Image compression in resource-constrained eye tracking devices

Pavel Morozkin\textsuperscript{a,b}, Marc Swynghedauw\textsuperscript{a} and Maria Trocan\textsuperscript{b}

\textsuperscript{a}SuriCog, Paris, France; \textsuperscript{b}Institut Supérieur d’Electronique de Paris, Paris, France

ABSTRACT
Resource-constrained embedded devices, operating with images, are becoming increasingly common. Examples include remote low-power smart sensors, wireless sensor networks, autonomous cameras, eye tracking devices, etc. The principal requirements of such devices are the operation in real time, low power, low heat as well as low MIPS. These requirements can be fulfilled with the use of approximated version of original image processing algorithms. The EyeDee\textsuperscript{TM} embedded eye tracking solution (developed by SuriCog) is the world’s first innovative solution using the eye as a real-time mobile digital cursor, while maintaining full mobility. Being an example of resource-constrained embedded device, the system consists of a wearable device (Weetsy\textsuperscript{TM} frame) capturing images on the human’s eye and an embedded pre-processing device (Weetsy\textsuperscript{TM} pre-processing board) sending these eye images over a transmission medium (wire/wireless transmission) to a remote processing unit for the further gaze reconstruction. This paper is aimed at introducing image compression approaches in the resource-constrained devices in general and some of their implementation in the Weetsy\textsuperscript{TM} pre-processing board in particular.

Introduction

Resource-constrained (also called resource-restrained) device is a small physical device that has limited processing and storage capabilities, and it often runs on batteries (Khosrow-Pour, 2005). Such a device is usually equipped with image sensors and employs image compression techniques to significantly reduce the amount of data to transmit over a selected medium (wire/wireless transmission). These devices are usually found themselves in a wide range of applications including monitoring conditions of the environment (using Wireless Sensor Networks or WSN (Hutton, 2005), Internet of Things (IoT) applications (Xia, Yang, Wang, & Vinel, 2012), wearable devices (smart watches, physical activity trackers, etc) (Case, Burwick, Volpp, & Patel, 2015), smart...
house automation (Sun, Yu, Kochurov, Hao, & Hu, 2013), eye tracking applications (Duchowski, 2007), etc. Due to the limited capabilities of these devices, different design trade-offs must be considered, in hardware design and software development. These trade-offs usually imply usage of several techniques:

1. Use of approximated versions of the original algorithms. This technique is aimed at achieving such an implementation, which provides near-the-same approximated results in comparison with results produced by original algorithms, but with the use of much less resources (processor instructions in case of an MCU) taking much less execution time.
2. Use of advanced application-dependent power-saving modes (for example, sleep mode, standby mode, etc).
3. Process less data locally. There is a well-known trade-off between the amount of processing to be done locally and the amount of data to be sent over a medium. The general assumption is that if there are more data to be processed locally, then there are less data to be sent and vice versa. The goal is to find an optimal point between application performance (usually based on data transmission speed) and battery consumption.

Usually each product involving resource-constrained device, which locally performs image compression, is based on several architectural, performance and power management decisions, leading to desired product characteristics, such as physical dimensions, performance and final price. Since image compression allows us to drastically reduce the amount of data to be transmitted to a remote unit, there is a natural interest (both industrial and scientific/academic) in the design and implementation of such image processing and compression algorithms and approaches for the resource-constrained devices. This paper discusses image compression approaches for the resource-constrained devices and then it focuses on the image compression approaches for the specific resource-constrained eye tracking system.

**Related work**

Resource-constrained embedded devices using image compression (like smart cameras or wireless image and video sensor networks) are widely reviewed in the literature. For example, in Lee, Kim, Rahimi, Estrin, & Villasenor, 2009 authors present quantitative comparison between the energy costs associated with direct transmission of uncompressed images and sensor platform-based JPEG (Wallace, 1991) compression followed by transmission of the compressed image. They also examine advanced applications of JPEG (such as region of interest coding and successive/progressive transmission) as well as provided detailed experimental results examining the trade-offs in processor resources, processing/transmission time, bandwidth utilization, image quality, and present overall energy consumption considerations.

Since Wireless Multimedia Sensor Networks (WMSNs) (Akyildiz, Melodia, & Chowdhury, 2007) differ from classical wired networks and wireless sensor networks, WMSNs are considered to be an independent research domain in which several surveys have been done. For example, the survey (ZainEldin, Elhosseini, & Ali, 2015) studies and analyses relevant
research directions and the most recent algorithms of image compression over WMSN, characterizes the benefits and shortcomings of recent efforts of such algorithms and provides an open research issue for each compression method and its potentials to WMSN. The final goal is to reduce consumed power and increase life time (main performance metric).

Since energy efficiency is one of the most challenging issues in multimedia applications involving resource-constrained devices benefiting on image compression, several surveys especially study this aspect. For example, in the survey Ma, Hempel, Peng, & Sharif, 2013 provide a broad picture of the state-of-the-art energy-efficient techniques that have been proposed in wireless multimedia communication for resource-constrained systems such as wireless sensor networks and mobile devices. Then they categorize these techniques into two groups: multimedia compression techniques and multimedia transmission techniques and for each group the authors perform analysis and evaluations on energy efficiency in applying these compression algorithms to resource-constrained multimedia transmission systems. Another work in the domain of energy-efficient image compression in resource-constrained, but this time applied to multihop wireless networks is presented in (Wu & Abouzeid, 2005). The goal is to propose two design alternatives for energy-efficient distributed image compression and then to investigate them with respect to energy consumption and image quality. Their simulation results show that the proposed scheme prolongs the system lifetime. Another work, which targets energy saving in the sensor network application, but with the use of general purpose data compression (instead if image compression) is presented by Sadler & Martonosi, 2006. The authors develop Sensor LZW (S-LZW) algorithm and show how different amounts of compression can lead to energy savings by up to a factor of x4.5.

Deep neural networks (DNNs) (Schmidhuber, 2015) have found themselves also in the image compression domain. For example, in Liu et al., 2018 authors develop an image compression framework tailored for DNN applications, which is named DeepN-JPEG, with the goal of embracing the nature of deep cascaded information process mechanism of DNN architecture. According to experimental results DeepN-JPEG can achieve \( \sim 3.5x \) higher compression rate over the popular JPEG.

In contrast with modern DNN-based approaches, application of well-known wavelet-based image compression in WSNs is studied in Nasri, Helali, Sghaier, & Maaref, 2011. Their approach is based on the use of wavelet image transform and distributed image compression by sharing the processing of tasks to extend the overall lifetime of the network.

One well-known tool used in the image compression done in the resource-constrained devices is so-called low complexity Discrete Cosine Transform (DCT) (Ahmed, Natarajan, & Rao, 1974; Amutha, 2013; Bayer & Cintra, 2012; Chandra & Chakrabarty, 2001; da Silveira, Oliveira, Bayer, Cintra, & Madanayake, 2017; Kaddachi et al., 2012; Pearlman, Islam, Nagaraj, & Said, 2004; Pennebaker, Mitchell, Langdon, & Arps, 1988), which was initially developed to be used in very resource-constrained platforms, for example low power processors in the WSNs. Certain applications involve real-time (Akter, Reaz, Mohd-Yasin, & Choong, 2008; Callahan, 1995) image compression performed directly on-board (Yu, Vladimirova, & Sweeting, 2009) used in several missions to meet the specialized needs of (for example) deep-space applications while achieving state-of-the-art compression effectiveness.
Since low-complexity DCT is aimed to trade-off between quality of decompressed image and amount of resources needed to perform the DCT, to assess the approximation in a fair manner, the authors consider the ratio between performance measures and arithmetic cost (Figure 1, results are taken from da Silveira et al., 2017).

Implementations of modern video coding standards, such as H.264 (MPEG-4 AVC) (Richardson, 2004; Richardson, 2011) or H.265 (HEVC) (Sullivan, Ohm, Han, & Wiegand, 2012), usually are computationally intensive, because of finding temporal redundancy in the sequence of images by applying motion estimation. In the final products (on top of PCB boards) these video coding standards usually are deployed in the form of the dedicated IP cores to assure the high performance and to offload the main CPU. In such systems usually ‘low latency property’ is reached with the use of multi-core CPU-based architectures (Tan & Barazesh, 2017) or FPGA-based massively parallel architectures. Such video coding algorithms usually target high-resolution images (2, 4, 8 K), because they are designed to be used in multimedia applications (HDTV, video streaming, video conferences, etc.). It should be noted that wearable resource-constrained eye tracking systems do not especially use images of such a high resolution. For example, SuriCog’s EyeDee™ eye tracking wearable system uses images of VGA resolution of 640 × 480 pixels (which are then cropped to 400 × 260 pixels images). In contrast, eye tracking systems focus on high frequency of the entire system (image acquisition, pre-processing, encoding, transmission, decoding, post-processing, eye tracking itself and optional gaze reconstruction). For example, the EyeDee™ supports up to 100 FPS frequency, which results in 10 milliseconds budget for the entire chain. Of these, only 2.5 (approx.) milliseconds are available for the encoding/decoding. Encoding/decoding delays highly depend on the implementation. Even though several literature sources (most fully technical forums) mentioned the fact that it is possible to run a particular H.264/5 codec implementation at 100–150 FPS, we did not find wide mention of running these products at frequencies used in the eye tracking such as 300–800 Hz. It should be also noted that the EyeDee™ is a general-purpose eye tracking system, which targets applications of quite different domains: from multimedia and entertainment to research and military. While more specifically oriented eye tracking

![Figure 1](image-url)  
**Figure 1.** Average (a) PSNR and (b) SSIM measurements per additive cost at compression ratios.
systems usually work on even higher frequencies, for example 300, 500, 800 Hz, which makes high FPS-requirements to the codecs that can be used on these eye tracking systems.

Human eye centre localization (also called eye pupil centre localization and eye iris centre localization) is a relatively challenging task, because of accuracy and robustness issues related mainly to low contrast eye images, noise in the eye images, changing of the external lightning conditions (glares of sun, etc.), eye blinks, etc. In general, eye centre localization techniques differ, because of different goals (performance, accuracy, implementation aspect, final costs). Here we briefly present several eye centre localization approaches.

One set of approaches is based on the use of means of gradients. For example, in Timm & Barth, 2011 authors derive a simple objective function, which only consists of dot products. Then the maximum of this function corresponds to the location where most gradient vectors intersect and thus to the eye’s centre. Another gradient-based approach is presented in Ahuja, Banerjee, Nagar, Dey, & Barbhuiya, 2016, September. Another technique is based on invariant so-called isocentric patterns (Valenti & Gevers, 2012), which uses of so-called isophote properties to gain invariance to linear lighting changes (contrast and brightness), to achieve in plane rotational invariance, and to keep low computational costs. Several approaches are based on the use of so-called deformable templates (Valenti, Yucel, & Gevers, 2009, June) or use the spherical model of the human eyeball model (Baek, Choi, Ma, Kim, & Ko, 2013) to perform the iris centre localization.

It should be noted that SuriCog’s EyeDee™ eye tracking wearable system uses human eye centre localization, which is based on (1) extraction of the useful eye tracking features (pupil contours) by eye image filtering with the goal to preserve edges of the pupil, (2) fitting of the pupil ellipse shape on the base of the pupil edges, (3) finding the centre of this pupil ellipse.

**Eyedee™ eye tracking solution**

In (Morozkin, Swynghedauw, & Trocan, 2017, September) the EyeDee™ (Figure 2) embedded eye tracking solution developed by SuriCog was introduced. The actual problem is the deployment of computationally intensive image processing-based eye tracking algorithms on a resource-constrained embedded platform, based on an MCU (Microcontroller Unit) and an FPGA (Field-Programmable Gate Array).

In the case of SuriCog’s EyeDee™ solution, the embedded system should be able to capture at high frequency (100 Hz) the grayscale 8bpp (bits-per-pixel) image of user’s eye of VGA (Video Graphics Array) resolution of $640 \times 480$ pixels and broadcast wirelessly in real time to the end application the result of the image processing-based eye tracking algorithm, which consists in the parametrization of a 3D model of the eye. The system should run continuously during more than 3 h, with the lowest latency possible (typically <10 ms).

To reproduce eye images needed for the research we have developed a simulator (Figure 3), which generates eye images closed to real ones (Figure 4). We simulate an eye of known geometry and a camera sensor of known resolution, focal and distortion. The real pupil is a disc in rotation. The image of the pupil viewed by the sensor is the
perspective projection of the refraction of the real pupil through the cornea of a known index of refraction.

The noise present in real images is simulated using a Gaussian noise (Luisier, Blu, & Unser, 2011) (based on normal/Gaussian distribution) or Poisson noise (Luisier et al., 2011) (also known as Shot noise and based on Poisson distribution) (see Listing 1 in Annex).

Herein and after we use the following terminology (Figure 5):

(a) Full size image – original eye image (resolution 640 × 640, 8bpp) taken from miniaturized camera sensor;
(b) Static ROI (Region of Interest) – cropped region (resolution 400 × 260, 8bpp) containing image of the eye; finding of the dynamic ROI I done on top of the static ROI;
(c) Dynamic ROI – region (resolution 120 × 120, 8bpp) containing image of the human’s pupil; because term ‘static ROI’ is relatively rarely referred in this work, term ‘dynamic ROI’ is assumed under term ‘ROI’;

Figure 2. EyeDee™ eye tracking solution: Weetsy™ frame, Weetsy™ pre-processing board, π-Box™ remote smart sensor and processing unit (running EyeDee™ software).

Figure 3. Ellipse reconstruction on simulator-based eye image.
Eye image compression classical approaches

The classical eye image compression approaches are based on well-known image compression systems such as JPEG (Wallace, 1991) and JPEG2000 (Skodras, Christopoulos, & Ebrahimi, 2001). Due to low performance of their software implementations these systems are usually implemented directly in the hardware: either in the form of an IP (Intellectual Property) core inside an MCU or in the form of a dedicated IP core running inside an FPGA.

For JPEG codec compression of dynamic ROI image (contains eye pupil) gives 0.50–0.55 bpp (Figure 6(a)) for PSNR-quality up to 36–38 dB (which is enough for the eye tracking), which gives compression ratio 14–16. Meaning that if the size of the ROI image is 14,400 bytes (120 × 120), then JPEG-compressed ROI has about 900 bytes to transmit over the wireless media. To achieve this ratio the 'JPEG quality' input parameter of the JPEG encoder can be set to 40 (Figure 6(b)). The described configuration practically results in 60 ... 70 FPS (Frames Per Second, exact number varies due to variable JPEG compressed data size).

The core idea is that the ROI is less quantized (results in higher quality of decompressed image), while the rest of the image is more quantized (results in lower quality of...
decompressed image). For example, JPEG2000 provides this feature (Bradley & Stentiford, 2002, January). The ROI can be selected manually or automatically. For automatic ROI selection (Wang, Wei, Zheng, Du, & Gao, 2007, September) it is possible to apply the SIFT (Lindeberg, 2012) for feature extraction, descriptor generation and point matching to locate the ROI. According to visual comparison of quality of static ROI images (Figure 7), compressed with JPEG 2000 with deferent bitrates, values less than 0.9 produce results with sufficiently degraded quality (11 dB). Based on this comparison, it also can be seen that quality of ROI borders is degraded as lower bpp was used. However, quality of centre of ROI is stayed acceptable. Therefore, to consider low bitrates (0.1–0.3), the size of ROI could be increased from 120×120 to 180×180 (for example) to keep needed quality of edges of the pupil. This is especially important due to sensitivity of eye tracking algorithms to quality of the source (i.e. decompressed) image.

There are also several works that aimed at introducing the ROI selection into the standard baseline JPEG (Varma & Bagadi, 2014) as well as in JPEG XR (Dufaux, Sullivan, & Ebrahimi, 2009), which is a new still image compression technique approved by the JPEG committee, adopted as a standard by ITU-T and aimed to reach the speed of JPEG and the quality of JPEG2000. Several efforts are aimed at adding the ROI selection into

![Figure 6. JPEG codec assessment results: JPEG quality vs. bpp vs. PSNR.](image)

![Figure 7. Visual comparison of quality of static ROI images with selected ROI.](image)
video coding standards, such as H.264 (MPEG-4 AVC) (Ferreira, Cruz, & Assunção, 2008; Van Leuven, Van Schevensteen, Dams, & Schelkens, 2008) and H.265 (HEVC) (Patel & Rao, 2015, November). It should be noted (to avoid misunderstanding) that SuriCog's EyeDee™ eye tracking solution does not involve ROI-selection-based compression of the eye images. This type of compression was not yet tested, but can be considered for the further usage. Instead EyeDee™ is based on finding dynamic ROI (on top of the static ROI) following by transmission of this dynamic ROI either in uncompressed form, compressed with standard codecs (JPEG/JPEG2000) or compressed with proposed in this paper NN-based approaches aimed on finding of the FOI on top of the ROI and compression of the FOI with standard codecs (JPEG/JPEG2000) or their optimized versions.

We selected JPEG2000 because its outperforms JPEG, has open source implementation, supports ROI selection feature, and allows precisely control real bpp. We use OpenJPEG (OpenJPEG, open-source JPEG, 2000 codec) (used in this research) and Kakadu (Kakadu, closed-source JPEG, 2000 codec), because it allows to more precisely control ‘real bpp’. We also tested several non-standard wavelets by replacing standard CDF97 (Cohen-Daubechies-Feauveau CDF 9/7 wavelet GGuangjun, Lizhi, & Huowang, 2001, October, also called ‘JPEG97’) and CDF53 (also called the LeGall 5/3 wavelet Le Gall & Tabatabai, 1988, April) wavelets by other biorthogonal wavelets from Cohen-Daubechies-Feauveau family. We found that standard wavelets are still the best in most cases (except very low bitrates such as 0.15). For the eye tracking system, the quality of decompressed eye image can be lower, as proved in Morozkin, Swynghedauw, & Trocan, 2016. To enable several users to use eye tracking system with acceptable precision, size of ROI can be also increased. This leads to changing of bpp. For example, if 0.20 bpp was used to compress static ROI with selected ROI of size 120 × 120 (leads to 36 dB, Table 1), then in case of ROI of size 180 × 180, bpp has to be increased to 0.40 (leads to 37 dB, Table 1).

Since PSNR can’t be the best metric all the time, visual comparison of images is an acceptable practice (Figure 8). For compression of ROI images (with the size of ROI 220 × 220), it is possible to get satisfied results even with low bitrates such as 0.2.

On comparison of JPEG 2000 codec vs. JPEG codec and also FLIF codec (Free Lossless Image Format (FLIF)) (Figure 9) JPEG 2000 codec produces better results in all resolutions at bitrates, less than 1.5 (FULL size image and static ROI), 1.25 (ROI image).

Eye image compression alternative approaches

Machine Learning (ML) (Cortes & Vapnik, 1995; Samuel, 1959) is a field of computer science that gives computers the ability to learn without being explicitly programmed. In this definition term 'learning' refers to a task of inferring a function from labelled training data. The training data consist of a set of training examples. Machine learning approaches are divided into two main types:

(a) Supervised learning (Moller, 1993) – each training sample is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). Such sample is also called a ‘labelled’ sample.

(b) Unsupervised learning (Hastie, Tibshirani, & Friedman, 2009) – each training sample is only an input object (typically a vector), i.e. without a desired output value. Such sample is also called an ‘unlabelled’ sample.
| Res. | bpp 0.10 | 0.20 | 0.30 | 0.40 | 0.50 | 0.60 | 0.70 | 0.80 | 0.90 | 1.00 | 1.10 | 1.20 | 1.30 | 1.40 |
|------|----------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 120 × 120 | 30.44 | 36.08 | 39.49 | 42.24 | 44.78 | 47.93 | 50.16 | 57.55 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 140 × 140 | 30.22 | 35.31 | 38.12 | 40.14 | 42.10 | 44.14 | 45.73 | 48.03 | 49.79 | 53.87 | 63.29 | 64.29 | 65.27 | 68.45 |
| 160 × 160 | 30.00 | 34.57 | 36.91 | 38.74 | 40.13 | 41.72 | 43.60 | 44.70 | 45.95 | 47.83 | 49.39 | 50.58 | 56.88 | 100.00 |
| 180 × 180 | 29.75 | 33.74 | 35.70 | 37.05 | 38.49 | 39.65 | 40.86 | 42.02 | 43.51 | 44.49 | 45.43 | 46.61 | 48.24 | 49.29 |
| 200 × 200 | 29.59 | 33.08 | 34.72 | 35.93 | 37.12 | 38.24 | 39.19 | 40.17 | 41.10 | 42.13 | 43.52 | 44.20 | 44.93 | 45.91 |
| 220 × 220 | 29.30 | 32.07 | 33.67 | 34.81 | 35.63 | 36.63 | 37.51 | 38.51 | 39.15 | 39.97 | 41.00 | 41.81 | 42.58 | 43.74 |
Artificial Neural Networks (ANNs) (Hassoun, 1995), also known as Connectionist Models (Hanson & Burr, 1990) or Parallel Distributed Processing (Rumelhart, McClelland, & Research Group, 1987) are models that are inspired by human brains for simulation of their biological equivalent. The behaviour of a biological neuron is modelled via a set of mathematical operations. In particular, to obtain an output signal the following operations are performed on input signals: weighting, summing and thresholding. Then these output signals act as an input for other neurons, which creates a network. By processing a set of input samples (training set), neural networks may adapt themselves (find optimal weight values) to recognize patterns or to classify data. Their ability to extract hidden correlations between patterns makes them a powerful tool to recognize patterns.

In Morozkin, Swynghedauw, & Trocan, 2018, March we proposed a deep learning method based on Artificial Neural Network (ANN) used as a function regression calculator. It consists in tuning the hyperparameters (Bergstra & Bengio, 2012; Bergstra, Bardenet, Figure 8. Visual comparison of quality of ROI images (220 × 220).

Figure 9. Comparison of image compression systems.
Bengio, & Kégl, 2011; Snoek, Larochelle, & Adams, 2012) of this network to relate the image of the pupils (inputs) to the 5 parameters of a geometrical ellipse (outputs). The approach we propose in this paper is to use an ANN-based deep learning method as a classifier. We train the ANN to learn the relevant areas of an image (i.e. the pupil edges) and irrelevant areas of an image (rest of the image). We then use the output of ANN classification followed by image data compression and its transmission in compressed form for further processing. In the previous publication (Morozkin et al., 2018, March), we presented the classical eye image approach and the NN-based eye image compression approach, neural network construction and training as well as testing framework. It should be noted that based on the neural network (Demuth, Beale, De Jess, & Hagan, 2014) the ROI image blocks are classified into 2 classes (blocks contain/does not contain pupil edges). To this purpose, Torch7 (Torch framework ) software (neural network ‘nn’ and optimization ‘optim’ packages) is used with ‘convnet’ (convolutions+2-layer mlp), where 2-layer mlp is multilayer perceptron, and ‘2-layer mlp’ (pure 2-layer mlp) models. This functionality was fully integrated into the EyeDee™ eye tracking software running on Windows x64 platform.

**Experimental results**

During testing we used two NN models: ‘2-layer mlp’ model (Table 2) and ‘convnet’ model (Table 3). Both models were trained on training dataset and tested on test dataset (to assess the performance, i.e. generalization property of the trained NN). Since, in this paper, we did not focus on the tuning of the hyperparameters of the NN, we did not use validation dataset.

As for the training dataset it is possible to use two modes: (1) train NN on the eye images generated by the simulator and (2) train NN on the real eye images from the

| Mode                  | Block Size | Epochs | e %  | p %  | ROI      | FOI | FOI (reordered) | Bits per pixel (bpp) |
|-----------------------|------------|--------|------|------|----------|-----|----------------|----------------------|
| **Training dataset**  |            |        |      |      |          |     |                |                      |
| Trained on video,     | 20         | 50     | 97.85| 1.24 | 2.15     | 1.33| (38.14%)       | 0.88 (59.07%)         |
| trained on video      | 100        | 50     | 98.89| 0.46 | 2.15     | 1.33| (38.14%)       | 0.89 (58.60%)         |
|                       | 200        | 50     | 99.19| 0.48 | 2.15     | 1.33| (38.14%)       | 0.89 (58.60%)         |
|                       | 400        | 50     | 99.58| 0.49 | 2.15     | 1.33| (38.14%)       | 0.89 (58.60%)         |
|                       | 10         | 50     | 70.64| 20.12| 2.15     | 1.06| (50.70%)       | 0.50 (76.74%)         |
|                       | 100        | 50     | 70.87| 17.54| 2.15     | 1.04| (51.63%)       | 0.49 (77.21%)         |
|                       | 200        | 50     | 76.69| 14.06| 2.15     | 1.07| (50.23%)       | 0.52 (75.81%)         |
|                       | 400        | 50     | 79.08| 20.33| 2.15     | 1.35| (37.20%)       | 0.90 (58.17%)         |
| **Validation dataset**|            |        |      |      |          |     |                |                      |
| Trained on simulator, | 20         | 50     | 98.89| 0.46 | 2.15     | 1.33| (38.14%)       | 0.89 (58.60%)         |
| tested on video       | 100        | 50     | 99.19| 0.48 | 2.15     | 1.33| (38.14%)       | 0.89 (58.60%)         |
|                       | 200        | 50     | 99.58| 0.49 | 2.15     | 1.33| (38.14%)       | 0.89 (58.60%)         |
|                       | 400        | 50     | 70.64| 20.12| 2.15     | 1.06| (50.70%)       | 0.50 (76.74%)         |
|                       | 10         | 50     | 70.87| 17.54| 2.15     | 1.04| (51.63%)       | 0.49 (77.21%)         |
|                       | 200        | 50     | 76.69| 14.06| 2.15     | 1.07| (50.23%)       | 0.52 (75.81%)         |
|                       | 400        | 50     | 79.08| 20.33| 2.15     | 1.35| (37.20%)       | 0.90 (58.17%)         |

**Table 2.** Average results of ROI image block classification quality (expressed as efficiency/purity) and average bit budget (expressed as bits per pixel) needed to transmit the compressed ROI images with extra blocks removed. Use of ‘2-layer mlp’ model.
video file (these eye images are obtained from the camera sensor installed in the Weetsy™ Frame). As for the test dataset it is possible to use two modes: (1) test NN on the eye images generated by the simulator and (2) test NN on the real eye images from the same video file or from the different video file (or set of video files stored in the special database). It should be noted that since training and testing on the simulator-based eye images does not produce interesting results (in this case NN classifies ROI blocks almost perfectly), we omit showing these results. Secondly, there is also not much interest to train NN on the video and test on the simulator. Lastly, since, in this paper, we do not focus on the improving of the generalization property of the trained NN, i.e. a case, when the NN was trained on the simulator-generated eye images and tested on the set of video files (real images) we do not yet provide the results and consider them as a future work. As a result, for both used NN models we present results of two modes: (1) for training dataset: train NN on the one video file and then test on the same video file (testing quality of the training) and (2) for test dataset: train NN on the simulator and test on the video file (quick preliminary testing of the generalization property).

According to the application results (Tables 2–3), both ‘convnet’ and ‘2-layer mlp’ models show relatively promising results. In particular, increasing the number of training iterations (from 50 to 400) results in higher classification efficiency and lower classification purity. This is because the number of N00 and N11 is higher meaning that the model is trained better. For example, using ‘convnet’ model with ROI blocks of size 10 × 10 at 100 training iterations is enough to reach 100% of both N00 and N01 and, therefore, to classify ROI blocks perfectly. Using ROI image blocks of size 20 produce better results in comparison with blocks of size 10. This can be explained by the fact that in the case of ROI image block size of 20 pixels, there are 36 blocks to classify in the ROI image, while in the case of ROI image block size of 10 pixels, there are 144 blocks to classify (75%
more). Therefore, the probability of classification errors (missing blocks with pupil edges, extra blocks without pupil edges) is much higher.

As for the bpp results we used JPEG and JPEG2000 codecs (in presented results JPEG codes were used with JPEG-quality set to 75) to compare the benefit of compressing on the FOI (optionally reordered) instead on the ROI. Reordered FOI means that the blocks of the FOI with useful features were taken out and put into the new FOI image with the side information about original position of these ROI blocks. Thus, if the ROI block size is 10 (for example) and if after the NN classification the FOI contains 25 blocks, then the reordered FOI has the size of $10 \times 250$ pixels. As we can see form the table that when the ‘2-layer mlp’ model is used, the typical gains of the FOI vs. ROI and FOI (reordered) vs. FOI are 35–38% and 56–58% for the ROI blocks of size 20 pixels and 40–50% and 70–76% for the ROI blocks of size 20 pixels. As for the ‘convnet’ model these values are 33–38% and 54–60%, and 38–49% and 69–75%.

For the final set-up involving the NN-based approach (feature-based ROI image compression) we used JPEG encoder because it works faster than JPEG2000. We configured the JPEG encoder to keep JPEG quality at 75 (max. value is 100), which allows to keep high PSNR (>36-38 dB), because the decompressed ROI blocks will be further used in the image processing-based eye tracking algorithm. As it can be seen in Figure 10 with the proposed approach it is possible to reach up to 99.70% of gain in terms of data size reduction even with standard baseline JPEG codec. To these results we should add the same note as to previous results: for the compression algorithms for the eye tracking system, the term ‘frame’ has more priority over the term ‘pixel’, because the final performance is measured in the FPS (Frames per Second). This is the reason for calculating the ‘size’ – size of compressed frame and ‘Mbps’ – bit rate at 100 FPS (Figure 10).

Generally professional eye tracking systems are commercial products. Usually their eye tracking algorithms are based on different principles and, moreover, are implemented in a

![Figure 10.](https://example.com/image.png)

**Figure 10.** Eye image compression configurations. Static ROI image resolution: $400 \times 260$, ROI image resolution: $120 \times 120$, compressor: JPEG (with JPEG quality set to 75), bpp – bits per pixel, size – size of compressed frame, FOI – Features of Interest, Mbps – bit rate at 100 FPS.
quite different way: starting from algorithmic part and ending in the concrete implementation. The eye tracking algorithm can be implemented 100% in the PC (like EyeBrain T2 eyeBrain T2 eye-tracker for diagnosis of the main neurological diseases and learning disabilities by SuriCog does) or split between embedded device and PC (like Weetsy™ eye tracking system by SuriCog, where 30% approx. of the eye tracking algorithm is done inside the head-mounted Weetsy™ pre-processing board and the rest 70% is done inside PC via eye tracking software), or even more, it can be 100% implemented inside the ASIC-chip (like EyeChip™ by Tobii, Inc. EyeChipTM or AEye Chip by EyeTech Digital Systems, Inc. AEye Chip). Because of intellectual property issues, it is difficult to know if these eye tracking products use image compression system at all, which compression system is used exactly, what is the exact configuration if this compression system and the low-level details of its implementation and deployment in the product. However, we can assume most of the eye tracking systems (both classes: remote and wearable) do not use any eye image compression, because these eye tracking systems usually execute eye tracking algorithms locally in the device and transmit to the target user application ready-to-use eye tracking results. At the time of writing we could not find on the market any wearable eye tracking system, in which the eye tracking processing is split between the external embedded device and the PC-based processing unit, and, therefore, in which image compression can drastically reduce the size of image data and hence increase the final system performance.

Conclusion

In this paper firstly, we describe peculiarities of introducing the image compression system into resource-constrained embedded devices. We show that generally used video compression standards are rarely used in such devices, because implementations of such standards are computationally intensive due to elimination of temporal redundancy from the input image data. Instead, resource-constrained embedded devices integrate specific implementations of image compression systems involving usage of approximated versions of original algorithms. Such approximated versions result in low computational complexity as well as in low power consumption.

Secondly, for SuriCog’s EyeDee™ solution including Weetsy™ pre-processing board, we propose a new ROI eye image compression approach based on ROI image block classification. A neural network was implemented with Torch7 software, which was integrated into EyeDee™ software. Two different models, ‘convnet’ (convolutions+2-layer mlp) and ‘2-layer mlp’ (pure 2-layer mlp), have been used in order to train the neural network, both with generated and real ROI images. It has been shown that the classification quality is high for these training models. The proposed compression method has a maximal bitrate gain of ~99% at the only expense of memory needed to store the trained neural network. Future work is targeted on implementation of the approach in the Weetsy™ pre-processing board as well as its generalization for usage in a wider class of imagery applications.

Disclosure statement

No potential conflict of interest was reported by the authors.
Notes on contributors

Pavel Morozkin received the Ph.D. degree in Computer Science, Telecommunication and Electronics from the University Pierre and Marie Curie, Paris, France, in 2018. His current research interests include image and video compression in embedded systems.

Marc Swynghedauw received the Ph.D. degree in High Energy Physics from the University of Paris XI Orsay, Orsay, France and LLR Ecole Polytechnique / CERN, Geneva Area, Switzerland, in 1999. His current research interests include spiking neural networks, neuromorphic computing and event-driven parallel computations.

Maria Trocan (M’09–SM’13) received the Ph.D. degree in Signal and Image Processing from the Telecom Paris-Tech, Paris, France, in 2007, and the Habilitation degree from the Pierre and Marie Curie University, Paris, France, in 2014. Since May 2009, she has been an Associate Professor with the Department of Signal, Image and Telecommunications, Institut Superieur d’Electronique de Paris, Paris, France. Her current research interests include image and video analysis, and compression and sparse signal representations.

References

AEye Chip. BoardLevelOEM – eye tracking technology for OEMs. Retrieved from https://www.eyetechds.com/boardleveloem.html
Ahmed, N., Natarajan, T., & Rao, K. R. (1974). Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1), 90–93.
Ahuja, K., Banerjee, R., Nagar, S., Dey, K., & Barbhuiya, F. (2016, September). *Eye center localization and detection using radial mapping*. 2016 IEEE International Conference on image processing (ICIP) (pp. 3121–3125). Phoenix, Arizona: IEEE.
Akter, M., Reaz, M. B. I., Mohd-Yasin, F., & Choong, F. (2008). A modified-set partitioning in hierarchical trees algorithm for real-time image compression. *Journal of Communications Technology and Electronics*, 53(6), 642–650.
Akyildiz, I. F., Melodia, T., & Chowdhury, K. R. (2007). A survey on wireless multimedia sensor networks. *Computer Networks*, 51(4), 921–960.
Amutha, R. (2013). Low complexity energy efficient very low bit-rate image compression scheme for wireless sensor network. *Information Processing Letters*, 113(18), 672–676.
Baek, S. J., Choi, K. A., Ma, C., Kim, Y. H., & Ko, S. J. (2013). Eyeball model-based iris center localization for visible image-based eye-gaze tracking systems. *IEEE Transactions on Consumer Electronics*, 59(2), 415–421.
Bayer, F. M., & Cintra, R. J. (2012). DCT-like transform for image compression requires 14 additions only. *Electronics Letters*, 48(15), 919–921.
Bergstra, J. S., Bardenet, R., Bengio, Y., & Kégl, B. (2011). *Algorithms for hyper-parameter optimization*. Advances in Neural Information Processing Systems (pp. 2546–2554). Granada, Spain: Curran Associates.
Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb), 281–305.
Bradley, A. P., & Stentiford, F. W. (2002, January). *JPEG 2000 and region of interest coding*. Digital image Computing techniques and applications (Vol. 2, pp. 1–6). Melbourne: Australian Pattern Recognition Society (APRS).
Callahan, S. M. (1995). *U.S. Patent No. 5,408,542*. Washington, DC: U.S. Patent and Trademark Office.
Case, M. A., Burwick, H. A., Volpp, K. G., & Patel, M. S. (2015). Accuracy of smartphone applications and wearable devices for tracking physical activity data. *Jama*, 313(6), 625–626.
Chandra, A., & Chakrabarty, K. (2001). System-on-a-chip test-data compression and decompression architectures based on Golomb codes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(3), 355–368.
Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297.
da Silveira, T. L., Oliveira, R. S., Bayer, F. M., Cintra, R. J., & Madanayake, A. (2017). Multiplierless 16-point DCT approximation for low-complexity image and video coding. *Signal, Image and Video Processing, 11*(2), 227–233.

Demuth, H. B., Beale, M. H., De Jess, O., & Hagan, M. T. (2014). *Neural network design*. Martin Hagan.

Duchowski, A. T. (2007). Eye tracking methodology. *Theory and Practice, 328*, 51–59.

Dufaux, F., Sullivan, G. J., & Ebrahimi, T. (2009). The JPEG XR image coding standard [standards in a Nutshell]. *IEEE Signal Processing Magazine, 26*(6), 195–204.

EyeBrain T2 eye-tracker for diagnosis of the main neurological diseases and learning disabilities. Retrieved from http://www.eyebrain.fr

EyeChip™. Components & System Design for IS4, Tobii Tech. Retrieved from https://www.tobii.com/tech/products/components-system-design

Ferreira, L., Cruz, L., & Assunção, P. A. (2008). H. 264/SVC ROI encoding with spatial scalability. International Conference on signal processing and multimedia applications. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10

Free Lossless Image Format (FLIF). www.flif.info

Guangjun, Z., Lizhi, C., & Huowang, C. (2001, October). *A simple 9/7-tap wavelet filter based on lifting scheme*. Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205), Thessaloniki, Greece (Vol. 2, pp. 249–252). IEEE. doi:10.1109/ICIP.2001.958471

Hanson, S. J., & Burr, D. J. (1990). What connectionist models learn: Learning and representation in connectionist networks. *Behavioral and Brain Sciences, 13*(3), 471–489.

Hassoun, M. H. (1995). *Fundamentals of artificial neural networks*. Cambridge: MIT Press.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Unsupervised learning. In *The elements of statistical learning* (pp. 485–585). New York, NY: Springer.

Hutton, D. M. (2005). Smart environments: Technology, protocols and applications. *Kybernetes, 34*(6), 903–904. doi:10.1108/03684920510595580

Kaddachi, M. L., Soudani, A., Lecuire, V., Makkaoui, L., Moureaux, J. M., & Torki, K. (2012). Design and performance analysis of a zonal DCT-based image encoder for wireless camera sensor networks. *Microelectronics Journal, 43*(11), 809–817.

Kakadu, closed-source JPEG 2000 codec. Retrieved from http://kakadusoftware.com

Khosrow-Pour, M. (Ed.). (2005). *Encyclopedia of information science and technology*. Hershey, PA: IGI Global.

Lee, D. U., Kim, H., Rahimi, M., Estrin, D., & Villasenor, J. D. (2009). Energy-efficient image compression for resource-constrained platforms. *IEEE Transactions on Image Processing, 18*(9), 2100–2113.

Le Gall, D., & Tabatabai, A. (1988, April). *Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques*. 1988 International Conference on Acoustics, Speech, and Signal Processing, 1988. ICASSP-88. (Vol. 2, pp. 761–764). IEEE. doi:10.1109/ICASSP.1988.196696

Lindeberg, T. (2012). Scale invariant feature transform.

Liu, Z., Liu, T., Wen, W., Jiang, L., Xu, J., Wang, Y., & Quan, G. (2018). DeepN-JPEG: A deep neural network favorable JPEG-based image compression framework. Proceedings of the 55th Annual Design Automation Conference, DAC 2018 (Vol. Part F137710). [a18] Institute of Electrical and Electronics Engineers Inc. doi:10.1145/3195970.3196022

Luisier, F., Blu, T., & Unser, M. (2011). Image denoising in mixed Poisson–Gaussian noise. *IEEE Transactions on Image Processing, 20*(3), 696–708.

Ma, T., Hempel, M., Peng, D., & Sharif, H. (2013). A survey of energy-efficient compression and communication techniques for multimedia in resource constrained systems. *IEEE Communications Surveys & Tutorials, 15*(3), 963–972.

Möller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks, 6*(4), 525–533.

Morozkin, P., Swynghedauw, M., & Trocan, M. (2016). *Image quality impact for eye tracking systems accuracy*. 2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS), Monte Carlo, pp. 429–431. doi:10.1109/ICECS.2016.7841226

Morozkin, P., Swynghedauw, M., & Trocan, M. (2017, September). Neural network based eye tracking. In *Conference on computational Collective Intelligence Technologies and applications* (pp. 600–609). Cham: Springer.
Morozkin, P., Swynghedauw, M., & Trocan, M. (2018, March). Feature-based image compression. In Asian conference on intelligent information and database systems (pp. 454–465). Cham: Springer.

Nasri, M., Helali, A., Sghaier, H., & Maaref, H. (2011). Adaptive image compression technique for wireless sensor networks. Computers & Electrical Engineering, 37(5), 798–810.

OpenJPEG, open-source JPEG 2000 codec. Retrieved from http://www.openjpeg.org

Patel, Z., & Rao, K. R. (2015). Image segmentation approach for realizing zoomable streaming HEVC video. 2015 International Conference on Science and Technology (TICST), Pathum Thani, pp. 76–82. doi:10.1109/TICST.2015.7369344

Pearlman, W. A., Islam, A., Nagaraj, N., & Said, A. (2004). Efficient, low-complexity image coding with a set-partitioning embedded block coder. IEEE Transactions on Circuits and Systems for Video Technology, 14(11), 1219–1235.

Pennebaker, W. B., Mitchell, J. L., Langdon, G. G., & Arps, R. B. (1988). An overview of the basic principles of the Q-coder adaptive binary arithmetic coder. IBM Journal of Research and Development, 32(6), 717–726.

Richardson, I. E. (2004). H. 264 and MPEG-4 video compression: Video coding for next-generation multimedia. Chichester: Wiley.

Richardson, I. E. (2011). The H. 264 advanced video compression standard. Hoboken : Wiley.

Rumelhart, D. E., McClelland, J. L., & Research Group, P. D. P. (1987). Parallel distributed processing (Vol. 1). Cambridge, MA: MIT press.

Sadler, C. M., & Martonosi, M. (2006). Data compression algorithms for energy-constrained devices in delay tolerant networks. Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys ’06, Boulder, Colorado (pp. 265–278). New York, NY: ACM.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, 3(3), 210–229.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85–117.

Skodras, A., Christopoulos, C., & Ebrahimi, T. (2001). The JPEG 2000 still image compression standard. IEEE Signal Processing Magazine, 18(5), 36–58.

Timm, F., & Barth, E. (2011). Accurate Eye Centre Localisation by means of gradients. Visapp, 11, 125–130.

Torch framework. www.torch.ch

Van Leuven, S., Van Schevensteen, K., Dams, T., & Schelkens, P. (2008). An implementation of multiple region-of-interest models in H. 264/AVC. In . In signal processing for image Enhancement and multimedia processing (pp. 215–225). Boston, MA: Springer.

Wallace, G. K. (1991). The JPEG still picture compression standard. Communications of the ACM, 34(4), 30–44.

Wang, F., Wei, J., Zheng, N., Du, S., & Gao, B. (2007). Automatic ROI selection for JPEG2000 compression of remote sensing images. International Conference on Semantic Computing (ICSC 2007), pp. 615–621.
Wu, H., & Abouzeid, A. A. (2005). Energy efficient distributed image compression in resource-constrained multihop wireless networks. *Computer Communications, 28*(14), 1658–1668.

Xia, F., Yang, L. T., Wang, L., & Vinel, A. (2012). Internet of things. *International Journal of Communication Systems, 25*(9), 1101–1102.

Yu, G., Vladimirova, T., & Sweeting, M. N. (2009). Image compression systems on board satellites. *Acta Astronautica, 64*(9–10), 988–1005.

ZainEldin, H., Elhosseini, M. A., & Ali, H. A. (2015). Image compression algorithms in wireless multimedia sensor networks: A survey. *Ain Shams Engineering Journal, 6*(2), 481–490.

### Appendix

```cpp
1 std::random_device rd;
2 gen = std::mt19937(rd());
3 m_bitResol = 255;
4 if (poissonNoise) {
5    std::poisson_distribution<> p(input/m_noise+1);
6    float value = ((float) p(gen)) * m_noise - 1.;
7    output = static_cast<unsigned char>(MAX(0, MIN(m_bitResol, value)));
8 } else {
9    std::normal_distribution<> d(0, m_noise);
10   float value = m_bitResol,v + (m_bitResol-v) *d(gen)/100.;
11   output = static_cast<unsigned char>(MAX(0, MIN(value)));
12 }

Legend:
input - original image pixel value
output - modified image pixel value (pixel with noise)
m_noise - noise coefficient (input parameter)
```

**Listing 1.** Gaussian noise and Poisson noise implementation (C++).