j for each module of the mobile device in each iteration. The number of module (n), cost coefficients ($\lambda_1, \lambda_2, \lambda_3$), r_j^t, E_j, S_{decrease}, and time spent using the cloud server (T_c) are considered as the input variables. T_c represents the computational complexity of each module and a specific environment in cloud computing. As a result, we achieve the offloading decision index (ρ_j) of each module as the output. First, ρ_j is initialized to 0, which means the task j is offloaded to the cloud server, and the algorithm finds the optimal value of ρ_j by sorting and resetting the ρ_j according to the constraints (2) and (3). The binary variable service flag ξ can be updated by Eq. (1). We repeat the process of the algorithm until ξ maintains the equal value in the iteration, which means the close-to-optimal offloading decisions at last. The results, the decision values of the modules obtained by the optimization problem, are represented in Fig. 2 and Tab. 1.

3.2 System architecture

In our proposed framework, we use cloud computing to enhance spatial quality of video by upsampling algorithm, which uses many computational resources, e.g., conversion of a standard-definition video to an HD video.

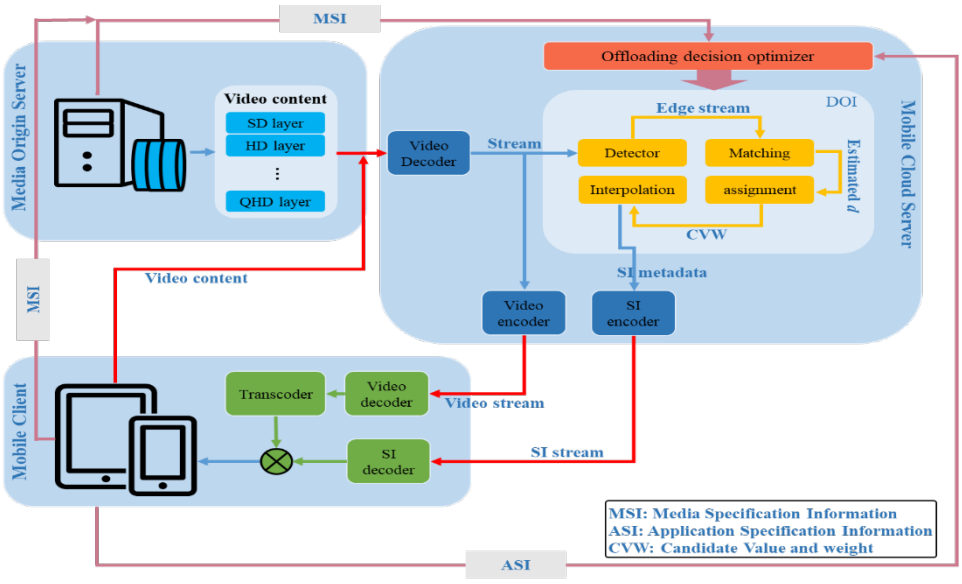


Figure 2: Architecture of the proposed framework

This upsampling process is separated into two parts, and the part requiring heavy computation is performed in the cloud, as illustrated in Fig. 2. A server transmits a LR video to the cloud. The cloud uses the open source Sobel algorithm to detect the edge of the LR video. Entropy coding and run-length coding are used to encode and decode edge information, as detailed in Subsection 3-3. The edge video is upsampled using DOI with region-matching algorithm and encoded, and is then sent together with the LR video to the client. The client decodes the LR video and combines it with the decoded edge video using a simple Bilinear algorithm, producing a high-quality, HR video. As shown in Fig. 1, the H.264/AVC international standard is used to encode and decode the video content.

Bilinear interpolation is used to interpolate the decoded videos at the client. The upsampling module in the cloud uses the DOI with region matching algorithm described in Subsection 3-4. As mentioned earlier, this module is an important operation to make the proposed video, which is considered an edge part of the video and has the most computational complexity, which is why we offload it to the cloud. The mobile client can then simply decode and interpolate the video, combining with the edge information to make a high-quality and HR video.

Tab. 1 depicts the computational complexity of each module used to implement the framework proposed in this paper. The cloud modules (edge detection and upsampling) account for 78.6% of the total computations required, whereas the modules processed by the client account for only 21.4%.

As illustrated in Tab. 1, we found that the edge upsampling module has the greatest amount computation. That is to say, the upsampling algorithm presented in Subsection 3-4 needs a complex process since the algorithm should make the edge video, which has a good quality. Therefore, the system proposed in this paper offloads the edge upsampling module to the cloud.

Table 1: Processing power required by each module (number of clocks×10^9)

Host	Modules	Video sequence			
		Aladdin	**Duck**	**Old Town**	**Park Run**
Cloud	**Edge detection**	5.47×10^2	6.11×10^2	6.09×10^2	6.46×10^2
	Edge upsampling	1.03×10^5	2.48×10^5	1.58×10^5	2.67×10^5
Client	**Non-edge upsampling**	6.53×10^3	6.58×10^3	6.68×10^3	6.56×10^3
	Video decoding	3.50×10^4	5.37×10^4	3.16×10^4	5.14×10^4
Total processing		1.45×10^5	3.08×10^5	1.97×10^5	3.25×10^5
Cloud offloading rate [%]		71.4	80.4	80.6	82.2
Client rate [%]		28.6	19.6	19.4	17.8

In addition, the module that does not requires to process in the client part is offloaded to cloud. On the client side, processing and energy have a tradeoff relationship. In Meng et al. [Meng, Wolter, Wu et al. (2018)], the author shows the tradeoff between shortening the execution time and extending the battery life of mobile devices and evaluates the performance improvement when making offloading decisions. In this paper, we selectively offload modules to the cloud, as in You et al. [You, Huang, Chae et al. (2017)]. As a result, the amount of processing required at the mobile client is reduced by 78.6% and the energy consumed is reduced proportionally.

3.3 SI processing

In the cloud environment, the edges are detected in the LR video using the Sobel algorithm. The edges are upsampled using DOI with region matching, and compressed using their locations and values.

In this section, to compress the HR edge video, we use the run-length encoding technique [Liaghati, Pan and Jiang (2017)] and the entropy coding technique for video compression [Manigandan and Deepa (2018)]. Our run-length encoding uses six types of edges $X_i's, (i = 0 \cdots N-1)$ in N edge directions. To apply the entropy coding technique, the total number of edges, X_T, and number of each edge type are calculated to draw the probability, $p_i = X_i/X_T$, of each type. The entropy H(X) is obtained as:

$$H(X) = -\sum_{i=0}^{N-1} p_i \log_2(p_i) \tag{4}$$

where $\log_2 p_i$ represents the optimal code-length of an edge type.

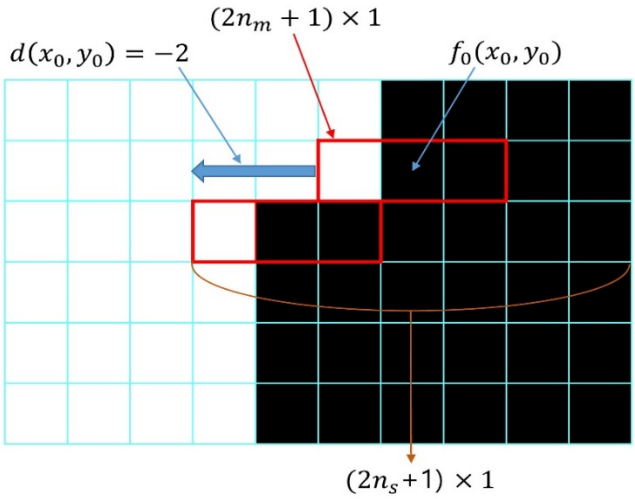


Figure 3: Region matching for estimating d

3.4 Direction-oriented interpolation with region matching

The upsampling algorithm proposed in this paper uses DOI with region matching. The weights and candidates of all pixels to be interpolated are saved into four steps, i.e., the down, up, left, and right directions.

For any pixel $f_0(x_0, y_0)$ in the down-sampled image $(W_0 \times H_0)$, a down direction $d(x_0, y_0)$ is estimated by region matching as follows [Junya, Takuya, Takayoshi et al. (2019)]:

$$d(x_0, y_0) = \arg\min_{x_0 - x} \left(\sum_{i=-n_m}^{n_m} |f_0(x_0 + i, y_0) - f_0(x_0 + x + i, y_0 + 1)| \right) \tag{5}$$

where $2(n_m + 1) \times 1$ is the block size for matching, $2(n_s + 1) \times 1$ is the maximum range of search, and the range of the search region is $-n_m + x_0 \leq x \leq n_s + x_0$, as shown in Fig. 3.

Candidate value and its weight of the upsampled image $f_1(x_1, y_1)$ are calculated and stored using $d(x_0, y_0)$, as shown in Fig. 4, where the positions of $f_1(x_1, y_1)$ are calculated as:

$$f_1\left(\gamma_x x_0 + \frac{d(x_0, y_0)\Delta\gamma_x}{\gamma_y} + n_g, \gamma_y y_0 + \Delta\right) \tag{6}$$

where f_1 is the upsampled image $(W_1 \times H_1)$, $\gamma_x = W_1/W_0$, $\gamma_y = H_1/H_0$, $0 \leq \Delta \leq \gamma_y$, and $(2n_g + 1) \times 1$ is the interpolation mask.

The candidate value and weight of $f_1(x_1, y_1)$ are calculated as:

$$c_i(f_1(x_1, y_1)) = f_0(x_0 + n_g/\gamma_x, y_0) \tag{7}$$

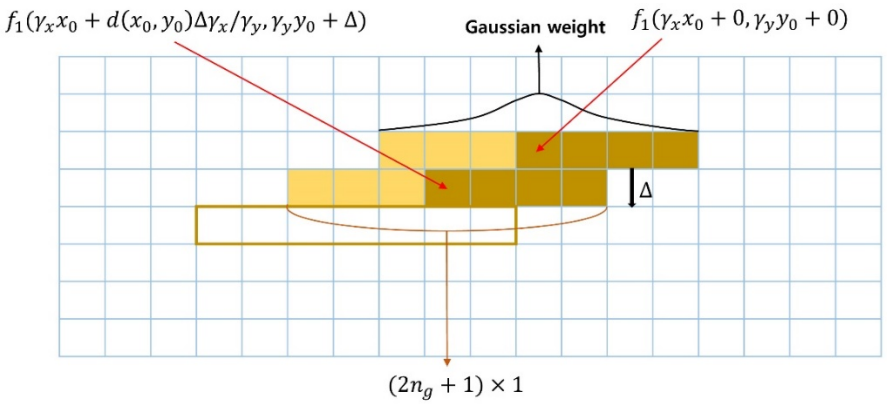


Figure 4: Assignment of candidate values and their weights in the horizontal direction

$$w_i\left(f_1(x_1, y_1)\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{n_g^2}{2\sigma^2}} \tag{8}$$

where c_i and w_i denote the i^{th} candidate and weight, respectively, and w is the Gaussian function of the magnitude of n_g, where the standard deviation is denoted by σ.

For any pixel $f_0(x_0, y_0)$, an up direction $u(x_0, y_0)$ is estimated by region matching as follows:

$$u(x_0, y_0) = \arg\min_{x_0 - x} \left(\sum_{i=-n_m}^{n_m} |f_0(x_0 + i, y_0) - f_0(x_0 + x + i, y_0 - 1)|\right) \tag{9}$$

and the method of calculation $f_1(x_1, y_1)$ equates to the down direction process, with a range of $-\gamma_y \leq \Delta \leq 0$.

For the horizontal axis, the left and right directions are estimated and candidate values and their weights are calculated in a manner similar to that used for the vertical axis.

For any pixel $f_0(x_0, y_0)$ of the down-sampled image $(W_0 \times H_0)$, a left direction $l(x_0, y_0)$ is estimated by region matching as follows:

$$l(x_0, y_0) = \arg\min_{y_0 - y} \left(\sum_{i=-n_m}^{n_m} |f_0(x_0, y_0 + i) - f_0(x_0, y_0 + y + i)|\right) \tag{10}$$

where $2(n_m + 1) \times 1$ is the block size for matching, and the range of the search region is $-n_m + y_0 \leq y \leq n_s + y_0$.

The candidate value and its weight for the up-sampled image $f_1(x_1, y_1)$ are calculated by:

$$f_1\left(\gamma_x x_0 + \Delta, \gamma_y y_0 + \frac{l(x_0, y_0)\Delta\gamma_y}{\gamma_x} + n_g\right) \tag{11}$$

where f_1 is the upsampled image $(W_1 \times H_1)$, $\gamma_x = W_1/W_0$, $\gamma_y = H_1/H_0$, $-\gamma_x \leq \Delta \leq 0$, and $(2n_g + 1) \times 1$ is the interpolation mask.

The candidate value and weight of $f_1(x_1, y_1)$ are calculated as:

$$c_i\left(f_1(x_1, y_1)\right) = f_0\left(x_0, y_0 + n_g/\gamma_y\right) \tag{12}$$

$$w_i\left(f_1(x_1, y_1)\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{n_g^2}{2\sigma^2}} \tag{13}$$

where c_i and w_i denote the i^{th} candidate and weight, respectively, and w is the Gaussian function of the magnitude of n_g, where the standard deviation is denoted by σ.

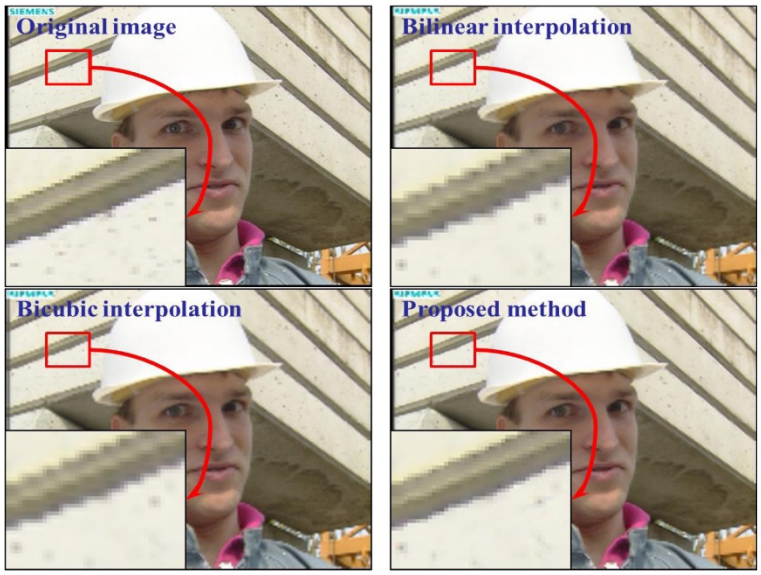


Figure 5: Interpolation results for the Foreman sequence

For any pixel $f_0(x_0, y_0)$, a right direction $r(x_0, y_0)$ is estimated by region matching as follows:

$$r(x_0, y_0) = \arg\min_{x_0 - x} \left(\sum_{i=-n_m}^{n_m} |f_0(x_0, y_0 + i) - f_0(x_0 + 1, y_0 + y + i)| \right) \quad (14)$$

and candidate values and their weights $f_1(x_1, y_1)$ are calculated in the range of $0 \leq \Delta \leq \gamma_x$. After calculating all of the candidate values and their weights for four directions, the interpolated values $f_1(x_1, y_1)$ are calculated using those candidate values and their weights. The $f_1(x_1, y_1)$ value is calculated using the weighted average of the candidate values by matching the four directions of the adjacent pixels, as follows:

$$f_1(x_1, y_1) = \frac{\sum_{i=1}^{n(x_1, y_1)} w_i(f_1(x_1, y_1)) c_i(f_1(x_1, y_1))}{\sum_{i=1}^{n(x_1, y_1)} w_i(f_1(x_1, y_1))} \quad (15)$$

where $n(x_1, y_1)$ is the number of candidates for $f_1(x_1, y_1)$. The sum of $w_i(f_1(x_1, y_1))$ is not normalized to 1, so the weighted sum should be divided by the sum of the weights.

This method can preserve high-frequency information with reduced blurring and jaggedness. As shown in Fig. 5, high-frequency information is preserved better with the proposed framework than with the other methods.

Although our upsampling algorithm provides higher quality video than existing upsampling algorithms, it requires the client to perform complicated operational processes. Therefore, our proposed framework offloads those complicated processes to the cloud to minimize the need for client computation.

4 Performance evaluation

4.1 Simulation environment

In this section, we conducted extensive simulation experiments to examine the performance of our proposed framework. The simulations were all performed on an IBM desktop computer (Intel® Core™2 Quad CPU Q6600 @2.40 GHz 2.40 GHz, 7 GB RAM, and 64 bit OS). We measured complexity as the number of clocks required for a certain computation because that is a generally meaningful metric in any computing environment. We used only a central processing unit (CPU) without a graphic processing unit (GPU) or any other special-purpose chips, which could have negatively affected on the performance. The characteristics of the videos used for the test are presented in Tab. 2.

To test diversity, we included an animation video (Aladdin). To ensure the validity of the test result, we used the test sequences also used in ISO/IEC WG11 SC29 MPEG: Duck, Old Town, and Park Run.

As stated earlier, the ultimate goal of this paper is to show how effective cloud offloading is in optimizing the energy, cost, and video quality of video streaming services. When used to change a LR video into a HR video, the transmission bit rate and computational burden in the client were noticeably reduced.

Table 2: Characteristics of the video sequences

Video sequence	Aladdin	Duck	Old Town	Park Run
Resolution (LR)	640×360	640×360	640×360	640×360
Resolution (HR)	1280×720	1280×720	1280×720	1280×720
Frame rate [fps]	32	32	32	32
Group of picture size	16	16	16	16
Intra period	8	8	8	8

Therefore, we compare the performance of the proposed framework with the conventional method, in which the client receives only the HR video from the server. More specifically, as performance evaluation metrics, we consider the transmission bit rate and decoding computational complexity on the client side. In addition, the video quality generated by the proposed framework is evaluated using MOS. We also considered throughput on the receiver side. The increase in the bandwidth required for the additional SI (i.e., edge information) proved to be much smaller than that required by the original HR video. Because we assumed that the computational resources of the cloud are infinite, we ignored the processing delay in the cloud.

4.2 Objective metric-based evaluation

We used the joint test model (JM) from H.264/AVC as the reference software to decode the four video sequences presented in Tab. 2, and we applied the Intel Parallel Studio program to measure the decoding time for each video and to predict its processing requirements. JM runs as a single thread, so the time is considered equal to the processing requirement.

Tab. 3 shows the transmission bit rate and operational requirement of the proposed framework. The data sent from the cloud to a client is the sum of LR video and edge information. The amount of a client's computational complexity is calculated by adding the computational complexity of decoding the transmitted LR video and the bilinear upsampling operation.

Table 3: Transmission bit rate and processing requirement of the proposed framework (the number of clocks×10^9)

Video sequence	Processing		Transmission	
	Bilinear upsampling [clocks×10^9]	Decoding [clocks×10^9]	LR [Mbps]	Edge information [Mbps]
Aladdin	6.53	10.09	0.75	0.4
Duck	6.56	15.5	2.56	4.23
Old Town	6.66	9.32	0.78	0.88
Park Run	6.56	15.03	2.54	9.86

Next, to test its validity, we compared the proposed framework with a conventional streaming method. In the conventional method, a client first receives a HR video from a server and then decodes the video. Tab. 4 shows the transmission bit rates and computational requirement for the conventional streaming method and the proposed framework.

As shown in Tab. 4, the proposed method ensures a higher efficiency than the conventional streaming method. The computational complexity at the client side is reduced by almost 68.2% on average and the client`s transmission bit rate is reduced by almost 45.2% on average.

Table 4: Transmission bit rates and computational requirements of the conventional streaming method and the proposed method (the number of clocks×10^9)

Video Sequence	Conventional method		Proposed method	
	Computation [clocks×10^9]	Data size [kbps]	Computation [clocks×10^9]	Data size [kbps]
Aladdin	47.5	2.10×10^3	16.6	1.13×10^3
Duck	74.5	1.08×10^4	22.0	6.78×10^3
Old Town	46.4	5.10×10^3	15.0	1.67×10^3
Park Run	76.4	1.79×10^4	21.6	1.24×10^4

Table 5: Efficiency of the proposed method compared with the conventional streaming method

Video Sequence	Computation (%)	Bit rate (%)
Aladdin	35.0	53.5
Duck	29.6	62.6
Old Town	34.4	32.7
Park Run	28.3	69.4

Tab. 5 presents the efficiency of the proposed method as a percentage, where the transmission bit rate and computational complexity for processing HR video content through the conventional streaming method are 100%.

As shown in Fig. 6, the computational efficiency and transmission bit rate differ according to the scene complexity in the video contents. For example, the animated Aladdin content has a comparatively low scene complexity. Therefore, both the computational complexity and transmission bit rate are highly efficient. Because the Old Town content also has lower complexity than the Park Run or Duck content, we got efficiencies of more than 50% for both computation and bit rate.

On the other hand, the Duck content is mostly a waveform, and the Park Run content has a well-wooded background. Because most of the computational complexity is offloaded to the cloud in those cases, both those videos show a higher computational efficiency than the Aladdin and Old Town videos (which are not complex). However, because the video content is complex in terms of scenes, the number of data to be transmitted increases, so both those video contents show a less efficient transmission bit rate of about 30%.

For another objective evaluation, Fig. 7 represents the structural similarity index (SSIM) for the proposed framework, and Fig. 8 presents value of the peak signal-to-noise ratio (PSNR) for the proposed framework.

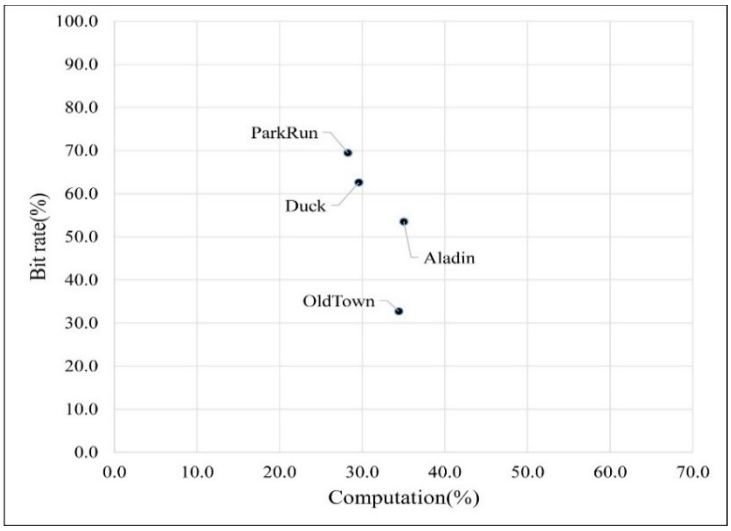


Figure 6: The efficiency of the proposed framework compared with the conventional one

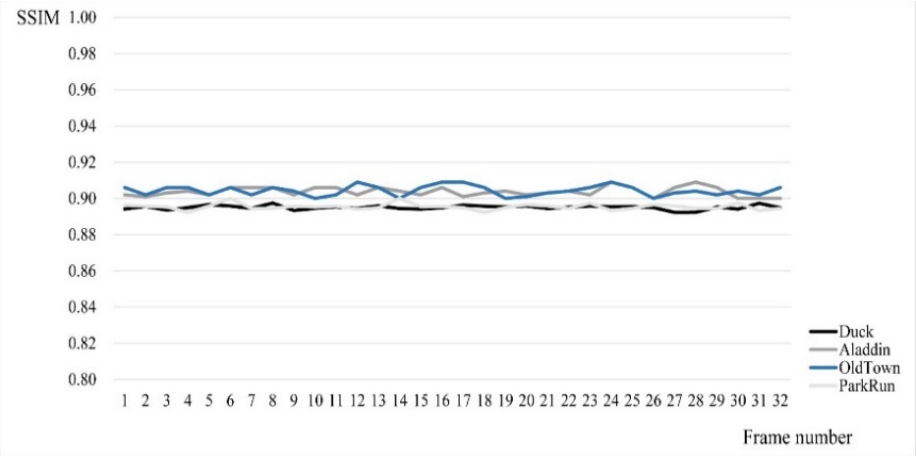


Figure 7: Structural similarity index measurement for each video

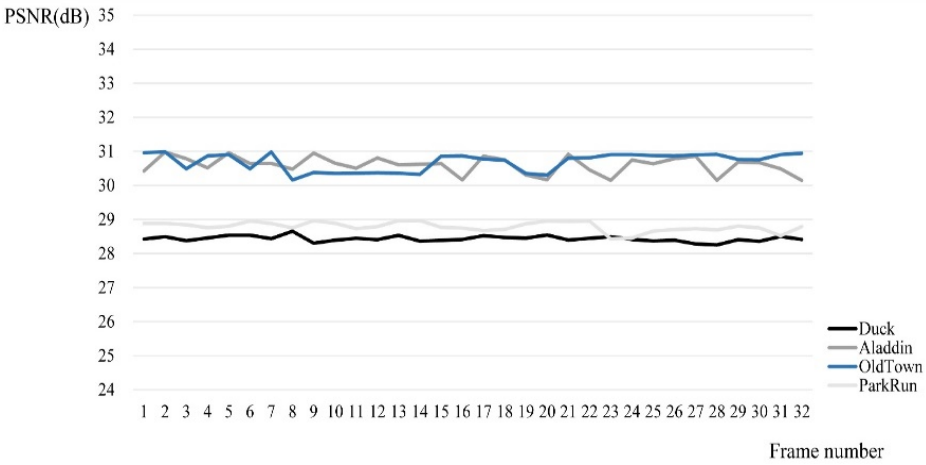


Figure 8: PSNR measurement for each video sequence

Tab. 6 presents the average SSIM and PSNR values for each video. The SSIM is a framework for predicting the perceived quality of digital images and videos [Wu, Li, Meng et al. (2018)]. As shown in Fig. 7 and Fig. 8 and Tab. 6, the average SSIM value is about 0.9, and the average PSNR value is about 29.63 dB.

Table 6: Measured SSIM and PSNR values

Metric	Video Sequence			
	Aladdin	**Duck**	**Old Town**	**Park Run**
SSIM	0.89	0.9	0.9	0.89
PSNR [dB]	30.6	28.43	30.69	28.79

So far, we have examined the improvement in computation and the transmission bit rate for the proposed framework and compared the quality of the different video contents in various ways. Next, we analyze the EE of the proposed framework, which is driven by the energy needed to generate and transmit the SI. In other words, the energy of the computational complexity that is offloaded from the mobile client to the cloud server is a profit, while the energy required to transmit the SI from the cloud server to the client is a loss. Thus, the sum of those energies is the total energy gain. CE is driven by the cost of using the cloud server to generate and transmit the SI and the cost of battery consumption at the mobile client [Sadooghi, Martin, Li et al. (2017); Nan, Li, Bao et al. (2017)].

Our simulation environment for the energy and CE analyses includes one mobile device (Samsung Galaxy S4) that is connected to LTE TCP network and an Amazon web services (AWS) on demand server. The parameters used in the equations that analyze EE and shows the energy consumption with three CPU specifications are presented in Tabs. 7 and 8 respectively.

First, we calculate the energy consumed (E_j) by the modules offloaded from the mobile client to the cloud server as:

$$E_j = \frac{C_j}{C_{cpu}^M} P_{Mobile} \tag{16}$$

Table 7: Parameter descriptions

Parameter	Description
$\boldsymbol{C_{cpu}^M}$	CPU specifications on the mobile device used for the simulation
$\boldsymbol{C_j}$	Computational amount for module j
$\boldsymbol{P_{Mobile}}$	Energy consumption by the CPU
$\boldsymbol{D_{SI}}$	Side information data size
$\boldsymbol{T_x}$	Transmission energy per 100 KB in LTE
$\boldsymbol{B}$	LTE network bandwidth in the simulation
$\boldsymbol{\lambda_1}$	Consumption power per dollar for the Galaxy S4
$\boldsymbol{\lambda_2}$	Transmission data per dollar
$\boldsymbol{\lambda_3}$	Usable time per dollar on the cloud server (AWS)
$\boldsymbol{S_{decrease}}$	Decrease in required data caused by the proposed framework
P_Δ	Increment of core energy consumption in the CPU

Table 8: Power consumption for different CPU specifications

$\boldsymbol{C_{cpu}^M}$	384 MHz		1026 MHz		1890 MHz	
Power (mW)	$\boldsymbol{P_B}$	$\boldsymbol{P_\Delta}$	$\boldsymbol{P_B}$	$\boldsymbol{P_\Delta}$	$\boldsymbol{P_B}$	$\boldsymbol{P_\Delta}$
	86	207	86	438	86	1358

Tab. 9 shows the power required to transmit 100 KB data across the LTE network.

Then, we calculate the transmission energy (r_j^t) required for module j to transmit the SI data, which is generated in the cloud server, as:

$$r_j^t = \frac{D_{SI}}{B} T_x \quad (17)$$

Finally, the total EE, E_{effi}, for the proposed framework is obtained as:

$$E_{effi} = \left\{\sum_{j=1}^{n} E_j - r_j^t\right\} \quad (18)$$

Table 9: Power required to transmit 100 KB across the cellular network

Samsung Galaxy S4 LTE (100KB)		
Power [dBm]	$\boldsymbol{T_x}$ [mW]	$\boldsymbol{R_x}$ [mW]
-85	1177	938
-95	1849	1110
-105	1699	1140

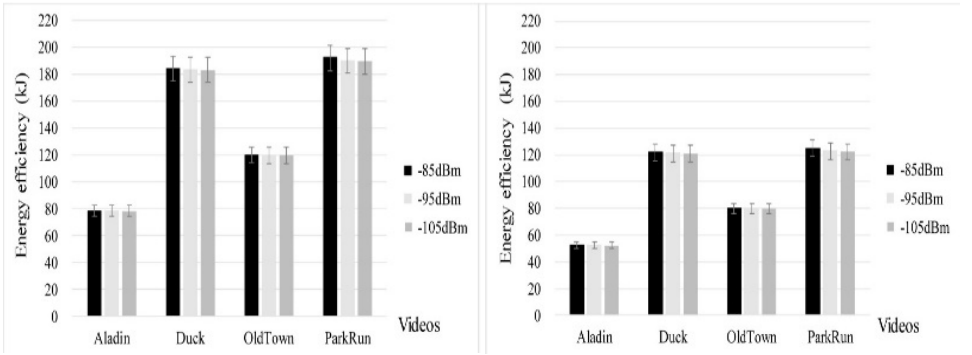
Fig. 9 and Tab. 10 depict the EE of the proposed framework at three CPU specifications (384 MHz, 1026 MHz, 1890 MHz) for the cloud server and three channel conditions (-85 dBm, -95 dBm, -105 dBm) for the cellular network. In terms of the CPU specifications, the EE depends on the CPU processing per second (C_{cpu}^M) and the consumed power (P_{Mobile}), which is caused by the CPU. In terms of the channel condition, better network conditions produce better EE by reducing the transmission energy.

Table 10: EE of the proposed framework

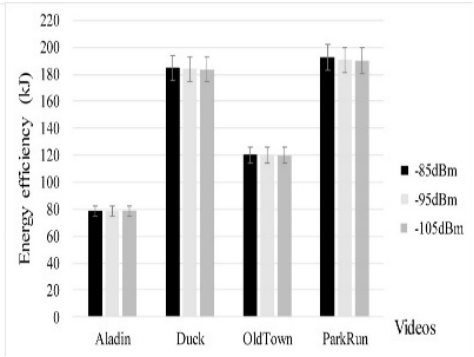
	$\boldsymbol{C_{cpu}^M}$ [MHz]	$\boldsymbol{T_x}$ [dBm]	**Video Sequence**			
			Aladdin	**Duck**	**Old Town**	**Park Run**
$\boldsymbol{E_{effi}}$ $[kJ]$	**384**	**-85**	78.53	184.62	119.97	192.38
		-95	78.44	183.69	119.77	190.21
		-105	78.42	183.52	119.74	189.83
	1026	**-85**	52.40	121.89	79.95	124.85
		-95	52.32	120.96	79.75	122.68
		-105	52.30	120.80	79.72	122.30
	1890	**-85**	78.63	184.87	120.12	192.65
		-95	78.54	183.93	119.93	190.48
		-105	78.53	183.77	119.90	190.10

Tab. 7 shows the cost coefficients, battery consumption at mobile terminal λ_1, data usage in cellular network λ_2, and spent time on the cloud server (AWS on demand) λ_3. Let $S_{decrease}$ be the decreased data resulted by the proposed method and T_c denotes the time spent using the cloud server. We can derive the CE (C_j) for module j as follows:

$$C_j = \left(\lambda_1 E_j + \lambda_2 S_{decrease}\right) - \lambda_3 T_c \quad (19)$$



(a) 384 MHz (b) 1026 MHz

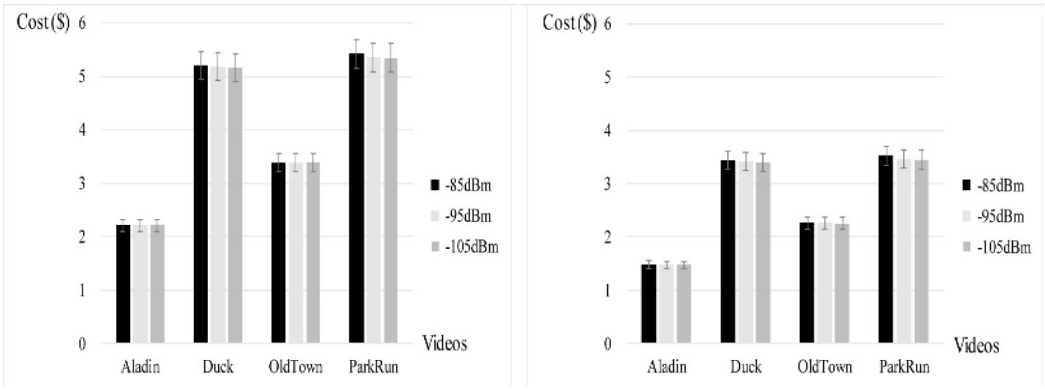


(c) 1890 MHz

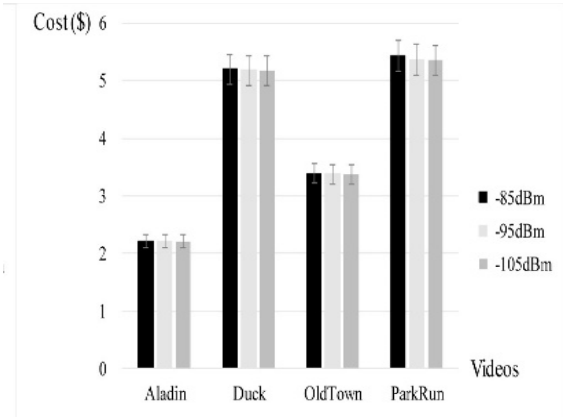
Figure 9: EE of the proposed framework examined by different mobile phones with different CPU capability

Then, the total CE is:

$$C = \sum_{j=1}^{n}(\lambda_1 E_j) + \lambda_2 S_{decrease} - \lambda_3 T_c \quad (20)$$



(a) 384 MHz (b) 1026 MHz



(c) 1890 MHz

Figure 10: CE of the proposed framework is examined by different mobile phones with different CPU capability

Fig. 10 and Tab. 11 show the CE calculated by Eq. (20). When we analyzed the CE according to the CPU specifications and channel conditions, the results show the same pattern as the EE.

Table 11: CE of the proposed framework

	C_{cpu}^{M} [MHz]	T_x [dBm]	Video Sequence			
			Aladdin	**Duck**	**Old Town**	**Park Run**
C[$]	384	-85	2.21	5.20	3.38	5.42
		-95	2.21	5.17	3.38	5.36
		-105	2.21	5.17	3.38	5.35
	1026	-85	1.48	3.44	2.26	3.52
		-95	1.47	3.41	2.25	3.46
		-105	1.47	3.41	2.25	3.45
	1890	-85	2.21	5.21	3.39	5.43
		-95	2.21	5.18	3.38	5.37
		-105	2.21	5.18	3.38	5.36

In the next subsection, we present the subjective evaluation for the proposed framework in terms of MOS.

4.3 Subjective evaluation

Typically, a LR video is upsampled to HR video, and then compared with original HR video. In this paper, we assume a specific simulation environment in which there is no original HR video, so we use the MOS metric for our subjective evaluation. In other words, our framework is intended for originally low-quality video content, such as social sharing by normal users.

We already have PSNR and SSIM metrics to evaluate the proposed framework and to acquire them, we have to use an original HR video to prove the performance of our proposed upsampling algorithm. In the MOS measurement, we use the same original HR video to evaluate the proposed framework, but our participants do not know what the original videos are.

We measured MOS to examine the video quality produced by our proposed framework. Tab. 12 shows the scale to determine the quality of each video [Lopes, Ascenso, Rodrigues et al. (2018); Fan, Jiang and Huang (2017)], based on which the HR videos that are generated by the proposed framework are measured by participants.

Table 12: Perceptual video quality scale for MOS

MOS	Quality	Impairment
5	Excellent	Very satisfied
4	Good	Satisfied
3	Fair	Some users dissatisfied
2	Poor	Many users dissatisfied
1	Bad	Nearly all user dissatisfied

We used the following conditions for the experiment:

- The participants were 20 image-process majors and 10 ordinary persons.
- The proposed HR and original HR video were played at the same time. Here, the participants did not recognize whether it is original HD or proposed HD videos.
- The participants scored each video using the scale in Tab. 12.

Table 13: MOS scores of the proposed HR videos

Video Sequence	Aladdin	Duck	Old Town	Park Run
Average MOS	3.5	3.7	2.9	3.5

Tab. 13 presents the average MOS for each video sequence. The overall MOS for the videos was 3.4, indicating good quality. Tab. 14 illustrates the MOS results for both the proposed HR and original HR videos. The original videos did not get a perfect score when the participants did not know which videos were original and proposed.

Table 14: MOS scores for both the proposed HR and original HR videos

Video Sequence		Aladdin	Duck	Old Town	Park Run
MOS	Original	4.3	4	3.9	3.7
	Proposed framework	4.5	3.9	4.1	3.9

The MOS values for the upsampled videos are almost the same as those of the original videos even though the transmission bit rate and computational requirements for the cloud-enhanced videos are much smaller than those for the originals.

5 Conclusion

In this paper, we have shown how effective cloud offloading can be for video streaming services. In terms of computational complexity on the client side, transmission bit rate, and evaluation of video quality, we found cloud offloading to be effective, with the proposed framework improving the QoE of the client. Cloud computing is currently developing, and the 5G network, which will be able to transmit more data more rapidly than existing technology, is on its way. At this time, the framework proposed in this paper is a future-oriented technology that is suitable for wireless communication and networking and next generation multimedia applications. Furthermore, various engineering fields are continuing to research deep learning technologies (especially deep reinforcement learning), as well as video codec and network management technology. In accordance with those trends, as our future work, we intend to improve the performance of our proposed framework in terms of cost efficiency, EE, latency, and QoE for the emerging services such as high-quality immersive multimedia services.

Acknowledgement: This research was supported by Korea Electric Power Corporation. (Grant No. R18XA02)

References

Akherfi, K.; Gerndt, M.; Harround, H. (2018): Mobile cloud computing for computation offloading: issues and challenges. *Applied Computing and Informatics*, vol. 14, no. 1, pp. 1-16.

Borodin, V.; Bourtembourg, J.; Hnajen, F.; Labadie, N. (2018): COTS software integration for simulation optimization coupling: case of ARENA and CPLEX products. *International Journal of Modelling and Simulation*, vol. 39, no. 3, pp. 178-189.

Boukerche, A.; Guan, S.; Grande, R. E. D. (2018): A task-centric mobile cloud-based system to enable energy-aware efficient offloading. *IEEE Transactions on Sustainable Computing*, vol. 3, no. 4, pp. 248-261.

Cheon, M.; Lee, J. S. (2018): Subjective and objective quality assessment of compressed 4K UHD videos for immersive experience. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 7, pp. 1467-1480.

CISCO (2019): CISCO visual networking index: global mobile data traffic forecast update, 2017-2022, White Paper. *CISCO Technical Report.*

Concolato, C.; Feuvre, J. L.; Denoual, F.; Mazé, F.; Nassor, E. et al. (2018): Adaptive streaming of HEVC tiled videos using MPEG-DASH. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 8, pp. 1981-1992.

Eswara, N.; Manasa, K.; Kommineni, A.; Chakraborty, S.; Sethuram, H. P. et al. (2018): A continuous QoE evaluation framework for video streaming over HTTP. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 11, pp. 3236-3250.

Fan, Q.; Yin, H.; Min, G.; Yang, P.; Luo, Y. et al. (2017): Video delivery networks: challenges, solutions and future directions. *Computers and Electrical Engineering*, vol. 66, no. 2018, pp. 332-341.

Fan, Z.; Jiang, T.; Huang, T. (2017): Active sampling exploiting reliable informativeness for subjective image quality assessment based on pairwise comparison. *IEEE Transactions on Multimedia*, vol. 19, no. 12, pp. 2720-2735.

Fischbach, M.; Wiebusch, D.; Latoschik, M. E. (2017): Semantic entity-component state management techniques to enhance software quality for multimodal VR-systems. *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 4, pp. 1407-1416.

Ghadiyaram, D.; Pan, J.; Bovik, A. C. (2019): A subjective and objective study of stalling events in mobile streaming videos. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 1, pp. 183-197.

Guan, Z.; Melodia, T. (2017): The value of cooperation: Minimizing user costs in multi-broker mobile cloud computing networks. *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 780-791.

Guo, S.; Liu, J.; Yang, Y.; Xiao, B.; Li, Z. (2019): Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing. *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 319-333.

Habiba, U.; Hossain, E. (2018): Auction mechanisms for virtualization in 5G cellular networks: basics, trends, and open challenges. *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2264-2293.

INTERNATIONAL STANDARD ISO/IEC 23008-1 Second Edition (2017): Information Technology-High Efficiency Coding and Media Delivery in Heterogeneous Environments-Part 1: MPEG Media Transport (MMT).

ISO/IEC JTC 1/SC 29/WG 11 (2019): N18401: Potential Enhancements for Network-Based Media Processing.

Jalil, M.; Tran, N. H.; Doug, Y. S.; Jo, B. S.; Hong, C. S. et al. (2017): QoE-driven channel allocation and handoff management for seamless multimedia in cognitive 5G cellular networks. *IEEE Transactions on Vehicular Technology*, vol. 66, no. 7, pp. 6569-6585.

Kumcu, A.; Bombeke, K.; Platisa, L.; Jovanov, L.; Looy, J. V. et al. (2017): Performance of four subjective video quality assessment protocols and impact of different rating preprocessing and analysis methods. *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 1, pp. 48-63.

Liaghati, A. L.; Pan, W. D.; Jiang, Z. (2017): Biased run-length coding of bilevel classification label maps of hyperspectral images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 10, pp. 4580-4588.

Lo, K. H.; Wang, Y. C. F.; Hua, K. L. (2018): Edge-preserving depth map upsampling by joint trilateral filter. *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 371-384.

Long, M.; Li, Z.; Xie, X.; Li, G.; Wang, Z. (2018): Adaptive image enhancement based on guide image and fraction-power transformation for wireless capsule endoscopy. *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 5, pp. 993-1003.

Lopes, F.; Ascenso, J.; Rodrigues, A.; Queluz, M. P. (2018): Subjective and objective quality assessment of omnidirectional video. *Proceedings of SPIE*, vol. 10752, no. 2018, pp. 1-17.

Lyu, X.; Tian, H.; Sengul, C.; Zhang, P. (2017): Multiuser joint task offloading and resource optimization in proximate clouds. *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435-3447.

Mahmoodi, S. E.; Uma, R. N.; Subbalakshmi, K. P. (2019): Optimal joint scheduling and cloud offloading for mobile applications. *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 301-313.

Manigandan, M. D.; Deepa, S. (2018): Comprehensive study on the effect of entropy encoding algorithms on medical image compression. *International Research Journal of Engineering and Technology*, vol. 5, no. 4, pp. 3460-3468.

Meng, T.; Wolter, K.; Wu, H.; Wang, Q. (2018): A secure and cost-efficient offloading policy for mobile cloud computing against timing attacks. *Pervasive and Mobile Computing*, vol. 45, no. 2018, pp. 4-18.

Nan, Y.; Li, W.; Bao, W.; Delicato, F. C.; Pires, P. F. et al. (2017): Adaptive energy-aware computation offloading for cloud of things systems. *IEEE Access*, vol. 5, no. 2017, pp. 23947-23957.

Oh, K. J.; Yea, S.; Vetro, A.; Ho, Y. S. (2009): Depth reconstruction filter and down/up sampling for depth coding in 3-D video. *IEEE Signal Processing Letters*, vol. 16, no. 9, pp. 747-750.

Panetta, K.; Samani, A.; Agaian, S. (2018): A robust no-reference, no-parameter, transform domain image quality metric for evaluating the quality of color images. *IEEE Access*, vol. 6, no. 2018, pp. 10979-10985.

Piran, M. J.; Islam, S. M.; Suh, D. Y. (2018): CASH: content- and network-context-aware streaming over 5G HetNets. *IEEE Access*, vol. 6, no. 1, pp. 46167-46178.

Raja, R. V.; Chitra, K.; Jonafark, M. (2018): A survey on mobile cloud computing. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 3, no. 3, pp. 2096-2100.

Sadooghi, I.; Martin, J. H.; Li, T.; Brandstatter, K.; Maheshwari, K. et al. (2017): Understanding the performance and potential of cloud computing for scientific applications. *IEEE Transactions on Cloud Computing*, vol. 5, no. 2, pp. 358-371.

Sato, J.; Akashi, T.; Yamada, T.; Ito, K. (2019): Affine template matching by differential evolution with adaptive two-part search. *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 14, no. 4, pp. 615-622.

Wang, Y.; Wang, L.; Wang, H.; Li, P. (2019): Resolution-aware network for image super-resolution. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 5, pp. 1259-1269.

Wang, Y.; Wu, L.; Yuan, X.; Liu, X.; Li, X. (2019): An energy-efficient and deadline-aware task offloading strategy based on channel constraint for mobile cloud workflows. *IEEE Access*, vol. 7, no. 2019, pp. 69858-69872.

Wu, Q.; Li, H.; Meng, F.; Ngan, K. N. (2018): A perceptually weighted rank correlation indicator for objective image quality assessment. *IEEE Transactions on Image Processing*, vol. 27, no. 5, pp. 2499-2513.

Xie, G.; Zeng, G.; Li, R.; Li, K. (2017): Energy-aware processor merging algorithms for deadline constrained parallel applications in heterogeneous cloud computing. *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 62-75.

Yadav, R.; Zhang, W.; Kaiwartya, O.; Singh, P. R.; Elgendy, I. A. et al. (2018): Adaptive energy-aware algorithms for minimizing energy consumption and SLA violation in cloud computing. *IEEE Access*, vol. 6, no. 2018, pp. 55923-55936.

You, C.; Huang, K.; Chae, H. (2016): Energy efficient mobile cloud computing powered by wireless energy transfer. *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1757-1771.

You, C.; Huang, K.; Chae, H.; Kim, B. H. (2017): Energy-efficient resource allocation for mobile edge computation offloading. *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397-1411.

Yun, J.; Jalil, P; Suh, D. Y. (2018): QoE-driven resource allocation for live video streaming over network-assisted D2D communications. *IEEE Access*, vol. 20, no. 1, pp. 72563-72580.

Zhou, B.; Dastjerdi, A. V.; Calheiros, R. N.; Srirama, S. N.; Buyya, R. (2017): mCloud: a context-aware offloading framework for heterogeneous mobile cloud. *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 797-810.

Zink, M.; Sitaraman, R.; Nahrstedt, K. (2019): Scalable 360º video stream delivery: challenges, solutions, and opportunities. *Proceedings of the IEEE*, vol. 107, no. 4, pp. 639-650.