Design and implementation of image processing and compression algorithms for a miniature embedded eye tracking system

Pavel Morozkin

Sorbonne Université

Institut supérieur d'électronique de Paris (ISEP)

École doctorale : « Informatique, télécommunications & électronique de Paris »

Design and Implementation of Image Processing and Compression Algorithms for a Miniature Embedded Eye Tracking System

Par Pavel MOROZKIN

Thèse de doctorat de Traitement du signal et de l'image

Présentée et soutenue publiquement le 29 juin 2018

Devant le jury composé de :

M.	Ales PROCHAZKA	Rapporteur
M.	François-Xavier COUDOUX	Rapporteur
M.	Basarab MATEI	Examinateur
M.	Habib MEHREZ	Examinateur
M^{me}	Maria TROCAN	Directrice de thèse
M.	Marc Winoc SWYNGHEDAUW	Encadrant

Abstract

Human-Machine Interaction (HMI) progressively becomes a part of coming future. Being an example of HMI, embedded eye tracking systems allow user to interact with objects placed in a known environment by using natural eye movements.

The EyeDee™ portable eye tracking solution (developed by SuriCog) is an example of an HMI-based product, which includes Weetsy™ portable wire/wireless system (including Weetsy™ frame and Weetsy™ board), π-Box™ remote smart sensor and PC-based processing unit running SuriDev eye/head tracking and gaze estimation software, delivering its result in real-time to a client's application through SuriSDK (Software Development Kit).

Due to wearable form factor developed eye-tracking system must conform to certain constraints, where the most important are low power consumption, low heat generation low electromagnetic radiation, low MIPS (Million Instructions per Second), as well as support wireless eye data transmission and be space efficient in general. Eye image acquisition, finding of the eye pupil ROI (Region Of Interest), compression of ROI and its wireless transmission in compressed form over a medium are very beginning steps of the entire eye tracking algorithm targeted on finding coordinates of human eye pupil. Therefore, it is necessary to reach the highest performance possible at each step in the entire chain.

Applying of image compression allows to drastically reduce amount of data to be sent to a remotely located PC-based processing unit performing eye tracking. However, to reach maximal performance image compression algorithms must be executed at the same frequency as eye image acquisition frequency (100 Hz minimum). Such a high frequency makes a certain constraints on several compression system parameters as well as compressed eye images, where the major ones are size of compressed eye image, quality of decompressed eye image, time needed to perform compression/decompression, computational complexity, operating memory requirements and some others.

In contrast with state-of-the-art general-purpose image compression systems, it is possible to construct an entire new eye tracking application-specific image processing and compression methods, approaches and algorithms, design and implementation of which are the goal of this thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

HMI	Human-Machine Interaction
HCI	Human-Computer Interaction
JPEG	Joint Photographic Experts Group
JPEG 2000	Joint Photographic Experts Group 2000
MPEG	Moving Picture Experts Group
HVS	Human Visual System
ROI	Region of Interest
FOI	Features of Interest
ANN	Artificial Neural Network
MLP	Multilayer Perceptron
CNN	Convolutional Neural Network
ML	Machine Learning
CMOS	Complementary Metal-Oxide-Semiconductor
LED	Light-Emitting Diode
MIPS	Million Instructions per Second
PSNR	Peak Signal-to-Noise Ratio
CODEC (codec)	Coder-Decoder
CRC	Cyclic Redundancy Check
Mbps	Megabit per second 1 Megabit = 1e6 bits
MBps	Megabyte per second 1 Megabyte = 1e6 bytes
BPP (bpp)	Bits Per Pixel
dB	Decibel
GOF	Group of Frames
SLE	System of Nonlinear Equations
JCU	JPEG Codec Unit
DoF	Degrees of Freedom
RDO	Rate-Distortion Optimization
MCU	Microcontroller Unit
FPGA	Field-Programmable Gate Array
RAM	Random-Access Memory
IC	Integrated Circuit
DCT	Discrete Cosine Transform
ET	Eye Tracking
HT	Head Tracking
FPU	Floating Point Unit
PCCR	Pupil Center Corneal Reflection

Chapter 1

Introduction

To understand the impact of human's vision during daily activities it is common practice nowadays to study natural eye movements obtained via eye tracking [1] solutions, which are implemented as a Human-Machine/Computer Interaction (HMI/HCI) [2] devices, which collect and exchange data with some processing units. One of SuriCog's flagship product is EyeDee™ (Figure 1) portable HMI-based eye tracking solution, which is the world's first solution using the eye as a real-time mobile digital cursor. The EyeDee™ targets industry grade applications (e.g., maintenance), multimedia applications (e.g., interaction with objects placed in a known environment), decision critical applications (e.g., control centers), ergonomic assessment [3] and also training applications (e.g., cockpit of an aircraft or a helicopter). The EyeDee™ maintains full mobility, which results in total freedom of user's movements allowing user to interact with objects placed in a known environment. The solution consists of Weetsy™ portable wire/wireless system including Weetsy™ frame (performs eye image acquisition), Weetsy™ board (performs pre-processing), π-Box™ remote smart sensor (performs head localization) and PC-based processing unit (performs post-processing) running SuriDev eye/head tracking and gaze estimation software, which provides its results via SuriSDK™ software to application side EyeDee™ Studio software targeted on a particular client's project/application.

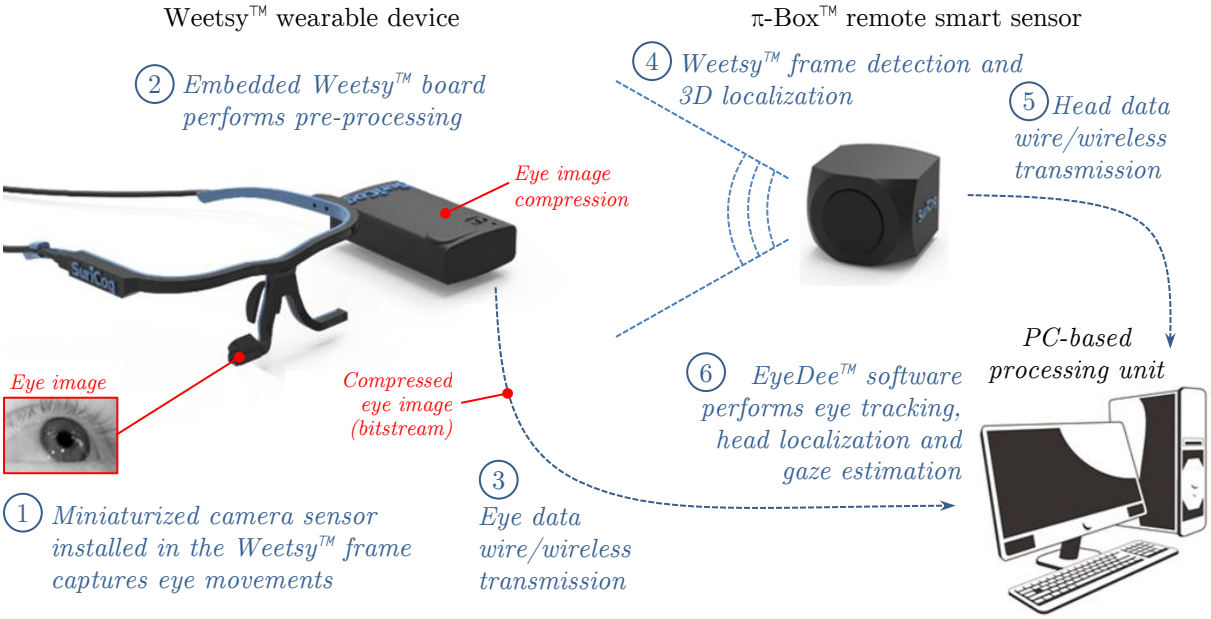


Figure 1. EyeDee™ eye tracking solution.

The EyeDee™ algorithms allow to reconstruct:

- Center of rotation of the eye in 3D space. This center is computed on the base of 5 parameters of the pupil's ellipse in 2D space coupled with position of the human's head,
- 6DoF-position (six Degrees of Freedom) of the human's head in the 3D space,
- gaze (direction at which human is looking, [4]) in the 3D space.

Since the eye movements are the fastest movements that the human body is able to produce, eye tracking solutions must be especially responsive. This immediately results in restrictions on the algorithms used to analyze at high frequencies (>100 Hz) an eye image to find center of the pupil, as

well as relatively hard constraints on the software/hardware implementation of these algorithms in a target physical hardware usually based on a low-power MCU coupled with an FPGA. One canonical factor, which has direct impact on system responsiveness and latency, is a transmission of an eye image obtained from a miniaturized digital camera sensor over a wireless media (Wi-Fi [5], Bluetooth [6] or others), which has to be done in a minimum time possible (<2-3 ms). To reduce the time of eye image transmission, it is a general practice to use image compression systems (also called 'image coding systems' and less often 'image compression techniques') [7]. To integrate a particular implementation of a selected image compression system into the product, a set of settings (so-called 'profile') has to be found, which leads in the highest overall performance of image compression system. This profile represents the results of the tradeoff between size of compressed image (measured in bits-per-pixel or bpp), visual quality of decompressed image (measured as PSNR in dB), time to perform image compression/decompression (measured in milliseconds or microseconds), computational complexity (measured in MIPS), operating memory requirements (measured in bytes) as well as overall resulted device autonomy (duration of using of the device powered by battery without any external power source). The research presented in this thesis is focused on solving this type of tradeoffs by accurate analysis of existed general-purpose compression systems, followed by gradual construction and optimization of an entire new eye image compression system especially developed for SuriCog's EyeDee™ product.

1.1 Objective and Scope of the Research

The objective of this thesis is the development of highly efficient image processing and compression algorithms for a miniature wireless resource constrained low-power embedded eye tracking system. The main goal is to show that new application-specific eye image compression algorithms can be more efficient than the general-purpose state-of-the-art image compression algorithms, used in such a standards as JPEG [8], JPEG 2000 [9] and some others.

This thesis has three main goals:

1. Select (or propose) the most efficient eye image compression system(s) for the EyeDee™.
2. Introduce an improvements of selected (or proposed) eye image compression system(s).
3. Implement of the eye image compression system(s) in the EyeDee™ and show benefits.

This thesis is divided on five parts:

1. Introduce a research domain and application specifics (Introduction).
2. Conduct a study of state-of-the-art in image compression systems (Theoretical Part).
3. Propose new approaches dedicated to eye image compression (Methodology).
4. Show benefits of previously proposed approaches (Experimental Results).
5. Implement proposed approaches in the EyeDee™ product (Conclusion and Future Work).

1.2 Summary of the Contributions

This thesis contains following proposed contributions:

- **Eye-based rate-distortion optimization.** One propose an eye-based rate distortion optimization algorithm for wavelet transform based eye image compression. It is shown that after wavelet transform several subbands contain eye details, which have very little impact on the eye tracking precision results. Therefore, such «less prioritized» subbands can be more quantized (or entirely removed), while «more prioritized» subbands are less quantized.
- **Eye image delivery chain performance analysis approach.** One propose an approach of performance analysis of all components of the entire «image delivery chain» (also called «image acquisition chain»), which includes several steps: acquisition of the eye image from the camera sensor (readout), reading of the eye image from the RAM of readout IC (FPGA based) into the RAM of host IC (MCU based), compression of the eye image, transmission of compressed eye image (bit stream) via several mediums to the remote side followed by decompression of

the compressed eye image. It is shown that based on assessment (accurate measurement of execution time) of each component used in each step it is possible to derive a mathematical model (represented as set of equations), further analysis of which permits to find bottlenecks of the «image delivery chain» and resolve them to improve the system overall performance.

- **Eye image compression based on low complexity DCTs.** One propose an eye image compression approach, which is based on using of low-complexity Discrete Cosine Transform (DCT) approximations, implemented in a multiplierless form (i.e., using only addition and shift operations). Compared with the original DCT, DCT approximations tend to maintain much less execution time at the expense of slight degradation of decompressed eye image quality, which directly contributes to degradation of the eye tracking system precision error. It is shown that it is possible to adjust level of DCT approximation, which permits to control eye tracking system precision error in a desired range (depending on the target application) while benefiting on the advantages of multiplierless implemented low-complexity DCT approximations.
- **Eye image compression system limits assessment.** One propose an assessment of the limits of eye image compression system. Such a limits include: minimal quality of decompressed image could be used for the eye tracking without *significant* eye tracking precision loss, minimal delay of eye image compression/decompression allowing to the eye tracking system stay *responsive*, minimal size of compressed eye image allowing to use less bitrate consuming data transmission technologies (e.g., Bluetooth or ZigBee), minimal MIPS (i.e., computational complexity) allowing to use low power (and usually small form factor and low price) hardware, and several other characteristics. It is shown that relying on such limits, several general purpose image compression approaches should be excluded from the consideration on the very early prototyping stage during design of portable wireless eye tracking system involving eye image compression.
- **Neural network based eye tracking approximation.** One propose an original eye tracking approach based on the use of neural network targeted on data regression. It was shown that using of such approach results in complete replacement of image processing based eye tracking algorithm coupled with geometric eye modeling by a precisely tuned and perfectly trained neural network, which directly transforms wirelessly transmitted floating-point values of decimated eye image (result of the 3D perspective projection of a model of rotating pupil disk) into five floating-point parameters of pupil's ellipse (result of the eye tracking). Such approach allows to drastically reduce the size of the transmitted data (from Weetsy™ board to PC-based processing unit) from the typical size of raw/compressed eye image to just 20 bytes of five floating-point values. However, this approach has a certain eye tracking quality issues (eye tracking system precision error).
- **Feature based ROI image compression.** One propose an improvement of compression of the Region of Interest (ROI, region containing image of the human's pupil). It was shown that it is possible to find and remove extra information from the ROI. Such extra information has *relatively* little impact on pupil's ellipse reconstruction, hence little impact on the eye tracking results quality. Finding of the «extra information» is done via trained neural network targeted on data classification, i.e., classification of blocks of the ROI into two classes: «block does contain pupil edges» (hence must be kept) and «block does not contain pupil edges» (hence considered as «extra information» and highly compressed or entirely removed). As a result of such approach, compressed ROI with highly compressed or entirely removed blocks has much less size compared to originally compressed ROI, because of higher compression ratio for «extra blocks» or less data to compress (in case of blocks removed).
- **Neural network based ROI finding.** One propose an original approach of the ROI finding, which is based on using of trained neural network targeted on data classification instead of using an original ROI finding algorithm. Original ROI finding algorithm performs multiple

threshold based block scanning of the integral image and provides multiple ROI 'candidates' as an output (hence, there is a need to select the best ROI 'candidate'), while neural network based approach is targeted on the same task, but without a threshold. It was shown that neural network based approach provides more stable results and executed more faster due since *generally* obtaining results from trained neural network is usually takes less time than execution of image processing algorithms. Potentially neural network based ROI finding can replace currently used image processing based ROI finding.

– **Foveated based eye image compression.** One propose an original approach of foveated based eye image compression, which is inspired by the 'foveated image rendering' i.e., a rendering, where information, which takes most user's attention, is rendered with maximum quality while information, which takes less user's attention, is rendered with minimum quality. The user's attention is determined via using of the eye tracking system. It was shown that this approach can be used in the eye tracking algorithm, which takes as an input static ROI image (cropped region of the full size image, which has such minimal vertical and horizontal dimensions to keep maximal pupil's vertical and horizontal positions). The static ROI image contains (at least) two parts: preliminary found (via neural network based approach) dynamic ROI image, which is not compressed and the rest of the image, which is compressed. In compare with «uniformly compressed» eye image, feature based eye image compression leads to «non-uniform compression», which immediately leads to variable bpp and less compressed eye image size in general.

1.3 Thesis Outline

This thesis includes five parts: an introduction, which is followed by four chapters and an appendix containing five publications. Chapter 2 is dedicated to theoretical part used in the thesis. Chapter 3 presents methodology used to conduct the research. Chapter 4 presents results, obtained on applying of previously presented theory coupled with defined methodology. Chapter 5 concludes the thesis.

1.4 Thesis Information

This thesis is completed under the CIFRE program [10], which stands for Industrial Agreement of Training through Research. The core idea of the program is conducting of scientific research in collaboration between academia and industry, where academia member is ISEP institute and industrial member is SuriCog company. The interest for the ISEP institute is research in the domain of image processing and image compression algorithms while interest for the SuriCog company is applying of these algorithms to the EyeDee™ eye tracking solution.

Chapter 2

Theoretical Part

2.1 Introduction

The EyeDee™ product (developed by SuriCog) is aimed at allowing user to interact with objects in a known environment by naturally using human's eye-movements. In particular, the complete system includes Eye Tracking (ET) system, which is based on computationally intensive image processing algorithms. These algorithms are aimed at reconstruction of the position of the pupil center [11] in 3D space. Input of the ET is a human eye image, obtained from a miniaturized digital camera sensor, installed in the Weetsy™ frame. Output the eye tracking system is five pupil ellipse parameters (floating-point values) defining pupil ellipse. Then these values are used by a Gaze software component (part of SuriDev software) to reconstruct user's gaze i.e., direction at which user is looking at each particular time moment. To accelerate performance of the ET, an amount eye image data, which is transmitted over a wire/wireless medium, can be drastically reduced by applying image compression. To properly introduce image compression in the entire EyeDee™ system it is necessary to understand the theory of image processing (in general) and image compression (in particular), as well as theory of main building blocks of the complete eye tracking algorithm of the EyeDee™. This chapter introduces this theory.

2.2 Eye Tracking Specifics

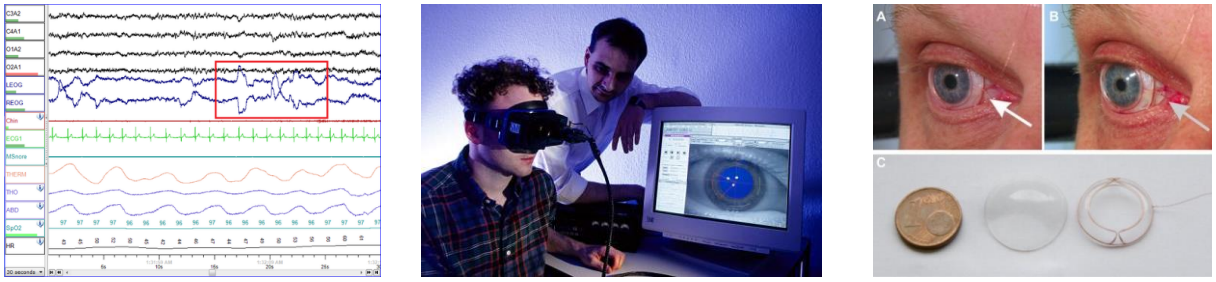
Eye tracking approach has long been utilized to study the visual attention of individuals. There are several key techniques to detect and track the movements of the eyes. The most commonly used eye tracking technique is Pupil Center Corneal Reflection (PCCR), which is based on the idea of eye illumination by a light source resulting in highly visible reflections. These reflections are captured by the camera sensor and resulted images are used to identify the reflection of the light source on the cornea (so-called 'glints') as well as the pupil. Reconstruction of gaze (direction at which user is looking at each particular moment) is based on a vector formed by the angle between the cornea and pupil reflections (the direction of the vector). Such a vector can be optionally coupled with other geometrical features of the reflections to increase precision of eye tracking results at the expense of computational complexity.

There are several eye tracking techniques (Figure 2):

- EOG (ElectroOculoGraphy) [12] – technique for measuring the corneo-retinal standing potential between the front and the back of the human eye. The resulting signal is called the electrooculogram. Primary applications are in ophthalmological diagnosis and recording eye movements. To measure eye movement, pairs of electrodes are typically placed above and below the eye or to the left and right of the eye. If the eye moves from center position toward one of the two electrodes, this electrode detects the positive side of the retina and the opposite electrode detects the negative side of the retina. Consequently, a potential difference occurs between the electrodes. Assuming that the resting potential is constant, the recorded potential is a measure of the eye's position.
- VOG (Video-OculoGraphy) [13] – well-established non-invasive, video-based technique of measuring horizontal, vertical and torsional position components of the movements of both eyes (via image processing algorithms applied to eye images) using a head-mounted device which is equipped with small cameras. Primary applications are medical applications. VOG techniques are applied in a wide field of scientific research related to visual development and cognitive science as well as to pathologies of the eyes and of the

visual system. The VOG technique can use eye images obtained with the 'bright pupil' or the 'dark pupil' illumination technique (presented below).

– Scleral coil technique [14] – technique for measuring the eye position, which is based on using a small coils of wire, which embedded in a modified contact lens. These coils are inserted into the eye after local anaesthetic has been introduced. When a coil of wire moves in a magnetic field, the field induces a voltage in the coil and then a signal of eye position will be produced. This allows horizontal eye movement to be recorded. If it is necessary to also monitor vertical eye movements, then a second set of field coils, usually set orthogonally to the first set, it used. The two signals (one for horizontal, one for vertical eye movement) generated in the eye coil can then be disentangled using appropriate electronics. If the eye coil is of an appropriate design, then torsional movements can also be recorded. In experiments on eye movements in animals, the eye coils are frequently implanted surgically. The advantage of this method is that it has a very high temporal and spatial resolution allowing even the smalled types of eye movements (e.g. microsaccades [15]) to be studied. The disadvantage is that it is an invasive method, requiring something to be placed into the eye. This method is rarely used clinically, but is an invaluable research tool.



EOG (ElectroOculoGraphy VOG (Video-OculoGraphy) Scleral coil

Figure 2. Eye tracking techniques: EOG, VOG, scleral coil.

There are two major pupil illumination techniques used with PCCR (Figure 3):

1. 'Bright pupil' – pupil illumination technique, where an illuminating LED is placed near to the optical axis of the imaging device resulting the pupil to appear lit up (same phenomenon that causes red eyes in photos), i.e., pupil is clearly demarcated as a bright region due to the photo reflective nature of the back of the eye.
2. 'Dark pupil' – pupil illumination technique, where an illuminating LED is placed at some distance from the optical axis resulting the pupil to appear darker than the iris, while the sclera, iris itself and eye lids all reflect more illumination.

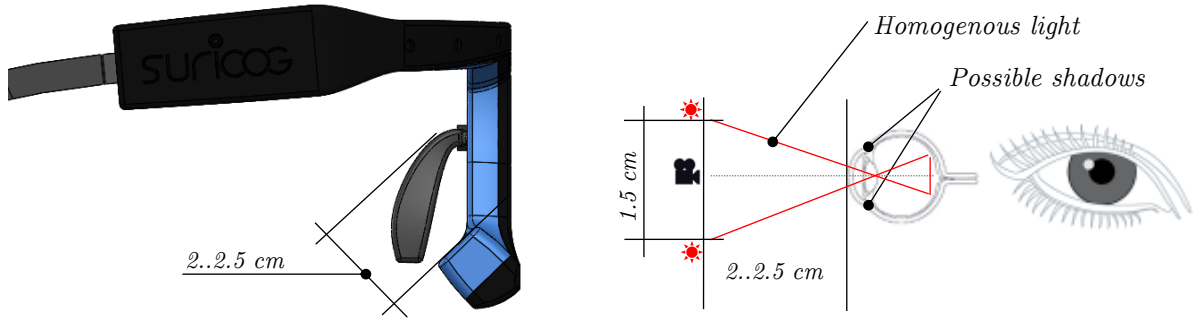
Bright pupil *Dark pupil*

Figure 3. Eye tracking illumination techniques: bright pupil and dark pupil.

The combined use of both bright and dark pupil techniques [16–18] usually is based on using the infrared LEDs when bright-pupil image and dark-pupil image are obtained by switching between either on and off the axis of the camera sensor. The resulted deferential pupil image is obtained by subtracting these images following by thresholding.

SuriCog's EyeDee™ eye tracking solution uses the VOG technique coupled with the 'dark pupil' illumination technique.

Eye tracking illumination conditions (Figure 4) have significant impact on the quality of the eye image. For example, Weetsy™ frame uses two IR LEDs placed on a distance of 1.5 cm while the distance between the miniaturized camera sensor and the human's eye is about 2..2.5 cm. If the IR LEDs distance is not optimal it is likely to have possible unwanted shadows (Figure 4).



Weetsy™ frame (profile view) *IR illumination scheme (dark pupil)*

Figure 4. Eye tracking illumination conditions and distances.

Different illumination conditions as well as several positions of the IR light sources were exhaustively tested at the prototyping stage of the product. Results of these tests are out of scope of this thesis.

Usually an eye tracking algorithm uses several digital filters, where the most widely applied are:

1. Gaussian blur (smoothing). Used to reduce the noise in the input eye images.
2. Histogram calculation. Histogram is used during the next image processing steps.
3. Binarization. Used to obtain black and white eye image.
4. Edge finding. Used to find pupil contours of the pupil following of their auto measuring (to reconstruct pupil's ellipse).

The Hough transform [19] is a well-known tool to identify geometrical objects in the image. Usually these objects are lines and circles (in this case Hough Circle Transform is used). Therefore, the question consists in the potential applicability of the Hough transform to the eye tracking. Since pupil contours have the form of an ellipse, the Hough transform must be used to detect ellipses. Since ellipse is defined via 5 parameters (Table 1), the Hough transform must perform the search in the 5D parameter space, which is computationally too expensive to be done on 100 Hz eye tracking frequency.

Table 1. Hough transform complexity to identify different objects.

Object	Object equation	Unknown parameters (complexity)
Line	$y = kx + b$	2
Circle	$(x-h)^2 + (y-k)^2 = r^2$	3
Ellipse	$\frac{((x-h)\cos\alpha + (y-k)\sin\alpha)^2}{a^2} + \frac{((x-h)\sin\alpha - (y-k)\cos\alpha)^2}{b^2} = 1$	5

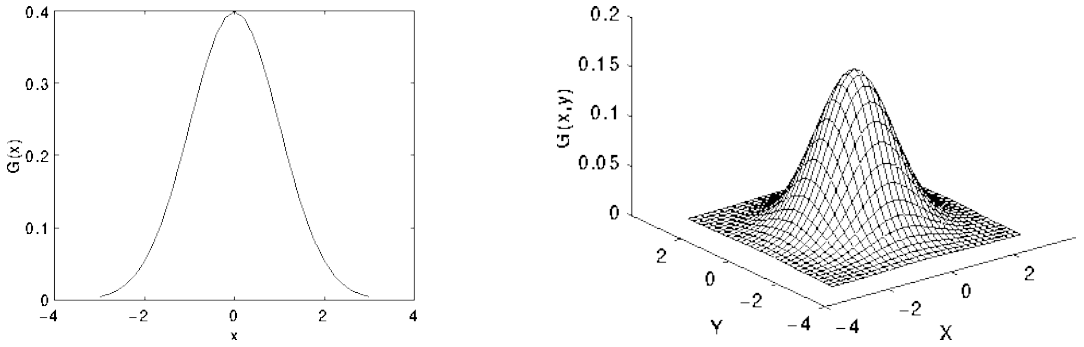
To reduce the noise in the input eye images Gaussian smoothing [20] is used. The Gaussian smoothing operator is a 2-D convolution operator that is used to 'blur' images and remove small details and noise. In this sense it is similar to the mean filter, but it uses a different kernel that represents the shape of a Gaussian convexity. This kernel has some special properties which are detailed below.

The Gaussian 1-D/2-D distributions have the form:

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}, \qquad (2.1)$$

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \qquad (2.2)$$

where σ is the standard deviation of the distribution. The corresponding distributions are illustrated in Figure 5.



1-D 2-D

Figure 5. 1-D/2D Gaussian distributions with mean 0/(0,0) and σ=1.

The idea of Gaussian smoothing is to use this 2-D distribution as a 'point-spread' function, and this is achieved by applying a convolution. Since the image is stored as a collection of discrete pixels there is a need to produce a discrete approximation to the Gaussian function before the convolution can be performed. In theory the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel, however in practice it is effectively zero more than about three standard deviations from the mean, so the kernel can be truncated at this point. Figure 6 shows a suitable integer-valued convolution kernel that approximates a Gaussian with σ of 1.0. However, it is not obvious how to select the values of the mask to approximate a Gaussian. For example, it is possible to use the value of the Gaussian at the center of a pixel in the mask, but this is not accurate, because the value of the Gaussian varies non-linearly across the pixel. Then the value of the Gaussian over the whole pixel was integrated (by summing the Gaussian at 0.001 increments). Since the integrals are not integers it is necessary to rescale the array so that the corners had the value 1. Finally, the 273 is the sum of all the values in the mask (Figure 6).

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Figure 6. Discrete approximation to Gaussian function with σ =1.

As soon as a suitable kernel has been calculated, then the Gaussian smoothing can be performed using standard convolution methods. The convolution can be performed fairly quickly since the equation for the 2-D isotropic Gaussian shown above is separable into x and y components. Therefore, the 2-D convolution can be performed by first convolving with a 1-D Gaussian in the x direction, and then convolving with another 1-D Gaussian in the y direction. (The Gaussian is in fact the only completely circularly symmetric operator which can be decomposed in such a way.) After eye image denoising via Gaussian smoothing SuriCog's EyeDee™ eye tracking algorithm does not use any image enhancement approaches.

2.3 Image Nature and Terminology Specifics

Eye image can be interpreted is a discrete 2D signal in spatial coordinates (x,y), which is a result of projection of the human's pupil onto an image matrix of the CMOS censor. This signal is represented as a 2D matrix with unsigned integer values. Since color information from the eye image is not used in the current implementation of the ET algorithm during the computation of the pupil ellipse parameters,

the eye image is grayscale 8bpp image with 256 levels of color gradation. Therefore, herein and after in this thesis term «grayscale 8bpp eye image» is assumed under term «eye image».

If pixel value itself is interpreted as a dimension, 2D image signal can be seen as 3D signal (Figure 7). In theory 2D signal values can be completely independent, hence there is no relationship between them. However, in practice due to nature of captured signals such values are usually dependent, meaning that changing/moving a pixel value in one (x,y)-position usually leads to changing/moving pixel values around this (x,y)-position. Observing 2D signals in 3D space gives a moving surface (Figure 7).

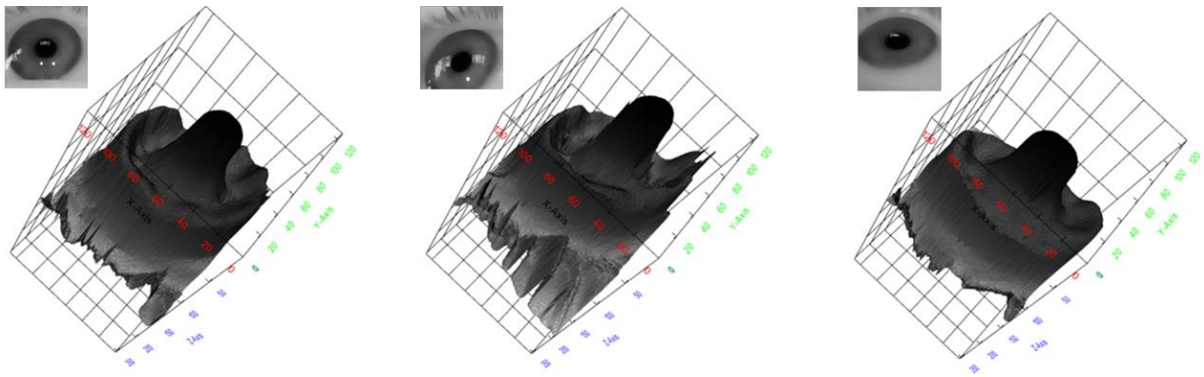


Figure 7. Examples of eye image represented as 3D signal (moving surface).

Since image is a discrete 2D signal, it is well known practice in the domain of image compression to progressively apply signal processing algorithms to each dimension of the image. In particular, such algorithms work in frequency domain [21] rather than in spatial domain. Frequency domain refers to the analysis of mathematical functions or signals with respect to frequency, rather than time. A given function or signal can be converted between the spatial (or time) domain and frequency domain with a pair of mathematical operators called a *transform*. Examples of such a transforms are: Fourier [22] transform, wavelet [23] transform and others. Therefore, description of the image compression algorithms often refer to term «frequency», because the input image signal is viewed in both the time and frequency domains simultaneously with use of several time–frequency representations in particular and time–frequency analysis in general.

2.4 Image Compression Fundamentals

All signals can be classified at least by number of dimensions needed to represent these signals:

- 1D signals – text, signals from certain sensors (temperature, speed, and distance measurement), and sound. Usually such signals have low difference between samples. Such 1D signals contain only spatial redundancy.
- 2D signals – images, photos of nature, natural scenes, etc. To compress these signals special methods are used, which address nature of the signals by using special transforms for example. Such 2D signals contain only spatial redundancy.
- 3D signals – images, with taking into account time domain. In such a case, sampling rate is more or equal to motion dynamics (i.e., moving of the objects in time). Such 3D signals contain both spatial and temporal redundancy.

SuriCog's EyeDee™ product involves processing of the eye images, which are acquired at very high frequency (>100 Hz). Most of the time (when user's eye motion is low) such eye images can be considered as being 3D signals, because most of the time they are highly correlated. However, it was shown that human is able to rapidly move the eyes on short distances (deltas) at frequencies as high as 500 Hz and more (so-called «rapid eye movements» [24]). Hence, in these cases eye image acquisition frequency is much lower than frequency of rapid eye movements and by definition such signals cannot be interpreted as 3D signals. (They can be interpreted though as a 3D signals during some time periods, when eye image acquisition frequency does not exceed eye maximal movement frequency.) Other aspect

is computational complexity. Due to additional computational complexity needed to process such 3D signals, eye images are considered in this thesis as being 2D signals, i.e., time notion (correlation in time) is not taken into the account during compression the of these eye images. Since transmission of uncompressed eye images requires a very high bitrate (in domain of resource constrained low-power embedded systems), image compression is aimed to extract and reduce natural redundancy from the eye image.

Generally image data compression approaches are split into two classes:

1. Image compression – data compression approach, aimed at eliminating of only spatial redundancy from the input image signal.
2. Video compression – data compression approach, aimed at eliminating of both spatial and temporal redundancies from the input image signal. Most video compression algorithms combine spatial image compression and temporal motion compensation.

The usual building blocks of the general-purpose image/video coding system (Figure 8) are:

1. Image acquisition – obtaining source images from digital camera sensor.
2. Pre-processing – operations on the raw uncompressed source images, such as trimming, color format conversion, color correction, or de-noising. In case of using ANN-based approaches bit depth reduction is used to decrease range of the input sample values and, therefore, to accelerate training.
3. Encoding – transformation of the pre-processed images into a coded bitstream. The goal of encoding is to generate a compact representation of the input images, which is suitable for the transmission medium in the given application scenario.
4. Transmission – packaging of the bitstream into an appropriate format and transmission over the selected medium (communication channel). Transmission optionally uses special approaches of data protection against data losses/corruptions and approaches of data loss recovery (also called «error concealment»).
5. Decoding – transformation of the received bitstream into a reconstructed images. Since image encoding usually requires lossy compression to achieve the target transmission bitrate constraints, the decoded images constitute an approximation of the original source images. If unrecoverable transmission losses have been occurred, the decoder applies error concealment strategies to recover the corrupted images as much as possible.
6. Post-processing – operations on the reconstructed image sequence for enhancement or for adaptation of the sequence for display or further processing. These operations can include color correction, trimming, or re-sampling. Also special effects may be applied as determined by the application. In case of SuriCog's EyeDee™ Gaussian blurring is used at the post-processing step to reduce the noise in the input eye images.
7. Displaying/processing – further usage of image sequence, such as viewing or processing.

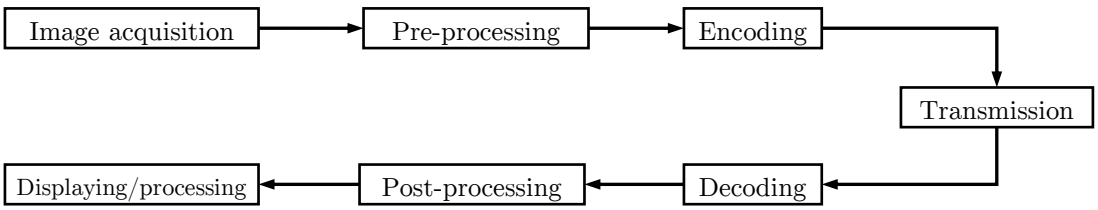


Figure 8. General block diagram of image/video coding system.

In general, encoding stage (image compression) consists of the three steps:

1. Transform – mapping of input image data (pixel values) to a set of coefficients of basis functions [25]. The goal of a transform is to keep (or at less to stay very close to) strong «energy compaction» property [26].
2. Quantization – setting closed to zero coefficients of transformed image to zero.

3. Entropy coding – transformation of quantized coefficients to achieve minimal size of the resulting bit sequence (bitstream) with ability to perform inverse transformation, i.e., to reconstruct quantized transform coefficients sequence followed by inverse transform to reconstruct original input image data (pixel values).

In general, a transmission stage consists of two following steps:

1. Channel coding – use of error correction codes [27] to allow decoded to reconstruct corrupted data. Since in SuriCog's application eye images are sent in a compressed form, it is expected to consider use of error correction codes. These codes will allow correction of the errors, which can be introduced due to unreliable transmission medium (best-effort delivery). However, usage of such codes requires additional computation resources, needed to perform encoding/decoding and (optionally) correction, as well as higher transmission bitrate, because these codes introduce additional extra information into the bitstream. Then this extra information is used for checking of error presence and error correction.
2. Error presence checking – usage of Cyclic Redundancy Check (CRC) [28]. CRC can be used to detect data integrity during transmission, i.e., to calculate if the data received is bitwise identical to data sent. If calculated CRC is not equal to received CRC, compressed image will be considered as «corrupted» image and skipped for further decompression and processing (or partially used in case of defined policy concerning «corrupted» images).

To get benefits (in terms of compression time; also used term 'compression duration') of eye image compression, following condition has to be true:

$$T_{compr} + T_{transf\_compr} + T_{decompr} < T_{transf\_uncompr} \tag{2.3}$$

where:

T_{compr} – time of image compression,

T_{transf_compr} – time of transmission of compressed image,

$T_{decompr}$ – time of image decompression,

$T_{transf\_uncompr}$ – time of transmission of uncompressed image.

By reason of specifics (maximal possible performance) of developed product, target goal consists in reaching minimal possible eye image delivery delay (i.e., delay between emitting eye image by the camera sensor and availability of the same eye image in the processing unit performing eye tracking) while maintaining acceptable image quality and (ideally) maintaining minimal compressed image size. Therefore, importance of compression factors has to follow this order (most prioritized are presented first):

1. Time of image compression (represented in milliseconds or in microseconds);
2. Quality of decompressed image (represented as PSNR in dB);
3. Size of compressed image (represented in bytes or bpp).

The final goal is to reach Pareto optimality between these metrics. The exact requirements for each of these metrics are presented further.

2.4.1 Using Advantages of Human Visual System

All applications, which involved image data compression to satisfy transmission medium requirements (reduction of the storage space or increase of the transmission speed), followed by its decompression, take advantage of Human Visual System (HVS) [29]. In these applications decompressed image data are then used directly by humans either by viewing (images/video), reading (text), or hearing (sounds). Usually, such data contain significant amount of redundancy, which can be reduced or entirely removed. The origins of this redundancy are different. For example, it is known fact that in general nature signals

(either images or sounds) contain significant amount of redundancy. Text, written in a particular language also contains certain amount of redundancy (so-called «natural language redundancy»; different languages have different average redundancy level). It is also common way of to think of «taking advantage of the HVS» to produce desired effects in a target application. Examples of « taking advantage of the HVS» include:

- Color squeezing – reduction of the depth of the chroma component by use of so-called chroma subsampling [30] technique. This reduction is possible, because according to HVS model, color resolution of the HVS is much lower than the brightness resolution. Another example of HVS-related effects is aftereffect of color, which depends on the orientation of several shapes (for example lines [31]);
- Small details removal – in is generally considered that small details in an image are taking much less attention of the user than edges. This is because HVS is more sensitive to edges rather than small details. Hence removal of these small details (via quantization) from the original image does not lead to perceptible loss of visual quality, which as a result, does not cause difficulties in image interpretation. This principle is widely used in lossy image compression technique, where output decompressed image does not have significant visual difference in compare with input uncompressed image.

It should be noted that SuriCog's EyeDee™ eye tracking system performs image processing of decompressed eye images rather than their direct displaying. The eye tracking system does not incorporate human feedback on viewing decompressed eye images. (Rather it incorporates human feedback on hearing audio signal, which is generated based on human eye gaze – the direction at which human is looking at each particular time moment.) Therefore, during selection, implementation, optimization and tuning of the image compression system, recommended practices on obtaining the best acceptable quality, based on usual advantages of HVS, cannot be directly applied due to application specifics.

Because the eye tracking algorithm is stayed relatively sensitive to quality of decompressed eye image, a special research was done (presented follow) on decompressed image quality adjustment. This research shows dependency of quality of decompressed eye image on the precision of the output eye tracking results.

2.4.2 Image Compression Building Blocks

Transform is a pair of mathematical operators used to convert a given function or signal between the spatial domain (also called 'time domain') and frequency domain [32]. An enormous efforts have been done over the last two decades and a half to achieve transforms, which satisfy a particular requirements (either energy compaction property, implementation complexity, parallelization ability, etc.). Example of such transforms are the following: wavelet [33–35] transforms and their applications [36–38], adaptive (optionally integer) wavelet transforms [39–41], bandelets [42–44], contourlets [45–47], directionlets [48–50], grouplets [51,52], Discrete Cosine Transforms (DCT) [53–55], low complexity DCTs [56–58], multiplierless DCTs [59–61], tetrolets [62–64] and relatively recently found shearlets [65–67]. To achieve low complexity implementation (hence to accelerate transform execution time), wavelet transforms has been implemented accordingly. One example includes lifting schemes [59,68,69], in which transforms usually are called 'lifting transforms' [70–72]. However, these transforms have a number of implementation issues. For example, lifting transforms have cache-related issues [73,74] or a certain issues related to their parallelization (for example with use SIMD [75] instructions). Wavelet transforms that map integers to integers can be implemented with use of integer arithmetic only [76], which allows to use such implementations in real-time (usually FPGA-based) systems.

In contrast, several image compression standards are successfully implemented with SIMD instructions (such as MMX, SSE2, AVX2, NEON, AltiVec) [77]. Several wavelet transforms can be implemented in parallel [78–80] fashion, requires powerful multicore CPUs to compress rapidly huge

images (resolution can be as much as 20,000x20,000 pixels and even more), fox example in satellite imagery – images of Earth or other planets collected by imaging satellites. In contrast with parallel implementations, low complexity DCT based transforms are initially developed to be used in a very resource constrained platforms, for example low power processors for Wireless Sensor Networks (WSN) [81–83]. Certain application involve real-time [84–86] image compression performed directly on-board [87–89] used in several missions to meet the specialized needs of (for example) deep-space applications while achieving state-of-the-art compression effectiveness. In particular, ICER [90], which is used in the Mars Exploration Rover (MER) mission, can provide lossless and lossy compression, and incorporates an error-containment scheme to limit the effects of data loss during transmission. Usually such missions incorporate hardware implementations [91] of real-time image data compression used in satellite remote sensing.

From mathematical point of view, the transforms have certain number of issues, which are usually related to the fact that input signals are discrete and not continuous. Another issue is use of integer arithmetic in a transform, initially implemented with floating-point arithmetic. This technique permits to accelerate execution time, but has certain issues. For example, integer wavelet transforms have boundary effects/artifacts [92–94] and several rounding operators [95] issues. Other approaches of increase of transform performance is changing internal form of data they operate with. For example, a rectangular wavelet transform [96] outperform the square one.

Several transforms are also found themselves to be directly used for edge detection: for example, wavelets [97], directionlets [98] and shearlets [66]. Adaptive wavelets are applied in domain of eye-gaze [99] video compression, which is based on the foveation behavior of the HVS, as well as in domain of eye iris [100] image compression, based upon the popular CDF 9/7 discrete wavelet transform. Quantization is a technique aimed on mapping a range of values to a single quantum value. In codec implementations usually quantization is parametrized with quantization matrixes, which acts as a devisor on which transform coefficients are divided to obtain such a mapping. Entropy coding [101–103] is the final step in the image compression, which includes modeling on the quantized near-to-zero transform coefficients to find an absolute minimum number of bits, needed to represent these coefficients. Depending on the entropy coder used, several codecs include so-called packetization module (also called 'packetizer') coupled with rate-distortion optimization [104] module, which forms a bitstream by adding into it already coded coefficients to not exceed available bits-per-pixel (bpp) budget (which is directly set as the input configuration parameter or calculated from compression ratio).

2.4.3 Image Compression Techniques (JPEG and JPEG 2000)

JPEG [8] is a commonly used standard of lossy compression for digital images. It was created by the Joint Photographic Experts Group committee in 1992. JPEG typically achieves 10:1 compression with little perceptible loss (usually invisible for human's eye) in image quality. JPEG encoder consists of several stages (Figure 9): from pre-processing, DCT transform and quantization to Huffman [105] entropy coding. The uncompressed image is first converted from its original color model (such as RGB) to a luminance-chrominance model (YCbCr). Each channel is then passed to the Discrete Cosine Transform (DCT). In the quantization step, a specified (according to special so-called quantization tables) number of DCT coefficients are set to zero. A special sampling scheme, zigzag sampling, creates long runs of zeros which readily compress in a Run-Length Encoding (RLE) stage. Finally, Huffman encoding optimizes entropy and creates a final bitstream.

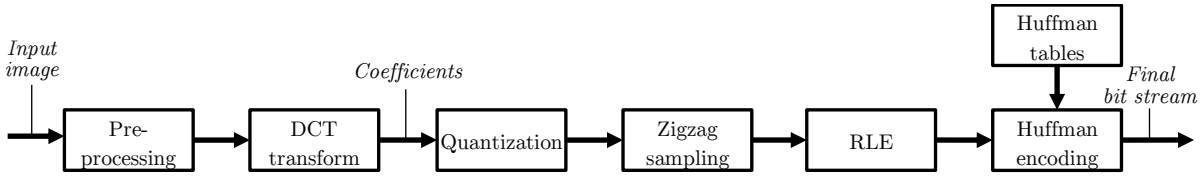


Figure 9. JPEG encoder scheme.

JPEG 2000 (usually abbreviated JP2) [9] is an image compression standard and coding system. It was created by the Joint Photographic Experts Group committee in 2000 with the intention of superseding their original discrete cosine transform-based JPEG standard with a newly designed, wavelet-based transform. JPEG 2000 encoder consists of several stages (Figure 10): from pre-processing, wavelet integer transform (which maps integers to integers and which is implemented according to lifting scheme) and quantization to entropy coding and packetization of the final bit stream (bit stream assembler).

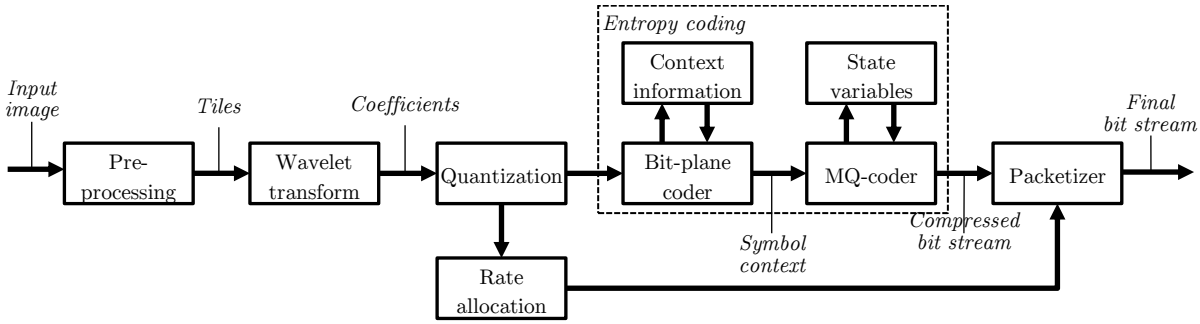


Figure 10. JPEG 2000 encoder scheme.

Within the pre-processing module, each color component of the source image passes through a multiple component transformation and is decomposed into non-overlapping rectangular tiles (tile components) of equal size. The wavelet transform can be configured to be lossy (9/7 filter) or lossless (5/3 filter). After transformation, all coefficients are then quantized. The entropy coder divides the quantized sub-bands into rectangular code blocks. The bit-plane coder categorizes each bit in the code block coefficient into 3 coding passes, each bit belonging to only one coding pass. It then parses the code block bit plane from most significant bit (MSB) to least significant bit (LSB). It arranges the bits in a zigzag order for each bit plane. The entropy coder also computes the context information needed by the MQ-coder as well as the distortion metrics, which will be used by the rate allocation unit of the Tier-2 coder. The bits and contexts output from the bit-plane coder are then processed by the MQ-coder, which generates the compressed bitstream. Rate allocation is used for post-compression rate-distortion optimization and to target a defined bit-rate or an arbitrary set of specified bit-rates. Based on the defined compression ratio, progression order and other configurable options of the JPEG 2000 standard based coder, the packetizer generates packets and places them into the final bitstream.

JPEG 2000 also provides possibility to select Region of Interest (ROI) – a part of the image, which will be further compressed. By utilizing all state-of-the-art in image compression JPEG 2000 is able to produce significantly better results in compare with its predecessor JPEG.

2.4.4 Video Compression Techniques (H.264 and H.265)

H.264 (MPEG-4 AVC) [106,107] and H.265 (HEVC) [108] are widely used standards in video compression domain. They can provide the best compression ratios other video compression standards. However, their usage for compression of the eye images transmitted from Weetsy™ board to PC-based processing unit is problematic due to (most prioritized are presented first):

1. High computation complexity (even with hardware implementations), which is usually leads to power consumption increase. A typical example is motion estimation internal module, which is considered to be the most computationally intensive module among other codec's internal

modules. Depending on a particular implementation an amount of processing dedicated to motion estimation can be as much as to 75%.

2. Perceptible encoding/decoding delays (even with special profiles applied), which directly contribute to eye tracking system responsiveness.
3. Need of guaranteed data delivery of data transmission, which is assured by network protocols such as TCP and implemented via acknowledgement mechanism including packets sequential numbering. As a result of unstable transmission medium, data corruption is possible, hence data re-transmissions can occur, which results in unknown transmission delays and which can be crucial for some time-critical applications involving eye tracking (for example military eye tracking systems installed in an HMD-HUD used by pilots of an aircraft or a helicopter to control moving targets). Using best-effort delivery, which is assured by network protocols such as UDP or RTP, with these codecs in case of data corruption or data lose will result in inability of decoding (best case) or unpredictable/undefined behavior (worst case).
4. Relatively high cost of available hardware implementations (IP cores), which is explained by relatively high implantation complexity as well as cost of integration in a particular hardware.
5. Possible licensing and legal issues due to specifics of the application.

Therefore, despite of their best compression ratios (result of contributions from academia and industry over a years), such products are not exactly dedicated (even with a special profiles applied) to SuriCog's EyeDee™ eye tracking application domain and requirements (real-time operation, minimal delays, low complexity implementation).

2.4.5 Image Compression Recent Improvements

Over the last decade an enormous effort has been done in the domain of improvement of efficiency of each component of image compression system: from spatial-to-frequency domain transform approximations to entire new data coding approaches. The reason of these improvements consists in reduction of the implementation complexity, while minimizing difference in compare with an original (non-approximated) algorithm versions (Figure 11). The main goal of this reduction is to find a profile (window), which conforms to principle of '20-80', i.e., where transform speed is 80% increased at the expense of 0..20% transform precision lost (due to approximation). As a result of such reduction of complexity, there are several advantages: performance acceleration and cost reduction of the target hardware. This section provides brief descriptions of some of research directions, aimed on improvement of image compression systems.

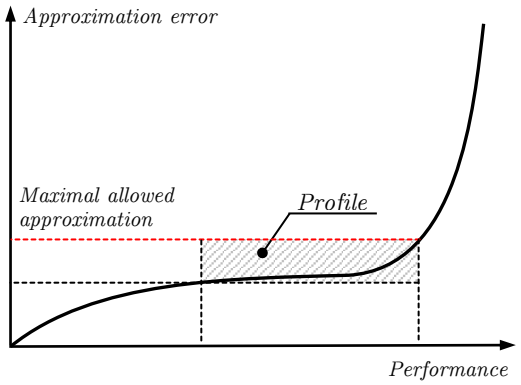


Figure 11. Maximal performance acceleration vs. maximal allowed approximated error.

The first research direction consists in improvement of already used components of image compression. *Data transform* is the first component, aimed on spatial-to-frequency domain conversion. Over the recent years an enormous effort has been done to find a DCT approximations, which do not compute the exact DCT coefficients, but still provide energy compaction property at a very low computational cost. In particular, in the year of 2012 there was proposed a scheme [109] of using Cordic Loeffler (CL) DCT coupled with MQ-coders [110] Golomb [111] and to achieve lowest possible

computational complexity leading to low power hardware. In the year of 2013 there was proposed a scheme [112] (targeted very low bit-rates: 0.14 bpp) of using Optimal Zonal BinDCT (based on Chen's factorization: computing only 4 significant coefficient in 8x8 block) with Golomb coder. In the year of 2012 there was proposed a low-complexity 8-point orthogonal DCT approximation, which requires 14 additions only [113]. In the year of 2016 there was proposed a low-complexity 16-point orthogonal DCT approximation, which requires 44 additions only [114]. *Quantization* [115] is the intermediate component, aimed on setting closed to zero transform coefficients to zero. Quantization is done based on a particular application (compression of images of knowing nature, hence aposteriorical knowledge about the signals). *Data coding* coupled with Rate-Distortion Optimization (RDO) [104] is the final component, aimed on coding of quantized coefficients after the transform into output sequence of bits (bitstream). There are several data coding algorithms are used in practice: Huffman coding [105], Run-Length Encoding (RLE) coding [116], arithmetic coding [117,118] and Q-Coder [110] family (including QM-coder, Q15-Coder, MQ-Coder, which is a series of binary arithmetic coders with multiplierless implementation, i.e., only shifting and adding operations) and Golomb coding [111]. Recent efforts in data coding include Finite State Entropy (FSE) [119] algorithm – a new breed of entropy coder, which is based on Asymmetric Numeral Systems (ANS) theory [120–123] and which is used in Zstandard [124] lossless data compression algorithm. FSE achieves precise compression accuracy (like Arithmetic coding) at much higher speeds. Recent efforts in rate-distortion optimization include applying of psychovisual [125] based approaches. For example, applying of psychovisual similarity estimations between the images to make rate-distortion decisions. In particular, Guetzli [126,127] algorithm was proposed by Google, Inc. [128], which uses introduced Butteraugli [126,127] psychovisual similarity metric to make rate/distortion decisions and which strikes a balance between minimal loss and file size by employing a special search algorithm. This algorithm tries to overcome the difference between the psychovisual modeling of JPEG's format, and Guetzli's psychovisual model, which approximates color perception and visual masking in a more thorough and detailed way than what is achievable by simpler color transforms and the discrete cosine transform. In particular, part of implementation of the Guetzli and Butteraugli relies on hard coded high-precision constants [129], which were possibly auto generated based on machine learning (ML) approach, where a huge corpus of images was used as an input for a neural network, which is targeted on generation/optimization of these high-precision constants of the psychovisual model. Another example from Google, Inc. is Brotli [130,131] lossless compressed data format that compresses data using a combination of the LZ77 algorithm and Huffman coding, with efficiency comparable to the best currently available general-purpose compression methods. Another example is Free Lossless Image Format (FLIF) [132–134] (previously JiF [135]), which directly operates in spatial domain (hence no DCT/DWT transform) and uses Meta-Adaptive Near-zero Integer Arithmetic Coding (MANIAC), a variant of Context-Adaptive Binary Arithmetic Coding (CABAC) [136], where the contexts are nodes of decision trees which are dynamically learned at the encode time.

The second research direction is proposition of entire new image compression approaches: either by applying different mathematics/computation models (for example fractal based [137]) followed by their software/hardware implementations, or applying new image compressive sensing [138] approaches, implemented directly in hardware. However, when the first ones usually require enormous time to compress an image, second ones are not yet (at the time of this research) implemented in the products, which are already available on the market, especially in a form of low-cost miniaturized digital camera sensor.

The third research direction is applying of Artificial Intelligence (AI) [139] / Machine Learning (ML) [140] based / Neural Network (NN) based approaches [141–143]. In particular, some of recently created companies (for example WaveOne, Inc. [144]), use these approaches to create custom-tailored, context-dependent solutions, which are targeted on outperforming all existing codecs, while running in real-time [145]. Usually detailed description or operating principle of these approaches are considered as a 'know-how' and not yet and/or fully revealed due to possible intellectual property issues.

There is also a visible trend over a recent years, when companies, business of which involves storage of massive client's image data, decide to create custom codecs instead of use standard ones. This decision is usually based on economic reasons (less data stored on a remote servers, less payment rent for these servers). In particular, Dropbox, Inc. [146] created Lepton [147] – tool and file format for losslessly compressing JPEG compressed images by an average of 22% [147], Facebook, Inc. created Zstandard [124] – lossless data compression algorithm, which compresses/decompresses 3-5x faster to obtain 10-15% smalled compressed files (compared with zlib library, which is de facto standard implementation of the deflate algorithm).

According to such a consideration it is possible to draw a conclusion that since SuriCog's EyeDee™ eye tracking application involves very low cost/complexity implementation and real-time operation (minimum possible compression delay), only image compression approaches, which target similar application can be considered for their further evaluation and potential usage followed by improvements.

2.5 Machine Learning

Machine Learning (ML) [148–150] 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 labeled training data. The training data consist of a set of training examples.

Machine learning approaches are divided into two main types (Figure 12):

- Supervised learning [151] – 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 also called «labeled» sample.
- Unsupervised learning [152] – each training sample is only an input object (typically a vector), i.e., without a desired output value. Such sample also called «unlabeled» sample.

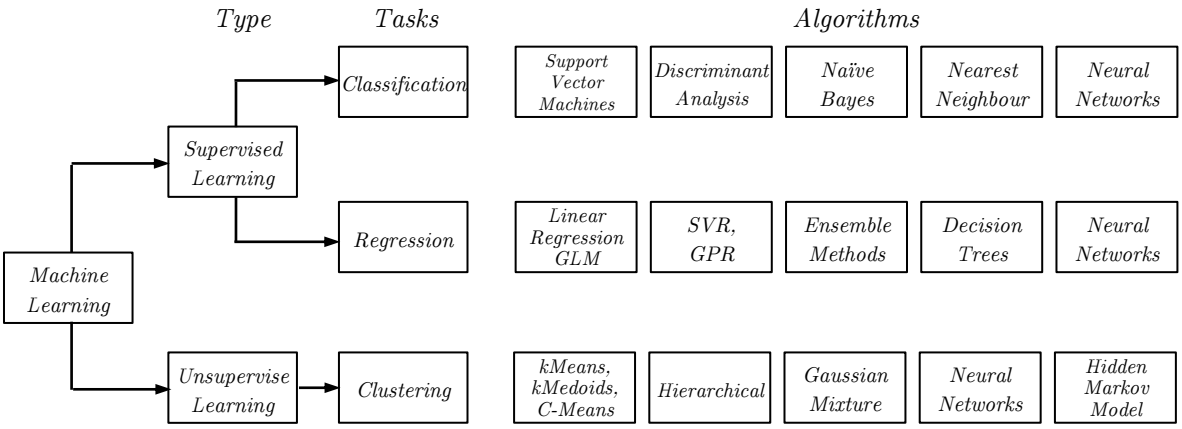


Figure 12. Machine learning algorithms classification.

These ML types are based on the following approaches:

- Support Vector Machines [153] (SVM, also Support Vector Networks) – supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.
- Linear Discriminant Analysis (LDA) [154] – generalization of Fisher's linear discriminant [155], a method used in statistics, pattern recognition and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events.
- Naive Bayes Classifiers (NBC) [156] – family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

- K-nearest Neighbors Algorithm (k-NN) [157] – non-parametric method used for classification and regression. In both cases, the input consists of the k-closest training examples in the feature space.
- Generalized Linear Model (GLM) [158] – flexible generalization of ordinary linear regression that allows for response variables that have error distribution models other than a normal distribution.
- Support Vector Regressor (SVR) [159] – regressor, which performs regression, predicting continuous ordered variables. In contrast with SVM (Support Vector Machine), which is a classifier, which performs classification, predicting discrete categorical labels. Both SVR and SVM use very similar algorithms, but predict different types of variables.
- Gaussian Process Regression (GPR) [160] – powerful nonparametric regression technique.
- Ensemble methods [161] – methods, based on use of multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.
- Decision trees [162] – decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.
- K-means clustering [163] – method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells [164].
- K-medoids clustering [165] – partitioning method commonly used in domains that require robustness to outlier data, arbitrary distance metrics, or ones for which the mean or median does not have a clear definition. It is similar to k-means, and the goal of both methods is to divide a set of measurements or observations into k subsets or clusters so that the subsets minimize the sum of distances between a measurement and a center of the measurement's cluster. In the k-means algorithm, the center of the subset is the mean of measurements in the subset, often called a centroid. In the k-medoids algorithm, the center of the subset is a member of the subset, called a medoid.
- Fuzzy C-means Clustering (FCM) [166–175] algorithm – one of the most widely used fuzzy clustering algorithms.
- Hierarchical clustering [176–184] (also called Hierarchical Cluster Analysis or HCA [185–194]) – method of cluster analysis which seeks to build a hierarchy of clusters. Used in data mining and statistics.
- Hidden Markov Model (HMM) [195–201] – statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved states (i.e., hidden states).
- Gaussian Mixture Model (GMM) [202–211] – probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters.

It should be noted that in general machine learning is considered to be difficult, because finding expected patterns is hard and especially in cases, where not enough training input data are available. As a consequence, machine-learning based approaches can produce incorrect results.

2.6 Neural Network Fundamentals

2.6.1 Neural Network Basics

Artificial Neural Networks (ANN) [212], also known as Connectionist Models [213] or Parallel Distributed Processing [214] are models are inspired by human brains with aim of simulation of their biological equivalent. The behavior of a biological neuron is modeled via a set of mathematical operations. In particular, to obtain an output signal the following operations are performed on input signals: weightening, summing and thresholding. Then these output signals are 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 make them a powerful tool to recognize patterns.

Mathematically [215] ANN can be interpreted as an *approximation* and *generalization* of an unknown multivariate (usually nonlinear) function (so-called objective function [216]), values of which are given only at some points, over a special mechanism of interconnected (usually nonlinear) activation functions [217]. *Approximation* here means finding of a *global minimum* [218,219] among several local minimums [219]. *Generalization* here means *minimization of the approximation error* of the result, obtained from the trained ANN on the input, which was not used during ANN training. In this mechanism activation function takes as an input linear combination [220] of connected signals and produces one output. The network is formed (Figure 13) by connecting the output of a certain building blocks, so-called *neurons*, to the input of other neurons forming a directed, *weighted graph* (Figure 14). Constants, which are used in this linear combination are so-called *weights*. Input neurons and hidden ones are different. For example, for pattern recognition task it is known practice to use (at least) 3-layer ANN, with linear activation function for the input layer units and a sigmoid activation function (namely a logistic activation function) for the hidden layer units. As for the output layer units, it is possible to use any of the both: either sigmoid or linear.

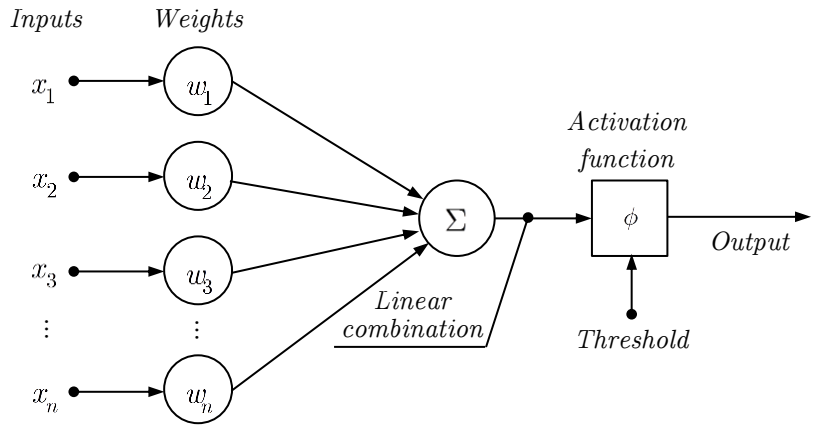


Figure 13. Neural network neuron scheme.

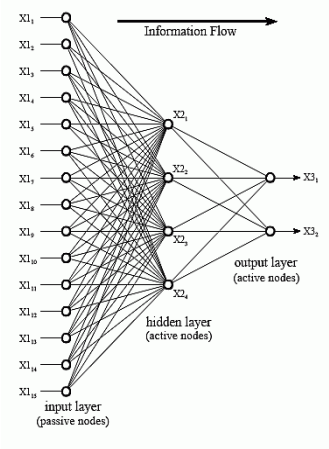


Figure 14. Example of neural network graph

There are two functions, which are referred during training:

1. Loss function (also called cost function or error function) – function that maps a values of one or more variables onto a real number intuitively representing some «cost» associated with the value.
2. Objective function – the most general term for any function that is optimized during training.

Objective function is approximated via an iterative technique like backpropagation [219] (commonly used by the gradient descent optimization algorithm [221] to adjust the weight of neurons by calculating the gradient of the loss function), which is a progressive updating (calibration) of the weights by repeating two key steps (Figure 15):

1. Forward propagation (also called «forward pass») – set of weights and bias values are applied to the input data to calculate an output. For the first forward propagation, the set of weights are selected randomly [222] (usually based on Gaussian distribution [150]) between 0 and 1.
2. Backward propagation (also called «backward pass») – the margin of error of the output is measured and the weights adjusted accordingly to decrease the error.

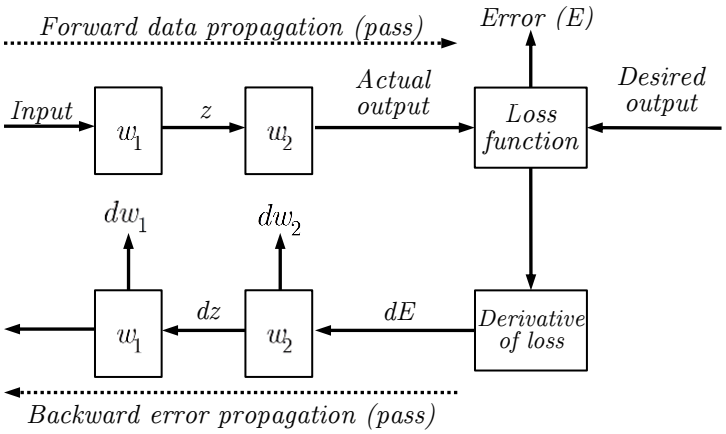


Figure 15. Gradient descent operation principle.

There are tree datasets used:

1. Training dataset – dataset of examples used during learning via forward/backward propagation.
2. Test dataset – dataset that is independent of the training dataset, but that follows the same probability distribution as the training dataset. If a model fit to the training dataset also fits the test dataset well, minimal overfitting [223,224] has taken place (Figure 16). A better fitting of the training dataset as opposed to the test dataset usually points to overfitting. A test set is therefore a set of examples used only to assess the performance (i.e., generalization) of a fully specified classifier.
3. Validation dataset – dataset of examples used to tune the hyperparameters. A hyperparameter is, for example, the number of hidden layers. As well as the test dataset, validation dataset should follow the same probability distribution as the training dataset.

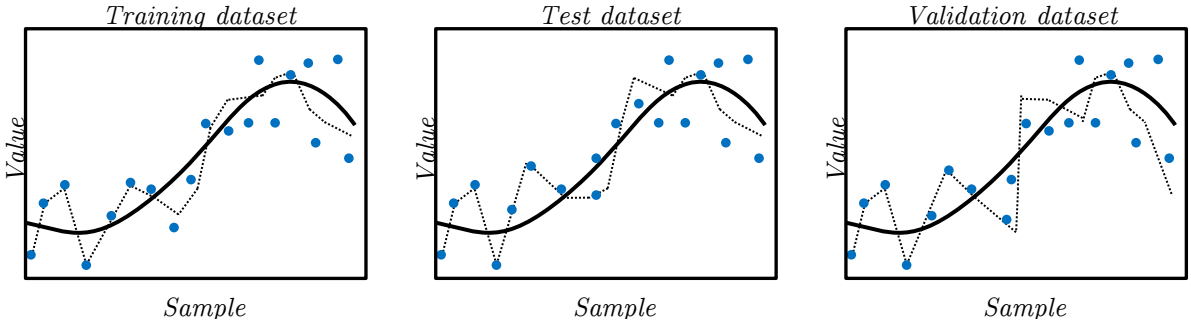


Figure 16. Datasets: training dataset, test dataset and validation dataset.

The accuracy of a neural network model is usually determined after the model hyperparameters are learned. Then the test samples are fed to the model and the number of mistakes (zero-one loss) the model makes are recorded, after comparison to the true targets. Then the percentage of misclassification is calculated. Accuracy on training set, test set and validation datasets (Figure 17) are differ.

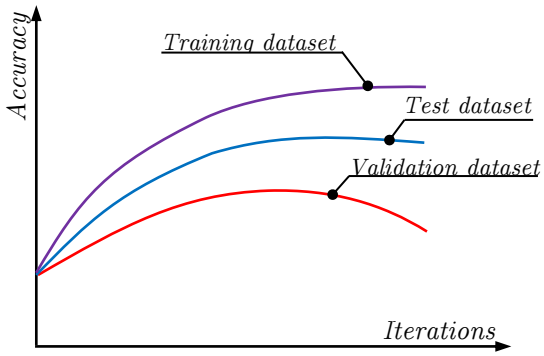


Figure 17. Accuracy on training and validation datasets.

The training procedure is parameterized by several variables, which are called hyperparameters [225,226]. The usual hyperparameters are (described more precisely in the next sections):

- Number of epochs – parameter, which represents the number of iterations of forward/backward propagation over the training dataset;
- Learning rate – parameter, which represents the size of the step taken at each stochastic estimate of the gradient; determines how fast weights are changed.
- Learning rate decay – parameter, which is used to decay the learning rate. Using of non-adaptive learning rates can be suboptimal. The recommended practice is to lower the learning rate as the training progresses. The decay is reduced by some constant factor every few epochs or either by *exponential decay*, in which the decay takes a mathematical form of the exponential every few epochs. Use of decay allows the learning algorithm to converge faster and with higher precision;
- Weight decay – parameter, which is used to L2-regularize the solution (model overfitting reduction [223,224]). Such a regularization prevents the weights from growing too large;
- Momentum – parameter, which is used to prevent the system from converging to a local minimum or saddle point.

There are two general approaches for neural network training (Figure 18):

- Online training – weights and bias values are adjusted for every training item based on the difference between computed outputs and the training data target outputs.
- Batch training – the adjustment delta values are accumulated over all training items, to give an aggregate set of deltas, and then the aggregated deltas are applied to each weight and bias.

However, the term «batch» itself is ambiguous: some researchers use it to refer to the entire training set, and some researchers use it to refer to the number of training examples in one forward/backward propagation. To avoid that ambiguity and make clear that batch corresponds to the number of training examples in one forward/backward propagation, one can use the term mini-batch.

Therefore, the following terminology are used to parameterize training:

- Number of epochs – the number of forward/backward propagations (passes) of all the training dataset.
- Size of batch – the number of training examples in one forward/backward propagation (pass).
- Number of iterations – number of epochs / size of batch.

Example: Dataset has 1000 training samples, size of batch (mini-batch) is 200. Number of iterations = 1000 / 200 = 5.

online training:

```
loop maxEpochs times
  for each training data item
    compute weights and bias deltas for curr item
    adjust weights and bias values using deltas
  end for
end loop
```

batch training:

```
loop maxEpochs times
  for each training item
    compute weights and bias deltas for curr item
    accumulate the deltas
  end for
  adjust weights and bias deltas using accumulated deltas
end loop
```

Figure 18. High-level pseudo-code: online vs. batch training.

Figure 19 summarizes all the relationships between ANN building blocks.

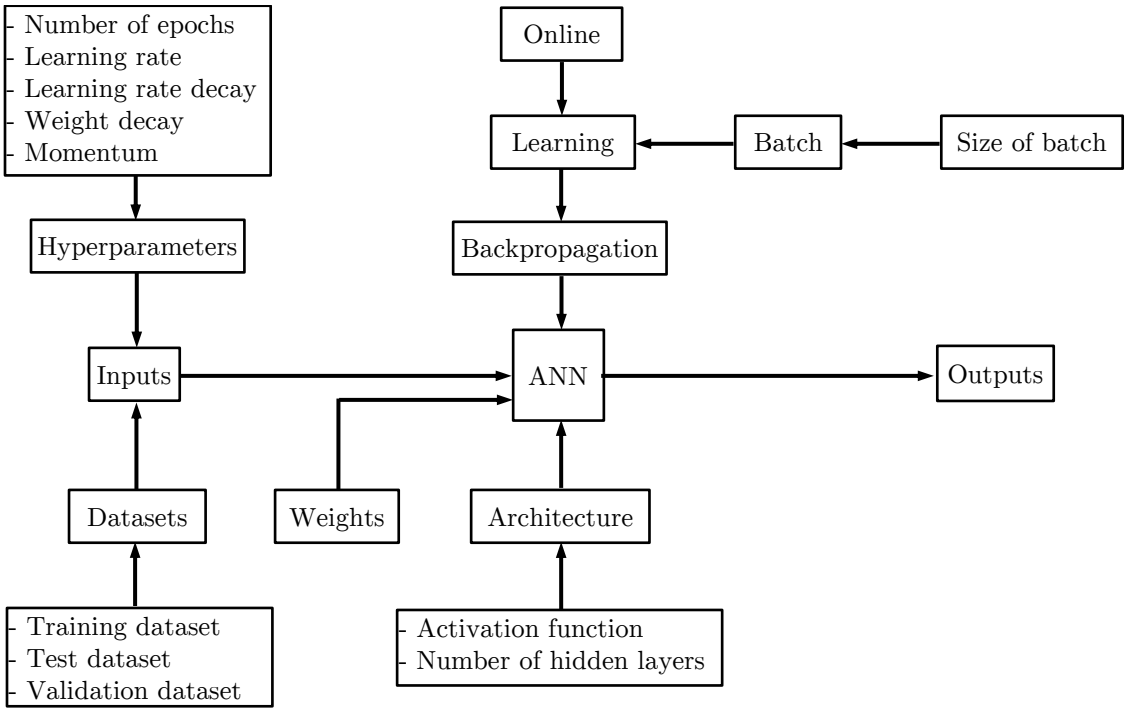
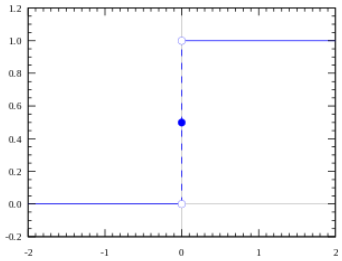


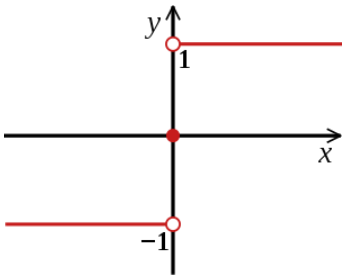
Figure 19. Relationships between ANN building blocks.

A neuron is connected to other neurons via its input and output links. Each incoming neuron has an activation value and each connection has a weight associated with it. The neuron sums the incoming weighted values and this value is input to an activation function. The output of the activation function defines the output from the neuron. Activation functions are differ: there are some common activation functions (Figure 20) and some other activation functions used in practice (Figure 21). For the input layer linear activation functions are generally used, while for the hidden layers non-linear activation functions are generally used, which for the output layer linear activation are recommended (however, it is also possible to use non-linear activation functions for the output layer).



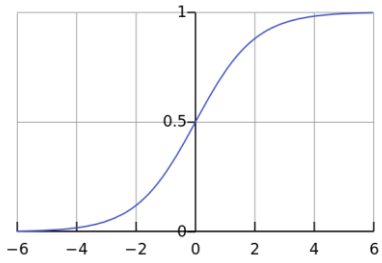
Step function

$$f(x) = \begin{cases} 1, \, if \, x \geq 0 \\ 0, \, if \, x < 0 \end{cases}$$



Sign function

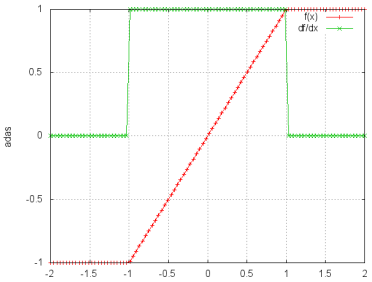
$$f(x) = \begin{cases} -1, \, if \, x < 0 \\ 0, \, if \, x = 0 \\ 1, \, if \, x > 0 \end{cases}$$



Sigmoid function

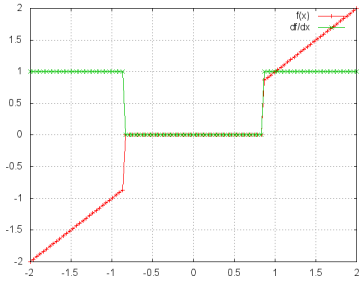
$$f(x) = \frac{1}{1 + e^{-x}}$$

Figure 20. Some common activation functions.



HardTanh

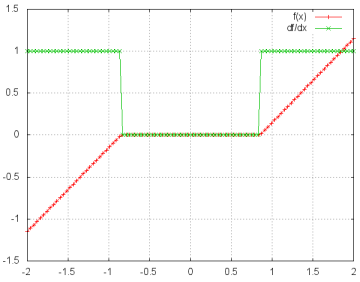
$$f(x) = \begin{cases} 1, \, if \, x > 1 \\ -1, \, if \, x < -1 \\ x, otherwise \end{cases}$$



HardShrink

$$f(x) = \begin{cases} x, \, if \, x > lambda \\ x, \, if \, x < -lambda \\ 0, otherwise \end{cases}$$

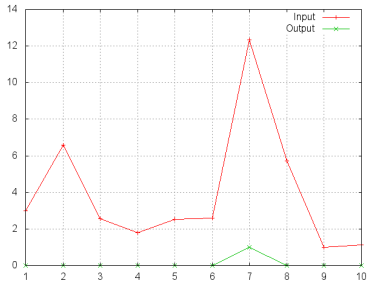
where *lambda* is 0.5 by default



SoftShrink

$$f(x) = \begin{cases} x - lambda, \, if \, x > lambda \\ x + lambda, \, if \, x < -lambda \\ 0, otherwise \end{cases}$$

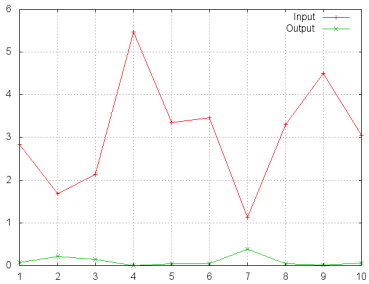
where *lambda* is 0.5 by default



SoftMax

$$f(x)_i = \frac{e^{x_i - shift}}{\Sigma_{j=1}^{N} e^{x_j - shift}},$$

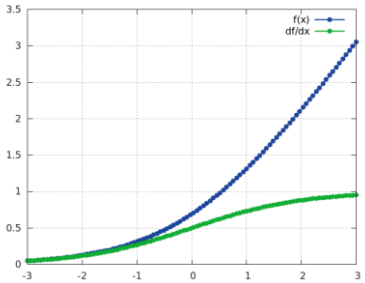
where $shift = \max_i(x_i)$



SoftMin

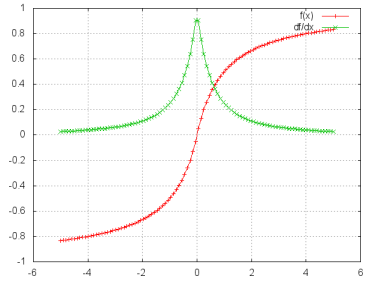
$$f(x)_i = \frac{e^{-x_i - shift}}{\Sigma_{j=1}^{N} e^{-x_j - shift}}$$

where $shift = \max_i(-x_i)$

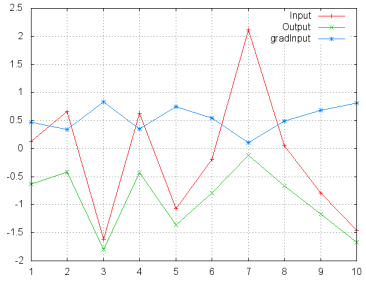


SoftPlus

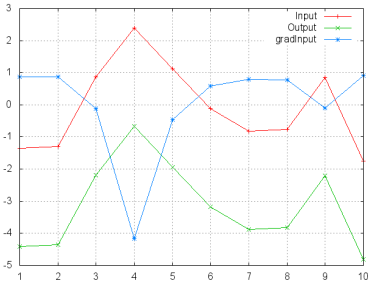
$$f(x) = \frac{1}{beta} * \log(1 + e^{beta * x_i})$$



SoftSign

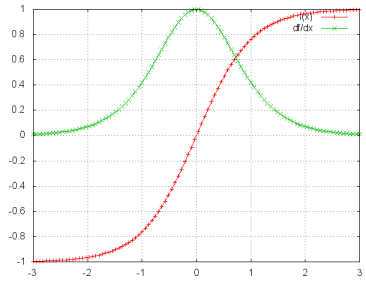


LogSigmoid



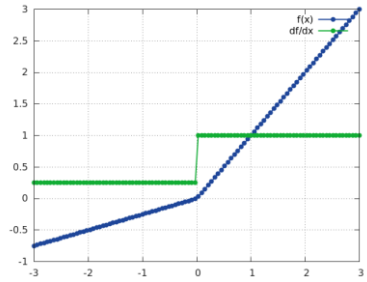
LogSoftMax

$$f(x) = \frac{x_i}{1 + |x_i|}$$



Tanh

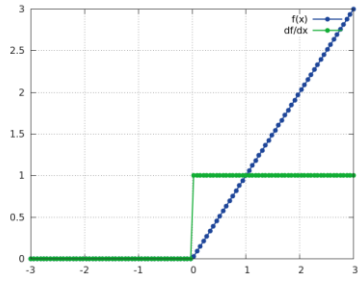
$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



PReLU

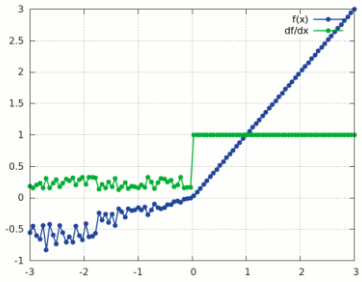
$f(x) = \max(0, x) + a * \min(0, x)$

$$f(x) = \log(\frac{1}{1 + e^{-x_i}})$$



ReLU

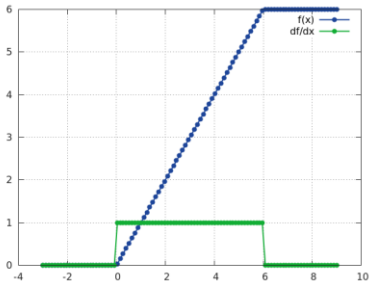
$f(x) = \max(0, x)$



RReLU

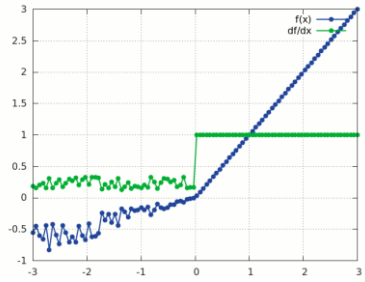
$f(x) = \max(0, x) + a * \min(0, x)$, where $a \approx U(l, u)$, U is random distribution, l=1/8 and u=1/3 by default

$$f(x)_i = \frac{1}{\Sigma_{j=1}^{N} e^{x_j} * e^{x_i}}$$



ReLU6

$f(x) = \min(\max(0, x), 6)$



RReLU

$f(x) = \max(0, x) + a * \min(0, x)$, where $a \approx U(l, u)$

Figure 21. Activation functions used in practice.

To select the best activation function there is ideally a need to know the certain characteristics of the function of the process, which is approximated by use of neural network. Then selected activation function will approximate the function of the process faster leading to faster training process. For example, a sigmoid activation function performs well for a classification, because approximating a classifier function as combinations of sigmoid is easier than other activation functions or their combinations [227]. Therefore, usage of the sigmoid activation function will lead to faster training process and convergence. If there is no (or very minimal) knowledge of the function of the process (nature of the process), then there is a general suggestion to select ReLu activation function, because it is considered that ReLu performs well most of the time as a general purpose approximator.

Learning is implemented with several algorithms. The most used algorithm is backpropagation learning algorithm. The backpropagation learning algorithm [219,228,229] was developed independently by Rumelhart [230] (simplest derivation), Le Cun (alternate derivation [231] and several variations of the original algorithm [232,233], which have been also reported earlier by Parker and Werbos [234,235]). Error backpropagation networks are the most widely used neural network model as they can be applied to almost any problem that requires pattern mapping. It was the discovery of this paradigm that brought neural networks out of the research area and into real world implementation.

ANN learning approaches are divided into two categories:

1. Supervised learning – approach, where 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 also called «labeled» sample.

2. Unsupervised learning – approach, where each training sample is only an input object (typically a vector), i.e., without a desired output value. Such sample also called «unlabeled» sample.

These learning approaches are used to solve the following general tasks (Figure 22):

- Regression – approximation and generalization of an unknown multivariate (usually nonlinear) function (so-called objective function), which values only at some points are given, over a mechanism of interconnected (usually nonlinear) activation functions.
- Classification (categorization) – organizing/splitting data into several categories (classes).
- Clustering – the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).
- Dimensionality reduction [236] – process of reducing the number of random variables under consideration by obtaining a set of principal variables.

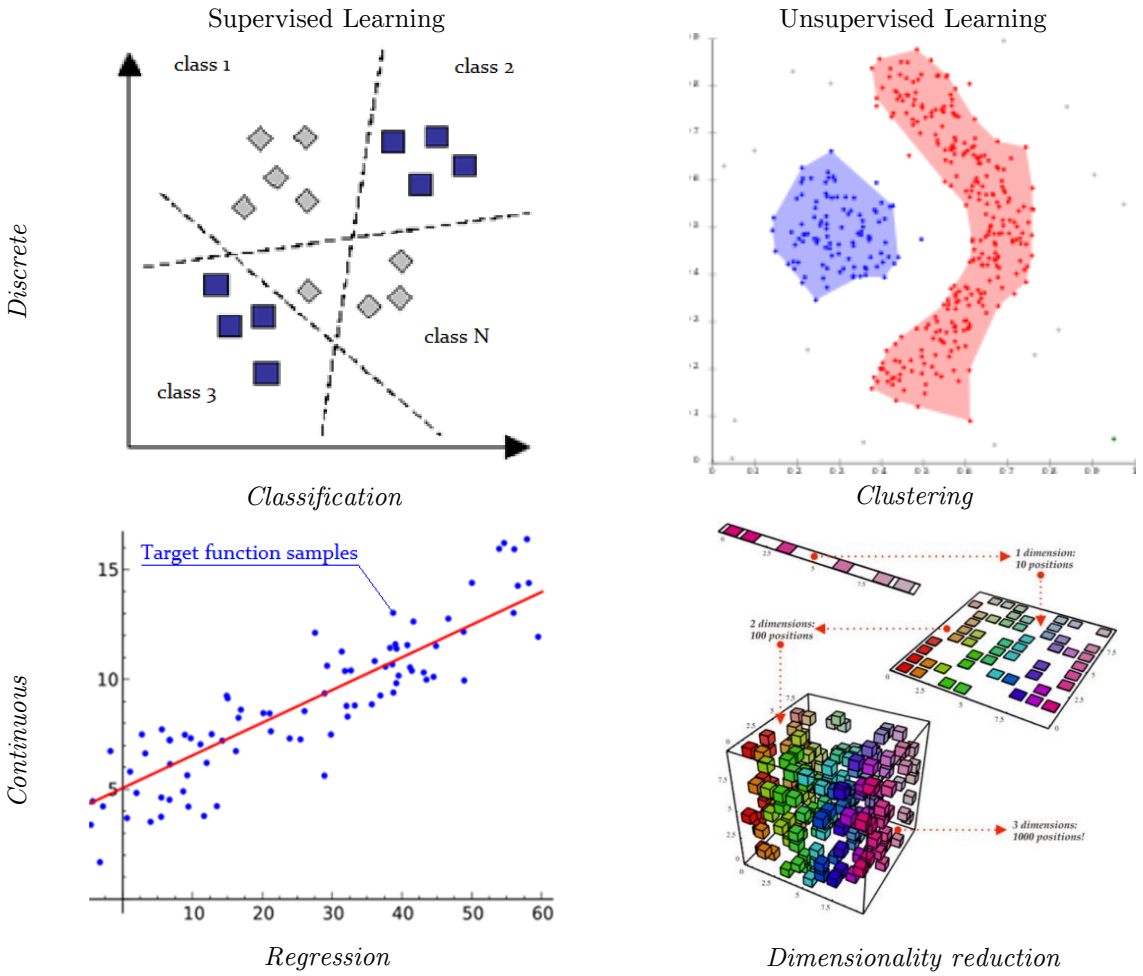


Figure 22. ANN tasks illustration.

Usually supervised learning of multilayered neural networks with conventional learning algorithms has a problem of local minimum. During training of the neural network using a training set of input-output pairs gradient descent-type learning algorithms (including backpropagation learning algorithm) update the connection weights of the neural network without any prior knowledge. Using a gradient descent algorithms to adjust the weights involves a situations, where training of neural network gets stuck in the local minima (Figure 23). A several studies address this problem by usually exploring the combination of neural network appearance and the learning parameters in local minima-free condition.

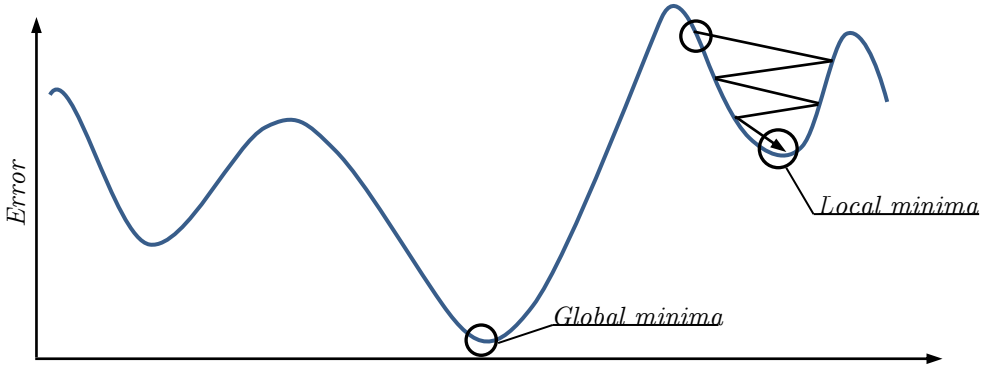


Figure 23. Global minima and local minima.

There are several categories of the ANNs, which are differ usually in their architecture and the tasks they are targeted. Some of the frequently used in practice ANN types (Figure 24) are:

- Feed-forward ANNs. These networks allow signals to travel one way only: from input to output. There are no feedback (loops); i.e., the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straightforward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred as 'bottom-up' or 'top-down' organization. The well-known types of feed-forward ANNs are:
  - Single Layer Perceptron (SLP) is an algorithm for supervised learning of binary classifiers (functions that can decide whether an input, represented by a vector of numbers, belongs to some specific class or not).
  - Multilayer Perceptron (MLP) is a multilayer version of the SLP.
- Feedback networks (also called 'recurrent networks' or 'interactive networks'). These networks can have signals traveling in both directions by introducing loops in the network. Feedback networks are powerful and can get extremely complicated. Computations derived from earlier input are fed back into the network, which gives them a kind of memory. Feedback networks are dynamic in a sense that their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. The well-known types of feedback networks are:
  - Self-Organizing Map (SOM) or Self-Organizing Feature Map (SOFM) is a type of ANN that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map, and is therefore a method to do dimensionality reduction.
  - Bayesian Regularized Artificial Neural Networks (BRANNs) are more robust than standard back-propagation neural networks and can reduce or eliminate the need for lengthy cross-validation. Bayesian regularization is a mathematical process that converts a nonlinear regression into a «well-posed» statistical problem in the manner of a ridge regression. The advantage of BRANNs is that the models are robust and the validation process, which scales as *O(N2)* in normal regression methods (such as back propagation) is unnecessary.

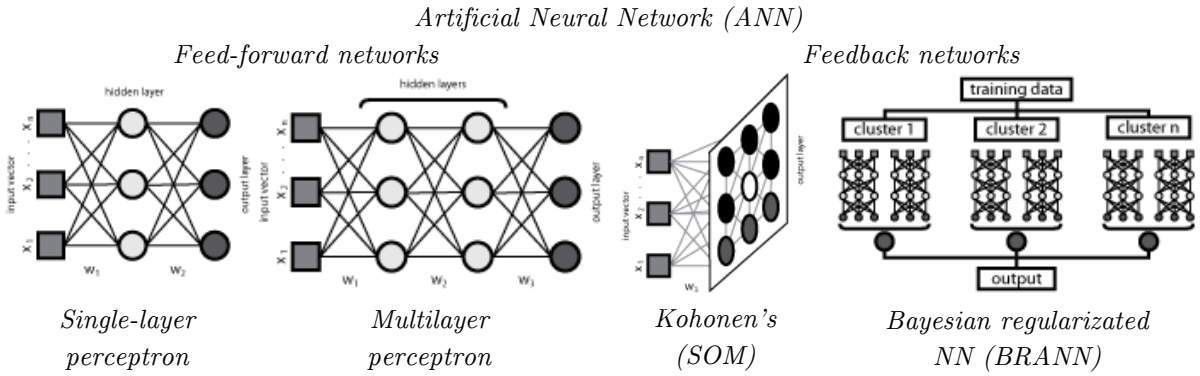


Figure 24. Artificial neural network types illustration.

Other more application-specific ANN types include:

- Convolutional Neural Networks (CNN) – class of deep, feed-forward ANNs that has successfully been applied to analyzing visual imagery.
- Deep Neural Networks (DNN) – ANN with multiple hidden layers between the input and output layers.
- Deep Belief Networks (DBN) – generative graphical model, or alternatively a class of deep neural network, composed of multiple layers of latent variables ('hidden units'), with connections between the layers but not between units within each layer.
- Convolutional Deep Belief Networks (CDBN) – type of DNN that is composed of multiple layers of convolutional restricted Boltzmann machines stacked together.
- Deep Boltzmann Machines (DBM) – type of binary pairwise Markov random field (undirected probabilistic graphical model) with multiple layers of hidden random variables.

A particular ANN, which has only 1 hidden layer can be considered as 'simple'. However, even with 1 hidden layer such ANNs can (for example) recognize handwritten digits with accuracy more than 98%. If such an ANNs has more than 1 hidden layer, it can be considered as 'complex' or so-called Deep Neural Network (DNN, Figure 25). Such networks use the intermediate hidden layers to solve complex pattern recognition problems. For example, in case of image processing such a DNNs can recognize more complex shapes as: edges, triangles or rectangles, built up from edges. However, in contrast with simple 1-hidden layer ANNs, training of DNNs is considered to be more complex and time-consuming [237].

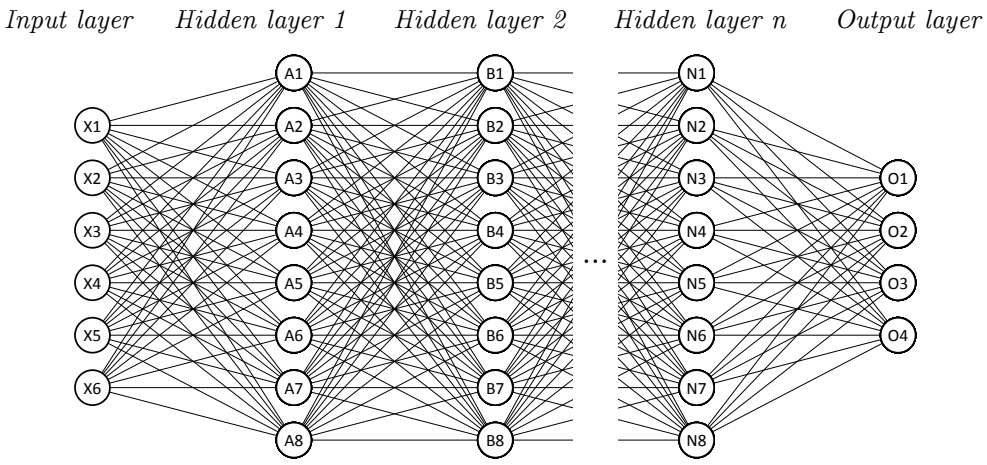


Figure 25. Deep neural network illustration.

One fundamental parameter in ANNs is 'learning rate'. During training of ANNs by gradient descent algorithm, each iteration applies backpropagation to calculate the derivative of the loss function with respect to each weight followed by subtraction of the derivative from this weight. However, if such

approach is directly applied, the weights will change far too much each training iteration, which will make them «overcorrect» and the loss will actually increase (Figure 26). Therefore, to prevent such effect, in practice each derivative is multiplied by a small value, so-called 'learning rate', before it is subtracted it from its corresponding weight. Other interpretation – learning rate determines *how fast* weights of an ANN change. If learning rate is constant for all the layers there is a potential problem of 'vanishing gradient', which consists in the fact that at some point weights may start changing very little or stop changing at all. To avoid this problem learning rate may vary from layer to layer.

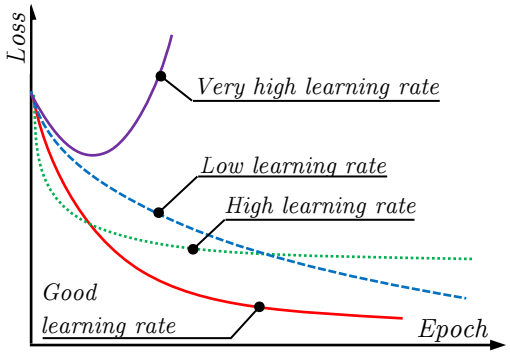


Figure 26. Learning rate illustration.

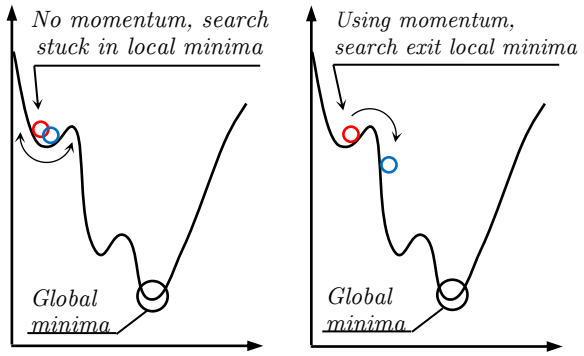


Figure 27. Momentum illustration.

Another fundamental parameter in ANNs is 'momentum'. If an ideal world the error function has no global minima, gradient descent optimization algorithm will always minimize the error function to reach this global minima. However, in practice the error surface is more complex and may contain several local minimas (Figure 27). In this case, gradient descent optimization algorithm can get stuck in a local minima, considering that global minima is reached, which will, of course, lead to sub-optimal results. To avoid this scenario, a special parameter 'momentum' is introduced. Momentum is a value between 0 and 1, which is used to increase the size of the step taken towards the minimum by trying to «jump» from a local minima. It is usual practice to make learning rate smaller, when momentum value is relatively large. A large momentum value also means that the convergence will happen fast. However, a small momentum value cannot reliably avoid local minima, and can also slow down the training. Momentum also targets in smoothing out the variations, in case if the gradient keeps changing direction. A right value of momentum can be obtained by 'hit and trial' or via cross-validation (split the entire dataset set into *training dataset* and a *validation dataset*).

Hyperparameters are considered to be «magic numbers», because they are values of the ANN model set before training. Examples include: number of hidden layers in a DNN and learning parameters (learning rate, momentum). Changing hyperparameter values by just a small amount can have (and usually has) a huge impact on the performance of the ANN. The process of searching the best combination of hyperparameters is commonly known as 'hyperparameter optimization'.

A several techniques are used for hyperparameter optimization (Figure 28):

- Grid search – technique, which is in basic sense, a brute force method to estimate hyperparameters. For example, if there are number hyperparameters, and each one of them have several possible values, then, performing grid search is basically taking a Cartesian product of these possible values, which makes a grid search quite costly, because expense grows exponentially with the number of hyper parameters and the number of discrete levels of each. However, grid search technique can be well-parallelized resulting in a time decrease.
- Random search – technique, which uses a random combinations of hyperparameter values. Due to the known fact that some of the hyperparameters actually have little effect (or no effect) on the model for certain data sets, it is useless to evaluate all their combinations, especially for high-dimensional hyperparameter spaces. It was demonstrated [225], that

given the disparity in the sensitivity of model accuracy to different hyperparameters, a set of candidates that incorporates a larger number of trial values for each hyperparameter will have a much greater chance of finding effective values for each hyperparameter. As a result, instead of focusing on Cartesian product of hyperparameter values, studying (or estimating) a several random combinations enables to explore more values of each hyperparameter at the same cost.

– Random Latin hypercube – technique, which is based on Latin hypercube sample (LHS) [238] and basically is an approach in which samples are exactly uniform across each hyperparameter, but still random in combinations. These so-called 'low-discrepancy' point sets attempt to ensure that points are approximately equidistant from one another to fill the space efficiently. This sampling allows for coverage across the entire range of each hyperparameter and is more likely to find good values of each hyperparameter.

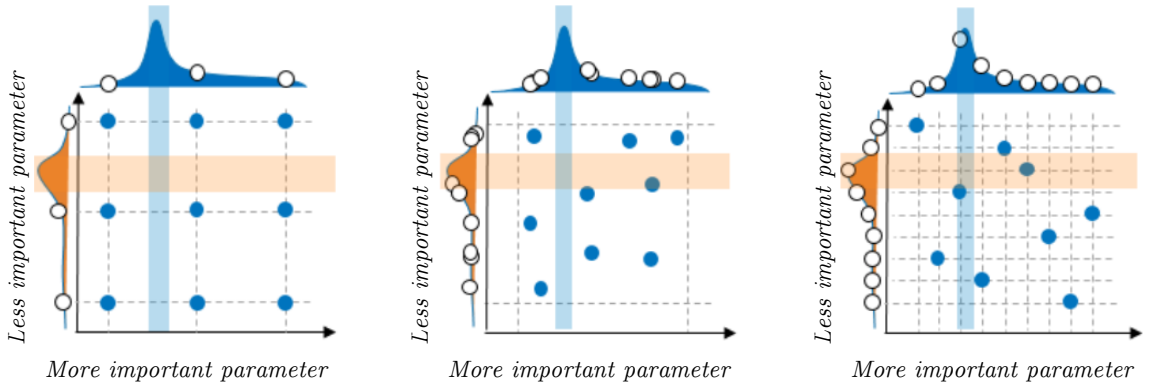


Figure 28. Hyperparameter optimization: grid search, random search, random Latin hypercube.

Weight initialization can have a profound impact on both the convergence rate and final quality of results produced by the ANN. There are several strategies to initialize ANN weights. To evaluate what happens under each particular weight initialization strategy, it is common to visualize outputs of each neuron as a dataset passes through the network. Usually most layers make it straightforward to initialize weights randomly from a uniform distribution over [-std, std]. However, the determination of the range to use has been the subject of a lot of research and often considered to be the key to efficient learning in a computational network. For example, in case of backpropagation learning algorithm, there is 'efficient backprop' [239] weight initialization approach for linear and convolutional layers, where initial weight values are randomly selected from $U[-1/\sqrt{fan\_in},1/\sqrt{fan\_in}]$, where in case of CNN $fan\_in = kW * kH * nInputPlane$, where kW/kH – convolution kernel width/height, $nInputPlane$ – number of expected input planes (explained in details in the next section) in the image given to *forward* function (implements forward pass).

Usually before perform training with the ANN there is some pre-processing applied to the input data. Such a pre-processing can be:

1. Input data normalization and scaling. This is intended for faster approaching to global minima at error surface.
2. Input data denoising. Raw data obtained from the physical sensors (for example, camera sensor) is usually noisy.
3. Bit depth reduction. Used to reduce the range of values used as an input for the ANN to accelerate the training.

Artificial neural networks can be viewed from a different angle. Since multilayer perceptrons are nonlinear regression and discriminant models they can be implemented via statistical methods and tools.

Therefore, neural network terms and definitions can be translated into statistical methods terms and definitions to establish a relationships between them (Table 2).

Table 2. Comparison between terms in literature: ANNs vs. statistical methods.

Term in ANN literature	Term in statistical methods literature
features	*variables*
inputs	*independent variables*
outputs	*predicted values*
targets (training values)	*dependent variables*
errors	*residuals*
higher-order neurons	*interactions*
generalization	*interpolation and extrapolation*
functional links	*transformations*
(synaptic) weights	*parameter estimates*
competitive learning or adaptive vector quantization	*cluster analysis*
patterns or training pairs	*observations*
training, learning, adaptation, or self-organization	*estimation*
error function, cost function, or Lyapunov function	*estimation criterion*
Supervised learning or heteroassociation	*regression and discriminant analysis*
unsupervised learning, encoding, or autoassociation	*data reduction*

Artificial intelligence has to some fundamental differences in philosophy between neural network engineers and statisticians. While neural network engineers see neural networks as a 'data in – predictions out black boxes' with no human intervention, statisticians are tend to deeply understand the process under study, generate hypotheses and models, create test assumptions, diagnose problems in the model and data, and show results in a comprehensible way, with the goal of explaining the phenomena being investigated. In general, neural networks include several models, such as MLPs, that are useful for statistical applications. Statistical methodology is directly applicable to neural networks in a variety of ways, including estimation criteria, optimization algorithms, confidence intervals, diagnostics, and graphical methods.

2.6.2 Multilayer Perceptron and Convolutional Neural Network

A multilayer perceptron (MLP) is a class of feedforward ANN. An MLP consists of at least three layers of nodes (Figure 29). Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training [240,241]. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

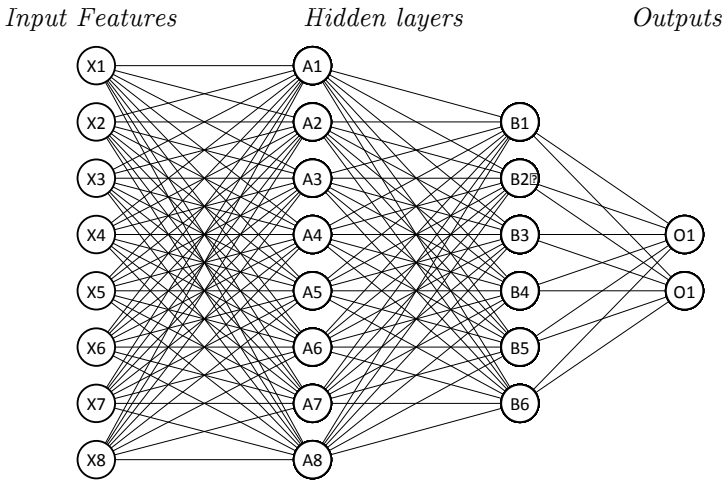


Figure 29. Multilayer perceptron illustration.

A convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward ANNs that has successfully been applied to analyzing visual imagery [242]. CNNs are very similar to ordinary NNs: they are made up of neurons that have learnable weights and biases. The main difference is that ConvNet architectures make the explicit assumption that the inputs are *images*, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network. In particular, unlike a regular NN, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. For example, the input images in CIFAR-10 [243] are an input volume of activations, and the volume has dimensions 32x32x3 (width, height, depth respectively). A simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. To build ConvNet architectures three main types of layers are used: convolutional layer, pooling layer, and fully-connected layer (exactly as seen in regular NNs).

A typical example of ConvNet architecture (Figure 30), classifying RGB images over 4 animal categories (horse, cat, dog, bird), consists of the following components:

- Input layer [32x32x3] – layer, which holds the raw pixel values of the image, in this case an image of width 32, height 32, and with 3 color channels (also called in ConvNet terminology 'input planes'): R, G, B.
- Convolution layer №1 – layer, which computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32x32x12] if there is a decision to use 12 feature maps (also called 'output planes').
- ReLU layer №1 – layer, which applies an ReLU activation function, defined as *max(0,x)*. This leaves the size of the volume unchanged: [32x32x12].
- Sub-sampling layer №1 (also called 'pooling layer') – layer, which performs a downsampling (decimation) operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].
- Convolution layer №2 – resulting in volume such as [16x16x24] if there is a decision to use 24 feature maps.
- ReLU layer №2 – leaves the size of the volume unchanged: [16x16x24].
- Sub-sampling layer №2: resulting in volume such as [1x1x24].
- Fully-connected layer – layer, which computes the class scores, resulting in volume of size [1x1x24], where each of the 6 numbers (24/4) correspond to a class score, such as among the 4 animal categories (or 10 categories in case of CIFAR-10): horse, cat, dog, bird.

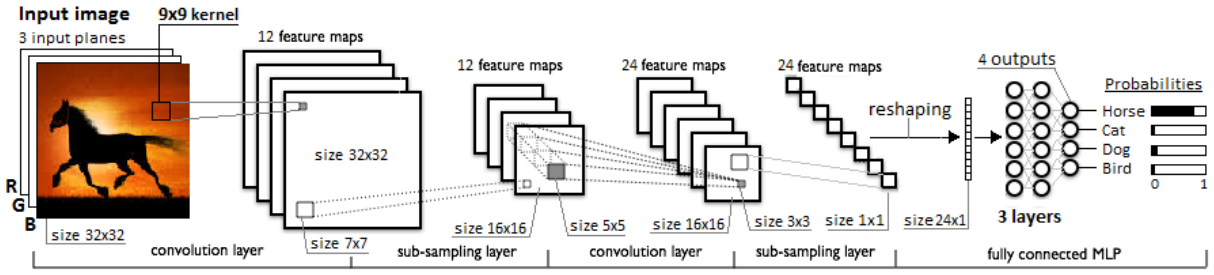
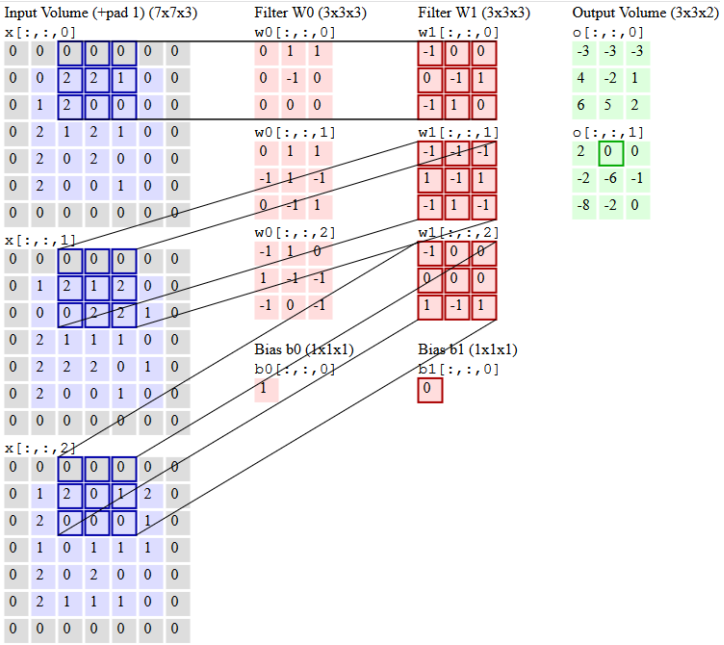


Figure 30. Convolutional neural network architecture illustration.

The *feature map* is the output of one filter (also called 'kernel') applied to the previous layer. A given filter is applied across the entire previous layer, moved one pixel at a time (Figure 31, a). Each position results in an activation of the neuron and the output is collected in the feature map, which is basically the output image of each convolutional layer (Figure 31, b).



a. Convolution visualization

b. Example of feature maps

Figure 31. Convolution visualization and example of feature maps.

2.6.3 Some Related Work in ANNs/CNNs/DNNs

It is known fact that deep CNNs have recently achieved enormous success in many visual recognition problems. However, there are still number of open issues:

1. Implementations of deep CNNs models are computationally intensive.
2. Implementations of deep CNNs models are memory consuming.
3. Training of DNNs takes significant time.
4. Overfitting and generalization issues.
5. Hyperparameters optimization takes significant time.
6. Need to collect huge volumes of training data.

During the past few years, tremendous progresses have been done to address these problems.

To address the issue of computational expensiveness and high memory consumption a tremendous progresses have been done during the past few years. The core ideas are concentrated around model compression and acceleration [244] without significantly decreasing its performance. For example, the recent advanced techniques for compacting and accelerating CNNs models are roughly categorized into

four schemes: parameter pruning and sharing, low-rank factorization, transfered/compact convolutional filters and knowledge distillation.

To address the issue of significant time needed to training, there are several approaches have been proposed, each if which addresses a particular issue. Example of such an issues include:

- Internal covariate shift – change in the distribution of network activations due to the change in network parameters during training [245]. If the input statistical distribution keeps changing, the hidden layers will keep trying to adapt to that new distribution hence slowing down convergence. To address this issue a special technique was proposed called 'batch normalization' (BN). BN normalizes the inputs for each hidden layer so that their distribution is fairly constant as training proceeds. As a result, convergence of the ANN is improved, which leads to training time reduction.
- Vanishing gradient – a situation, when during weight update (proportional to the gradient of the error function with respect to the current weight) gradient is vanishingly small, effectively preventing the weight from changing its value. It was shown that saturating nonlinearities (like *tanh* or *sigmoid*) cannot be used for DNNs as they tend to get stuck in the saturation region as the network grows deeper. Some solutions include: use of nonlinearities which do not saturate like ReLU, use of smaller learning rates, use of several weight initialization approaches.
- Normalization (or scaling) – transposition of the input variables into the data range that the particular activation functions lie in. It was shown [246] that input data normalization with certain criteria, prior to a training process, is crucial to obtain good results as well as to fasten significantly the calculations.

Overfitting and generalization is a phenomenon, when trained ANN produces high quality output results on the training dataset, but produces low quality output results on a new data. In other words, the model does not generalize well its output results. Overfitting becomes an even more significant issue in DNNs, where NN has large numbers of layers containing many neurons. Regularization is one method to minimize overfitting issue. Regularization modifies the objective function that is minimized by adding additional terms that penalize large weights. The most common type of regularization is so-called 'L2 regularization' [247]. It can be implemented by augmenting the error function with the squared magnitude of all weights in the neural network. Overfitting can be also reduced by using 'dropout' to prevent complex co-adaptations on the training data [224,248].

In terms of ANNs implementation there is also visible progress done. At the beginning ANNs were implemented using single core CPUs followed by gradual migration to multicore CPUs. However, to accelerate training process of neural network even more, a several alternative platforms are widely used nowadays such as general-purpose GPUs [249] or an FPGAs [250] as well as application-specific ASICs [251]. ANN, which is implemented in the embedded platform (such as MCUs/FPGAs/ASICs) is called 'embedded artificial neural network' (or 'embedded ANN'). There are also several implementations of embedded ANNs reported [252,253] over the last years.

2.7 Eye Tracking System Application Specifics

The complete EyeDee™ eye tracking system must be responsive (duration of one eye image processing is $<$10 ms), because the speed of eye movements is considered [254] to be the fastest among other human body movements. This system includes transmission of eye images over a medium (wire/wireless communication channel). Such transmission takes a certain amount of time, which directly contributes to increase of EyeDee™ responsiveness, i.e., delay between eye image acquisition and output results from the eye tracking algorithm. To reduce eye image transmission time, initial eye image (taken from a miniaturized digital camera sensor) can be compressed, which results in less bytes to transmit over the selected medium. Also initial eye image contains (due to nature of capturing) spatial redundancy [255,256]. This spatial redundancy can be reduced to a certain level, which results in slight eye tracking

precision degradation, but at the same time does not exceed a maximum allowed eye tracking precision error (Figure 32). The spatial redundancy can be reduced by applying lossy image compression (i.e., compression with information loss). The main goal of this reduction is to find a profile (window), which conforms to '20-80' principle, i.e., a point where eye tracking responsibility is 80% increased at the expense of 20% eye tracking precision loss.

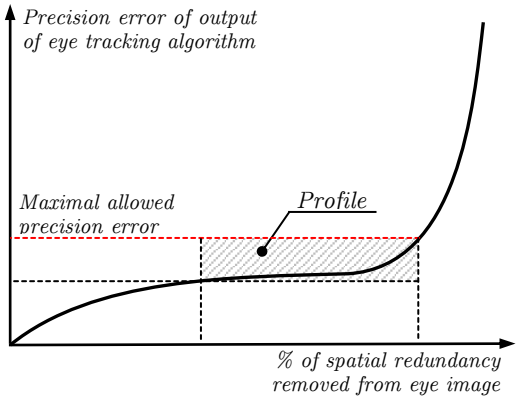


Figure 32. Profile: removing eye image spatial redundancy vs. eye tracking precision error.

Therefore, in case of EyeDee™ eye image compression is aimed to:

- Reduce (or eliminate) spatial redundancy from eye image (input signal) without losing needed details (pupil edges) to keep eye tracking precision in a desired range. As a result, images will be sent in compressed (compacted) form and less bytes will be transmitted over the selected medium.
- Reduce time of image transmission from Weetsy™ board to PC-based processing unit over the selected medium.

Each image compression system can be characterized by following main criteria (most prioritized are presented first):

- Time of image compression – delay between loading input image into the compression system and obtaining output compressed image, which is represented in a form of sequence of bits (so-called bitstream).
- Size of compressed image – amount of bits, which are used to represent compressed image (bitstream). Usually size of bitstream is measured in bytes. Division of bitstream size by number of pixels of the source input image gives bits-per-pixel (bpp) metric, which is usually used to characterize compressed image. The bpp metric can be applied as an input of a compression system instead of compression ratio. The bpp metric is used during designing of the data transmission systems involving compression, especially during estimation of so-called bit-budget (usually represented in a form of bits-per-second, usually Kbps or Mbps are used) which is needed to be reserved in a communication channel to transmit compressed images at a desired frequency (represented as number of image frames per second or FPS metric).
- Quality of decompressed image – measured similarly between original (uncompressed) and reconstructed (decompressed) image. Usually quality is represented via peak signal-to-noise ratio (PSNR) [257] metric or mean square error (MSE) [258]. However, these metrics are not always precisely characterize similarity (also used 'similarity difference' or 'similarity delta') between the images. Thus, to solve this issue, other similarity metrics are also applied, such as SSIM, MS SSIM [259], hamming distance between two hashes of the images (this approach does not require images to be the same size).

An additional criteria, which are also considered during image transmission system design include:

- Computational complexity [260] – number of processor instructions (operations), which is needed to perform image compression. Usually computational complexity is represented via million instructions per second (MIPS) metric.
- Memory consumption – amount (in bytes) of operating memory (RAM), which is needed to execute image compression algorithm.
- Parallelization – ability of parallel execution [261] of certain algorithms of image compression building blocks due to presence of independent data flows inside image compression system. A well-known example is parallel execution of discrete cosine transforms (DCT) due to its independent applying to image blocks.
- Ability of implementation in hardware – potential presence of properties, which are considered during hardware implementation, i.e., parallel execution of independent building blocks, absence of floating-point arithmetic (only integer arithmetic is used). In the last case implementations of floating-point arithmetic based algorithms are converted to implementations of integer arithmetic based algorithms, while providing the same results.

Therefore, applying of a particular image compression system can be viewed as an optimization problem, i.e., performing codecs comparison (Figure 33) to find the best combination (for a particular application) of their main criteria.

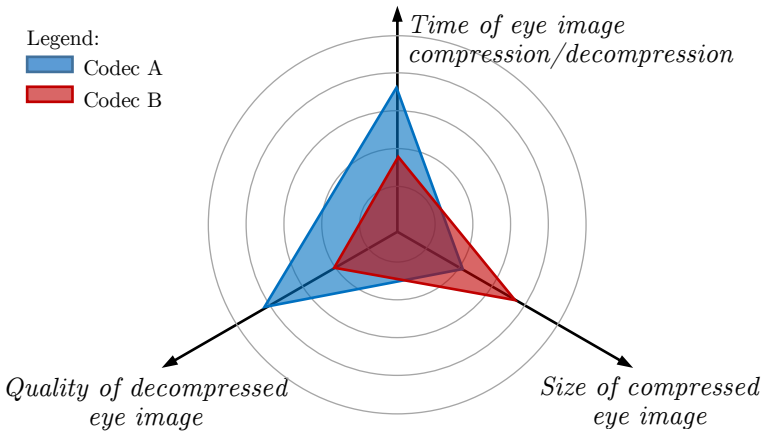


Figure 33. Comparison of several codecs in terms of time/size/quality.

2.8 Computational Complexity and Implementation Aspect

Nowadays, most embedded and portable image processing applications require low power consumption and wireless communication and face a common problem of limited CPU resources and limited battery. In most cases, these image processing applications have a following processing chain:

- Image retrieval from the digital camera sensors (readout);
- Image processing in real-time to extract relevant features by applying computer vision algorithms;
- Application-specific processing of the extracted relevant features. For example, finding of the parameters of a physical model relevant to the application (for example, SLAM for reconstruction of the 6 degrees of freedom of an object, ellipse fitting for eye tracking, model fitting, etc.);
- Send the results of application-specific processing to the end user application.

Image processing algorithms can be highly demanding in computational resources and can often be implemented in different platforms (CPU/GPU/FPGA) to gain the benefit of each platform. For example, FPGA/GPU are primarily used for parallel execution of independent building blocks, which usually contains such an implementation (usually approximation of original algorithm) of algorithms, which uses integer arithmetic rather than floating-point arithmetic. In contrast, CPU has the Floating

Point Unit (FPU), which is used to perform floating-point operations of these image processing algorithms. Communication stacks (such as USB or TCP/IP) can introduce bottlenecks usually due their limited implementations. Usually embedded IPs of communication standards inside an MCU are relatively simple or limited functionality, which does not allow take advantage of their full versions (which are usually positioned on the market as an independent standalone IPs). Another reason of potential communication bottlenecks is 'misbalance of bitrate of IOs', i.e., a situations inside an MCU, when bitrate of data reading external RAM is much less than bitrate of data transmission via particular communication IP. Hence in such cases, since communication IP operate in parallel with CPU (via issuing an asynchronous request, which is followed by waiting on the confirmation of the operation completion), after physical completion of the operation by IP, it stays in standby mode waiting for completion of the CPU operation. As a result, final performance is significantly degraded. Usually these bottlenecks are relatively hard to find and not trivial to resolve. Hard to find because they tend to appear in execution of combination of the different IPs with asynchronous API, while their independent execution can give expected results. Not trivial to resolve, because there is a need of careful and accurate performance analysis (usually requires use of commercial tools), which is followed by such an implementation, which leads to balanced/optimal resources utilization (optimal hardware utilization). Several example of embedded image acquisition/processing systems presented in [262,263].

In the case of SuriCog's EyeDee™ solution, the embedded system should be able to capture at high frequency (100Hz) the image of user's eye (VGA resolution, 8bpp), and broadcast wirelessly in real-time to the end application the result of the processing algorithm, which consists in the parametrization of a 3D model of the eye. The system should run continuously during more than 3 hours, with the lowest latency possible (typically <10ms). Three options are possible:

1. Locally read the sensors and send the resulting image to the end application, which executes of the full algorithm on the client's machine;
2. Locally read the sensors, locally execute the full algorithm and send the results to the client's machine;
3. Locally read the sensors, pre-process the image and send these preprocessed images to the client machine for final processing.

The first option is constraint by the limited bandwidth of the wireless channel (Wi-Fi, Bluetooth) and the latency/quality loss introduced by standard compression/decompression algorithms. The second option is constraint by the limited resources of CPU, battery and power dissipation required to run the algorithms at full speed. The third option, described in this thesis, introduces a preprocessing phase, which can be viewed as a "smart compression", i.e., compressing of the image to select its relevant features, which are required by the final algorithm. In the case when an eye tracker uses so-called dark pupil technique the features of interest are the points that lie on the edges of the pupil's quasi-ellipse. SuriCog's EyeDee™ eye image acquisition system is based on Weetsy™ board hardware (Figure 34), which includes the following components:

1. Microcontroller Unit (MCU) – used for input/output communication, eye image pre-processing, using of FPU for floating-point arithmetic.
2. Field-Programmable Gate Array (FPGA) – used for digital camera sensor readout, hardware processing, acceleration of the parts of the eye tracking algorithms initially implemented in PC-based processing unit.
3. Wireless module – used for wireless sensing of eye images over a network to PC-based processing unit following eye tracking via EyeDee™ software. Several data transmission standards are supported: Wi-Fi, Bluetooth (including Bluetooth LE) and ZigBee.
4. External RAM – used to store intermediate results of the eye image pre-processing.

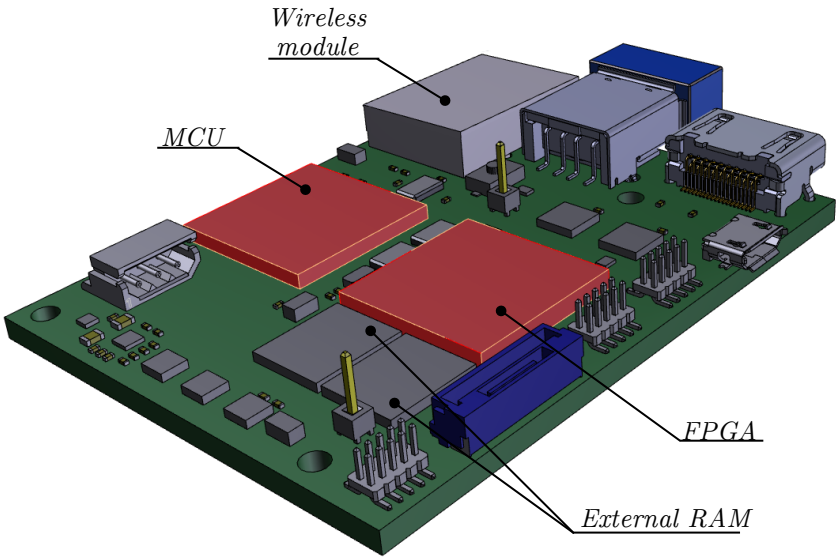


Figure 34. Weetsy™ board hardware.

The EyeDee™ embedded eye tracking solution developed by SuriCog is aimed to conform to several requirements, where the most important among others are low-power consumption [264], low-heat generation, low EM radiation [265], low-MIPS [266], as well as support of wireless data transmission and space efficient form factor. The problem consists in the deployment of computationally intensive image processing based eye tracking algorithm on a combination of resources-restrained embedded platform and a personal computer (PC). Therefore, the problem consists in finding such a balance, which leads to optimal computing resources utilization. This concept can be illustrated (Figure 35) as a «slider» principle between splitting of the complete eye tracking algorithm computations over two resources: embedded software (firmware) and PC-based desktop software.

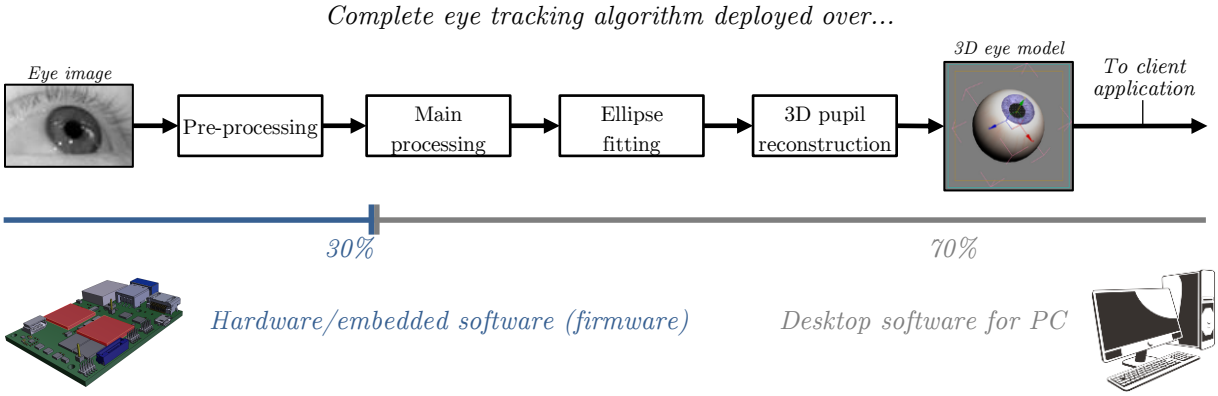


Figure 35. «Slider» principle of deployment of the complete eye tracking algorithm.

2.9 Conclusion

The EyeDee™ embedded eye tracking solution developed by SuriCog contains a wearable device which performs acquisitions of the eye image at very high frame rate. Since eye image is a discrete 2D signal in spatial coordinates (x,y), it is possible to apply image processing tools to compact energy of this signal: such as spatial-to-frequency domain wavelet based transforms, followed by compression of the resulted coefficients. Since EyeDee™ does not involve viewing of the eye image by the user, usual advantages of human visual system (HVS), such as color squeezing (reduction) or small details removal, cannot be directly applied because they will result of eye tracking quality degradation. Modern Video compression techniques such as H.264 and H.265 are difficult to apply, because of several issues related to high computation complexity, high (for real-time eye tracking) encoding/decoding delays, need of

guaranteed data delivery of data transmission, relatively high cost of available hardware implementations (IP cores) and some others. Image compression techniques such as JPEG and JPEG 2000 allow images to be compressed independently, hence it is possible to use best effort data delivery (UDP for example). At the same bitrate JPEG 2000 gives better decompressed image quality over JPEG due to use of wavelet transform over DCT transform. However, wavelet transform takes more instructions to execute. Since EyeDee™ embedded eye tracking solution targets low power embedded hardware there is a need of use low-complexity image compressing/compression algorithms. Low-complexity multiplierless (based only on additions and bit shift operations) DCT approximations are potentially appropriate candidate to be execute in Weetsy™ board hardware. However, since DCT approximations do not compute the exact DCT coefficients, but still provide energy compaction property at a very low computational cost, there is a possible eye tracking quality degradation issues, which will be investigated in the next chapters. The goal of use of DCT approximations is to find a profile (window), which conforms to '20-80' principle, i.e., a point where eye tracking responsibility is 80% increased at the expense of 0..20% eye tracking precision loss. There are several approaches, which can be potentially executed even faster than low-complexity image compressing/compression algorithms. These approaches can be based on the use of artificial neural networks (ANNs) such as multilayer perceptrons (MLPs) or convolutional neural networks (CNNs or ConvNets). Such ANNs can be implemented in the resource constrained embedded devices, which is Weetsy™ board. Detailed review of the ANN-based approaches is presented in the following chapters.

Chapter 3

Methodology

3.1 Introduction

Because the eye tracking systems present class of devices, which are relatively hard to reproduce, the methodology used to conduct research in this thesis is based on the following approaches:

- «Success & failures» principle – technique based on estimation the results produced by an algorithm: positive results (success hence algorithm can be applied to the product) or negative results (failure, hence algorithm is excluded from the consideration).
- «Implementation & evaluation» principle – technique assumed initial implementation (or most likely 'porting') of the image processing algorithms to rapidly evaluate quality their results and their overall performance.
- Approximation of the original algorithms – technique aimed to achieve such an implementations, which provides *near-the-same results*, but with use of much less processor instructions leading in much less execution time.
- Rapid prototyping – rough implementation of the algorithm ideas in the software to estimate applicability of their full algorithm versions.
- Qualitative and quantitative research methods [267] – aimed at finding bugs or errors quickly by rough qualitative evaluation of expected numerical result or quantitative evaluation of its equivalent graphical representation or a pure graphical result (image).

The following thesis chapters often refer to several industry-adopted techniques:

- Complexity split/division – technique, where instead of one component implementing 100% of overall 'complexity' (either computational complexity or/and development complexity) there are N components, each of which implementing x% of complexity in total reaching 100% of complexity.
- Managing tradeoffs between computational complexity, execution time, quality of delivered results, target hardware costs and development time and costs.
- Algorithms approximations – finding of such an implementations of original algorithms, which are 'efficient' to be implemented in the embedded software (MCU) or directly in the hardware (FPGA).
- Bit depth reduction – in case of image processing, a quick way to reduce data size without changing image processing/compression algorithms. Due to reduced bit depth (reduced data values) these algorithms provide better results (less compressed size). Bit depth reduction can be seen as a *thin layer* before/after input/output data, which is not affected undelaying processing.

It should be noted that the thesis is done under CIFRE program [10], and therefore during description of used methods, approaches and algorithms author tries to establish and maintain a strong connection between theory (academia) and practice (industry), which is the core idea of the CIFRE.

3.2 SuriCog's Eye Tracking Application Specifics

3.2.1 SuriCog's Eye Tracking Algorithm

Not available in the public version of the thesis.

3.2.2 SuriCog's Application-Specific Image Compression

In this thesis term «image compression» should be understood in eye tracking application-specific sense in compare with classical definition (application) of image compression.

The difference is in the following:

- Classical image compression – compaction of an *unknown* type image, where decompressed image is used for its direct viewing (either on a screen or printed on a material). In this application quality of decompressed image is selected based on considerations, obtained according to human visual system (HVS) properties. For example, according to HVS, quality of a photo has to be about 45 dB, quality of video has to be also about 40 dB. If quality of this content is less than these recommended values, the human feels a certain discomfort (especially during long time interaction with this content). In such applications term «lossy» reefers to removing of the small details from the image to minimize compressed image size, while maximizing of the visual quality, i.e., making decompressed image looks closer to original image as much as possible. The classical image compression standards are JPEG, JPEG 2000, PNG [268,269] and many others.
- SuriCog's application-specific image compression – compaction of a *known* type image (eye image), where decompressed image is used as an input of an image processing-based eye tracking algorithm. In this case decompressed eye image is not viewed by the human, which makes unusable applying of considerations, obtained according to human visual system (HVS) properties. In SuriCog's eye tracking application term «lossy» relates to minimizing (or keeping in a desired range depending on a particular target application) of the eye tracking error (caused by lossy compression) while maximizing of the amount of extra data removed from the eye image. This application does not strictly involve decompressed eye image look close to original image. Rather, this application involves decompressed image *contain as minimum data as needed* to provide desired *eye tracking results quality*.

Term «quality of eye tracking results» itself is composed of the following metrics:

- Precision – similarity of results produced by the eye tracking system with «real results» (ground truth).
- Accuracy (also called 'stability') – ability of the eye tracking system to maintain the same precision of eye tracking results from experiment to experiment.

3.2.3 Finding of the Eye Image Compression Algorithm Requirements

To select or propose application specific eye image compression system, several minimal requirements to this system have to be estimated (possibly theoretically) and found (usually experimentally). Then these requirements (values) are used to compare several potentially applicable image compression approaches with the goal to find the most performant one. A several potential image compression approaches have to be excluded from further consideration on the very early stage (prototyping, initial performance estimation) of the product development. Especially if considering of the mass production (thousands of devices per year) of the product, then each building block (image processing algorithm, third-party physical IC component, etc.) has to be accurately and completely assessed in terms of its properties: overall performance, MIPS-budget, physical dimensions, power consumption, cost and others. Therefore, product creation can be viewed as an optimization problem, which is converged to finding of a crossing point (optimal point) between the several levels (factors): industrial design, product performance, economical factor and others.

The case of the eye tracking system used in multimedia applications (in particular, interaction with the objects placed in a known environment) the following criteria are taken into account:

1. Responsiveness of the system – ability of the eye tracking system to react on the change of the user movements as fast as possible, i.e., the delay between the acquisition of the new eye image by the camera sensor and eye results obtained from the eye tracking system.
2. Quality of the eye tracking – multi parameter, which includes the following underlying parameters: precision and accuracy.

In the case above responsiveness is more prioritized over quality, because it is considered that it is possible to find such a profile (Figure 36), where responsiveness varies in wider range of values while quality varies in more narrow range of values. For example, maximal increase (80%) of responsiveness of the system from the certain baseline is followed by minimal degrease (20%) of the eye tracking quality.

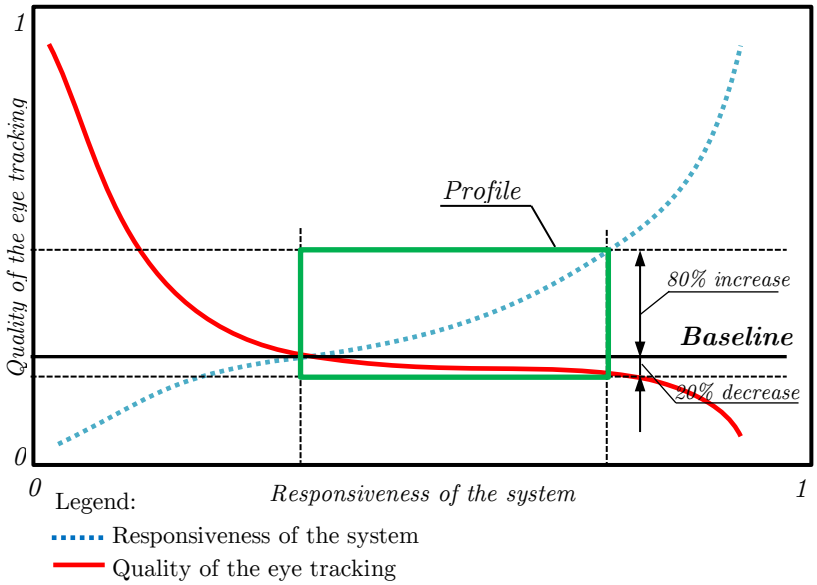


Figure 36. Profile: maximal system responsiveness vs. minimal degrease of the ET results quality.

Therefore, in such multimedia applications the following criteria are used for estimation of the eye image compression system (most prioritized goes first):

1. Time of eye image compression/decompression – parameter, which has direct impact on responsiveness of the eye tracking system.
2. Size of compressed eye image – parameter, which has direct impact on responsiveness of the eye tracking system.
3. Quality of decompressed eye image – parameter, which has direct impact on the eye tracking results.
4. Ability to operate in lossy transmission medium – ability to partially decompress compressed eye image, which was partially corrupted and/or partially lost due to using of transmission medium with best-effort data delivery, where no retransmissions used.
5. Implementation aspect and costs – estimated costs of R&D, hardware, third-party IPs, etc.

3.2.3.1 Finding Maximal Time of Eye Image Compression/Decompression

To estimate maximal time of eye image compression/decompression it is needed to understand the processing delays of each component in the entire processing chain: from eye image readout to obtaining results from eye tracking algorithm. Due to initial requirement of 100 Hz processing frequency, the period of processing of 1 eye image frame is 10 ms. This time of 10 ms is split between several components, each of which contributes to overall processing delay of 10 ms. According to practical measurements (Figure 37, Table 3) the summarized delay dedicated to compression/decompression is 2.5 ms approx.

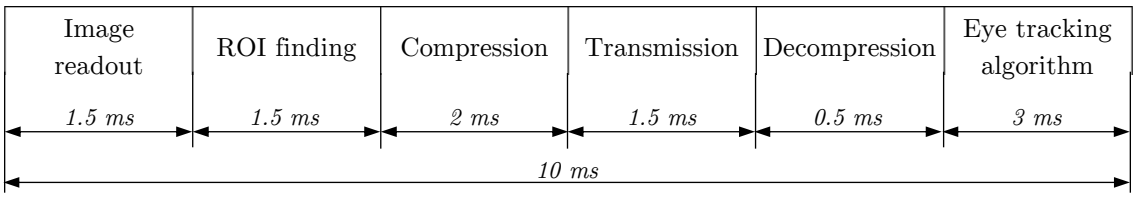


Figure 37. Hardware utilization during one iteration of the eye image processing.

Table 3. Additional details of exact configuration used during delays measurement

Image readout	FPGA-based readout of VGA image 640x480
ROI finding	FPGA-based ROI finder, ROI scanning step 12 pix
Compression	-
Transmission	Wi-Fi 802.11n, bitrate 6.3 Mbps, freq. 5 GHz, distance 3..5 m
Decompression	-
Eye tracking algorithm	Proprietary SuriQat ET algorithm

The estimated time of 2.5 ms of ROI compression/decompression can be approximately split as:

- Time of ROI compression: 2 ms.
- Time of ROI decompression: 0.5 ms.

3.2.3.2 Finding Minimal Size of Compressed Eye Image

To find minimal acceptable size of compressed eye image a special research was done, which was based on estimation of the constraints provided by used hardware (combination of ICs: Wi-Fi module controlled by host MCU, which is Renesas RZ). In particular it was estimated (theoretically) and proved (practically) that maximal available bitrate (also called data rate, bandwidth, and bit budget) is 6.33 Mbps (120h*120w*55FPS*8bpp=6336000 bps). Based on this value it is possible to theoretically estimate FPS on the remote PC-based side after decompression. However, time needed to perform compression/decompression must be taken into account. This time depends on the size of the source input image. Therefore, in case of using JPEG-based codec (JCU+libjpeg-turbo) a System of Linear Equations (SLE) (presented in the following chapter) can be derived (via natural cubic spline interpolation for example, which was used) from previously obtained results of codec comparison (Figure 70):

- SLE 'real bpp-PSNR'– defines a relationship between real bpp of the compressed image and quality of decompressed image (in PSNR);
- SLE 'real bpp-PSNR'– defines a relationship between real bpp of the compressed image and JPEG quality (1..100).
- Function (for both compressor/decompressor), which defines a relationship between size of input image (in pixels) and time needed to compress this image (in us/ms).
- Function (for both USB/Wi-Fi), which defines a relationship between size of transmitted image (in pixels) and measured bitrate (in Mbps).

3.2.3.3 Finding Minimal Quality of Decompressed Eye Image

Finding of the minimal quality of decompressed eye image is needed to know the range, where quality of eye tracking system results start significantly degrade (>10%). Then it is possible to keep quality of decompressed eye image at the level close to the found, but not exceeding it. This approach allows to control eye tracking quality by controlling decompressed eye image quality via lossy compression. The test setup (Figure 38) of the eye tracking system uses two configurations. In the first setup no image compression is applied, hence ET operates directly on raw image data obtained from a

miniaturized digital camera sensor. In the second configuration lossy image compression-decompression is used to reduce decompressed image quality.

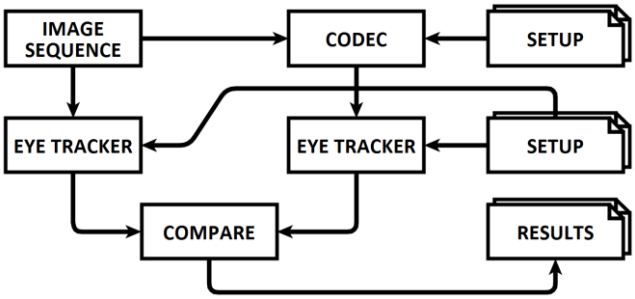


Figure 38. Test setup for finding minimal quality of decompressed eye image.

Both components (eye trackers and image codec) are setup before the comparison operation. As an input, several prerecorded sequences of eye images were used to reproduce exact the same conditions during testing. All images have 400x256 pixel resolution, 8bpp color depth. As codec OPENJPEG [[270]] was used, which is implementation of JPEG 2000 [271] image compression standard. The input image sequence contained 256 frames, but only 20 of them (fragment) were used for demonstration purposes.

Another technique is eye image pixel bit depth reduction. To potentially reduce the size of compressed eye image it is possible to reduce bpp of the original input eye image. From signal processing point of view this operation can be interpreted as passing of the signal through a High-Pass Filter (HPF), which cuts high pixels values ('high frequencies' in signal processing terminology) values and keeps only low pixel values ('low frequencies' in signal processing terminology).

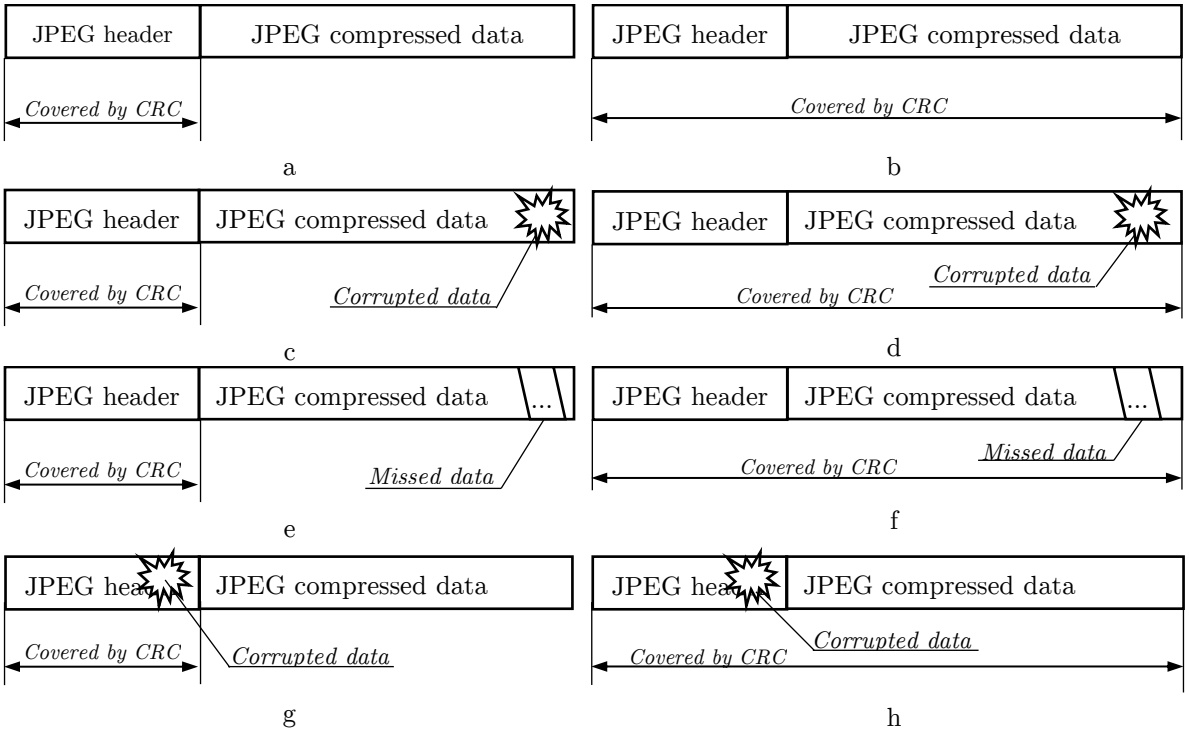
3.2.3.4 Considerations on Ability to Operate in Lossy Transmission Medium

Ability to operate in lossy transmission medium is based on:

- using of transmission medium with best-effort data delivery (no retransmissions used);
- partial decompression of compressed eye image, which was partially corrupted and/or partially lost.

It is possible to apply error correction codes to increase level of compressed data integrity at the expense of slight increase (depends on correction capability) of the amount of transmitted compressed data. In this case corrupted compressed image bitstream will be corrected by the channel decoder. However, using channel coding (encoder-decoder) will certainly take an amount of time (especially decoding and possible error correction), which will result on final FPS decrease. Even after correction of corrupted bitstream there, is no guarantee that image decoder (decompressor) will be able to decompress it, because if (for example) corruption was also took place in the header (contains sensitive information about the image: resolution, format, etc.) of compressed stream, then decoder is not able even to decode (interpret/parse) corrupted header or even more it will incorrectly interpret corrupted header, which will lead to incorrect decoding in the best case (for example, decoding according to wrong format) and unexpected/undefined behavior in the worst case (for example, allocation of memory for incorrectly computed image size). Usually such worst cases lead to run-time crash of the decoder software. Therefore, to avoid such cases it is possible to apply CRC as follows (Figure 39):

- protect entire bitsteram with CRC to let decoder completely skip decoding if the CRC of received compressed bitstream is not equal to expected CRC;
- protect only the header of compressed image with CRC to let decoder correctly decode of the header (calculated header CRC is equal to expected CRC), correctly initialize its internal buffers from header's values and 'try' to decode the rest of the stream (possibly corrupted).



Legend:

	Decoding		Decoded image	Decoder crash
	Header	Data		
a	Yes	Yes	OK	Impossible
b	Yes	Yes	OK	Impossible
c	Yes	Yes	Artefacts	Impossible
d	No	No	-	-
e	Yes	Yes	Artefacts	Impossible
f	No	No	-	-
g	No	No	-	-
h	No	No	-	-

a. Without CRC

	Decoding		Decoded image	Decoder crash
	Header	Data		
a	Yes	Yes	OK	Impossible
b	Yes	Yes	OK	Impossible
c	Yes	Yes	Artefacts	Impossible
d	Yes	Yes	Artefacts	Impossible
e	Yes	Yes	Artefacts	Impossible
f	Yes	Yes	Artefacts	Impossible
g	Yes	Yes	-	Possible
h	Yes	Yes	-	Possible

b. With CRC

Figure 39. Different approaches of JPEG bit stream protection via CRC.

It should be noted that video compression techniques are especially sensitive to compressed bit stream corruption or loss, because in case of decoding of corrupted bit stream it is very possible that entire group of frames (GOF) will be corrupted. Therefore, these video compression techniques usually relies on use of guaranteed delivery communication channels, where retransmissions occur time-to-time.

To target best-effort delivery communication channels, video compression techniques support a special so-called 'profiles', which are basically modified versions of original codec implementation, but where modified version contains an additional logic executed in case of internal errors detection: 'early exit' policies, checking sizes validity before memory allocation, etc. However, according to several sources, recommended amount of bitsteram corruption and/or bitsteram loss with which such profiles can natively operate (without error correction codes) is not more than 20% only.

3.3 Eye Image Compression Alternative Approaches

3.3.1 3D Modeling of Pupil Surface

When input images are represented in 3D (Figure 40) there are surfaces, progressively moving in time. The idea is to describe such surfaces (or their approximated versions) by applying equation of the surface (Figure 41) with number of coefficients, much less than number of pixels in input image.

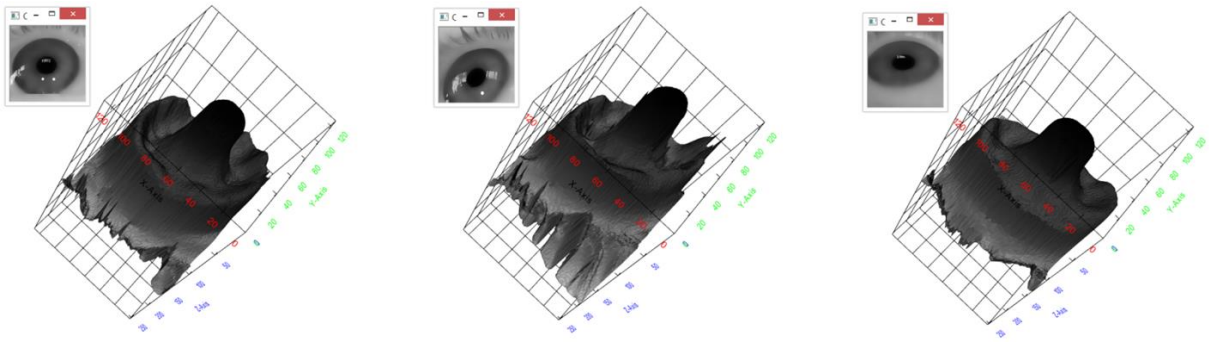


Figure 40. Input images represented in 3D.

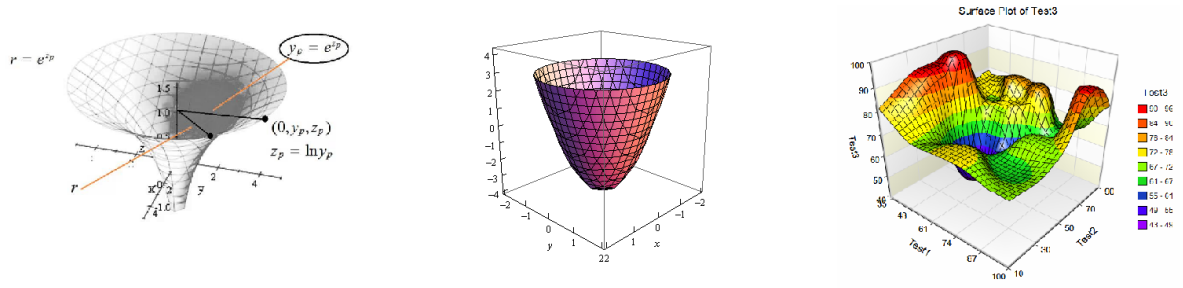


Figure 41. Few examples of surface.

Therefore, to find this coefficients a form of modeling on input image has to be done, which leads to high number of computations. The model has to be parameterized by N coefficients (Figure 42), which implies approximation of the reconstructed surface to real one (with the same approach of compression of residual image, i.e., compression of prediction error). Therefore, it is possible to compress only prediction error i.e., difference between real surface and predicted (approximated) by model surface. If the predication is accurate then prediction error will be small leading to small size of it in compressed form. However, finding of such surface in real-time takes significant computation resources, which makes it problematic to apply in the developed product.

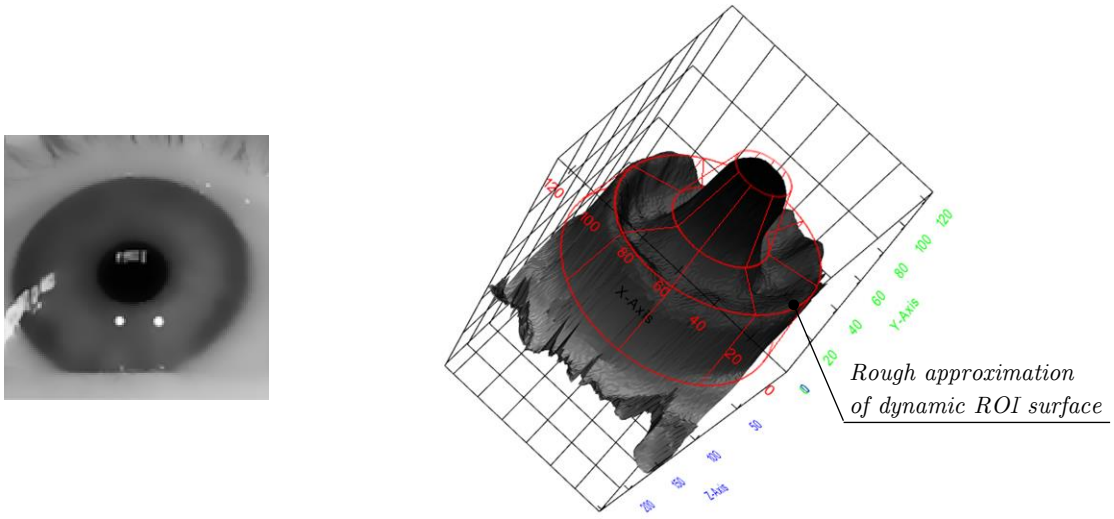


Figure 42. Example of 3D pupil surface rough approximation.

Maintaining maximal and minimal possible values (Figure 43) cab help to reduce bpp needed to represent pixel values (for example from 8 bpp to 7-6 bpp).

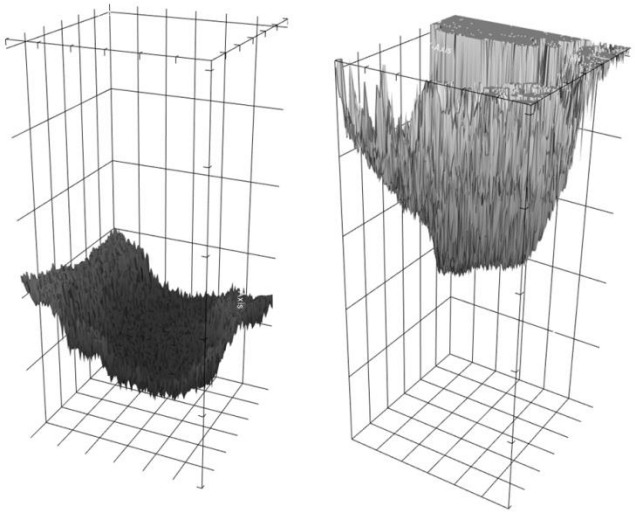


Figure 43. 3D visualization of minimal and maximal values of moving pupil image.

3.3.2 Dictionary Based Eye Image Compression

The idea behind the dictionary based eye image compression is that encoder and decoder share a dictionary [272] full of block samples (patches). Each input image is split on blocks. For each block the most corresponding block from the dictionary is found and residual image (difference) is computed. Encoder sends indexes of blocks and compressed residual image. Before operation dictionary has to be created and optimized (near similar blocks removal). This procedure is known as 'dictionary training'.

The dictionary based eye image compression idea correlates with basic hybrid codec scheme. Most of today's video coding standards based on idea of exploiting temporal redundancy between highly-correlated consecutive frames. Since they are very similar it is natural way to send only difference between them. However, it is possible to obtain even higher compression ratio by sending combination of so-called *motion vectors* and compressed *residual image*. The basic component of modern video codec is the hybrid video encoder (Figure 44). Encoder includes an entire decoder loop especially to predict the subsequent frame. It uses technique, which is called *motion estimation* to find motion vectors that describe the transformation from one frame to another one. For this frame is decomposed into number of blocks that usually have form of rectangle. Then each block of the current frame is compared with blocks in some area of reference frame to find the most similar block. Based on coordinates of such block motion vectors are computed. Then compensated frame is subtracted from the current frame to obtain difference between the two consecutive frames. This so-called *residual image* is needed by decoder to keep the details of the current frame during its reconstruction.

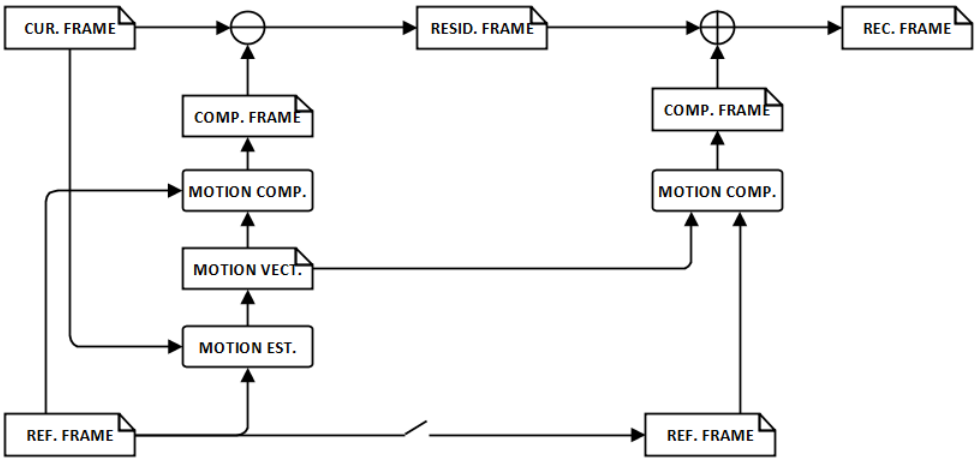


Figure 44. Basic hybrid codec scheme.

Especially for encoding images that have objects with known form, special video compression techniques could be applied. Dictionary-based video coding (Figure 45) is one of them. The main difference from the previous scheme is absence of sending reference frames. These frames are replaced by dictionary, which has predefined number of blocks, which are used for comparison to find «best match». Block indexes are result of motion estimation, they are transmitted to decoder with statically compressed residual image.

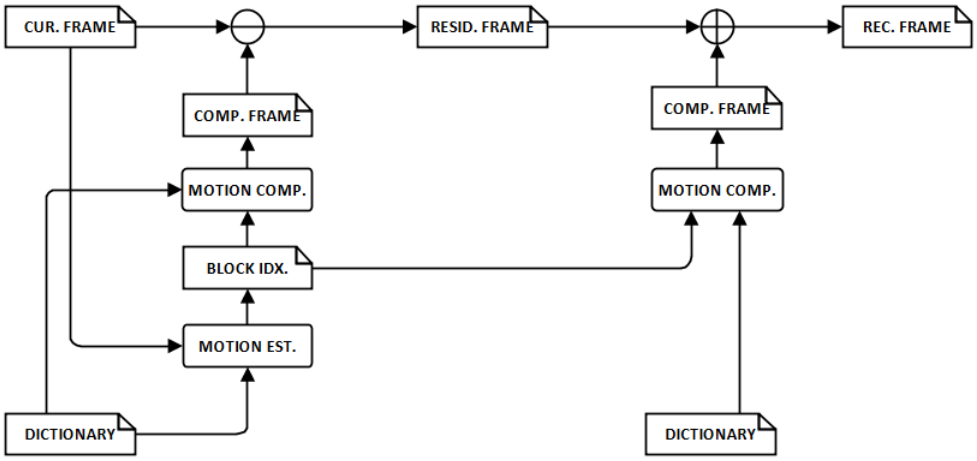


Figure 45. Dictionary-based codec scheme.

The idea of applying such scheme is increasing compression ratio based on assumption that motion of objects on the each next frame has low level of dynamics (i.e., they are highly correlated) or they are placed on known area. Therefore, to exploit such nature of video sequence, it is possible to define and «train» dictionary to reconstruct the coming frame. However, to keep the details of objects and to react on fluctuations of motion dynamics it is still necessary to obtain and send residual frame, which is be used by decoder for frame reconstruction.

As an input dictionary takes image sequence, size of block and total number of blocks, which will be used for comparison. This way is used in MATLAB version. There are other ways to create dictionary and even more to update it at run-time. For example, in C++ version of this approach dictionary is created by setting number of frames that will be stored in it. Then these frames are split into the blocks, which are and optionally go through optimization to remove similar ones.

For dictionary-based video encoder motion estimation consists in finding indexes of blocks, which are the most similar to block from current frame. One of the most trivial approaches to find such indexes is to apply exhaustive search for each block of the current frame. Depth of such search is defined by so-called search area coefficient (usually is represented as number of pixels), which sets both horizontal and

vertical offsets that finally defined some area. When another block is selected in the loop, it is compared with all the blocks of dictionary that fit the area. Moreover, if dictionary contains number of blocks, which is greater than number of all possible blocks of image, it is possible to use additional blocks for comparison, which can be interpreted as «alternative blocks». Motion compensation consists in generation of compensated frame by choosing blocks from the dictionary by their indexes and adding them to the frame. Residual image is obtained as difference between current frame and compensated frame. Energy of residual image can be used as an indicator to estimate impact of changing encoder's parameters.

3.3.3 Dictionary + DCT Based Eye Image Compression

The idea behind dictionary + DCT based eye image compression is to split eye image into blocks, then to perform Discrete Cosine Transform (DCT) to each block. Then to use the same idea described before (dictionary full of samples) to replace block of pixels to index of the near similar block from dictionary.

A discrete cosine transform (DCT) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. DCT is employed in JPEG standard. DCT can be applied to all image (Figure 46) or to each block of the image. Then DCT coefficients can be quantized with different thresholds (Figure 46, Figure 47).

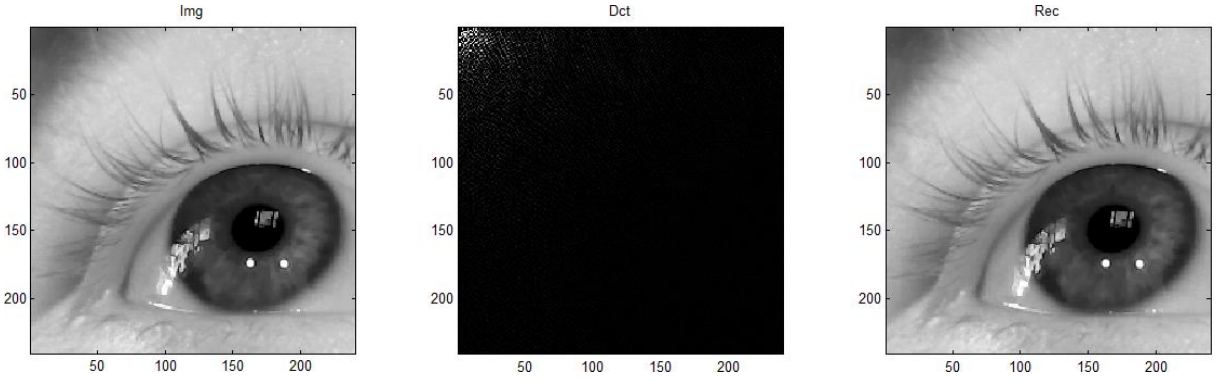


Figure 46. DCT applied to all image.

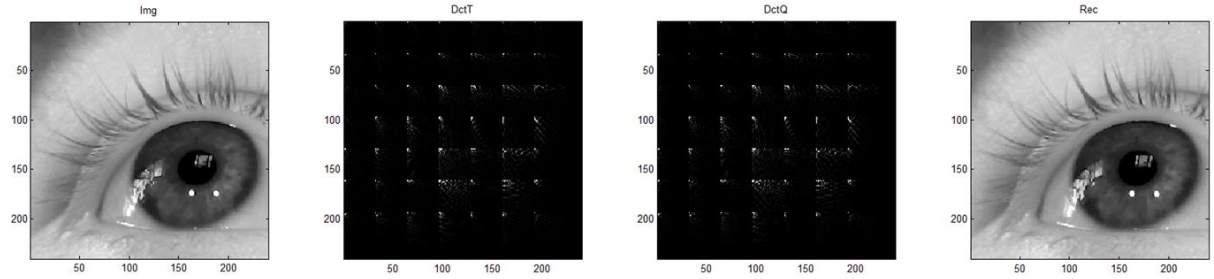


Figure 47. DCT applied to blocks of pixels (threshold 1e-2).

The idea is to study dynamics of change of such coefficients and synthesize L-system (or set of L-systems), which could re-produce such coefficients with low cost. This idea is correlated with plane filling curves (Figure 48) and 2D/3D-cellular automata (Figure 49).

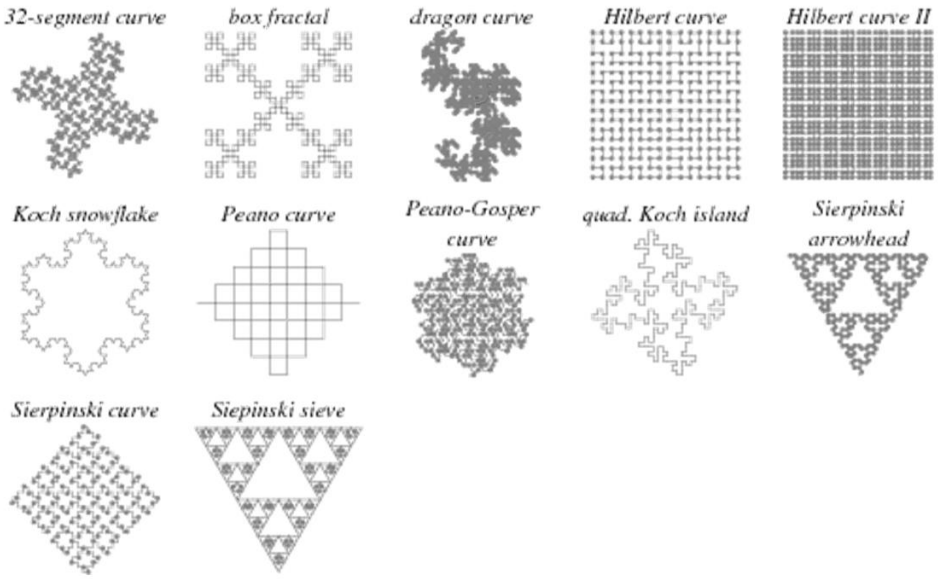


Figure 48. Plane filling curves.

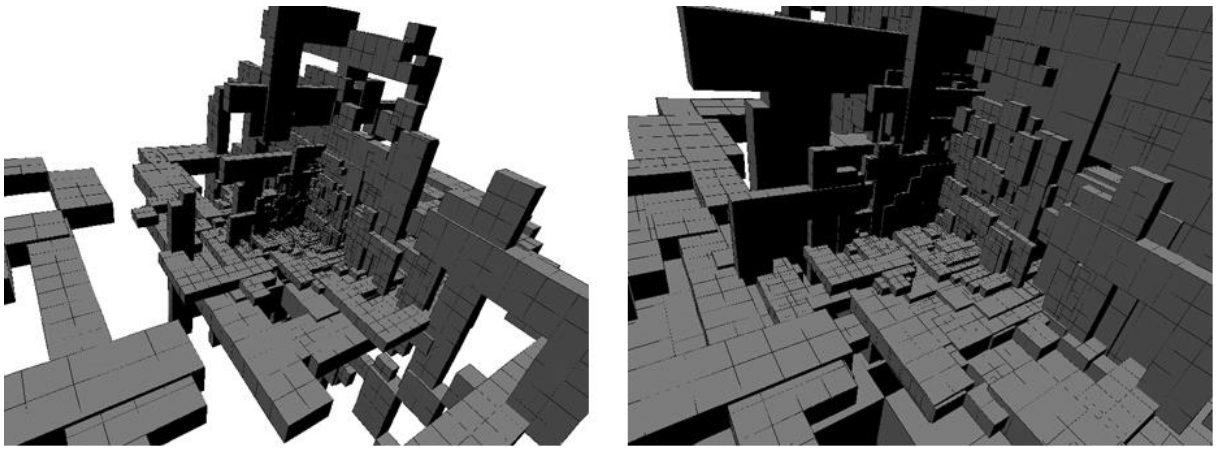


Figure 49. 3D cellular automata examples.

Usage of L-systems and cellular automata could produce good results (leading to high compression ratio), but on limited set on well-known inputs. Even for this approach preliminary classification of possible inputs should be done. Then for each input (block of pixels) such L-system is found, which can produce near the same input (threshold is applied). Therefore, this approach can be interpreted as previously described dictionary-based compression, but instead of dictionary full of samples, L-systems are used to generate an input based on few rules (should be send to decoder). However, this approach has the same problems as fractal-based compression, i.e., problem of finding Iterated Function System (IFS) by having attractor (input image), which leads to delay not competitive to delay, closed to real-time terminology.

3.3.4 Schemes with Shared Model

The idea behind schemes with shared model is to use the same model on transmitted (TX) and receiver (RX) size and transmit only prediction errors. Hence decoder can reconstruct data based on the model and add received prediction errors to fully reconstruct the data. Expression «sharing of the same model» means that from time to time and based on accumulated statistics encoder updates its local model and sends to decoder information to update its local model used for decoding. As a result, both TX and RX sides share the same model.

However, all schemes with shared model require quarantined data delivery and do not work correctly if the data was corrupted (or lost) during transmission due to lossy channel. For example, in case of data corruption RX will use corrupted date to update its model and starting from this moment TX and RX do not share the same model anymore, which will lead to RX data reconstruction errors.

Distributed source coding (DSC) [273] is based on principle of the compression of multiple correlated information sources which do not communicate with each other. By modeling the correlation between multiple sources at the decoder side together with channel codes, DSC is able to shift the computational complexity from encoder side to decoder side. Because of this complexity shift it is possible to implement sender in resource constrained embedded low-power platforms. In general, DSC involves coding of two or more dependent sources with separate encoders and joint decoder. DSC can be symmetric (same bitrate is used in coding the input sources) and asymmetric (different bitrates are used in coding the input sources). Typical applications of DSC are sensor networks and video/multimedia compression.

Distributed video coding (DVC) is based on DSC. One important aspect of DVC is that many of DVC systems described in the literature use of a so-called 'feedback channel' (Figure 50) from the decoder to the encoder to determine the rate. However, the number of requests issued through the feedback channel is often high, and as a result the overall delay of the system could be unacceptable in practical video compression and streaming applications. There were several solutions have been introduced to address this issue. For example, feedback-free DVC systems have been proposed with the incorporation of a difficult trade-off between encoder complexity and compression performance. As a result, a method was proposed for constraining the number of feedback requests to a fixed maximum number of N requests for an entire Wyner-Ziv (WZ) frame [274].

In case of SuriCog's EyeDee™ eye tracking system transmission medium is a best-effort delivery communication channel. Therefore, use of any approach involving feedback channel is problematic.

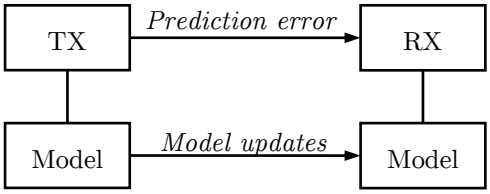


Figure 50. Compression scheme with shared model.

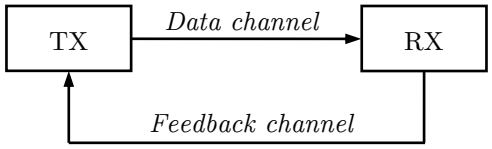
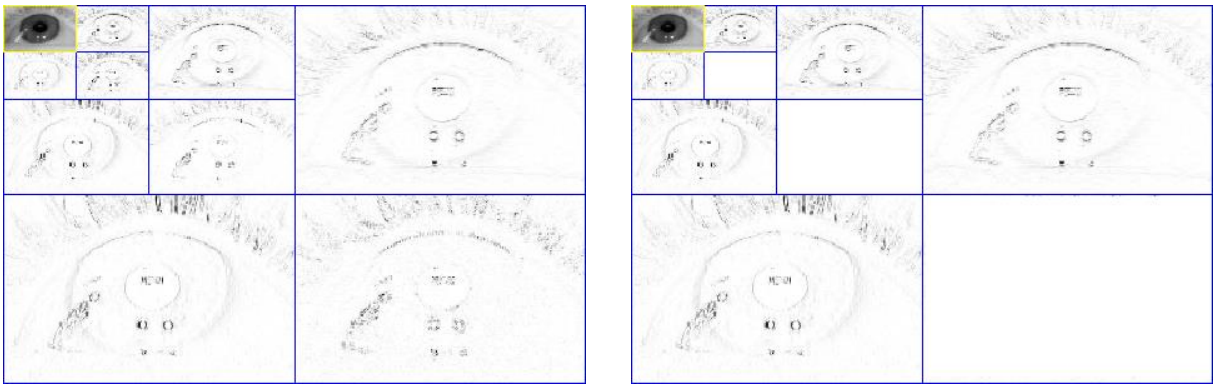


Figure 51. Distributed video coding.

3.4 Eye Based Rate Distortion Optimization

Rate-distortion optimization (RDO) is an approach of improving video quality in video compression. The name itself refers to the optimization of the amount of distortion (loss of perceptual video quality) against the amount of data required to encode the video, the rate. RDO is aimed to improve existed variable bit allocation techniques and to get, as a result, higher compression ratios while maintaining the same (or near-the-same) computational complexity. This allows to use predefined (aposteriorical) knowledge of information source (eye images) to keep important to eye tracking regions of pixels uncompressed (cornea-pupil edges with highest quality) and to compress unimportant regions (lowest quality). The idea (Figure 52) is to find a particular subbands (for example, subbands, containing diagonal details) and then compress these subband coefficients with higher ratio leading to less bpp or remove these subband coefficients, leading to even lesser bpp.



Diagonal details more compressed *Diagonal details entirely removed*

Figure 52. Idea of RDO based on diagonal coefficient processing.

3.5 Neural Network Based Approaches

3.5.1 Applying Neural Networks to the EyeDee™ Solution

Over the last years there is a tendency of applying neural networks in industry [275–278] to the parts of complex algorithms or even more a complete *replacement* of complex algorithms by trained neural networks. There are several major advantages behind this tendency:

- Simplify development of complex image processing based eye tracking algorithm. Usually development of complex algorithm takes significant amount of time needed for optimization, precision, accuracy, implementation, and parallelization.
- Accelerate of the eye tracking algorithm. Taking results from neural network is usually considered as faster operation in compare with execution of the algorithms because trained neural network is an «offline» approach (pre-trained set of weights) of complex objective function while classical algorithms search this objective function in run-time («online» approach).
- Constant time response. Due to conditional logic, (and, in consequence, not constant number of operations executed in run-time) the execution time of image processing based eye tracking algorithm is not constant, which can be essential for especially time-critical applications. Usage of neural trained network provides guaranty of constant time response [279], because of constant number of operations to be executed to get a response.
- Implementation aspect: implementation can be seen as a set of pre-defined floating-point numbers (trained weights) coupled with neural network architecture implementation (MLP or ConvNet). These parts are implemented many times directly in hardware [280,281]. Hence, hardware implementation costs go down.

The minor disadvantages of using neural networks are:

- Quality of training of neural network is still a challenge. The common issues are generalization and coverage. Which usually results in collecting eye tracking data from several sources (users) into huge databases and training of differently configured neural networks on these databases. Several approaches are aimed to find the best hyperparameter combination, including basic grid search [282], random search [225] and others.
- Time of training is high especially in case of using DNN. Recent solutions are based on using of the GPUs to accelerate training time as well as .usage of advanced, more optimized training algorithms [245] aimed on training time reduction.
- Approval issues. Usually trained ANNs are considered as ‘input-output black boxes’. While image processing algorithms are relatively easy to trace in step-by-step mode and verify certain

core values of the algorithm, trained ANNs are cannot be traced in the same manner. Therefore, due to lack of clear traceability it is hard to approve ANN in the product and even more to certify ANNs in contrast with constantly proved image processing algorithms.

However, in an applications, where usage of ANN results cannot imply on an industrial catastrophe of or a live loss (for example, multimedia applications, entertainment applications), usage of trained ANNs has a certain potential.

The EyeDee™ *eye tracking* can be interpreted as a multivariate function, which unambiguously associates an ROI image (input of the eye tracking algorithm) with five ellipse parameters (output of the eye tracking algorithm). Such a function is implemented via image processing (extraction of contours of pupil ellipse followed by its automatic measuring). It is shown that it is possible to apply ANN based approach to find an approximation this function.

The EyeDee™ *head tracking* can be interpreted as a multivariate function, which unambiguously associates an filtered image of the Weetsy™ Frame (input of the head tracking algorithm) with 6DoF parameters describing position of the Weetsy™ Frame in the 3D space (output of the head tracking algorithm). It is shown as well that it is possible to apply ANN based approach to find an approximation this function.

The EyeDee™ *calibration* [283] is a process aligning a person's gaze estimation to a particular scene, when the geometric characteristics of a subject's eyes are estimated as the basis for a fully-customized and accurate gaze point. The calibration can be interpreted as the process of determining the equations used to map angles and radii between the pupil and glint centers of the user's eye to radii and angles on the screen, with the origin of the coordinate system for the screen being just below the bottom center of the monitor's screen. Result of the calibration is an improved gaze estimation for a selected subject. It is shown as well that it is possible to apply ANN based approach to the calibration task: either completely (ANN is used to perform an entire calibration process) or partially (default calibration algorithm cross-checks its calibration results with results obtained from ANN).

Due to the thesis scope the following research focuses only on applying ANN based approaches to the eye tracking.

3.5.2 Neural Network Based Eye Tracking (Based on Function Regression)

One potentially applicable example of using ANN in the eye tracking domain consists in the complete replacement of currently used eye tracking image processing based algorithm coupled with geometrical eye modeling by a precisely tuned and perfectly trained ANN, which directly transforms wirelessly transmitted floating-point values of decimated eye image (result of the 3D perspective projection of a model of rotating pupil disk) into five floating-point parameters of pupil's ellipse (result of the eye tracking). Hence the implementation of the eye tracking algorithm is reduced to a neural network construction and training approach, which is proposed in this thesis.

To create eye images for experimentation a special eye simulator was developed based on a 3D eye model projected onto a plane (CMOS sensor). It simulates an eye of known geometry and a camera sensor of known resolution, focal and distortion. Simulator settings permit to simulate only needed eye features (Figure 53) at different detalization levels. The noise present in real images is simulated using a Gaussian kernel.

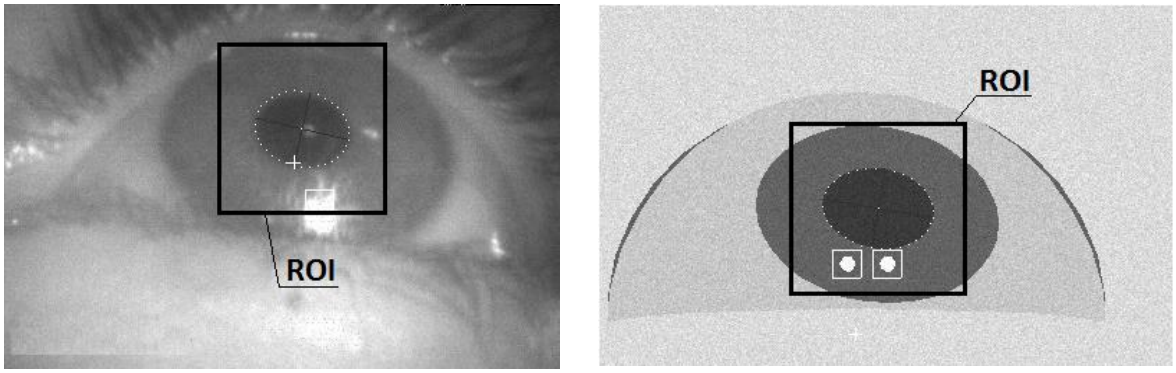


Figure 53. Eye images: real one and generated by simulator.

It can be seen that 3D projection of the rotating pupil disk has a geometrical shape of an ellipse. The reconstruction of the gaze (i.e., direction in which the user is looking) is based on the center of rotation of the eye coupled with the coordinates of pupil's center, which are calculated based on five ellipse parameters (Figure 54), obtained by automatic ellipse shape measuring.

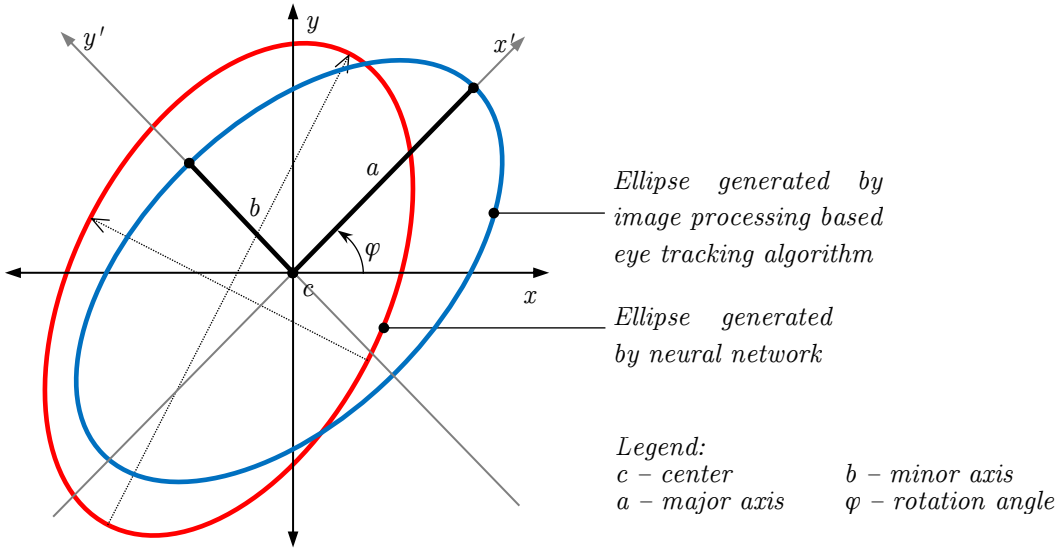
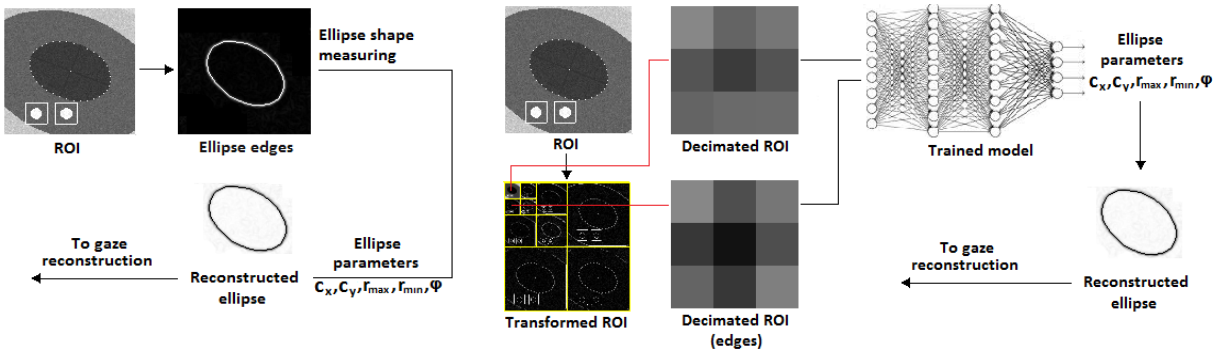


Figure 54. Five ellipse parameters: center (x, y), major/minor (a/b) axis, rotation angle (φ).

The process of eye tracking can be interpreted as a multivariate function, which unambiguously associates a dynamic ROI image (region that contains the centered image of the pupil) with five ellipse parameters. This function can be based on (Figure 55):

- Image processing – filtering the eye image aimed on pupil ellipse shape preservation, followed by its automatic measuring;
- Training of the ANN – the ANN inputs are the decimated ROI coupled with the decimated ROI edges – both can be optionally high-frequency subbands of the 2D wavelet transform – and outputs are the ellipse parameters, followed by loss function optimization (minimization of the average ellipse reconstruction error).



a. Image processing based b. Neural network based

Figure 55. Eye tracking approaches: image processing based and neural network based.

The neural network is aimed on finding a correlation (Figure 56) between the floating-point values of a decimated ROI image (decimated to reduce the amount of wirelessly transferred data) and the five floating-point parameters of the pupil's ellipse (necessary for gaze reconstruction). To estimate the relationships among these variables, a regression analysis function is applied using Torch7 [284] software (neural network 'nn' and optimization 'optim' packages). This function is further integrated into the EyeDee™ eye tracking software running on Windows 8 Win32 platform. During an initial testing we decided to keep only one layer for the network, and since the decimation reduces 120x120 pixels 8bpp ROI image into 3x3/5x5 pixels 32bpp image, the neural network has 9/25 inputs (18-50 inputs in case of using ellipse edges) and 5 outputs.

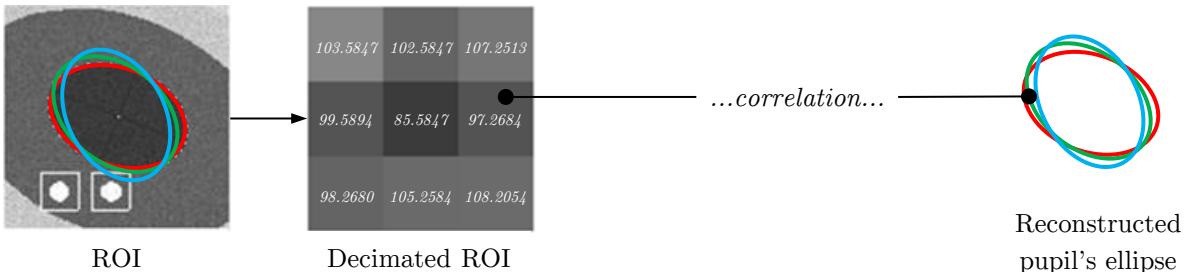


Figure 56. Correlation between decimated ROI image and pupil's ellipse.

The training of the neural network is based on the well-known back-propagation approach coupled with a gradient descent optimization method. The output of the network is compared to the desired output using a loss function. Therefore, the training can be interpreted as the loss function optimization (error minimization). The training is done with the following hyper-parameters:

- Number of epochs – the number of iterations over the training dataset;
- Learning rate – the size of the step taken at each stochastic estimate of the gradient;
- Learning rate decay – allows the algorithm to converge with high precision (it is often recommended to lower the learning rate as the training progresses);
- Weight decay – used to L2-regularize the solution (model overfitting reduction [285]), which prevents the weights from growing too large;
- Momentum – used to prevent the system from converging to a local minimum or saddle point.

It should be noted also that to ensure correct ellipse parameters reconstruction based on decimated ROI images of all possible eye positions, it is essential to provide all possible input-output pairs for the training. Since number of such pairs can be sufficiently large (number of all possible ellipse centers c_x and c_y multiplied by number of number of all possible ellipse sizes r_{max} and r_{min} multiplied by number of all possible ellipse rotation angles φ) training can take an amount of time.

To reduce this complexity, it is possible to reduce number of input-output pairs by removing such a pairs that cannot practically appear, because ellipse sizes and ellipse center positions are fluctuated (user moves the eye) in a certain range. Another way is to use several modern techniques aimed on

training acceleration, such as dropout [224] or batch normalization [245], use of ReLU [242] (or its improved versions PReLU [286] or RReLU [287]).

3.5.3 Feature Based Eye Image Compression (Based on Data Classification)

This approach is based on idea of using ANN as a classifier. The ANN is trained to classify the areas of the dynamic ROI image: relevant for eye tracking features (edges in this case) and not relevant features. Then output classification result from the ANN is used as an input of image compressor to define which area of the image should be broadcasted over a wireless medium for further processing by the PC-based desktop eye tracking software.

Herein and after we refer to the following terminology:

– ROI (Region of Interest) – region containing image of the human's pupil;
– FOI (Features of Interest) – image containing useful (for eye tracking algorithm) features of ROI.

It is necessary to understand the difference of the proposed neural network based approach of ROI eye image compression with respect to the classical one. Classical approach of image compression (Figure 57), consists in the compression of the ROI image, sending thus the compressed image (bitstream) over a channel, followed by decompression of the bitstream on the remote side to get the original ROI image, which is further used as an input for the eye tracking algorithm. The neural network based approach of image compression (Figure 58), consisted in the following steps:

a. Training a neural network to classify the blocks of a ROI image. The training is based on a set of samples: ROI block itself and a Boolean value that indicates if this block contains pupil edges. During the training this value is obtained from the default image processing based eye tracking algorithm.
b. Use the trained neural network to perform ROI blocks classification, i.e., to decide if a particular block contains pupil edges.

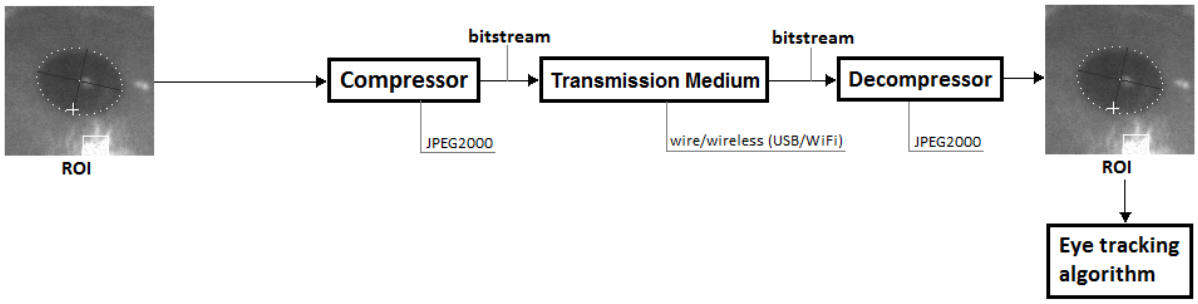


Figure 57. Image compression: classical approach.

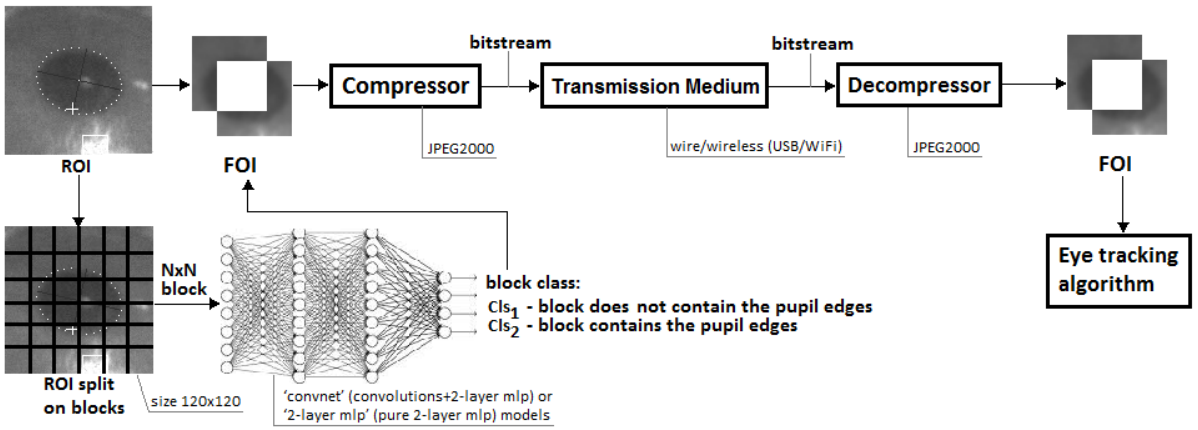


Figure 58. Image compression: neural network based approach.

The neural network is aimed on classification of the ROI image blocks into 2 classes (blocks contain/does not contain pupil edges). To this purpose, we also used Torch7 [284] software (neural network 'nn' and optimization 'optim' packages) with 'convnet' (convolutions+2-layer mlp, where 2-layer mlp is *multilayer perceptron*, Figure 59) and '2-layer mlp' (pure 2-layer mlp, Figure 60) models. This functionality was further integrated into the EyeDee™ eye tracking software running on Windows 10 x64 platform. During an initial testing we decided to split the ROI image into blocks of size 20x20 and 10x10 (the block sizes in range 10..20 lead to conditions resulting in maximal benefit of using the presented approach). In case of 'convnet' model we used 2 convolution layers (with max pooling), followed by reshaping, and standard 2-layer mlp model. In case of '2-layer mlp' model we directly used reshaping followed by 2-layer mlp model.

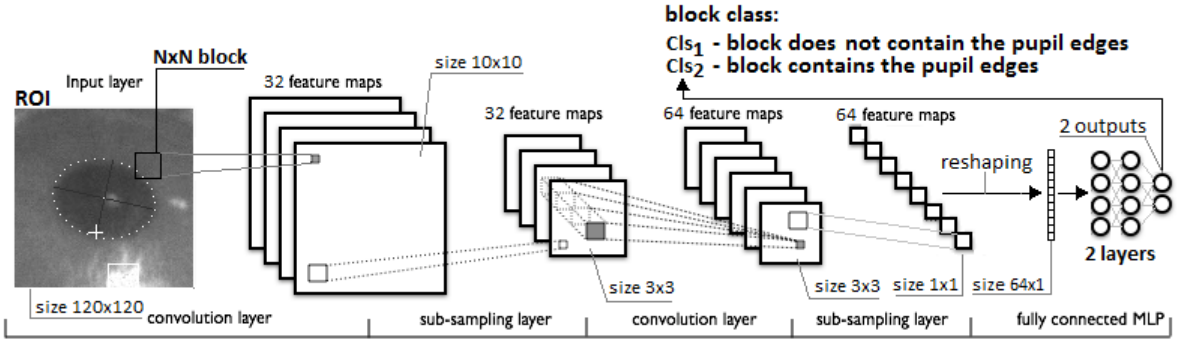


Figure 59. ROI image block classification using ConvNet (convolutions + 2-layer mlp).

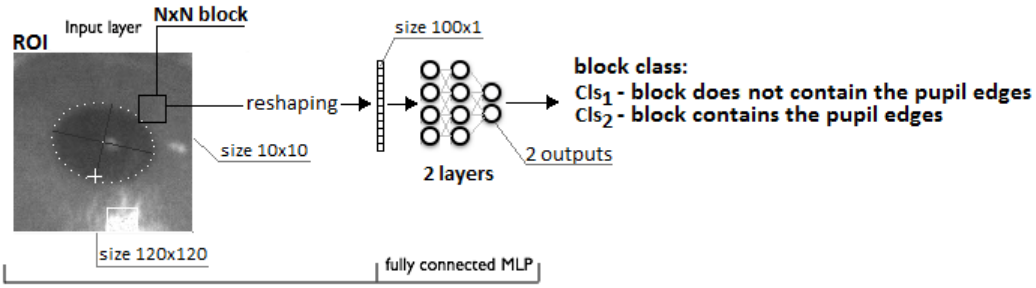


Figure 60. ROI image block classification using 2-layer mlp neural network.

We used batch learning, i.e., learning on the entire training data set at once. In all tests batch size was set to 10. The training of the neural network is based on the well-known back-propagation approach coupled with a gradient descent optimization method. The output of the network is compared to the desired output using a loss criterion. Therefore, the training can be interpreted as the loss function optimization (error minimization).

Final decision on which class a particular ROI block belongs to can be based on:

1. Probability from the last neural network SoftMax layer (default selection method often used in a neural network-based classifiers).
2. Applying a threshold (Figure 61) on probability density function (Figure 62). In this selection method statistics on probabilities distribution is taken into account.

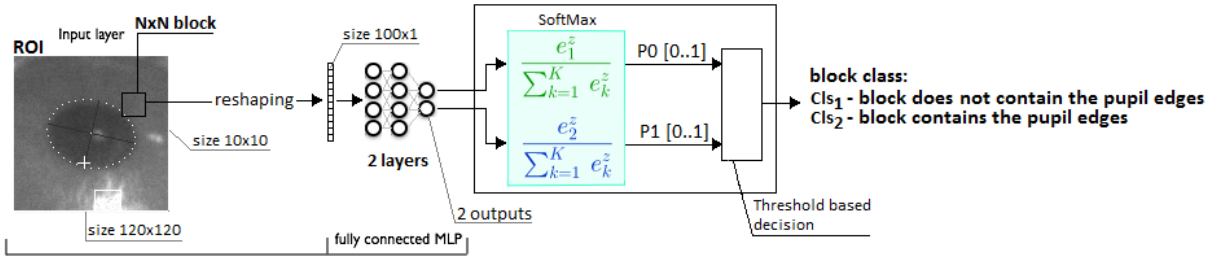


Figure 61. SoftMax last layer and threshold based decision.

SoftMax explained in details (+its relation with Bayes theorem)

The softmax function, or normalized exponential function is a generalization of the logistic function that «squashes» a K-dimensional vector z of arbitrary real values to a K-dimensional vector $\sigma(z)$ of real values, where each entry is in the range (0, 1], and all the entries add up to 1. The function is given by

$$\sigma : \mathbb{R}^K \rightarrow \left\{ \sigma \in \mathbb{R}^K \mid \sigma_i > 0, \sum_{i=1}^{K} \sigma_i = 1 \right\}, \tag{3.1}$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \text{ for } j = 1, ..., K. \tag{3.2}$$

The softmax function is used in various multiclass classification methods, such as multinomial logistic regression (also known as softmax regression), multiclass linear discriminant analysis, naive Bayes classifiers, and artificial neural networks. Specifically, in multinomial logistic regression and linear discriminant analysis, the input to the function is the result of K distinct linear functions, and the predicted probability for the j^{th} class given a sample vector x and a weighting vector w is:

$$P(y = j|\ \mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}. \tag{3.3}$$

An explanation of the relationship between the softmax function and the Bayes theorem is provided in [288].

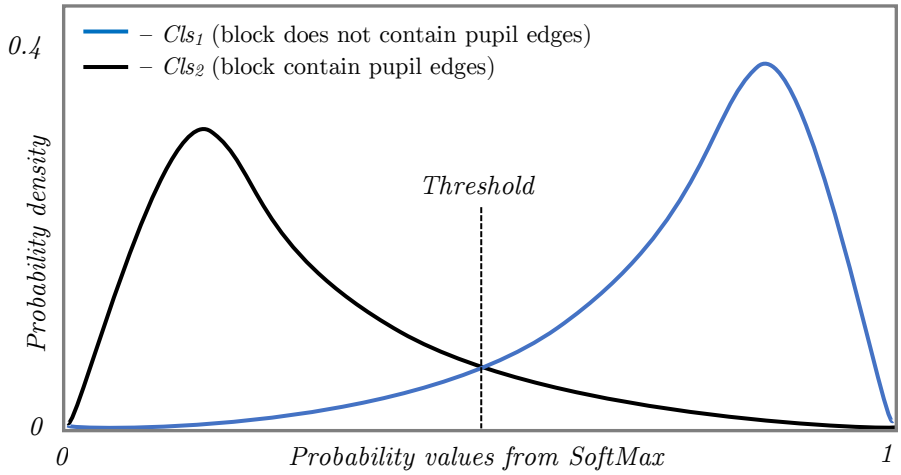


Figure 62. Probability density illustration.

Based on the final ANN outputs it is possible to measure two key metrics used to define quality of blocks classification: *efficiency* (ε) and *purity* (p). These metrics were selected, because terminology 'efficiency' and 'purity' is more pertinent (in comparison with commonly used 'precision' and 'recall') in characterizing results of blocks classification. These metrics are calculated as follows:

$$\varepsilon = \frac{N_{11}}{N_{10} + N_{11}}, \tag{3.4}$$

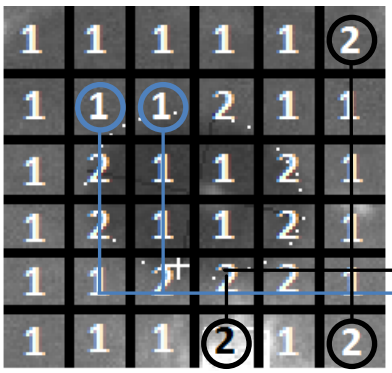
$$p = \frac{N_{01}}{N_{01} + N_{11}}, \tag{3.5}$$

where:

– N_{00} – number of blocks predicted not to contain pupil edges which do not contain edges in reality;
– N_{01} – number of blocks predicted to contain pupil edges which do not contain edges in reality;
– N_{10} – number of blocks predicted not to contain pupil edges which do contain edges in reality;
– N_{11} – number of blocks predicted to contain pupil edges which do contain edges in reality.

Efficiency metric (ε) can be interpreted as a *recall* (also called *sensitivity*) and shows how good trained NN at detecting the *Cls2*-clocks (block contains pupil edges). Ideally this values should be 100%.

Efficiency metric (p) can be interpreted as a *precision* and shows many of the positively classified *Cls2*-clocks (block contains pupil edges) were irrelevant (i.e., do not contain pupil edges in reality). Ideally this values should be 0%.



ROI image split on blocks (example)

General notion

		Prediction	
		Cls_1	Cls_2
Truth	Cls_1	N_{00}	N_{01}
	Cls_2	N_{10}	N_{11}

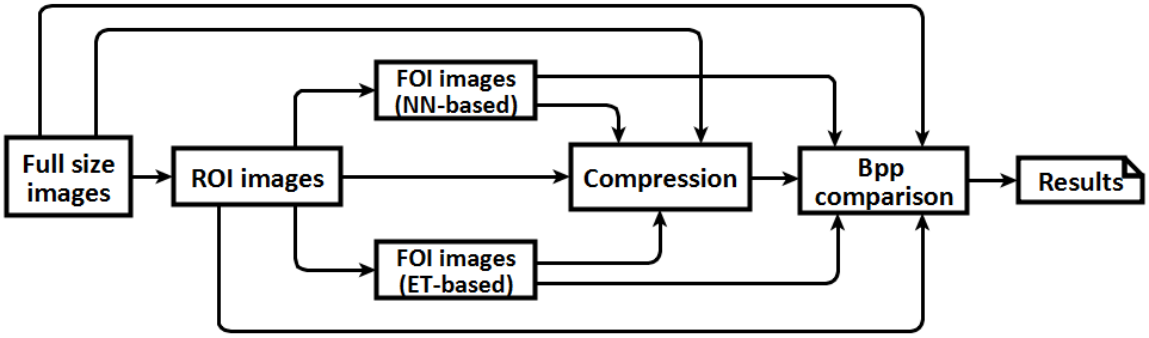
Particular example

		Prediction	
		Cls_1	Cls_2
Truth	Cls_1	N_{00}	N_{01}
	Cls_2	N_{10}	N_{11}

Legend:
Cls_1 – block does not contain pupil edges
Cls_2 – block contain pupil edges

Figure 63. Efficiency/purity illustrated explanation.

After obtaining these two metrics and compression of the FOI images according the approach it is possible to measure compression performance at least in terms of bitrate (bits per pixel, bpp) of uncompressed/compressed ROI/FOI images (Figure 64).



Legend:
FOI images (NN-based) – FOI images obtained with neural network based approach,
FOI images (ET-based) – FOI images obtained with general ET (Eye Tracking) algorithm

Figure 64. Compressed image comparison scheme.

Due to imperfect classification, it is very likely to have *missing blocks* with pupil edges and *extra blocks* without edges. Processing of such FOI images will lead to degradation of eye tracking quality. There are two strategies to reach high efficiency (>98%) and low purity (<5%):

1. Train ANN better – this strategy involves tuning of the hyperparameters (with random search for example). This approach takes a lot of time.
2. «Complexity split» principle (can be viewed as 'composite result', Figure 65) – this strategy involves split of final complexity of reaching desired percentages (>98% and <5%) between several components, which are in summary give the desired percentages. For example, after rapid estimation of a *typical* efficiency/purity percentages can be obtained from a middle-quality trained ANN (for example, typical values are <75-80% and >15-25%), it is possible to *fix* ANN parameters, stop time-consuming training and create a second component (FOI corrector, Figure 66), which takes ANN intermediate results and use them to reach expected percentages.

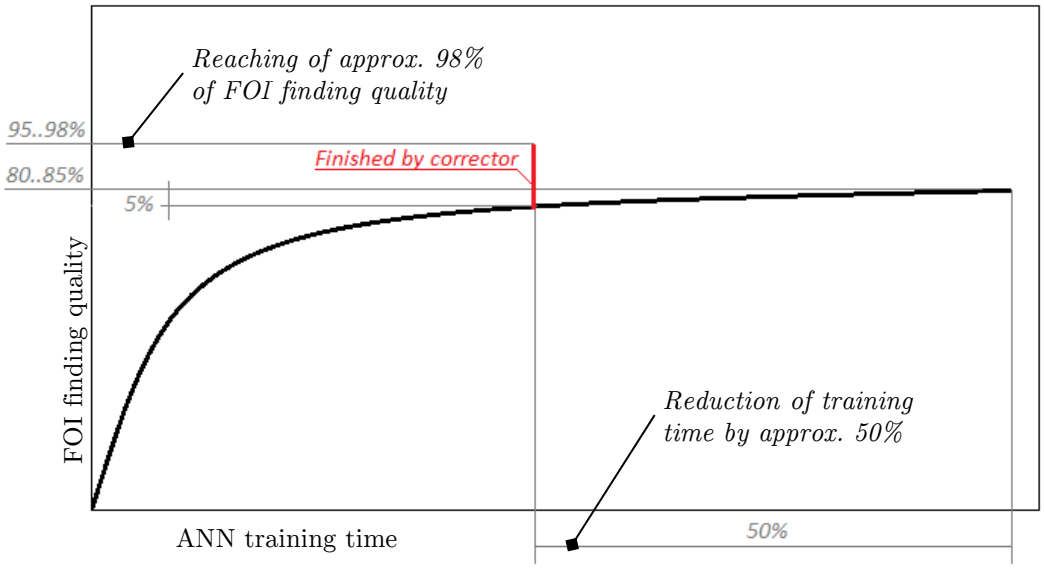


Figure 65. ANN training time reduction by introducing FOI corrector.

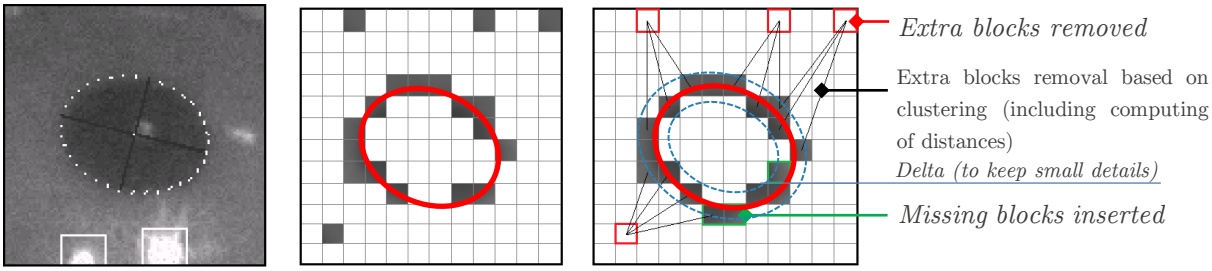


Figure 66. FOI correction + FOI compression illustration.

It should be noted that the FOI correction is considered as the future work. However, the FOI correction algorithm can be based on *K*-means clustering [163,289] or fuzzy logic [290].

3.6 Conclusion

SuriCog's EyeDee™ eye tracking algorithm is based on eye modelling. It consists in reconstructing a 3D model of an eye out of multiple 2D images taken by the CMOS sensor ah a very high acquisition frequency (100Hz). To introduce eye image compression into the system, the thesis term «image compression» itself should be understood in an eye tracking application-specific sense in compare with classical definition (application) of image compression. In particular, the eye tracking system does not incorporate any human feedback on viewing decompressed eye images. Therefore, during selection, implementation, optimization and tuning of the eye image compression system, usually used recommended practices on obtaining the best acceptable quality (based on usual advantages of HVS, for example) cannot be directly applied due to the eye tracking application specifics. Therefore, a special research was done (experimental results are presented in the following chapter) on studying the impact of eye image compression system parameters on the final precision of the eye tracking output results. These parameters include: execution time of eye image compression/decompression time, size of resulted compressed eye image, quality of decompressed eye image and ability for the compression system to operate in lossy transmission medium.

In contrast with classical general purpose image compression algorithms, a several eye tracking application specific alternative image compression approaches were proposed. For example, applying of Artificial Neural Networks (ANN) to the EyeDee™ solution has a certain potential. With use of ANNs it is possible to consider, rapidly implement and evaluate several approaches as: regression based eye

tracking, classification based eye image compression, classification based ROI finding and some others advanced approaches, experimental results of which are presented in the following chapter.

Chapter 4

Experimental Results

4.1 Introduction

This chapter contains experimental results of research, which was carried out based on a particular product (Chapter 1, Introduction) in the domain of defined theory (Chapter 2, Theoretical Part) and according to applied methodology (Chapter 3, Methodology).

4.2 Reproducibility of the Results

The results presented in this thesis were obtained based on EyeDee™ eye tracking solution, which consist of Weetsy™ portable wire/wireless system including Weetsy™ frame, Weetsy™ board, π-Box™ remote smart sensor and PC-based processing unit running SuriDev eye tracking software.

The experimental research results presented in this chapter can be reproduced by deploying modified version (with implemented methods, approaches and algorithms) of EyeDee™ eye tracking solution on a particular PC. All hardware and software used during the research (Table 4), as well as test video sequences are available upon a dedicated request to SuriCog via:

SuriCog
130 rue de Lourmel 75015 Paris France
+33 01 40 60 70 60
contact@suricog.com

Due to the fact that research conducted in this thesis is not a medical research, there is no test cases, where results were obtained on multiple users/patients, dedicated patient groups and similar methods involving participation of the individuals. Rather the results were obtained by the tests, in which only author himself was participated.

Table 4. Hardware/software used during the research.

Product	Description
Hardware	
Intel Core i7 3.6 GHz 64 bit, 4 physical cores, 8 logical cores	General purpose processor
Renesas RZ/A1H	MCU used in the Weetsy™ board
Altera Cyclone V FPGA	FPGA used in the Weetsy™ board
OV7251 CMOS sensor	CMOS image sensor used in the Weetsy™ frame
Software	
Windows 10 64 bit	General purpose OS
Visual Studio Community 2017	IDE for C/C++
MATLAB R2018a	Environment for numerical computing
ARM DS-5 5.15.0	IDE for Renesas RZ/A1H
Quartus Prime Lite Edition 15.0	IDE for Altera Cyclone V FPGA
Signal Tap software (part of Quartus Prime)	Used for measuring Renesas RZ/A1H performance
Torch7 scientific computing framework	Machine learning

4.3 SuriCog's Eye Tracking Algorithm Improvements

4.3.1 Hardware Based ROI Finder

Finding of dynamic ROI is considered as being 1st step of total eye image compression. Results produces by both PC-based desktop software and FPGA-based hardware ROI-finders were experimentally compared: FPGA-based ROI-finder shows near-the-same accuracy (Figure 67) as default software-based ROI-finder. Hence, FPGA ROI-finder can be deployed to Weetsy™ board leading to bypass of ROI finder in processing unit.

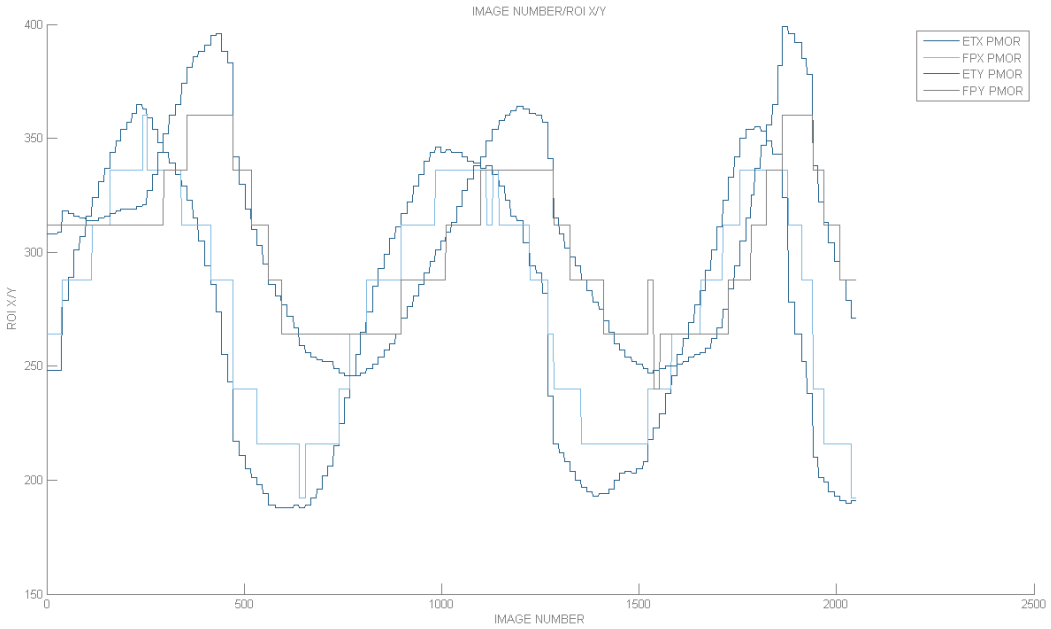


Figure 67. Visual representation of ROI finding in Weetsy™ board (FPGA).

Ideally full pupil's finding algorithm is implemented inside the FPGA. However, porting of the complete eye tracking algorithm is a time consuming task (eye tracking algorithm's optimization, reduction of operating memory needed, minimizing usage floating-point operations, etc.). A special operation mode was introduced, where FPGA-based ROI finder works in co-existence with default PC-based software ROI finder sending some diagnostic information about its operation to the processing unit. This technique allows to progressively improve quality of FPGA-based ROI finder while continuing usage of the default one. FPGA ROI step was configured for 24 and 12 pixels (Table 5).

Table 5. Comparison of FPGA ROI finder precision.

Video sequence Reference ET	Eye tracking error, %		
	SuriQuat ET ROI finder	FPGA ROI finder (24 pix step)	FPGA ROI finder (12 pix step)
PMOR_640x640_01.AVI, 640x640 (750 frames)	1.25	2.48	1.98
PMOR_400x260_01.AVI, 400x260 (750 frames)	1.32	2.67	2.02
MS_640x640_01.AVI, 640x640 (750 frames)	1.17	2.15	1.83
MS_400x260_01.AVI, 400x260 (750 frames)	1.18	2.39	1.99
EGAFF_640x640_01.AVI, 640x640 (750 frames)	1.89	3.05	2.27
EGAFF_400x260_01.AVI, 400x260 (750 frames)	1.96	3.28	2.46

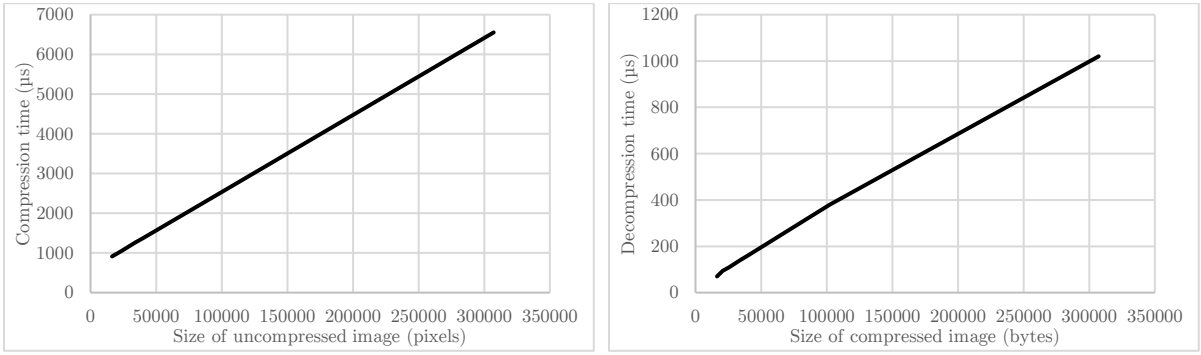
4.4 Finding of the Eye Image Compression Algorithm Requirements

To find system performance bottlenecks we decided to accurately measure entire image acquisition system (also called 'image delivery chain') to represent an entire system via set of System of Linear Equations (SLE), which are derived in this section.

4.4.1 Finding Maximal Time of Eye Image Compression/Decompression

Results show that dependency of uncompressed image size on compression time for compressor JCU grows linearly (Figure 68, a) as well as dependency of compressed image size-decompression for decompressor libjpeg-turbo (Figure 68, b).

NOTE: Since compression of eye images of resolutions 400x256 and 640*480 takes more time than maximal time of acquisition with Signal Tap (2.73 us) we extrapolated compression time results for eye images of these resolutions.



a. JCU b. libjpeg-turbo

Figure 68. Compression/decompression time assessment: JCU vs. libjpeg-turbo.

According to the results (Figure 68) in both cases time grows linearly, so their representation is an equation of a straight line $y = kx + b$ (Table 6).

NOTE: Changing of JCU JPEG quality (1..100) does not affect image compression time (because in fact JPEG quality defines only quantization tables content), so we consider that these parameters are totally independent.

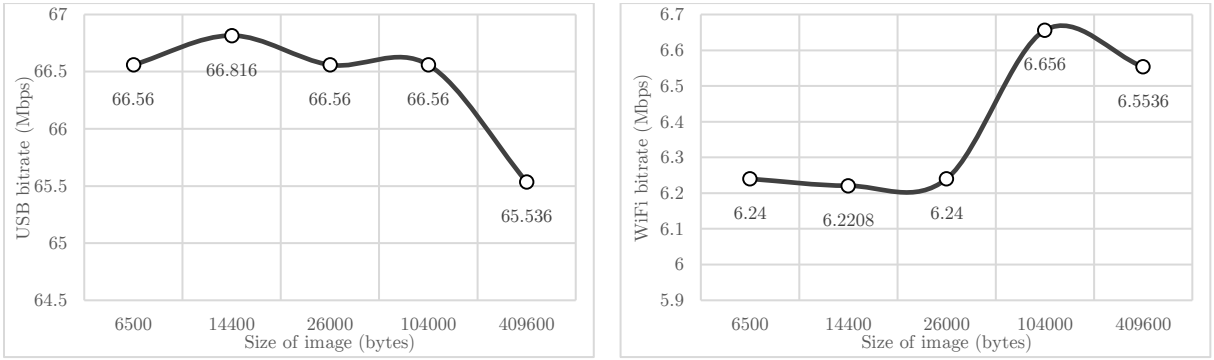
Table 6. Derived equations for uncompressed/compressed image size/time.

Compressor: Renesas RZ/A1H JCU	
Dependency	Uncompressed image size-compression time', where x is uncompressed image size (in 16bpp YUV pixels) and f(x) is compression time
Derived equation	f(x) = 0,019393631x + 594,254743
Decompressor: libjpeg-turbo	
Dependency	Compressed image size-decompression time' SLE, where x is compressed image size (in bytes) and f(x) is decompression time
Derived equation	f(x) = 0,00326667x + 16,4788732

The next step consists of measuring of the capabilities of the transmission mediums used: USB 2.0 (limited implementation) and Wi-Fi (WLAN 802.11n, full implementation).

Since in FPGA+MCU based readout images are read in packet-by-packet mode, there is a need to verify that reading cycle delivers the same performance across difference eye image resolutions. In a certain cases (high quantity of small packets, we call in 'small data blocks') due to internal implementation *overhead* related to increased number of variable comparisons on for/while loops the overall performance can degrade.

According to our measurements, resulting dependency of size of transmitted image over measured bitrate for both USB and Wi-Fi varies insignificantly (Figure 69) and hence can be neglected taking the consideration that both USB and Wi-Fi deliver the same bitrate during reading eye images of any resolution supported (full size image, static ROI, dynamic ROI).



a. USB b. Wi-Fi

Figure 69. Size of transmitted image over measured bitrate: USB vs. Wi-Fi.

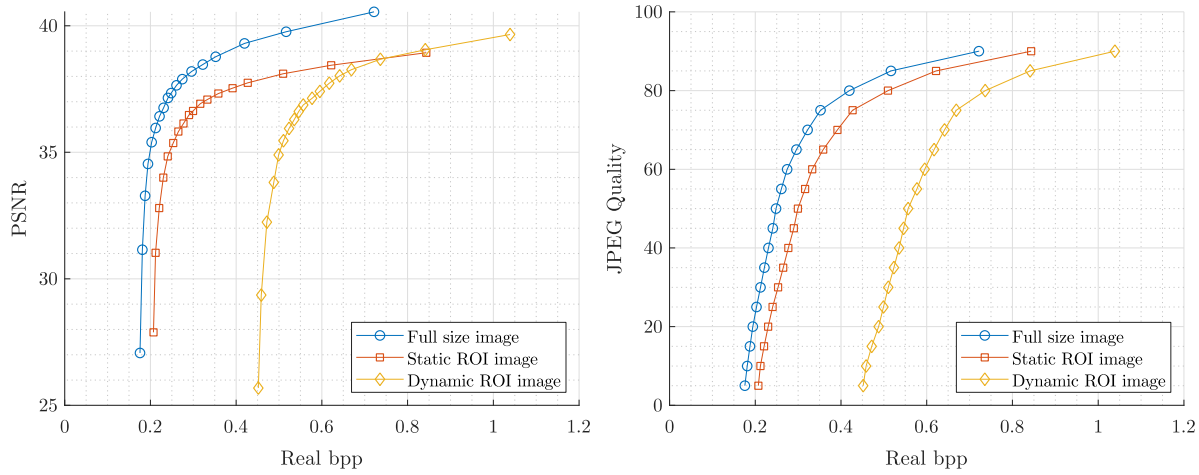
Conclusion based on the measurements:

- USB approx. bitrate is constant for and image sizes and about 66 Mbps;
- Wi-Fi approx. bitrate is constant for dynamic ROI image sizes (80x80...160x160) and about 6.2 Mbps.
- Wi-Fi approx. bitrate is differs for static ROI image and full size image (400x260, 640x640) and about 6.5 Mbps. This variation is clearly illustrates 'small data blocks' issue.

4.4.2 Finding Minimal Size of Compressed Eye Image

Because eye image of each different resolution contains more eye features, compressor results in terms of real bpp will be different across image resolutions with the same compressor's settings. To avoid this pitfall we measure compressor's behavior for each resolution independently. According to results of measurement of the JPEG quality vs. bpp vs. PSNR, decompressed image quality over real bpp (Figure 70, a) as well as JPEG quality over real bpp (Figure 70, b) continuously grows as expected according to well-known performance of JPEG standard.

NOTE: Since JCU uses internal 16bpp YUV2 representation, the 'bpp' should be interpreted as 'bits-per-16bpp pixel in YUV2 format'.



a. Decompressed image quality over real bpp b. JPEG quality over real bpp

Figure 70. JCU assessment results: JPEG quality vs. bpp vs. PSNR.

Then we use a linear interpolation to derive needed SLEs (Table 7).

Table 7. Derived SLEs for bpp and PSNR/JPEG quality.

Derived 'real bpp-PSNR' SLE, where x is real bpp and f(x) is PSNR	Derived 'real bpp-QUAL' SLE, where x is real bpp and f(x) is JPEG quality
Full size image (640x640)	
$f(x)=\begin{cases} 6.7833\cdot10^{2}\cdot x+-9.1638\cdot10^{1}, & \text{if } x\in[0.175,0.181], \\ 3.5667\cdot10^{2}\cdot x+-3.3417\cdot10^{1}, & \text{if } x\in(0.181,0.187], \\ 1.7857\cdot10^{2}\cdot x+-1.1286\cdot10^{-1}, & \text{if } x\in(0.187,0.194], \\ 1.0750\cdot10^{2}\cdot x+1.3675\cdot10^{1}, & \text{if } x\in(0.194,0.202], \\ 5.6000\cdot10^{1}\cdot x+2.4078\cdot10^{1}, & \text{if } x\in(0.202,0.212], \\ 5.2222\cdot10^{1}\cdot x+2.4879\cdot10^{1}, & \text{if } x\in(0.212,0.221], \\ 3.6667\cdot10^{1}\cdot x+2.8317\cdot10^{1}, & \text{if } x\in(0.221,0.23], \\ 4.0000\cdot10^{1}\cdot x+2.7550\cdot10^{1}, & \text{if } x\in(0.23,0.24], \\ 2.2500\cdot10^{1}\cdot x+3.1750\cdot10^{1}, & \text{if } x\in(0.24,0.248], \\ 2.5833\cdot10^{1}\cdot x+3.0923\cdot10^{1}, & \text{if } x\in(0.248,0.26], \\ 1.7143\cdot10^{1}\cdot x+3.3183\cdot10^{1}, & \text{if } x\in(0.26,0.274]. \end{cases}$	$f(x)=\begin{cases} 8.3333\cdot10^{2}\cdot x+-1.4083\cdot10^{2}, & \text{if } x\in[0.175,0.181], \\ 8.3333\cdot10^{2}\cdot x+-1.4083\cdot10^{2}, & \text{if } x\in(0.181,0.187], \\ 7.1429\cdot10^{2}\cdot x+-1.1857\cdot10^{2}, & \text{if } x\in(0.187,0.194], \\ 6.2500\cdot10^{2}\cdot x+-1.0125\cdot10^{2}, & \text{if } x\in(0.194,0.202], \\ 5.0000\cdot10^{2}\cdot x+-7.6000\cdot10^{1}, & \text{if } x\in(0.202,0.212], \\ 5.5556\cdot10^{2}\cdot x+-8.7778\cdot10^{1}, & \text{if } x\in(0.212,0.221], \\ 5.5556\cdot10^{2}\cdot x+-8.7778\cdot10^{1}, & \text{if } x\in(0.221,0.23], \\ 5.0000\cdot10^{2}\cdot x+-7.5000\cdot10^{1}, & \text{if } x\in(0.23,0.24], \\ 6.2500\cdot10^{2}\cdot x+-1.0500\cdot10^{2}, & \text{if } x\in(0.24,0.248], \\ 4.1667\cdot10^{2}\cdot x+-5.3333\cdot10^{1}, & \text{if } x\in(0.248,0.26], \\ 3.5714\cdot10^{2}\cdot x+-3.7857\cdot10^{1}, & \text{if } x\in(0.26,0.274]. \end{cases}$
$f(x)=\begin{cases} 1.4762\cdot10^{1}\cdot x+3.3835\cdot10^{1}, & \text{if } x\in[0.274,0.295], \\ 1.0000\cdot10^{1}\cdot x+3.5240\cdot10^{1}, & \text{if } x\in(0.295,0.322], \\ 1.0333\cdot10^{1}\cdot x+3.5133\cdot10^{1}, & \text{if } x\in(0.322,0.352], \\ 7.9104\cdot x+3.5986\cdot10^{1}, & \text{if } x\in(0.352,0.419], \\ 4.7423\cdot x+3.7313\cdot10^{1}, & \text{if } x\in(0.419,0.516], \\ 3.8049\cdot x+3.7797\cdot10^{1}, & \text{if } x\in(0.516,0.721], \\ 2.5052\cdot x+3.8734\cdot10^{1}, & \text{if } x\in(0.721,1.204], \\ 3.6480\cdot10^{-1}\cdot x+4.1311\cdot10^{1}, & \text{if } x\in(1.204,4.329]. \end{cases}$	$f(x)=\begin{cases} 2.3810\cdot10^{2}\cdot x+-5.2381, & \text{if } x\in[0.274,0.295], \\ 1.8519\cdot10^{2}\cdot x+1.0370\cdot10^{1}, & \text{if } x\in(0.295,0.322], \\ 1.6667\cdot10^{2}\cdot x+1.6333\cdot10^{1}, & \text{if } x\in(0.322,0.352], \\ 7.4627\cdot10^{1}\cdot x+4.8731\cdot10^{1}, & \text{if } x\in(0.352,0.419], \\ 5.1546\cdot10^{1}\cdot x+5.8402\cdot10^{1}, & \text{if } x\in(0.419,0.516], \\ 2.4390\cdot10^{1}\cdot x+7.2415\cdot10^{1}, & \text{if } x\in(0.516,0.721], \\ 1.0352\cdot10^{1}\cdot x+8.2536\cdot10^{1}, & \text{if } x\in(0.721,1.204], \\ 1.6000\cdot x+9.3074\cdot10^{1}, & \text{if } x\in(1.204,4.329]. \end{cases}$
Static ROI image (400x260)	

$$f(x)=\begin{cases}6.2800\cdot10^{2}\cdot x+-1.0212\cdot10^{2}, & \text{if } x\in[0.207,0.212],\\ 2.2125\cdot10^{2}\cdot x+-1.5885\cdot10^{1}, & \text{if } x\in(0.212,0.22],\\ 1.2000\cdot10^{2}\cdot x+6.3900, & \text{if } x\in(0.22,0.23],\\ 8.4000\cdot10^{1}\cdot x+1.4670\cdot10^{1}, & \text{if } x\in(0.23,0.24],\\ 4.0769\cdot10^{1}\cdot x+2.5045\cdot10^{1}, & \text{if } x\in(0.24,0.253],\\ 3.7500\cdot10^{1}\cdot x+2.5872\cdot10^{1}, & \text{if } x\in(0.253,0.265],\\ 2.7500\cdot10^{1}\cdot x+2.8522\cdot10^{1}, & \text{if } x\in(0.265,0.277],\\ 2.7500\cdot10^{1}\cdot x+2.8523\cdot10^{1}, & \text{if } x\in(0.277,0.289],\\ 1.6000\cdot10^{1}\cdot x+3.1846\cdot10^{1}, & \text{if } x\in(0.289,0.299],\\ 1.6471\cdot10^{1}\cdot x+3.1705\cdot10^{1}, & \text{if } x\in(0.299,0.316],\\ 1.0625\cdot10^{1}\cdot x+3.3553\cdot10^{1}, & \text{if } x\in(0.316,0.332].\end{cases}$$

$$f(x)=\begin{cases}1.0000\cdot10^{3}\cdot x+-2.0200\cdot10^{2}, & \text{if } x\in[0.207,0.212],\\ 6.2500\cdot10^{2}\cdot x+-1.2250\cdot10^{2}, & \text{if } x\in(0.212,0.22],\\ 5.0000\cdot10^{2}\cdot x+-9.5000\cdot10^{1}, & \text{if } x\in(0.22,0.23],\\ 5.0000\cdot10^{2}\cdot x+-9.5000\cdot10^{1}, & \text{if } x\in(0.23,0.24],\\ 3.8462\cdot10^{2}\cdot x+-6.7308\cdot10^{1}, & \text{if } x\in(0.24,0.253],\\ 4.1667\cdot10^{2}\cdot x+-7.5417\cdot10^{1}, & \text{if } x\in(0.253,0.265],\\ 4.1667\cdot10^{2}\cdot x+-7.5417\cdot10^{1}, & \text{if } x\in(0.265,0.277],\\ 4.1667\cdot10^{2}\cdot x+-7.5417\cdot10^{1}, & \text{if } x\in(0.277,0.289],\\ 5.0000\cdot10^{2}\cdot x+-9.9500\cdot10^{1}, & \text{if } x\in(0.289,0.299],\\ 2.9412\cdot10^{2}\cdot x+-3.7941\cdot10^{1}, & \text{if } x\in(0.299,0.316],\\ 3.1250\cdot10^{2}\cdot x+-4.3750\cdot10^{1}, & \text{if } x\in(0.316,0.332].\end{cases}$$

$$f(x)=\begin{cases}9.2308\cdot x+3.4015\cdot10^{1}, & \text{if } x\in[0.332,0.358],\\ 6.3636\cdot x+3.5042\cdot10^{1}, & \text{if } x\in(0.358,0.391],\\ 5.8333\cdot x+3.5249\cdot10^{1}, & \text{if } x\in(0.391,0.427],\\ 4.3902\cdot x+3.5865\cdot10^{1}, & \text{if } x\in(0.427,0.509],\\ 3.0357\cdot x+3.6555\cdot10^{1}, & \text{if } x\in(0.509,0.621],\\ 2.2072\cdot x+3.7069\cdot10^{1}, & \text{if } x\in(0.621,0.843],\\ 1.7524\cdot x+3.7453\cdot10^{1}, & \text{if } x\in(0.843,1.368],\\ 6.7332\cdot10^{-1}\cdot x+3.8929\cdot10^{1}, & \text{if } x\in(1.368,4.576].\end{cases}$$

$$f(x)=\begin{cases}1.9231\cdot10^{2}\cdot x+-3.8462, & \text{if } x\in[0.332,0.358],\\ 1.5152\cdot10^{2}\cdot x+1.0758\cdot10^{1}, & \text{if } x\in(0.358,0.391],\\ 1.3889\cdot10^{2}\cdot x+1.5694\cdot10^{1}, & \text{if } x\in(0.391,0.427],\\ 6.0976\cdot10^{1}\cdot x+4.8963\cdot10^{1}, & \text{if } x\in(0.427,0.509],\\ 4.4643\cdot10^{1}\cdot x+5.7277\cdot10^{1}, & \text{if } x\in(0.509,0.621],\\ 2.2523\cdot10^{1}\cdot x+7.1014\cdot10^{1}, & \text{if } x\in(0.621,0.843],\\ 9.5238\cdot x+8.1971\cdot10^{1}, & \text{if } x\in(0.843,1.368],\\ 1.5586\cdot x+9.2868\cdot10^{1}, & \text{if } x\in(1.368,4.576].\end{cases}$$

Dynamic ROI image (120x120)

$$f(x)=\begin{cases}5.2429\cdot10^{2}\cdot x+-2.1077\cdot10^{2}, & \text{if } x\in[0.451,0.458],\\ 2.2231\cdot10^{2}\cdot x+-7.2467\cdot10^{1}, & \text{if } x\in(0.458,0.471],\\ 9.8125\cdot10^{1}\cdot x+-1.3977\cdot10^{1}, & \text{if } x\in(0.471,0.487],\\ 9.0000\cdot10^{1}\cdot x+-1.0020\cdot10^{1}, & \text{if } x\in(0.487,0.499],\\ 5.0909\cdot10^{1}\cdot x+9.4864, & \text{if } x\in(0.499,0.51],\\ 3.7692\cdot10^{1}\cdot x+1.6227\cdot10^{1}, & \text{if } x\in(0.51,0.523],\\ 2.9167\cdot10^{1}\cdot x+2.0686\cdot10^{1}, & \text{if } x\in(0.523,0.535],\\ 3.2000\cdot10^{1}\cdot x+1.9170\cdot10^{1}, & \text{if } x\in(0.535,0.545],\\ 2.2727\cdot10^{1}\cdot x+2.4224\cdot10^{1}, & \text{if } x\in(0.545,0.556],\\ 1.2857\cdot10^{1}\cdot x+2.9711\cdot10^{1}, & \text{if } x\in(0.556,0.577],\\ 1.5000\cdot10^{1}\cdot x+2.8475\cdot10^{1}, & \text{if } x\in(0.577,0.595].\end{cases}$$

$$f(x)=\begin{cases}7.1429\cdot10^{2}\cdot x+-3.1714\cdot10^{2}, & \text{if } x\in[0.451,0.458],\\ 3.8462\cdot10^{2}\cdot x+-1.6615\cdot10^{2}, & \text{if } x\in(0.458,0.471],\\ 3.1250\cdot10^{2}\cdot x+-1.3219\cdot10^{2}, & \text{if } x\in(0.471,0.487],\\ 4.1667\cdot10^{2}\cdot x+-1.8292\cdot10^{2}, & \text{if } x\in(0.487,0.499],\\ 4.5455\cdot10^{2}\cdot x+-2.0182\cdot10^{2}, & \text{if } x\in(0.499,0.51],\\ 3.8462\cdot10^{2}\cdot x+-1.6615\cdot10^{2}, & \text{if } x\in(0.51,0.523],\\ 4.1667\cdot10^{2}\cdot x+-1.8292\cdot10^{2}, & \text{if } x\in(0.523,0.535],\\ 5.0000\cdot10^{2}\cdot x+-2.2750\cdot10^{2}, & \text{if } x\in(0.535,0.545],\\ 4.5455\cdot10^{2}\cdot x+-2.0273\cdot10^{2}, & \text{if } x\in(0.545,0.556],\\ 2.3810\cdot10^{2}\cdot x+-8.2381\cdot10^{1}, & \text{if } x\in(0.556,0.577],\\ 2.7778\cdot10^{2}\cdot x+-1.0528\cdot10^{2}, & \text{if } x\in(0.577,0.595].\end{cases}$$

$$f(x)=\begin{cases}1.5000\cdot10^{1}\cdot x+2.8475\cdot10^{1}, & \text{if } x\in[0.595,0.617],\\ 1.2083\cdot10^{1}\cdot x+3.0275\cdot10^{1}, & \text{if } x\in(0.617,0.641],\\ 8.8889\cdot x+3.2322\cdot10^{1}, & \text{if } x\in(0.641,0.668],\\ 6.0294\cdot x+3.4232\cdot10^{1}, & \text{if } x\in(0.668,0.736],\\ 3.6190\cdot x+3.6006\cdot10^{1}, & \text{if } x\in(0.736,0.841],\\ 3.0303\cdot x+3.6502\cdot10^{1}, & \text{if } x\in(0.841,1.039],\\ 2.1308\cdot x+3.7436\cdot10^{1}, & \text{if } x\in(1.039,1.513],\\ 5.6640\cdot10^{-1}\cdot x+3.9803\cdot10^{1}, & \text{if } x\in(1.513,4.638].\end{cases}$$

$$f(x)=\begin{cases}2.2727\cdot10^{2}\cdot x+-7.5227\cdot10^{1}, & \text{if } x\in[0.595,0.617],\\ 2.0833\cdot10^{2}\cdot x+-6.3542\cdot10^{1}, & \text{if } x\in(0.617,0.641],\\ 1.8519\cdot10^{2}\cdot x+-4.8704\cdot10^{1}, & \text{if } x\in(0.641,0.668],\\ 7.3529\cdot10^{1}\cdot x+2.5882\cdot10^{1}, & \text{if } x\in(0.668,0.736],\\ 4.7619\cdot10^{1}\cdot x+4.4952\cdot10^{1}, & \text{if } x\in(0.736,0.841],\\ 2.5253\cdot10^{1}\cdot x+6.3763\cdot10^{1}, & \text{if } x\in(0.841,1.039],\\ 1.0549\cdot10^{1}\cdot x+7.9040\cdot10^{1}, & \text{if } x\in(1.039,1.513],\\ 1.6000\cdot x+9.2579\cdot10^{1}, & \text{if } x\in(1.513,4.638].\end{cases}$$

4.4.3 Finding Minimal Quality of Decompressed Eye Image

To measure tracking accuracy, a reference (i.e., ground truth) has to be obtained. For this purpose tracking has to be done on uncompressed images. For high compression ratios (Figure 71, Figure 72, Figure 73, Figure 74) we are starting to lose the singularities and irregular structures in original image, which contain most sensitive information for the tracking system.

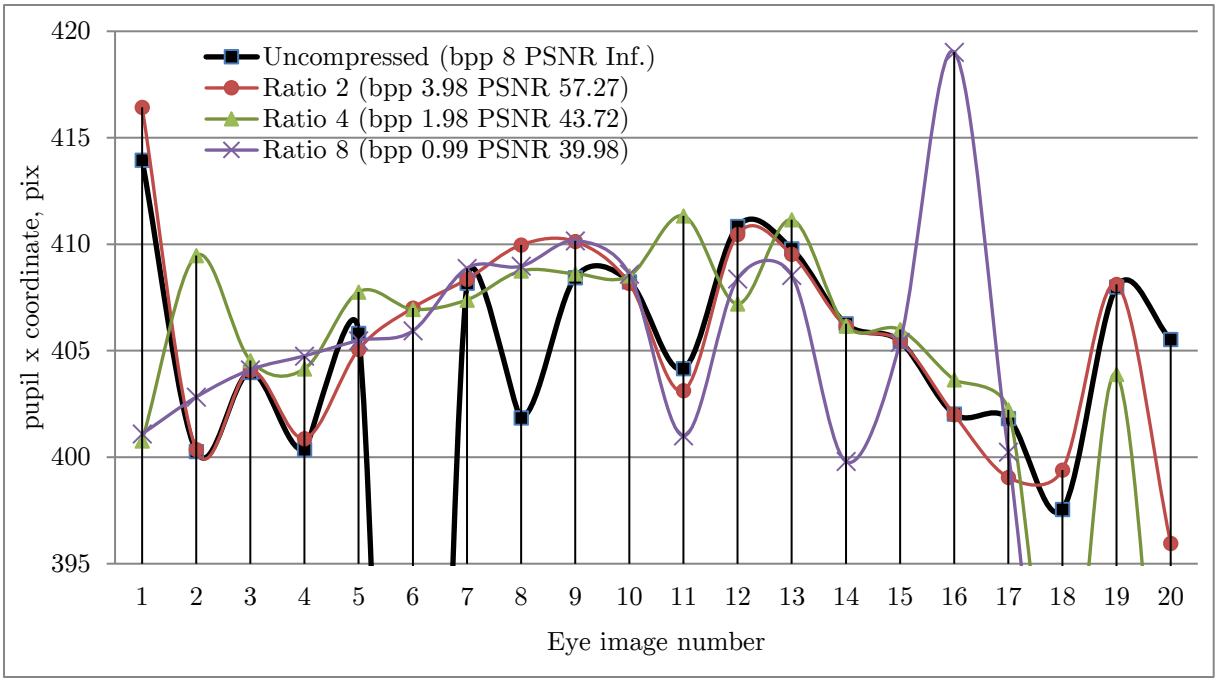


Figure 71. Eye image number vs. pupil x coordinate (low compression ratios).

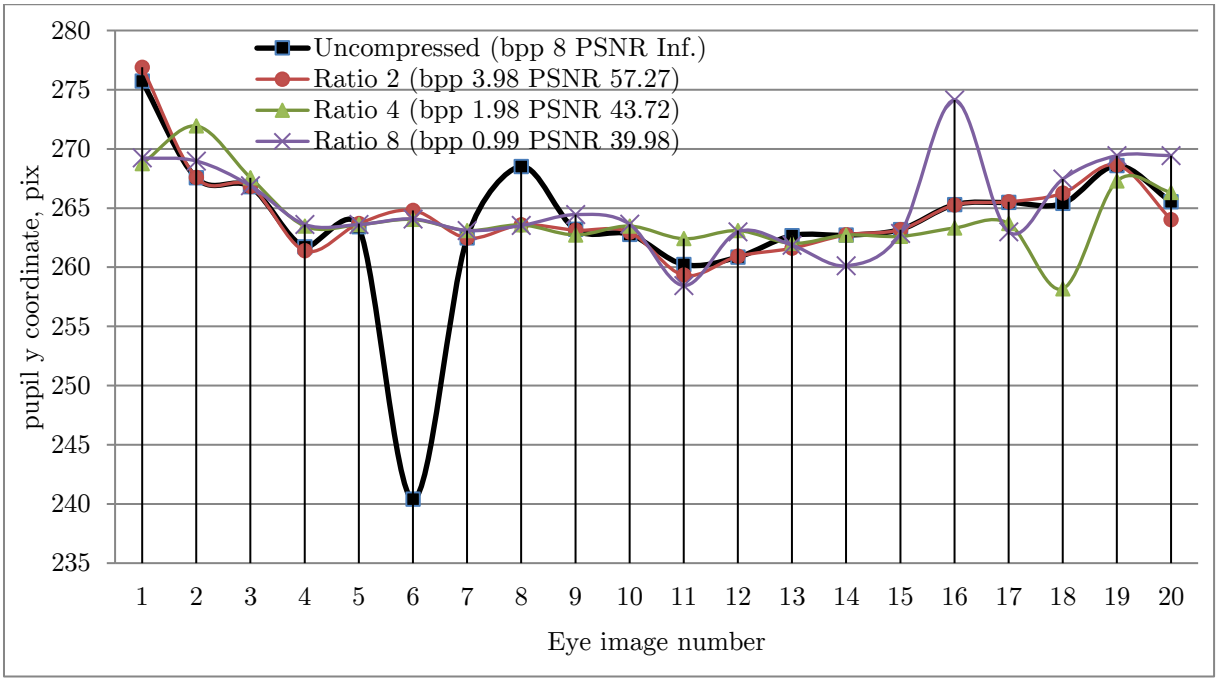


Figure 72. Eye image number vs. pupil y coordinate (low compression ratios).

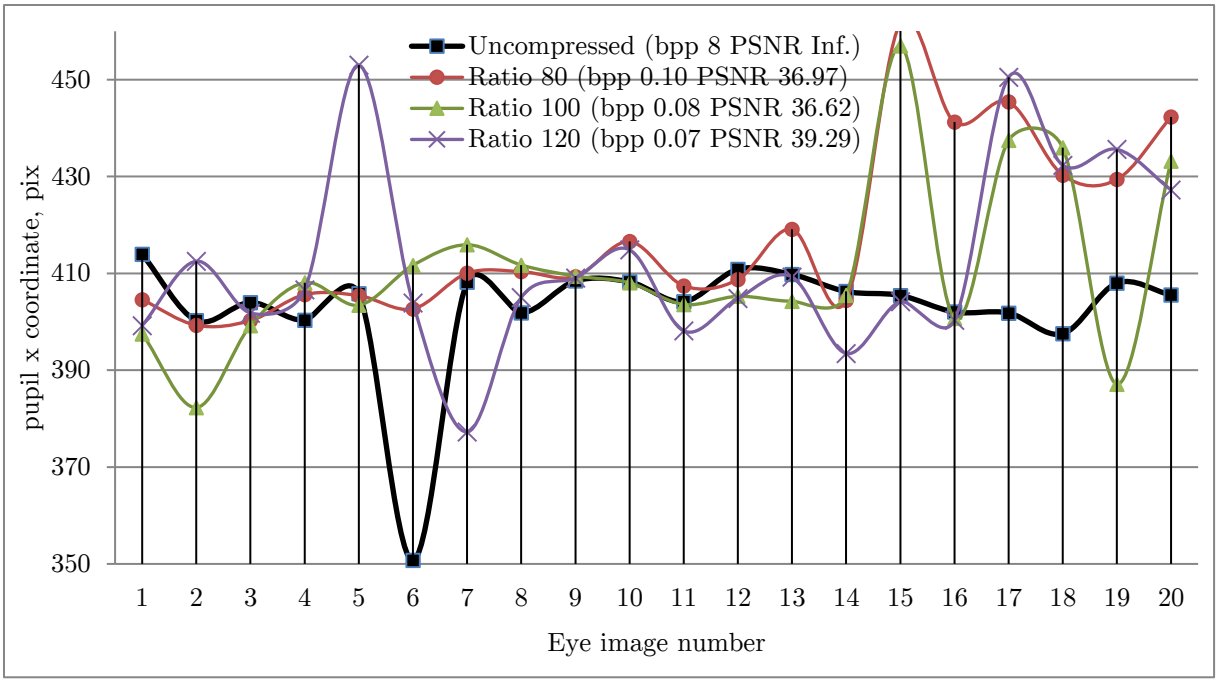


Figure 73. Eye image number vs. pupil x coordinate (high compression ratios).

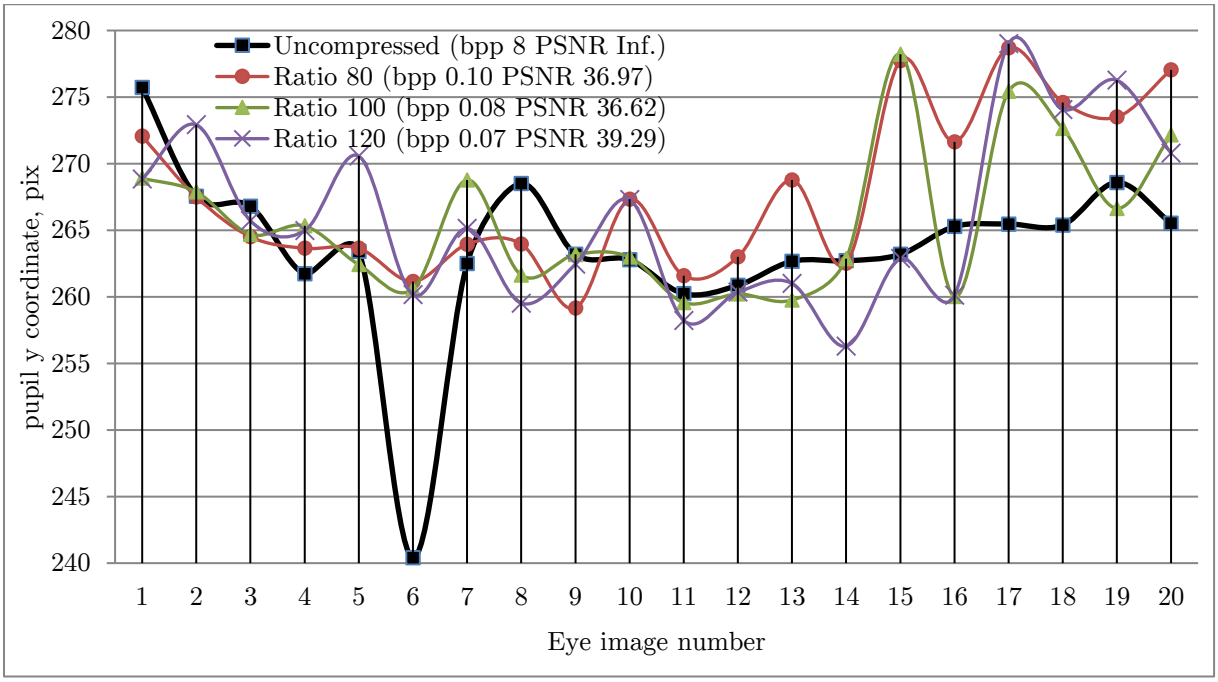


Figure 74. Eye image number vs. pupil y coordinate (high compression ratios).

Compression with ratio 2 leads to glossy image and, therefore, can be interpreted as part of a denoising algorithm. Ratios starting with 8 leads to tracking errors (human's eye is maximally turned in each direction), which are directly linked to the compression ratios. However, there are regions where the results are very close to the results obtained on uncompressed image. High compression ratios lead to errors that cannot be neglected. As a result so-called «mistracking» (Figure 77) occurs. To find the boundary, where compression can be used without significant degradation of eye tracking results, it is needed to measure the drift [291], (i.e., average systematic error) between eye tracking algorithm output result – pupil x/y coordinate (Figure 75) on the original (uncompressed) and decompressed images as well as drift between eye image segmentation thresholds THi/TLo (Figure 76) on the original (uncompressed) and decompressed images.

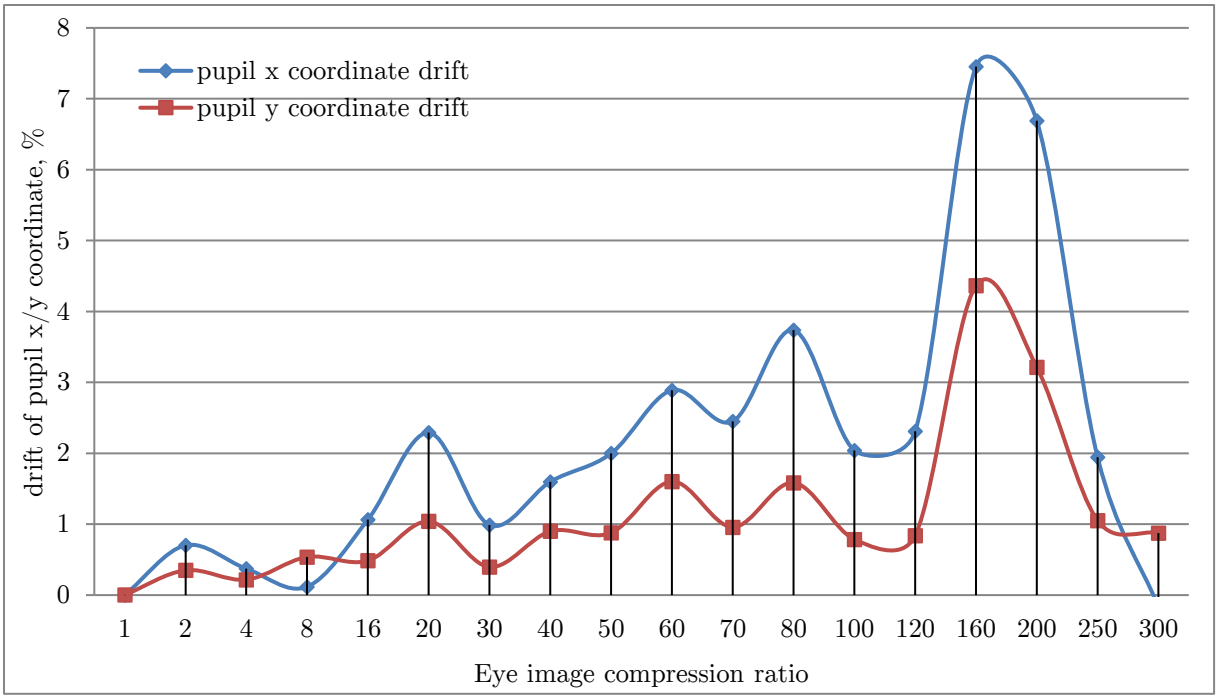


Figure 75. Drift of pupil x/y coordinate vs. eye image compassion.

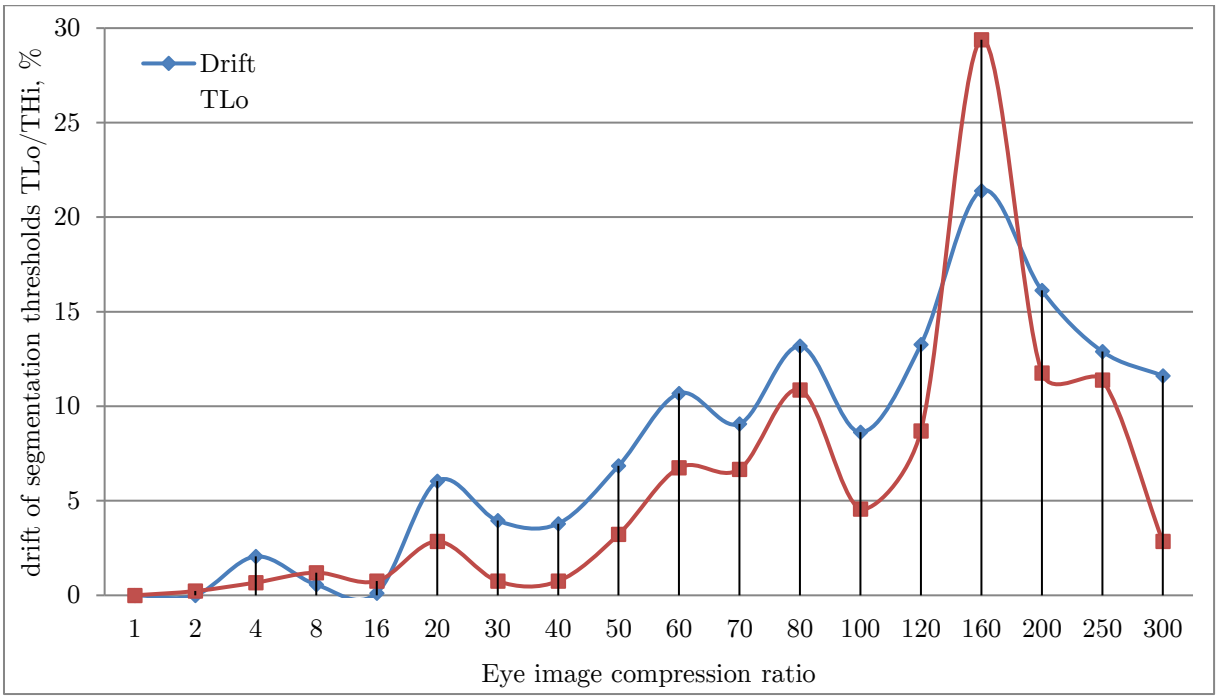
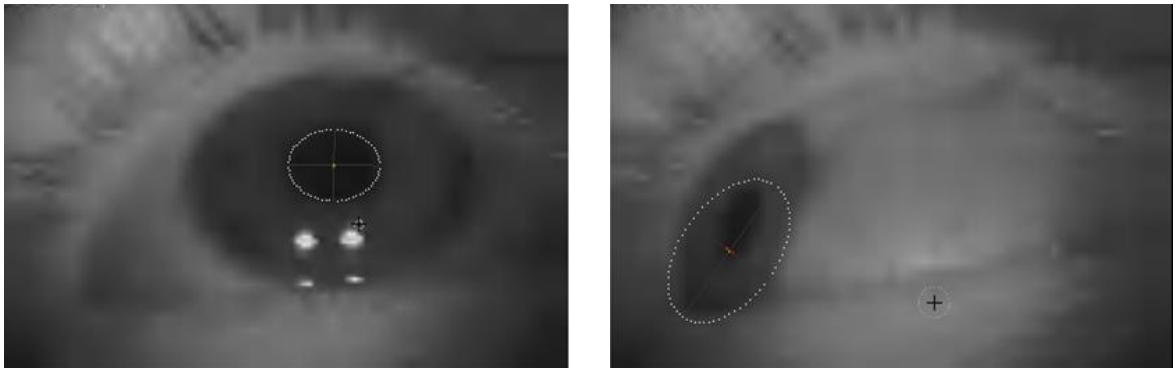


Figure 76. Drift of segmentation thresholds TLo/THi vs. eye image compassion ratio.

As it can be seen there is no proportional increase of the error with the increase of compression rations. This is due to the fact that different ratios lead to different types of distortion for the decompressed image. For example, ratios less or equal to 16 lead to less than 1% tracking errors, while ratios less or equal 50 lead to 2% tracking errors. Ratios higher than 120 lead to significant tracking errors (more than 4-5%, *mistracking*, as can be seen in (Figure 77).

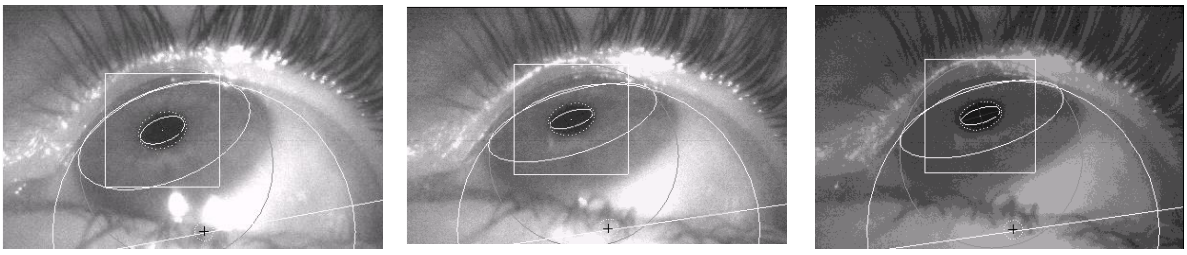


a. Successful tracking b. Mistracking

Legend: Eye image compression ratio 160, bpp 0.05, PSNR 34.73 dB, codec JPEG 2000

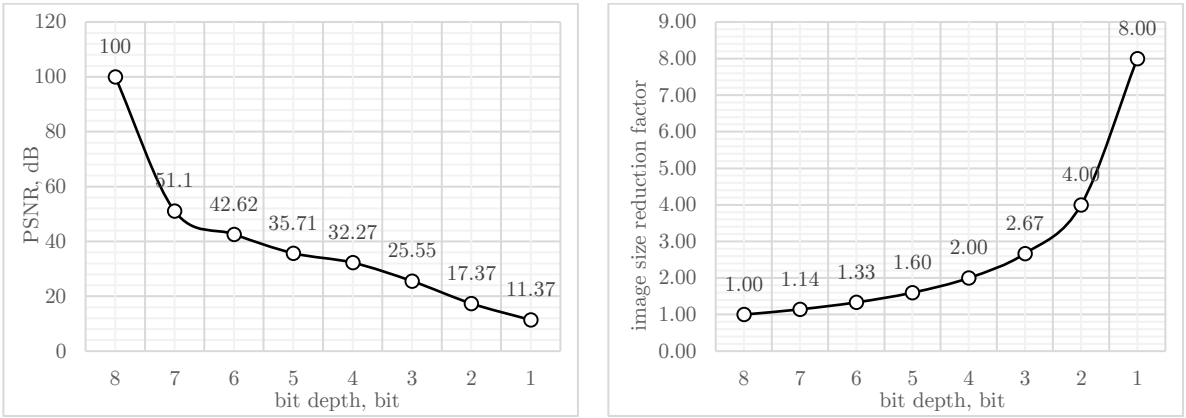
Figure 77. Successful tracking and mistracking.

Another simple, but effective technique is eye image pixel bit depth reduction. According to conducted research (Figure 78, Figure 79, Figure 80) usage of eye images of 4bpp commits on 1.25..8.78% of eye tracking precision errors, while immediately 50%-increasing eye image transmission performance, because of 50% less data to transmit from Weetsy™ board to PC based processing unit performing the eye tracking. Which makes this 'bit depth reduction' approach be acceptable in the wide range of multimedia applications (for example, interaction with the objects placed in a known 3D environment), where slight decreasing of eye tracking precision will not result in for example industrial disaster, life loss or misdiagnosis.



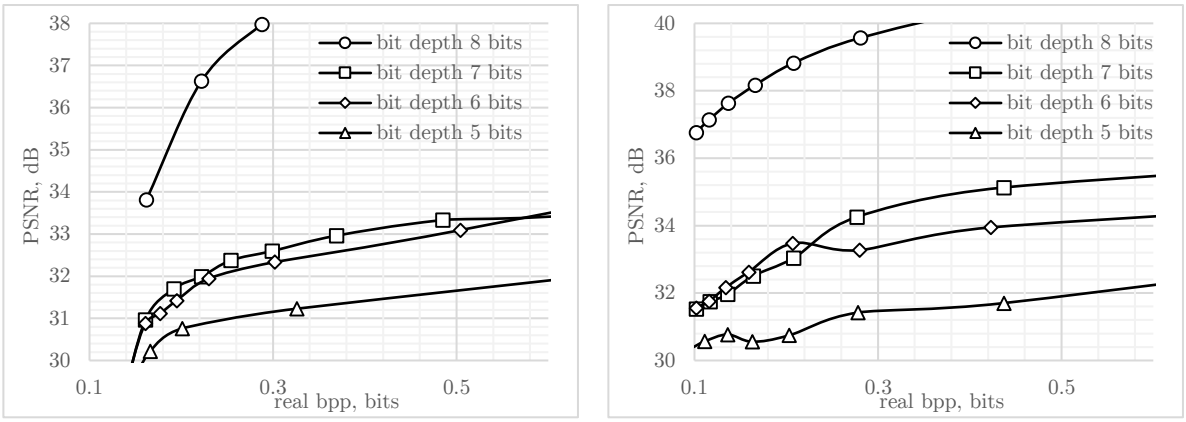
a. 8 bpp, PSNR Inf dB b. 4bpp, PSNR 32.27 dB c. 3 bpp, PSNR 25.55 dB

Figure 78. Visual comparison of eye images with reduced bit depth.



a. Image quality over bit depth b. Image size reduction factor over bit depth

Figure 79. Pixel bit depth reduction: uncompressed images.



a. Image quality over real bpp (JPEG) b. Image quality over real bpp (JPEG 2000)

Figure 80. Pixel bit depth reduction: compressed images (JPEG vs. JPEG 2000).

4.4.4 Considerations on Ability to Operate in Lossy Transmission Medium

To protect compressed image data (bitstream) against errors or loses a CRC16 was initially used following by its replacement to highly optimized CRC8, which was however used for data blocks even more than 256 bytes (max 1500 bytes) with the assumption that probability of obtaining the same CRC for corrupted data as for non-corrupted data stays low hence can be neglected. CRC8 was implemented at the application layer, during reading eye image pixels from the FPGA RAM into the MCU RAM via 16 bit data bus. Such an implementation 'in-place' leads to about 0.002% of slow-down of the reading, which can be neglected during FPS estimation.

```
97  void SCGmemcpyCRC8(char *pDst, char *pSrc, int len, unsigned char *pCrc8)
98  //-------------------------------------------------------------
99  //-------------------------------------------------------------
100 {   register short *pwd, *pws, word;     register int i, nbw;   unsigned char crc8;
101
102     nbw = (len + 1) / 2;     pwd = (short*)pDst;     pws = (short*)pSrc; crc8 = 0;
103     do {
104         *pwd = *pws;
105         //*pwd = 0xFFFF;
106         CRC8(crc8, (unsigned char)(*pwd & 0xFF));
107         CRC8(crc8, (unsigned char)(*pwd >> 8));
108         ++pwd;  ++pws;
109     } while (--nbw > 0);
110     *pCrc8 = crc8;
111 }
```

Figure 81. Computing of the CRC8 'in-place' during reading of the pixels.

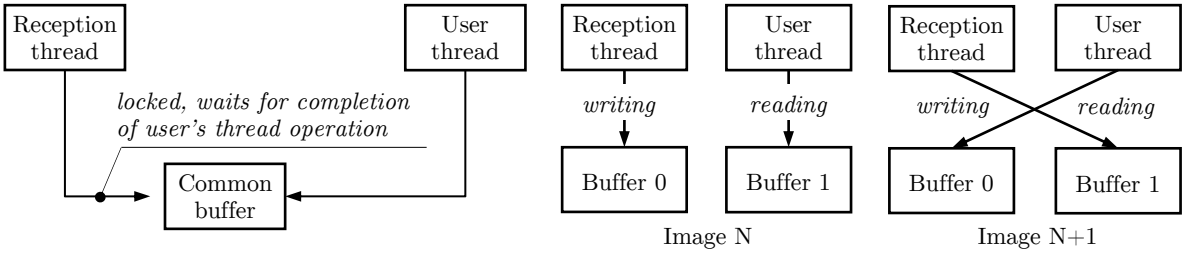
According to 6 hours continuous performance test of UDP transmission (Table 8) of packets with compressed image data from Weetsy™ board to PC based processing unit running Wi-Fi 2.4 GHz Access Point (AP) in a form of dedicated USB device (USB dongle) at the distance 2..8 meters there are about 0% of packets (Table 8, pkt_rcvd_with_crc_error) on which CRC8 checking was failed. Because of this we decided to avoid usage of this however optimized CRC8 done 'in-place' to accelerate the processing as much as possible. Such statistics even allows to use in Wi-Fi/RF stations-free environments some video compression systems with special profiles installed allowing in patricidal decoding of a corrupted video bitstream.

Table 8. Six hours continuous performance test of UDP transmission.

pkt_rcvd	2264306
pkt_rcvd_with_crc_error	0 (0.00 %)
img_rcvd	780200
img_rcvd_with_seq_error	619 (0.08 %)
pkt/img	2.90
total_bytes_rcvd	2545609516

payload_bytes_rcvd	2416395668
headers overhead	5.08 %
compression ratio	36.78
pkt_skip_client_busy	13745
bytes_skip_client_busy	17507426 (0.72 % of payload_bytes_rcvd)
datarate	0.11 Mbytes/sec 0.86 Mbits/sec

For a possible cases when data is corrupted we decided to protect JPEG header by sending only coded JPEG data while remote PC based processing unit uses its local copy of JPEG header (because JPEG header stays constant from image to image). In this case even if coded JPEG data is corrupted there is no risk that JPEG decoder will crash due to corrupted header. Before the transmission, PC based processing unit generates its local copy of JPEG header based on image resolution and JPEG quality (based on which quantization table is generated). According to statistics size of JPEG header takes in average about 12.5% of JPEG compressed static ROI eye image at JPEG quality 75, i.e., sending only JPEG coded data without JPEG header allows to reduce amount of transmitted data by 12.5% at minimum. Another issue is the packets lost due to UDP best-effort delivery (Table 8, img_rcvd_with_seq_error). There is no so much approaches to reduce this result, except acceleration of the reception as much as possible. On our case we decided to use multithreaded implementation, where data reception thread is running as much as possible without any locks (even copying of received data to 'user' buffer with is considered as 'lock', because it takes a certain amount of time). We decided to use classical multiple-buffering technique (in our case double-buffering, Figure 82) also called 'flip-flop technique', where reception thread stores currently received image data into buffer N while user thread reads previously received image data from buffer N-1. This technique allows to unblock reception thread in a moments when user thread does copying of the image data to its local user buffer.



Reception thread is locked until user thread is finished data copying of the data received

Reception thread keeps running while user thread does data copying of the data received

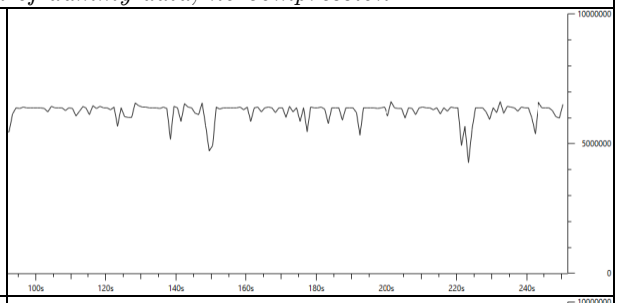
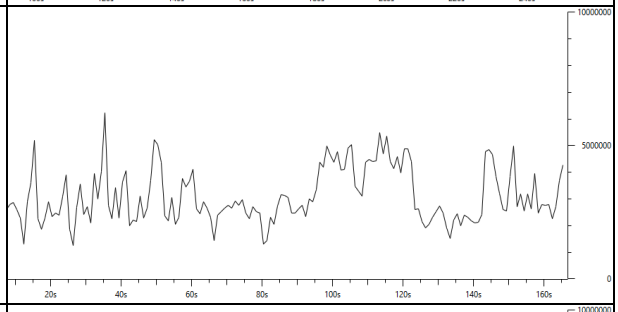
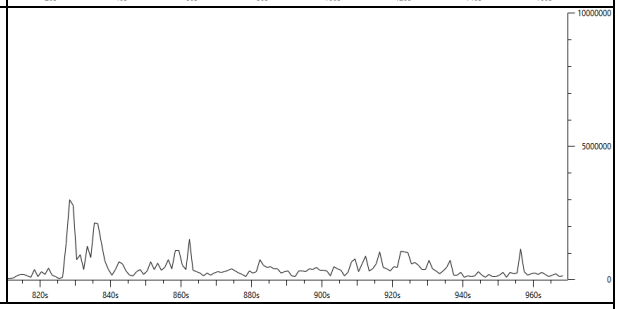
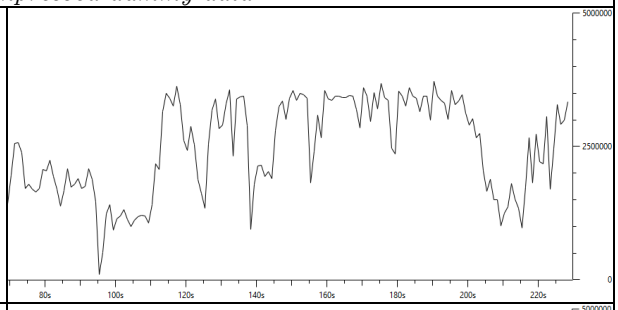
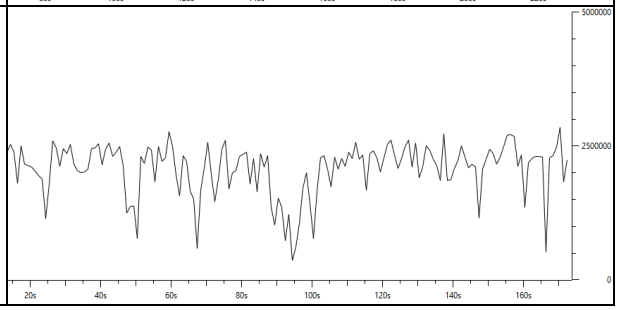
Figure 82. Single bufferring vs. multiple bufferring.

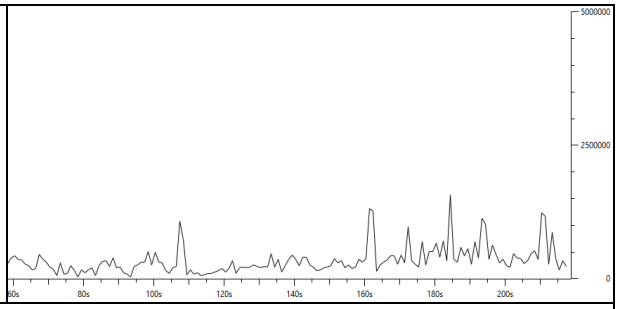
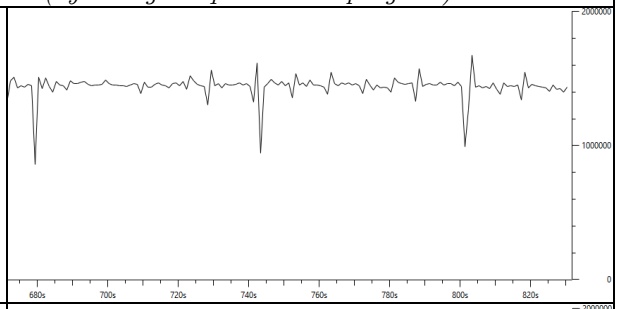
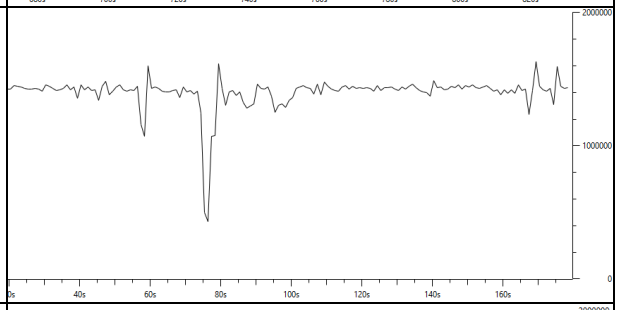
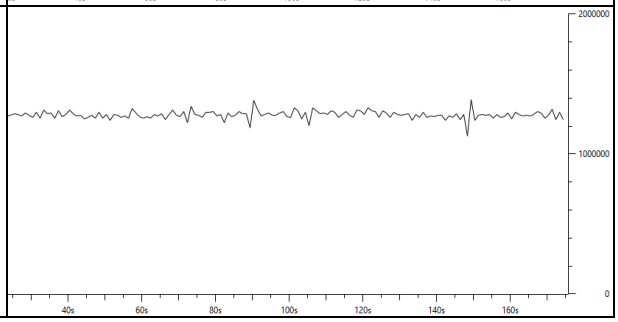
Currently used wireless hardware (Wi-Fi chip with RF module inside connected to external u.FL antenna) is very sensitive to change of environment conditions during data transmission (metallic objects between the antenna and the access point, presence of other stations in the same room, etc.), the amount of bitsteram corrupted and/or bitsteram lost is shown to be much higher than 20%. Which makes almost impossible to *constantly use* video compression techniques, aimed to remove both spatial and temporal redundancies from the input eye images.

Wireless transmission bitrate measurement (Table 9) on several distances between Weetsy™ board and PC based processing unit running Wi-Fi 2.4 GHz (AP) Access Point gives the following results on a distance 1 meter (typical usage distance):

1. Bitrate about 5.69..6.24 Mbit/sec (transmission of dummy data, no compression).
2. Bitrate about 2.21..2.23 Mbit/sec (transmission of compressed dummy data).
3. Bitrate about 1.41..1.45 Mbit/sec (transmission of compressed image data (eye image acquisition in progress).

Table 9. Results of wireless transmission bitrate measurement.

Distance, meters	Bitrate, Mbit/sec, measured with Tester	Bitrate, Mbit/sec, measured with Wireshark	Graph, obtained with Wireshark
Peak performance, transmission of dummy data, no compression			
1	5.69	6.24	
4	3.25	2.82	
8	1.15	2.85	
Transmission of compressed dummy data			
1	2.21	2.23	
4	1.93	2.00	

8	1.39	0.55	
Transmission of compressed image data (eye image acquisition in progress)			
1	1.41	1.45	
4	1.40	1.40	
8	0.54	1.27	

4.4.5 Image Compression Basic Configurations

There are two basic image compression configurations used in the Weetsy™ board:

- Compression of static ROI (Figure 83).
- Compression of the dynamic ROI (Figure 84).

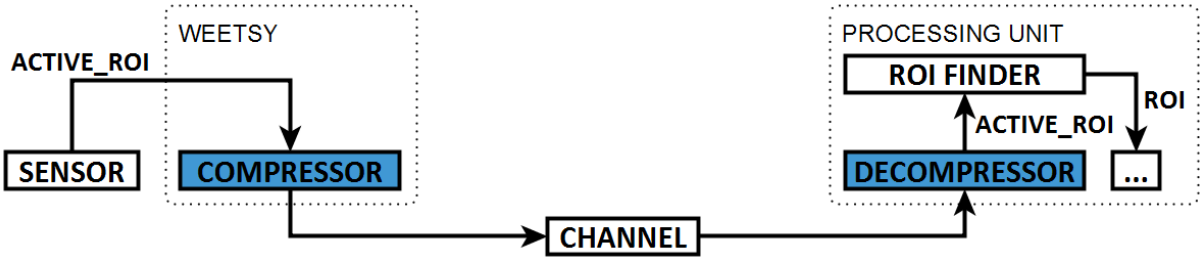


Figure 83. Configuration for compression of static ROI.

Compressed ROI Configuration

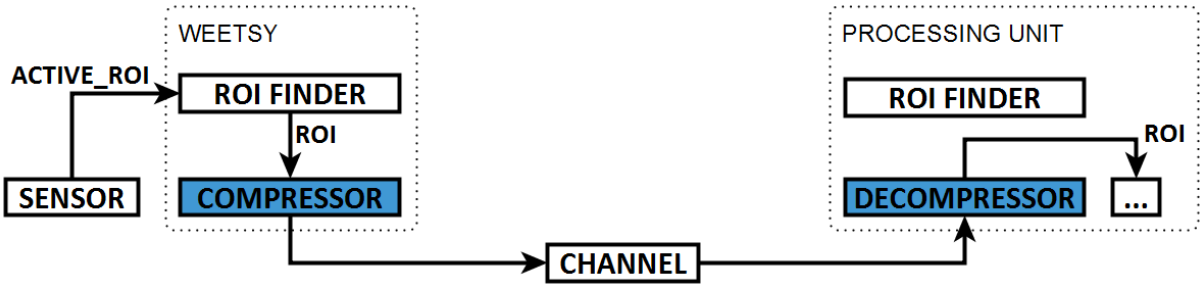


Figure 84. Configuration for compression of dynamic ROI.

4.5 Experimentations with Image/Video Compression Systems

4.5.1 Available Products

We tested several available ready-to-use products:

1. OpenJPEG [270] is an open-source library to encode and decode JPEG 2000 images. As of version 2.1 released in April 2014, it is officially conformant with the JPEG 2000 Part-1 standard. It was subsequently adopted by ImageMagick instead of JasPer in 6.8.8-2 and approved as new reference software for this standard in July 2015. Author: Université Catholique de Louvain (UCL). License: BSD.
2. Kakadu [292] is a closed-source library to encode and decode JPEG 2000 images. It implements the ISO/IEC 15444-1 standard fully in part 1, and partly in parts 2-3. Kakadu is a trademark of NewSouth Innovations Ltd. License: proprietary.
3. JasPer [293] is a project to create a reference implementation of the codec specified in the JPEG-2000 Part-1 standard (i.e., ISO/IEC 15444-1) - started in 1997 at Image Power Inc. and at the University of British Columbia.[2] License: JasPer License Version 2.0.
4. JJ2000 [294] is the Java reference implementation (part 5) of JPEG 2000. This project was the result of a partnership between EPFL, Canon Research France and Ericsson. It ended in September 2001 with a complete implementation of the normative parts of the JPEG 2000 core coding system (part 1).
5. x264 [295] is a free software library developed by VideoLAN for encoding video streams into the H.264/MPEG-4 AVC format. License: GNU General Public License (version 2.0).
6. x265 [296] is an open source free software and library for encoding video using the High Efficiency Video Coding (HEVC/H.265) standard. License: GNU General Public License (version 2.0) or commercial license.
7. Xvid [297] is a video codec library following the MPEG-4 video coding standard, specifically MPEG-4 Part 2 Advanced Simple Profile (ASP). License: GNU General Public License.
8. Custom solution: custom designed codec compatible with SuriCog's application specifics.

4.5.2 Comparison of Image Compression Systems: JPEG, JPEG 2000 and FLIF

Several image compression systems were compared: JPEG, JPEG 2000, and FLIF. As for video codecs, there was comparison of the following products: XVID (MPEG-4), X264 (H.264/MPEG-4 AVC), X265 (HEVC/H.265). Several software-based encoder products were deployed to RZ-microcontroller (used to drive Weetsy™ board): OPENJPEG (JPEG 2000 standard) and XVID (MPEG-4 standard). Based on results of several experimentations with image compression systems it was concluded that JPEG 2000 is the best image coding system among others compared, because it provides higher PSNR at lower bitrates, natively support selection of ROI (with higher bpp allocated for ROI in contrast with non-ROI)

JPEG 2000 and its predecessor JPEG are commonly used image compression systems. To evaluate JPEG 2000 codec's ROI selection feature (compression with higher bpp allocated for ROI), set of experiments was launched. For this purpose Kakadu JPEG 2000 codec was used. Since codec allows to control number of DWT decompositions for ROI, optimal number (leads to higher PSNR) was selected. Then ROI was extracted from decompressed image and compared with ROI obtained from original image. Results are presented in Table 10. PSNR values, presented in the table, are the average PSNR values for the reported ROI sizes and at a given bitrate.

Table 10. JPEG 2000 PSNR/bpp for different ROI sizes.

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	0.80
120x120	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	57.55
140x140	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	48.03
160x160	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	44.70
180x180	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	42.02
200x200	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	40.17
220x220	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	38.51

According to visual comparison of quality of static ROI images (Figure 85), compressed with deferent bitrates, values less than 0.9 produce results with sufficiently degraded quality (11 dB).

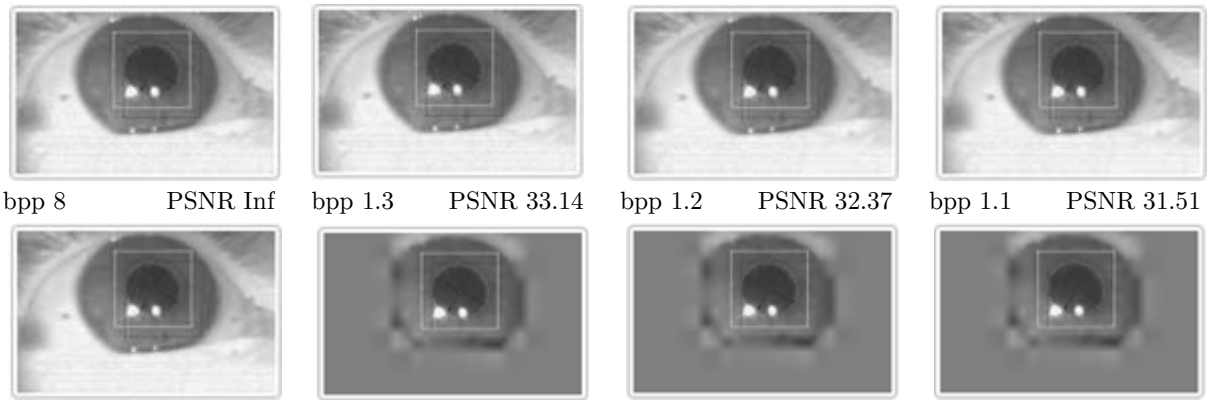
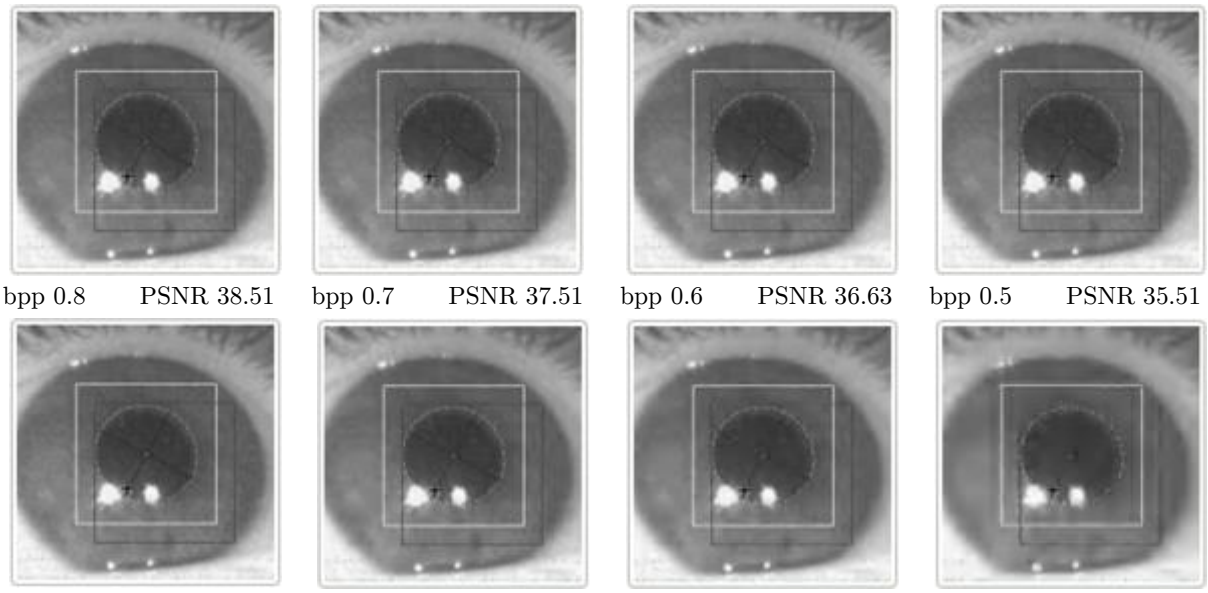


Figure 85. Visual comparison of quality of static ROI images with selected ROI.

Based on this comparison, it also can be seen that quality of ROI borders is degraded as lower bpp was used. However, quality of center of ROI is stayed acceptable. Therefore, to consider low bitrates (0.1-0.3), size of ROI could be increased from120x120 to 180x180 (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. To keep ability of using eye tracking system by several users 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 120x120 (leads to 36 dB, Table 10), then in case of ROI of size 180x180, bpp has to be increased to 0.40 (leads to 37 dB, Table 10). Since PSNR can't be the best metric all the time, acceptable practice is to perform also visual comparison

of images (Figure 86). For compression of ROI images with size of ROI 220x220 it is possible to get satisfied results even with low bitrates, such as 0.2.

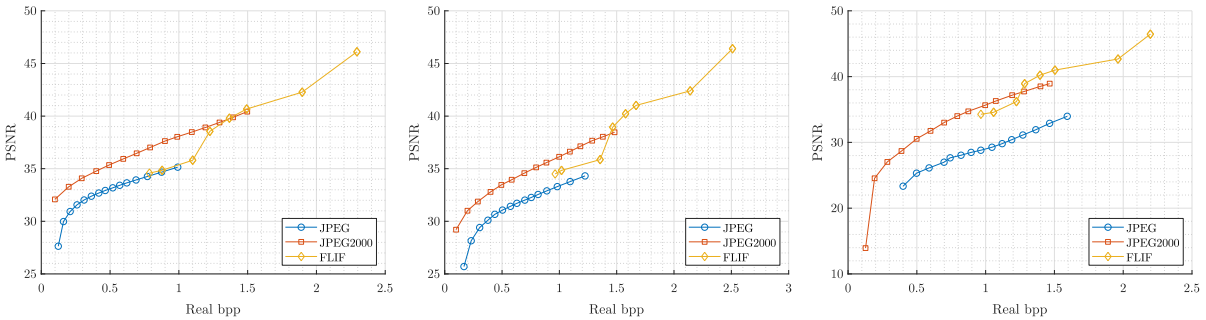


bpp 0.8 PSNR 38.51 bpp 0.7 PSNR 37.51 bpp 0.6 PSNR 36.63 bpp 0.5 PSNR 35.51

bpp 0.4 PSNR 34.81 bpp 0.3 PSNR 33.67 bpp 0.2 PSNR 32.07 bpp 0.1 PSNR 29.51

Figure 86. Visual comparison of quality of ROI images (220x220)

According to comparison of JPEG 2000 codec vs. JPEG codec and FLIF codec [133] (Figure 87, a-c) 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).



a. FULL resolution image (640x640) b. Static ROI image (400x256) c. Dynamic ROI image (120x120)

Figure 87. Image compression systems comparison.

Based on this comparison, it can be concluded that JPEG 2000 is the best image coding system among others compared, because it provides higher PSNR at lower bitrates, natively support selection of ROI (with higher bpp allocated for ROI in contrast with non-ROI). Therefore, JPEG 2000 image coding system can be considered for usage and possible future improvements.

4.5.3 JPEG 2000: Comparison of Different DWTs

The wavelet transform is similar to the Fourier transform (or much more to the windowed Fourier transform [22]) with a completely different merit function. The main difference is that the Fourier transform decomposes the signal into sines and cosines (i.e., the functions localized in Fourier space). In contrary the wavelet transform uses functions that are localized in both the real and Fourier space.

Generally, the wavelet transform can be expressed by the following equation [298]:

$$F(a,b) = \int_{-\infty}^{\infty} f(x)\psi^*_{(a,b)}(x)dx\,, \tag{4.1}$$

where the * is the complex conjugate symbol and function ψ is some function. This function can be chosen arbitrarily provided that it obeys certain rules. The wavelet transform is an infinite set of various transforms, depending on the merit function used for its computation. (Due to this reason the term 'wavelet transform' is used in very different applications.) There are many ways how to classify the types of the wavelet transforms. Since it is possible to use orthogonal wavelets for discrete wavelet transform development and non-orthogonal wavelets for continuous wavelet transform development, one classification is based on the wavelet orthogonality.

The discrete wavelet transform (DWT) is an implementation of the wavelet transform using a discrete set of the wavelet scales and translations obeying some defined rules. In other words, this transform decomposes the signal into mutually orthogonal set of wavelets, which is the main difference from the continuous wavelet transform (CWT), or its implementation for the discrete time series sometimes called discrete-time continuous wavelet transform (DT-CWT).

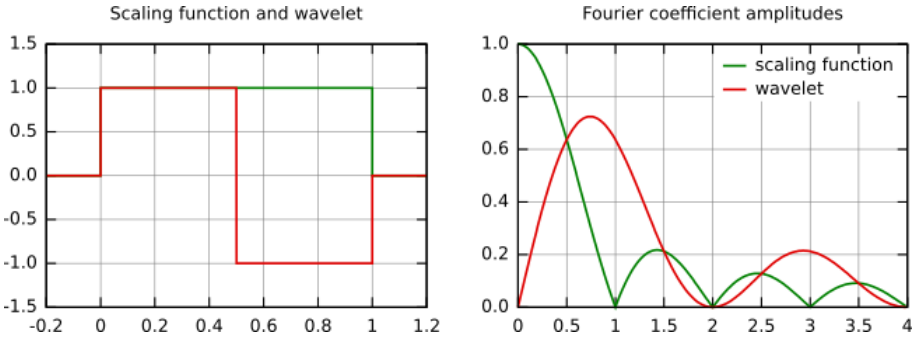
The wavelet can be constructed from a scaling function which describes its scaling properties. The restriction that the scaling functions must be orthogonal to its discrete translations implies some mathematical conditions on them which are mentioned everywhere, e.g. the dilation equation

$$\phi(x) = \sum_{k=-\infty}^{\infty} a_k \phi(S_x - k) \,, \tag{4.2}$$

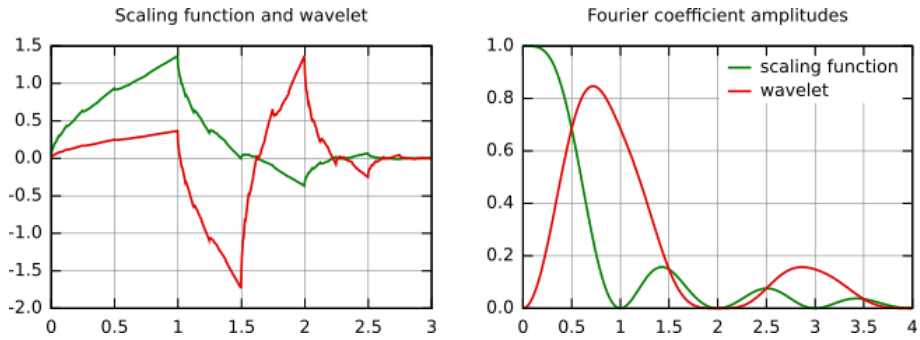
where S is a scaling factor (usually chosen as 2). Moreover, the area between the function must be normalized and scaling function must be orthogonal to its integer translations, i.e.

$$\int_{-\infty}^{\infty} \phi(x)\phi(x+l)dx = \delta_{0,l} \,, \tag{4.3}$$

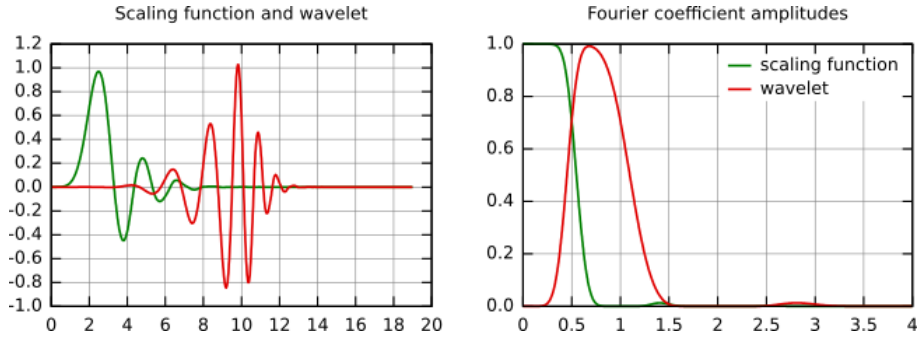
After introducing some more conditions (as the restrictions above does not produce a unique solution) we can obtain results of all these equations, i.e. the finite set of coefficients a_k that define the scaling function and also the wavelet. The wavelet is obtained from the scaling function as N where N is an even integer. The set of wavelets then forms an orthonormal basis which we use to decompose the signal. Note that usually only few of the coefficients a_k are nonzero, which simplifies the calculations. The most known family of orthonormal wavelets is the family of Daubechies. These wavelets are usually denominated by the number of nonzero coefficients a_k (for example, Daubechies 4, Daubechies 6, etc). Roughly said, with the increasing number of wavelet coefficients the functions become smoother. Below is the comparison (Figure 88) of several wavelets: Haar (simplest wavelet, which uses a box function as the scaling function), Daubechies 4 and Daubechies 20.



Haar scaling function and wavelet (left) and their frequency content (right).



Daubechies 4 scaling function and wavelet (left) and their frequency content (right).



Daubechies 20 scaling function and wavelet (left) and their frequency content (right).

Figure 88. Scaling functions, wavelets and Fourier coefficient amplitudes

There are several types of implementation of the DWT algorithm. The oldest and most known one is the Mallat's pyramidal algorithm [299]. In this algorithm two filters (smoothing and non-smoothing one) are constructed from the wavelet coefficients and those filters are recurrently used to obtain data for all the scales. If the total number of data $D = 2^N$ is used and the signal length is L, first $D/2$ data at scale $L/2^{N-1}$ are computed, then $(D/2)/2$ data at scale $L/2^{N-2}$, ... until to finally obtaining 2 data at scale $L/2$. The result of this algorithm is an array of the same length as the input one, where the data are usually sorted from the largest scales to the smallest ones. To simplify implementation several techniques were introduced over the time: lifting scheme and wavelet transforms that map integers-to-integers as well as its version using only integer arithmetic. These techniques were referenced in the Theoretical Part of this thesis.

To get familiar with typical performance of different DWT we launched a set of experiments. We used wavelet implementations obtained from WAILI (wavelet transform library) [300], because it uses integer wavelet transforms based on the lifting scheme [301] and provides various wavelet transforms of the Cohen-Daubechies-Feauveau family of biorthogonal wavelets. Results show (Table 11) that standard CDF97 (Cohen-Daubechies-Feauveau CDF 9/7 wavelet [302], also called 'JPEG97') and CDF53 (also called the LeGall 5/3 wavelet [303]) filters are the best in the most cases (except very low bitrates such as 0.15). It should be noted that dual tree complex wavelet transform [35] was not used in this experiment.

In Table 11 'CDF' means 'Cohen-Daubechies-Feauveau' biorthogonal family wavelet and '1_1' are the numbers of vanishing moments for the primal respectively dual wavelet function. It should be noted that '1_1' is the Haar basis, and '1_3' is the wavelet basis used by Ricoh's reversible embedded wavelets (CREW) [304,305]. 'CRF_13_7' and 'SWE_13_7' are other integer wavelet transforms (using the lifting scheme) supported by JPEG2000 [306]. It should be also noted that there are two concurring numbering schemes for wavelets of the CDF family:

1. The number of smoothness factors of the lowpass filters, or equivalently the number of vanishing moments of the highpass filters, e.g. '2, 2';
2. The sizes of the lowpass filters, or equivalently the sizes of the highpass filters, e.g. '5, 3'.

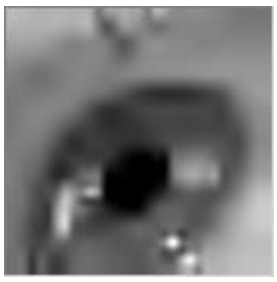
The first numbering was used in Daubechies book «Ten lectures on wavelets» [307]. Neither of this numbering is unique. The number of vanishing moments does not tell about the chosen factorization. A filter bank with filter sizes 7 and 9 can have 6 and 2 vanishing moments when using the trivial factorization, or 4 and 4 vanishing moments as it is the case for the JPEG 2000 wavelet. Therefore, the same wavelet may therefore be referred to as 'CDF 9/7' (based on the filter sizes) or 'biorthogonal 4, 4' (based on the vanishing moments).

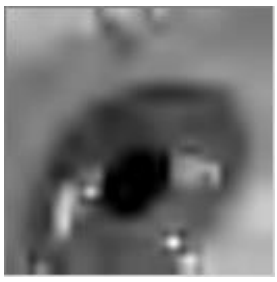
Table 11. Comparison of different DWTs.

Transf./ratio (bpp)	10 (0.80)	20 (0.42)	30 (0.29)	40 (0.22)	50 (0.18)	60 (0.15)
JPEG53	45.15	38.81	35.27	32.98	31.21	29.89
JPEG97	46.61	39.54	35.77	33.23	31.37	29.84
CDF_1_1	40.7	35.14	32.11	30.26	28.68	27.58
CDF_1_3	40.07	34.68	31.73	29.8	28.33	27.22
CDF_1_5	39.82	34.42	31.48	29.55	28.09	27.02
CDF_2_2	45.41	38.91	35.38	33.03	31.3	29.91
CDF_2_4	45.12	38.76	35.27	32.99	31.28	29.96
CDF_2_6	44.91	38.6	35.16	32.89	31.22	29.93
CDF_4_2	28.17	25.91	24.15	22.63	21.56	20.3
CDF_4_4	37.57	32.78	29.61	27.6	25.86	24.63
CDF_4_6	39.01	33.81	30.61	28.52	26.82	25.58
CRF_13_7	44.84	38.59	35.19	32.99	31.32	30.02
SWE_13_7	45.03	38.68	35.29	33.01	31.34	29.97

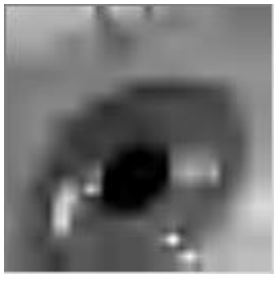
It is possible also to perform visual comparison of decompressed dynamic ROI images, which were compressed with use of different wavelet transforms at very high compression ratios (low target bitrates). In the example below (Figure 89), these dynamic ROI images were compressed with JPEG 2000 encoder with ratio 70. The average real bpp of compressed images is 0.14 bpp.

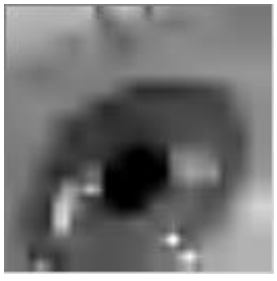

Uncompressed, PSNR Inf dB


SWE_13_7, PSNR 31.15 dB

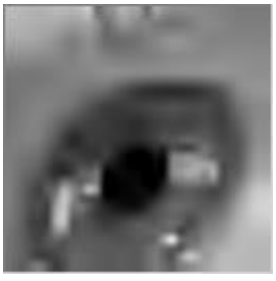

CRF_13_7, PSNR 31.10 dB


CDF_2_4, PSNR 27.60 dB


CDF_2_6, PSNR 27.60 dB


CDF_2_2, PSNR 27.30 dB


JPEG53, PSNR 27.10 dB


JPEG97, PSNR 27.05 dB

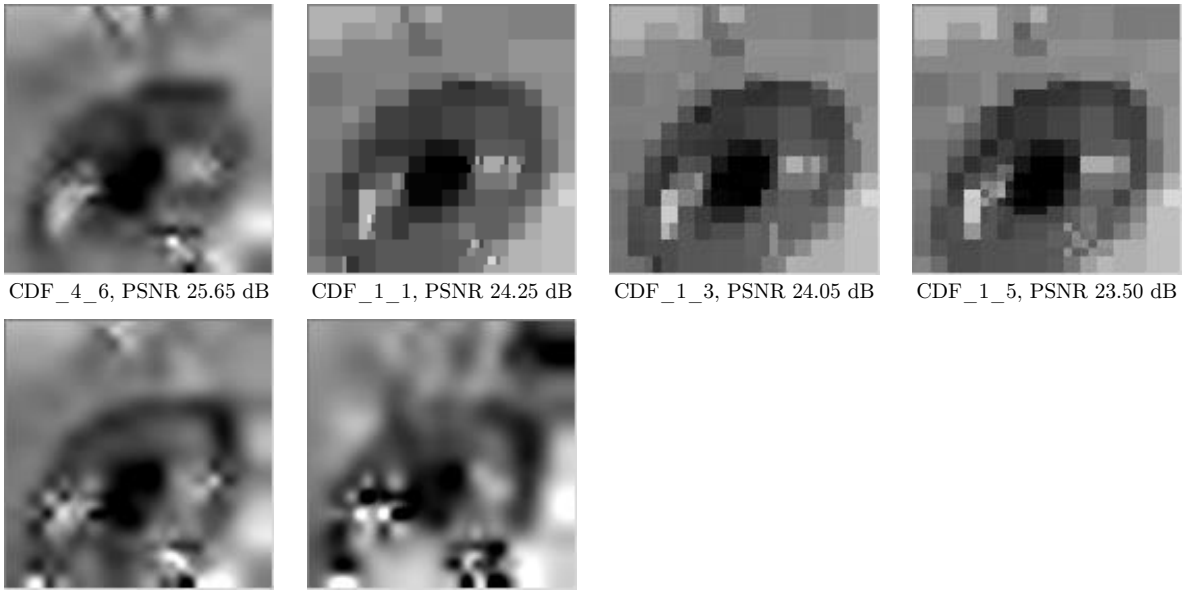
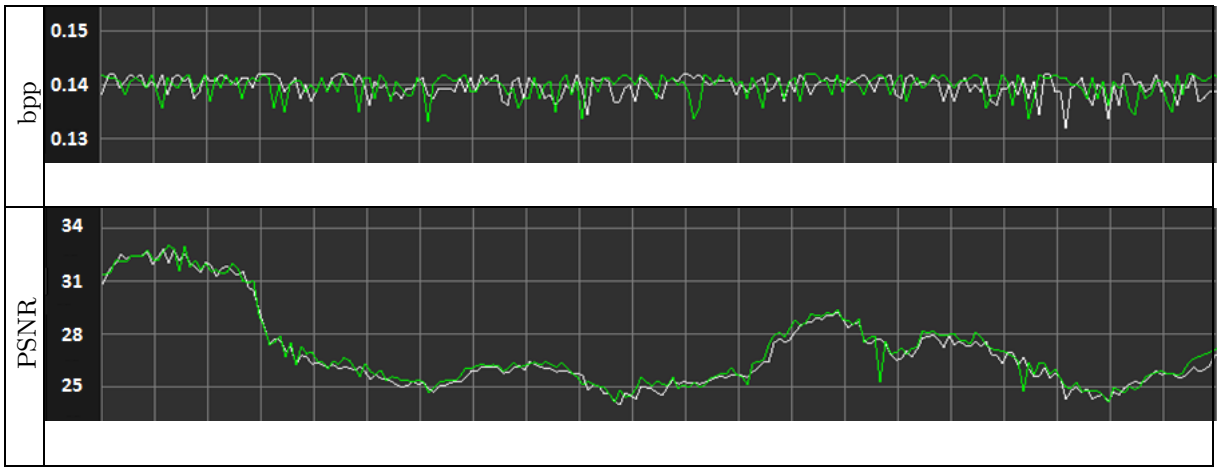


Figure 89. Visual comparison of ROIs compressed with JPEG 2000 with different transform used.

A comparison of JPEG97 (irreversible filter) vs JPEG53 (reversible filter) shows that they have near-the-similar behavior (Figure 90) and JPEG97 performs slightly better.



Legend: JPEG97 irreversible filter – white, JPEG53 reversible filter – green.

Figure 90. Comparison of change of PSNR and bpp in time: JPEG97 vs. JPEG53.

Comparison of wavelet-similar transforms (Figure 91) shows that shearlet transform has the best performance among other transforms.

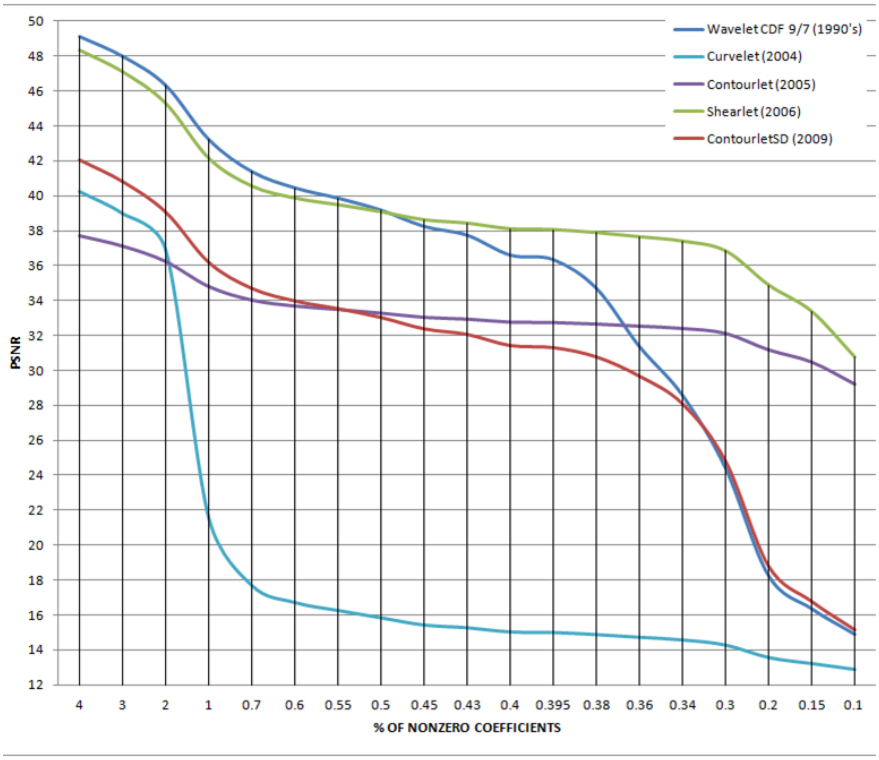


Figure 91. Comparison of wavelet-similar transforms.

4.5.4 Applying of the H.264/AVC Spatial Intra-only Compression

The H.264/AVC uses both spatial and temporal predictions to increase its coding gain. The intra-only compression uses spatial prediction and the prediction only occurs within a slice. For example, in case of 8x8-block intra prediction (Figure 92) each luminance sample in an 8×8 block is predicted from the neighboring constructed reference samples, where 8 different prediction directions and a DC prediction can be selected by the encoder. In the case of chrominance block, four different prediction methods are available. The key to improving coding performance when using spatial intra prediction is to select the proper prediction mode for each block.

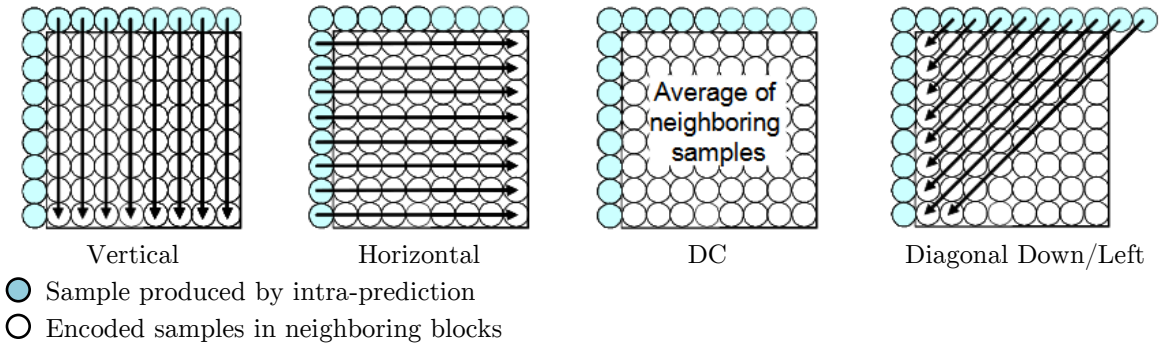
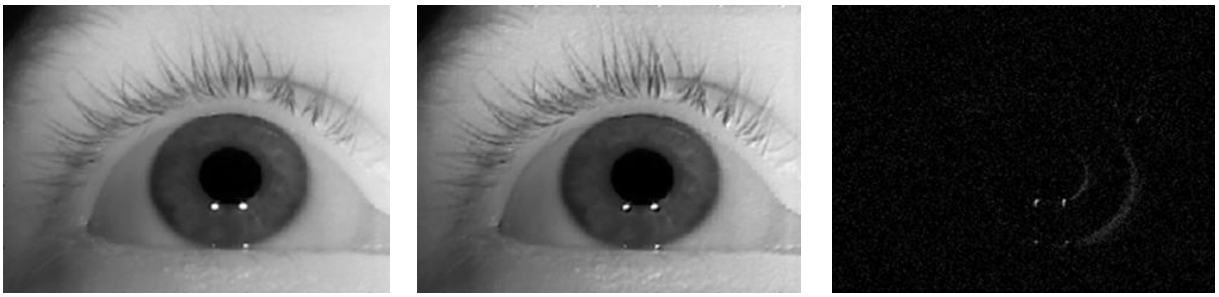


Figure 92. Examples of H.264/AVC spatial intra prediction modes.

The original input image and the intra-predicted image are very similar (Figure 93) due to the high accuracy of the prediction. Then ny subtracting the intra frame predicted image from original input image, the resulted 'difference' image (or residual image) is generated (Figure 93). An integer transform (with similar properties as a 4x4 DCT) is applied to this residual image with the resulting coefficients being adaptively quantized and entropy coded. The prediction mapping information is stored in a bit stream along with compressed residual image. Because the amount of data required for the residual image can be reduced by highly accurate intra-prediction, higher efficiency compression can be achieved even when using intra-only compression.



Original Intra-predicted Residual

Figure 93. Examples of H.264/AVC images: original, intra-predicted and residual.

However, taking into the account advantages of the H.264/AVC spatial intra-only compression, this approach is not planned to be used in the EyeDee™ eye tracking solution mainly because:

1. Hardware implementation of feature based ROI image compression (NN based approach) followed by low-complexity DCT approximations (applied in parallel) coupled with low-complexity entropy coding is expected to be executed faster.
2. Need to use proprietary IP core (either in form of compiled FPGA bitstream or in form of dedicated hardware integrated circuit).

4.6 Neural Network Based Approaches

4.6.1 Neural Network Based Eye Tracking (Based on Function Regression)

After the integration of the Torch7 software into the EyeDee™ eye tracking solution several experiments were completed (Figure 94).

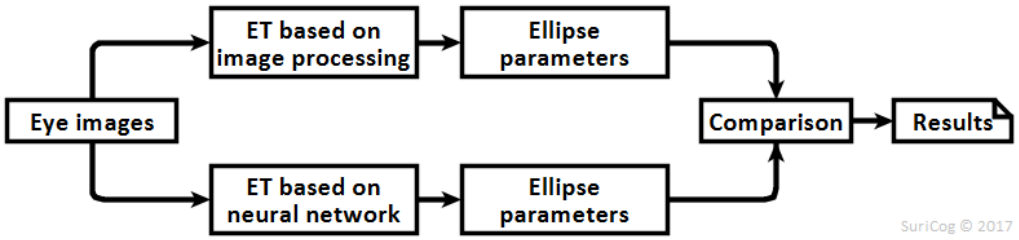


Figure 94. Testing eye tracking approaches: image processing based vs. neural network based.

In a first time we compute two ellipses – one calculated using image processing and one obtained using the trained neural network, and we measure their similarity, based on three coefficients: distance (ε_{d}), shape (ε_{s}) and orientation (ε_{0}) defined below.

1. Distance coefficient (ε_{d}):

$$\varepsilon_{\mathrm{d}} = e^{-\frac{|c_1 - c_2|}{|c_1 \| c_2|}}, \tag{4.4}$$

where c denotes the ellipse center.

2. Shape coefficient (ε_{s}):

For each ellipse equation:

$$ax^2 + 2bxy + cy^2 + 2dx + 2ey + 1 = 0 \tag{4.5}$$

we can form the matrix M:

$$M = \begin{bmatrix} a & b & d \\ b & c & e \\ d & e & 1 \end{bmatrix} \tag{4.6}$$

and calculate the 3 Eigen values: d_1, d_2, d_3 for the first ellipse and $\delta_1, \delta_2, \delta_3$ for the second one, which are further used for vectors v and w computation:

$$\mathrm{v} = \left(\frac{sign(d_3)}{\sqrt{|d_3|}}, \frac{sign(d_2)}{\sqrt{|d_2|}}, \frac{sign(d_1)}{\sqrt{|d_1|}}\right), \quad |d_3| \geq |d_2| \geq |d_1| > 0\,, \tag{4.7}$$

$$\mathrm{w} = \left(\frac{sign(\delta_3)}{\sqrt{|\delta_3|}}, \frac{sign(\delta_2)}{\sqrt{|\delta_2|}}, \frac{sign(\delta_1)}{\sqrt{|\delta_1|}}\right), \quad |\delta_3| \geq |\delta_2| \geq |\delta_1| > 0\,. \tag{4.8}$$

Finally, the shape coefficient is given by:

$$\varepsilon_s = e^{-\frac{|w-v|}{|w||v|}}. \tag{4.9}$$

3. Orientation coefficient (ε_0):

$$R_{1,2} = \begin{bmatrix} \cos\phi_{1,2} & -\sin\phi_{1,2} \\ \sin\phi_{1,2} & \cos\phi_{1,2} \end{bmatrix}, R = R_1^{-1} \times R_2\,, \tag{4.10}$$

$$\Theta_0 = \arccos(R_{00}), \Theta_1 = \arccos(R_{11})\,, \tag{4.11}$$

$$\varepsilon_0 = e^{-\sqrt{\sin^2\Theta_0 + \sin^2\Theta_1}}, \tag{4.12}$$

where $\phi_{1,2}$ denotes the ellipse rotation angles, $R_{1,2}$ the rotation matrices, R_{00} and R_{11} the first two components of the diagonal matrix R used to obtain the orientation angles Θ_0 and Θ_1.

Results of finding the optimal number of training iterations and optimal number of hidden layers (Figure 95, Figure 96) show these values are 40.000 and 2 respectively.

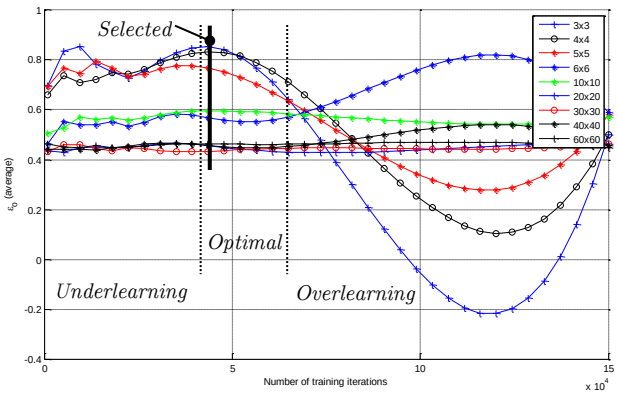


Figure 95. Finding the optimal number of training iterations.

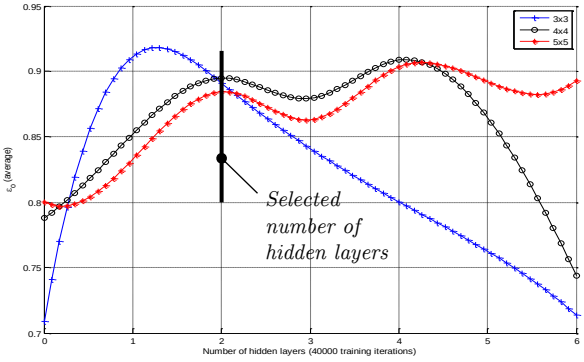


Figure 96. Finding the optimal number of hidden layers.

According to experimentation results, the use of a trained neural network has a good potential. Increasing the number of training iterations results in increasing the average accuracy of the reconstructed ellipse (Table 12). In our experimental framework we use the following hyper-parameters: the learning rate is set to 1e-5, learning rate decay to 1e-4 and both weight decay and momentum are set to 0.

Table 12. Average ellipse similarity for different ROI sizes.

Decimated ROI size	Training iterations	ε_{d}	ε_{o}	ε_{s}
3x3	1000	0.985	0.698	0.912
	4000	0.983	0.773	0.932
	10000	0.984	0.834	0.936
4x4	1000	0.991	0.660	0.974
	4000	0.997	0.736	0.904
	10000	0.997	0.700	0.925
5x5	1000	0.991	0.695	0.946
	4000	0.995	0.776	0.921
	10000	0.993	0.755	0.902

Increasing the number of decimation points results into lower average error (Figure 97) i.e., computed as cumulated difference between the original (expected) result and the approximated one (generated by the neural network during the training), measured on each epoch. For example, changing the decimated ROI size from 3x3 to 4x4 immediately decreased the reconstruction error, providing better quality of the output results c_x, c_y, $r_{\max}$, $r_{\min}$ and φ, and thus increasing the similarity between the original and reconstructed ellipses (Table 1).

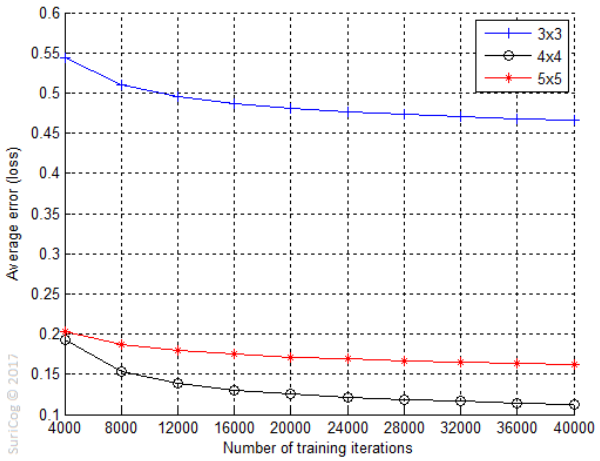


Figure 97. ANN training (average error decrease), 40000 training iterations.

Increasing the number of training iterations (epochs) results into lower average orientation coefficient (Figure 98), computed as a result of comparison of original and reconstructed ellipses for all samples used in the training dataset. Distance coefficient (ε_{d}) and shape coefficient (ε_{s}) are fluctuating in more narrow range (see Table 1). It can be shown then that the best average error of orientation coefficient (ε_{o}) over all sizes of decimated ROI is reached, when the number of training iterations is equal to 40000. To find an optimal number of hidden layers (Figure 99) we kept this number of iterations. It can be shown that best average error of the orientation coefficient (ε_{o}) is reached, when the number of hidden layers is equal to 2.

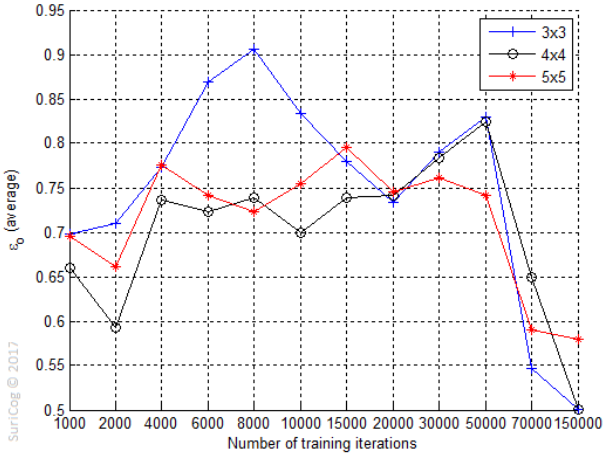


Figure 98. Trained ANN: average orientation coefficient (ε_0) vs. number of training iterations.

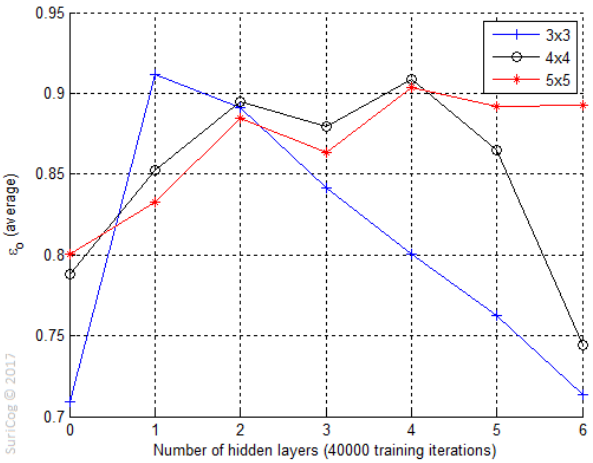
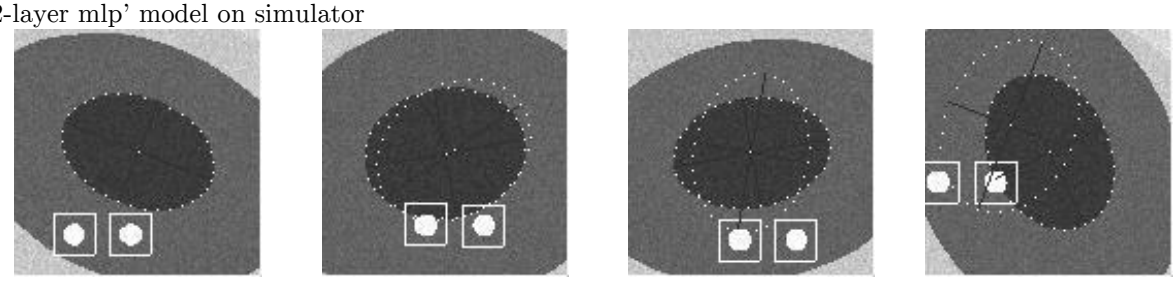


Figure 99. Trained ANN: average orientation coefficient (ε_0) vs. number of hidden layers.

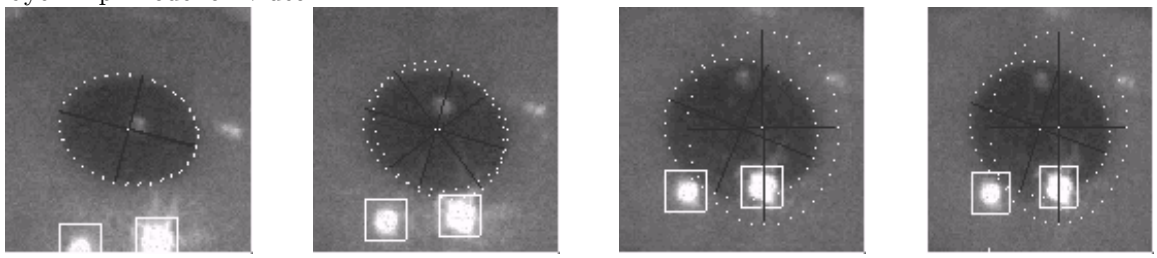
The visual comparison of ellipse reconstruction (Figure 100) shows that there are some accuracy issues, i.e., some ellipses generated by the trained neural network do not perfectly fit into the original ones generated by the simulator (although some are very close to, Figure 100, a-b). Even more, for new eye positions, not present in the training data set, unexpected results are generated by the trained neural network (Figure 100, c-d). To improve the accuracy, it is necessary to find the best neural network configuration and define the training strategy to avoid common issues such as overtraining, finding a local minimum (instead of global one) etc.

'2-layer mlp' model on simulator



a b c d

'2-layer mlp' model on video

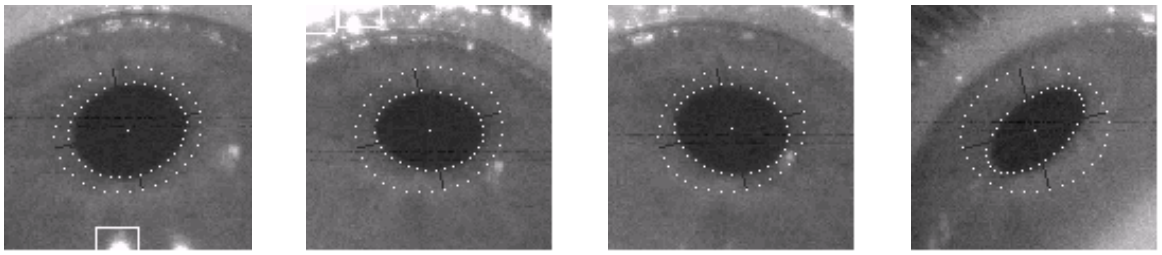


a b c d

Figure 100. Visual comparison of ellipse reconstruction (increasing degradation order).

Using one trained model on different user (generalization property validation) leads to almost the same canter and shape, but different radius (Figure 101).

'2-layer mlp' model on video



a b c d

Figure 101. Visual comparison of ellipse reconstruction (generalization property validation).

4.6.2 Feature Based Eye Image Compression (Based on Data Classification)

To evaluate the learning quality, we have tested the training of the neural network with images obtained from a simulator and from previously saved video sequences of eye movements. We finally selected 400 training iterations (epochs), as this amount is reasonable to get satisfying results (i.e., confusion matrixes values). With the same setup we obtained the results for two tested models: 'convnet' (convolutions+2-layer mlp) and '2-layer mlp' (pure 2-layer mlp).

According to the classification quality results (Table 13), the use of a trained neural network for ROI image blocks classification has a good potential in general. In particular, 'convnet' model shows better results over '2-layer mlp' model. This can be explained by the features extractions followed by the '2-layer mlp' model (instead of pure '2-layer mlp').

Table 13. Confusion matrixes: '2-layer mlp' model vs. 'convnet' model.

			Prediction							
			'2-layer mlp' model				'convnet' (convolutions+2-layer mlp) model			
			Simulator		Video		Simulator		Video	
	Block size		Cls_1	Cls_2	Cls_1	Cls_2	Cls_1	Cls_2	Cls_1	Cls_2
Truth	20	Cls_1	67.70%	0.02%	68.13%	0.14%	67.78%	0.00%	68.27%	0.00%
		Cls_2	0.15%	32.05%	0.08%	31.64%	00.00%	32.21%	0.00%	31.72%
	10	Cls_1	83.72%	0.53%	83.62%	0.88%	83.87%	0.36%	83.88%	0.62%
		Cls_2	1.13%	14.60%	1.48%	14.01%	0.98%	14.77%	0.92%	14.56%

According to the application results (Table 14, Table 15), 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 efficiency. This is because the number of N_{00} and N_{11} is higher meaning that the model is trained better. For example, using 'convnet' model with ROI blocks of size 10x10 at 100 training iterations is enough to reach 100% of both N_{00} and N_{11}.

Table 14. Average results of ROI image blocks classification quality (use of '2-layer mlp' model).

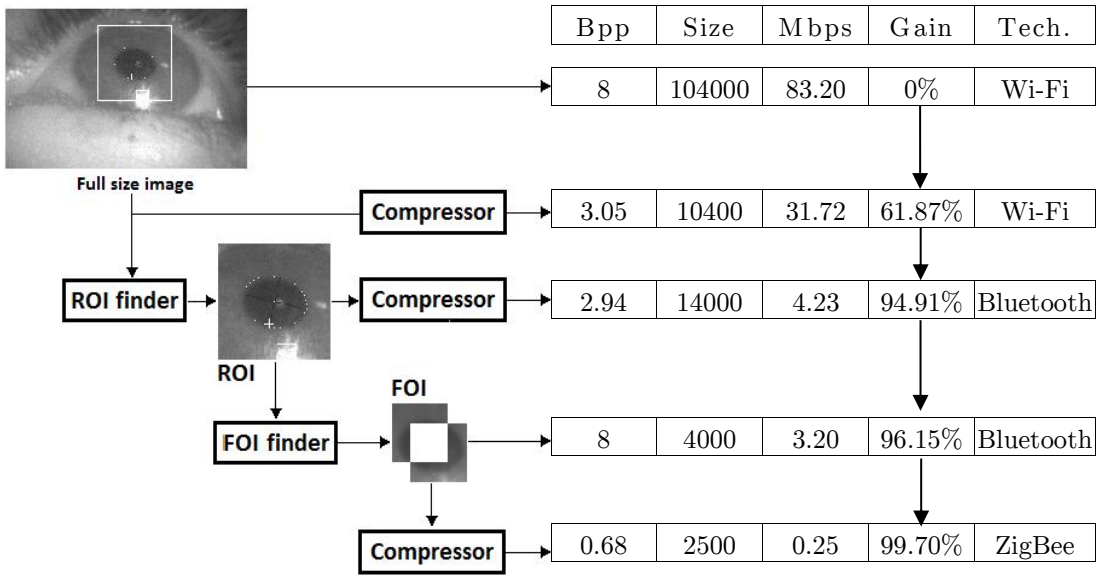
Set	Mode	Block Size	Epochs	ε %	p %	Bits per pixel (bpp)		
						Original ROI	ROI with removed blocks	ROI with removed blocks (reordered)
Training set	Trained on video, tested on video	20	50	97.85	1.24	2.15	1.33 (38.14%)	0.88 (59.07%)
			100	98.89	0.46	2.15	1.33 (38.14%)	0.89 (58.60%)
			200	99.19	0.48	2.15	1.33 (38.14%)	0.89 (58.60%)
			400	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	70.14	19.65	2.15	1.04 (51.63%)	0.50 (76.74%)
			200	70.87	17.54	2.15	1.04 (51.63%)	0.49 (77.21%)
			400	76.69	14.06	2.15	1.07 (50.23%)	0.52 (75.81%)

Validation set	Trained on simulator, tested on video	20	50	79.08	20.33	2.15	1.35 (37.20%)	0.90 (58.17%)
			100	81.09	17.747	2.15	1.35 (36.91%)	0.89 (58.49%)
			200	82.05	16.87	2.15	1.35 (36.95%)	0.89 (58.49%)
			400	85.50	20.79	2.15	1.38 (35.55%)	0.94 (56.03%)
		10	50	61.45	28.12	2.15	1.13 (47.41%)	0.51 (76.05%)
			100	63.55	29.64	2.15	1.16 (46.00%)	0.53 (75.12%)
			200	68.42	34.16	2.15	1.23 (42.61%)	0.59 (72.67%)
			400	72.08	37.94	2.15	1.29 (39.80%)	0.63 (70.86%)

Table 15. Average results of ROI image blocks classification quality (use of 'convnet' model).

Set	Mode	Block Size	Epochs	ε %	p %	Bits per pixel (bpp)		
						Original ROI	ROI with removed blocks	ROI with removed blocks (reordered)
Training set	Trained on video, tested on video	20	50	92.69	2.93	2.15	1.32 (38.66%)	0.86 (60.03%)
			100	93.93	0.27	2.15	1.31 (39.25%)	0.85 (60.48%)
			200	95.44	0.00	2.15	1.31 (38.90%)	0.86 (59.99%)
			400	100.00	0.00	2.15	1.33 (38.10%)	0.89 (58.68%)
		10	50	72.94	28.20	2.15	1.11 (48.50%)	0.55 (74.51%)
			100	70.04	26.66	2.15	1.10 (48.99%)	0.53 (75.45%)
			200	70.48	25.03	2.15	1.09 (49.14%)	0.53 (75.55%)
			400	71.31	23.10	2.15	1.31 (38.90%)	0.86 (59.99%)
Validation set	Trained on simulator, tested on video	20	50	89.90	11.68	2.15	1.37 (36.26%)	0.90 (58.05%)
			100	92.83	16.09	2.15	1.42 (33.67%)	0.96 (55.07%)
			200	93.07	16.51	2.15	1.43 (33.49%)	0.97 (54.81%)
			400	93.15	16.77	2.15	1.43 (33.37%)	0.97 (54.65%)
		10	50	62.06	38.45	2.15	1.18 (44.92%)	0.55 (74.51%)
			100	62.71	44.13	2.15	1.25 (41.94%)	0.59 (72.64%)
			200	62.58	43.91	2.15	1.26 (41.25%)	0.59 (72.57%)
			400	67.72	48.18	2.15	1.32 (38.43%)	0.65 (69.65%)

For the compression of ROI images, we used a JPEG 2000 encoder. We configured the encoder to keep a relatively high PSNR (>45 dBs) as the decompressed blocks will be further used in an image processing-based eye tracking algorithm. However, this quality can be lower as 32 dB, as it was proved earlier. Results show that with the proposed approach it is possible to reach 99.47% of gain in terms of data size reduction (Figure 102).

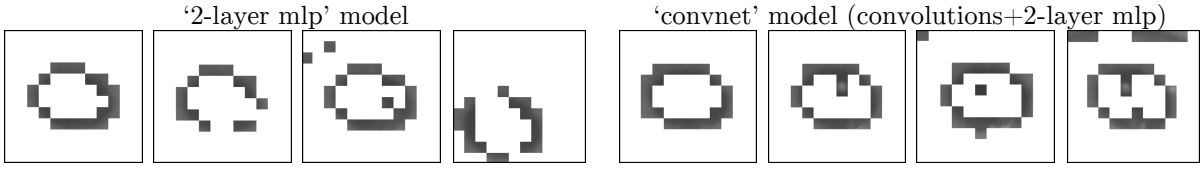


Bpp	Size	Mbps	Gain	Tech.
8	104000	83.20	0%	Wi-Fi
3.05	10400	31.72	61.87%	Wi-Fi
2.94	14000	4.23	94.91%	Bluetooth
8	4000	3.20	96.15%	Bluetooth
0.68	2500	0.25	99.70%	ZigBee

Legend: Full image resolution: 400x260, ROI image resolution: 120x120, compressor: JPEG 2000, bpp – bits per pixel, Mbps – megabits per second at 100 FPS, FOI – Features of Interest.

Figure 102. Eye image compression configurations.

The visual comparison of ROI image block removal quality (Figure 103) on the validation set shows that there are more accuracy issues, in comparison with the training set, i.e., the number of N_{01} is increased (Fig. 13, b-c). If the neural network is not enough trained the quality is significantly degraded (Fig. 13, d) in case of both 'mlp' and 'convnet' models.



Legend: Use of validation dataset, block size is 10.

Figure 103. Visual comparison of blocks removal quality.

Increasing the number of ROI image blocks will result in a more accurate preservation of pupil ellipse edges (less data to transmit). However, for the validation set there are issues of preserved blocks situated in the corners of the ROI images. These issues can be solved by applying some additional logic. For example, if the block is located M-blocks far from the blocks containing pupil edges, this block can be considered as incorrect and should not be used for further image compression.

From the computational point of view (computational complexity), the ROI image compression based on neural network requires just the resources needed to obtain the classification results (i.e., input: NxN ROI image blocks, output: value indicating whether a block contains pupil edges). The neural network has also constant time response, which can be taken into account for the overall performance estimation. However, the use of neural network has a well-known challenge of its training (hyper-parameters tuning, training time reduction, output results quality maximization), as the cost of each training session is time expensive.

From the implementation point of view (implementation complexity), the proposed neural-network eye tracking approach reduces the amount of data to compress by either a standard image compression system (JPEG 2000 or other) or an application-specific image compression system. It is also expected that the next version of the Weetsy™ board will have a memory space large enough to store

the trained neural network, which will make possible to apply/deploy the final eye image feature based compression configuration (Figure 104).

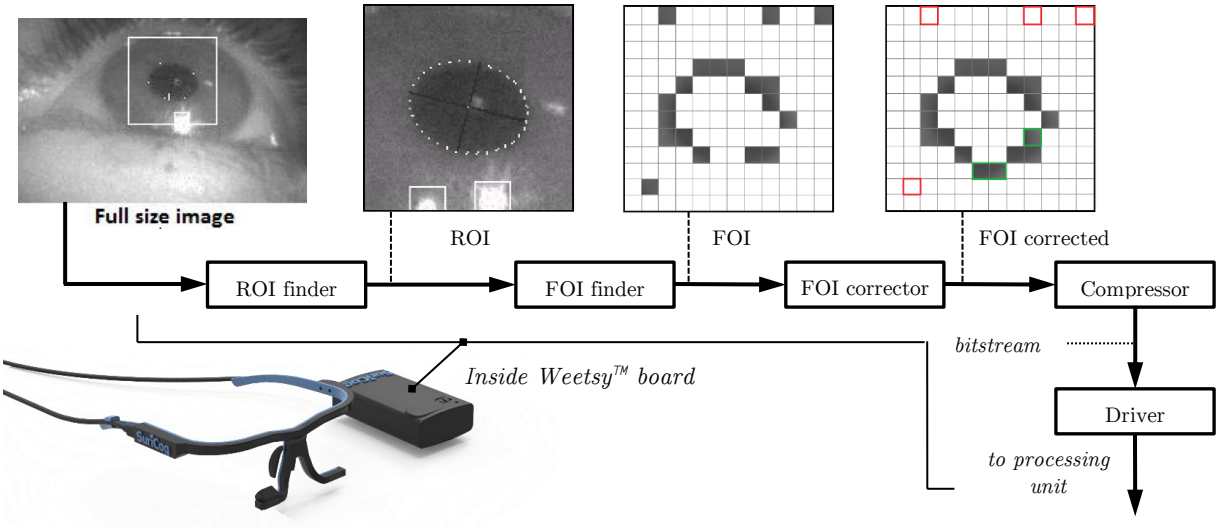


Figure 104. Final eye image feature based compression configuration.

From the conceptual point of view, employing a neural network before the ROI image compression can be interpreted as preprocessing, i.e., keeping only the regions in the ROI image that are important for the eye tracking algorithm, prior to compression (reducing thus data size and, therefore, speeding-up data transmission). To accelerate the compression of the FOI it is planned to use low-complexity DCT approximations (applied in parallel) followed by low-complexity entropy coding (for example, truncated golomb-rice codes [308], FSE encoding [119] or others approaches [309]), both implemented directly in the hardware (FPGA).

4.7 Conclusion

Experimental results show applicability of variety of proposed approaches, which can be coupled together. For example, FPGA-based hardware ROI-finder provides results closed to software based ROI finder, which makes possible to use it even in current version of the product. Due to additional extra non-ROI information needed, PC-based eye tracking processing unit software should be updated to be able take as an input only dynamic ROI image: either re-centered by the FPGA or extended to perform re-centering on the remote processing unit side.

Experimental results on finding of the eye image compression algorithm requirements show that minimal decompressed image quality has be about 32 dB, maximal time of eye image compression/decompression depends on size of the eye image compressed. In particular a set of System of Linear Equations (SLE) was found, which permits to model and analyze an entire eye image acquisition system without frequent performance tests. (These performance tests are done, when a particular profile is selected and properly implemented.) According to considerations on ability to operate in lossy transmission medium it is recommended to protect coded image data via CRC8, which has the fastest implementation possible. Another approach is to transmit coded bitstream without original header since this header does not contain any temporal information added for decompressor. In this case, decompressor has a local copy of the header, which is used do decompress received bitstream.

According to comparison of image compression systems such as JPEG, JPEG 2000 and FLIF it was shown that JPEG 2000 provides better results among others and can be recommended to use in form only in a form of hardware based IP core. Our deployment tests show that almost any software based image/video compression system has very slow execution frequency about 3-8 FPS maximum and therefor is not recommended for the usage in the product.

Artificial Neural Network (ANN) based approaches, such as ANN based eye tracking, feature based eye image compression or either ANN based ROI finder show prominent results and can be recommended for further experimentations and especially implementation in the target embedded software or directly in the hardware.

In case when due to several reasons processing of locally found and transmitted dynamic ROI is not possible, there is an approach of «foveated eye image compression», where static ROI image contains «less prioritized» and «more prioritized» regions, which are compressed accordingly leading to overall bpp reduction and, as a result, transmission performance increase. It should be noted that proposed eye image compression approaches and tools can be combined, leading to undiscovered yet compression systems.

Chapter 5

Conclusion and Future Work

In this thesis we proposed an multiple approaches to design and implementation of image processing and compression algorithms for a miniature embedded EyeDee™ eye tracking system containing Weetsy™ portable wire/wireless system (Weetsy™ frame and Weetsy™ board), π-Box™ remote smart sensor and PC-based processing unit running SuriDev eye tracking software.

During the thesis we studied the impact of almost each component of the eye image compression system: from image pre-processing, following by spatial-to-frequency transform, quantization and data coding to transmission of the coded image (bitstream) over unreliable transmission medium and decoding of received bitsteram following to applying eye tracking algorithms to decoded eye images. To benefit of compression system consists in improvement of performance by sending less data through physical transmission link (wire/wireless transmission) while disadvantage of compression is loss of original image quality. Hence in this thesis we studied an impact of decompressed eye image quality degradation over the quality of the eye tracking results (including precision and accuracy).

In particular, proposed eye-based rate-distortion optimization allows to achieve higher lower real bpp due to the fact several subbands contain eye details, which have very little impact on the eye tracking precision results and therefore can be more compressed or entirely removed. An analysis approach of image delivery chain (image acquisition system) permits to reduce of time-consuming performance testing of the physical hardware to analysis of the mathematical model represented as a set of System of Linear Equations (SLE), which were obtained in the thesis. Eye image compression based on low complexity multiplierless DCTs can be applied in addition to ANN based approach to achieve the lowest possible time of compression.

A several limits of the eye image compression system were found, such as: minimal quality of decompressed image could be used for the eye tracking without significant eye tracking precision loss, minimal delay of eye image compression/decompression allowing to the eye tracking system stay responsive, minimal size of compressed eye image allowing to use less bitrate consuming data transmission technologies (Bluetooth LE and ZigBee).

Proposed ANN based eye tracking approximation (targeted on a complete replacement of image processing based eye tracking algorithm coupled with geometric eye modeling) has a potential but has a generalization issue, related to pupil ellipse diameter. Such approach allows to reduce the size of the transmitted data from Weetsy™ board to PC-based processing unit to minimal possible 20 bytes, which contain five floating-point values defining pupil ellipse in a 2D space.

Feature based ROI image compression has a strong potential and targeted on an improvement of compression of the Region of Interest (ROI, region containing image of the human's pupil). It was shown in the thesis that it is possible to find and remove extra information from the dynamic ROI, because such extra information has relatively little impact on pupil's ellipse reconstruction, and hence little impact on the eye tracking results quality. On order to find such an «extra information» we applied ANN classifier.

Another proposed approach is ANN based ROI finding, which is aimed on an improvement of the ROI finding quality. Since an original ROI finding algorithm performs multiple threshold based block scanning across the integral image it provides multiple ROI 'candidates' as its output. In contrast, ANN based approach is targeted on the same task, but without a threshold leading to only one ROI 'candidate'.

To minimize real bpp of compressed static ROI a foveated based eye image compression which is inspired by the 'foveated image rendering' i.e., a rendering, where information, which takes most user's attention, is rendered with maximum quality while information, which takes less user's attention, is rendered with minimum quality. In this approach FPGA/ANN based ROI finder determines approximate position of the dynamic ROI, then ANN feature based approach splits dynamic ROI on blocks and determines which blocks contain useful for the eye tracking pupil edges. Then it performs image compression on the found eye image parts. The final compressed image has variable bits-per-pixel value (bpp), which allows to transmit this compressed image faster across wireless medium.

Future work consists in improvement and especially implementation of the ANN based approaches in the Weetsy™ board following by performance tests. Another aspect is an adaptation of the eye tracking algorithm to operation only on dynamic ROI found locally on the Weetsy™ board side by use of either FPGA based ROI finder or ANN based ROI finder.

Publications

Invited Journal Publications:

1. P. Morozkin, M. Swynghedauw, and M. Trocan, "Image Compression Approaches for Resource Constrained Devices," in Journal of Information and Telecommunication (TJIT), Taylor & Francis (in review)

Conference Publications:

2. P. Morozkin, M. Swynghedauw, and M. Trocan, "Design of an Embedded Image Acquisition System," in proceedings of 2015 IEEE International Conference on Electronics, Circuits and Systems (ICECS). (hal-01528546)
3. P. Morozkin, M. Swynghedauw, and M. Trocan, "An Image Compression for Embedded Eye-Tracking Applications," in proceedings of 2016 International Symposium on INnovations in Intelligent SysTems and Applications (INISTA). (hal-01528557)
4. P. Morozkin, M. Swynghedauw, and M. Trocan, "Image Quality Impact for Eye Tracking Systems Accuracy," in proceedings of 2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS). (hal-01528560)
5. P. Morozkin, M. Swynghedauw, and M. Trocan, "Neural Network Based Eye Tracking," in proceedings of 9th International Conference on Computational Collective Intelligence (ICCCI 2017). (hal-01528569)
6. P. Morozkin, M. Swynghedauw, and M. Trocan, "Feature-based Image Compression," submitted to 10th Asian Conference on Intelligent Information and Database Systems (ACIIDS 2018). (hal-01784048)

Appendix

Existing Head-Mounted Eye Tracking Solutions

There are several head mounted eye tracking solutions, which were existed on the market during the period of conducting this research. The several examples are presented in Figure 105.

Tobii Pro Glasses 2 (Dynavox)

SMI eye tracking glasses 2

ASL mobile eye-5 Glasses

Pupil Mobile

Dikablis Glasses 3 (Ergoneers)

SR Research EyeLink II (casque)

Figure 105. Head mounted eye tracking solutions.

Bibliography

1. Duchowski, A.T., 2007. Eye tracking methodology. Theory and practice, 328. Referenced in section: 1 (page 1).

2. Johannsen, G., 2009. Human-machine interaction. Control Systems, Robotics and Automation, 21, pp.132-62. Referenced in section: 1 (page 1).

3. Introduction to Ergonomics, Third EditionAug 14, 2008. Referenced in section: 1 (page 1).

4. Adams Jr, R.B. and Kleck, R.E., 2003. Perceived gaze direction and the processing of facial displays of emotion. Psychological science, 14(6), pp.644-647. Referenced in section: 1 (page 1).

5. Wi-Fi (802.11) Network HandbookDec 5, 2002. Referenced in section: 1 (page 2).

6. Bluetooth Low Energy: The Developer's HandbookNov 7, 2012. Referenced in section: 1 (page 2).

7. Image and Video Compression: Fundamentals, Techniques, and ApplicationsNov 17, 2014. Referenced in section: 1 (page 2).

8. Wallace, Gregory K. "The JPEG still picture compression standard." IEEE transactions on consumer electronics 38.1 (1992): xviii-xxxiv. Referenced in sections: 1.1 (page 2), 2.4.3 (page 13).

9. ISO/IEC 15444-1:2004, Information technology - JPEG 2000 image coding system: Core coding system. Referenced in sections: 1.1 (page 2), 2.4.3 (page 14).

10. CIFRE doctoral fellowships program, http://www.anrt.asso.fr/fr/cifre-7843, (access time 9-July-2018). Referenced in sections: 1.4 (page 4), 3.1 (page 39).

11. Asadifard, M. and Shanbezadeh, J., 2010, January. Automatic adaptive center of pupil detection using face detection and cdf analysis. In Proceedings of the International MultiConference of Engineers and Computer Scientists (Vol. 1, p. 3). Referenced in section: 2.1 (page 5).

12. Arthi, S. V., and Norman, S. R. (2016). Interface and Control of Appliances by the Analysis of Electrooculography Signals. In Artificial Intelligence and Evolutionary Computations in Engineering Systems (pp. 1075-1084). Springer, New Delhi. Referenced in section: 2.2 (page 5).

13. Van der Geest, J. N., and Frens, M. A. (2002). Recording eye movements with video-oculography and scleral search coils: a direct comparison of two methods. Journal of neuroscience methods, 114(2), 185-195. Referenced in section: 2.2 (page 5).

14. Whitmire, E., Trutoiu, L., Cavin, R., Perek, D., Scally, B., Phillips, J., and Patel, S. (2016, September). EyeContact: scleral coil eye tracking for virtual reality. In Proceedings of the 2016 ACM International Symposium on Wearable Computers (pp. 184-191). ACM. Referenced in section: 2.2 (page 6).

15. Engbert, R., and Kliegl, R. (2003). Microsaccades uncover the orientation of covert attention. Vision research, 43(9), 1035-1045. Referenced in section: 2.2 (page 6).

16. Zhai, S., Morimoto, C., & Ihde, S. (1999, May). Manual and gaze input cascaded (MAGIC) pointing. In Proceedings of the SIGCHI conference on Human Factors in Computing Systems (pp. 246-253). ACM. Referenced in section: 2.2 (page 6).

17. Morimoto, C. H., Koons, D., Amir, A., & Flickner, M. (2000). Pupil detection and tracking using multiple light sources. Image and vision computing, 18(4), 331-335. Referenced in section: 2.2 (page 6).

18. Zhu, Z., & Ji, Q. (2005). Robust real-time eye detection and tracking under variable lighting conditions and various face orientations. Computer Vision and Image Understanding, 98(1), 124-154. Referenced in section: 2.2 (page 6).

19. Duda, R. O., & Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM, 15(1), 11-15. Referenced in section: 2.2 (page 7).

20. Gaussian smoothing, https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm (access time 20-Feb-2018). Referenced in section: 2.2 (page 7).

21. Bourne, R., 2010. The Spatial and Frequency Domains. In Fundamentals of Digital Imaging in Medicine (pp. 55-86). Springer London. Referenced in section: 2.3 (page 9).

22. Bracewell, R.N. and Bracewell, R.N., 1986. The Fourier transform and its applications (Vol. 31999). New York: McGraw-Hill. Referenced in sections: 2.3 (page 9), 4.5.3 (page 78).

23. Torrence, C. and Compo, G.P., 1998. A practical guide to wavelet analysis. Bulletin of the American Meteorological society, 79(1), pp.61-78. Referenced in section: 2.3 (page 9).

24. Empson, J. A. C., & Clarke, P. R. F. (1970). Rapid eye movements and remembering. Nature, 227(5255), 287. Referenced in section: 2.4 (page 9).

25. Rafiee, J., Rafiee, M.A., Prause, N. and Schoen, M.P., 2011. Wavelet basis functions in biomedical signal processing. Expert systems with Applications, 38(5), pp.6190-6201. Referenced in section: 2.4 (page 10).

26. Wang, Y., Vilermo, M. and Yaroslavsky, L., 2000, September. Energy compaction property of the MDCT in comparison with other transforms. In Audio Engineering Society Convention 109. Audio Engineering Society. Referenced in section: 2.4 (page 10).

27. Johnson, S.J., 2009. Iterative error correction: Turbo, low-density parity-check and repeat-accumulate codes. Cambridge university press. Referenced in section: 2.4 (page 11).

28. Koopman, P. and Chakravarty, T., 2004, June. Cyclic redundancy code (CRC) polynomial selection for embedded networks. In Dependable Systems and Networks, 2004 International Conference on (pp. 145-154). IEEE. Referenced in section: 2.4 (page 11).

29. Thorpe, S., Fize, D. and Marlot, C., 1996. Speed of processing in the human visual system. nature, 381(6582), p.520. Referenced in section: 2.4.1 (page 11).

30. Poynton, C., 2002. Chroma subsampling notation. Retrieved June, 19, p.2004. Referenced in section: 2.4.1 (page 12).

31. McCollough, C., 1965. Color adaptation of edge-detectors in the human visual system. Science, 149(3688), pp.1115-1116. Referenced in section: 2.4.1 (page 12).

32. Tagliasacchi, M., Trapanese, A., Tubaro, S., Ascenso, J., Brites, C. and Pereira, F., 2006, October. Exploiting spatial redundancy in pixel domain Wyner-Ziv video coding. In Image Processing, 2006 IEEE International Conference on (pp. 253-256). IEEE. Referenced in section: 2.4.2 (page 12).

33. Lewis, A.S. and Knowles, G., 1992. Image compression using the 2-D wavelet transform. IEEE transactions on image processing, 1(2), pp.244-250. Referenced in section: 2.4.2 (page 12).

34. Antonini, M., Barlaud, M., Mathieu, P. and Daubechies, I., 1992. Image coding using wavelet transform. IEEE Transactions on image processing, 1(2), pp.205-220. Referenced in section: 2.4.2 (page 12).

35. Selesnick, I.W., Baraniuk, R.G. and Kingsbury, N.C., 2005. The dual-tree complex wavelet transform. IEEE signal processing magazine, 22(6), pp.123-151. Referenced in sections: 2.4.2 (page 12), 4.5.3 (page 80).

36. Li, H., Manjunath, B.S. and Mitra, S.K., 1995. Multisensor image fusion using the wavelet transform. Graphical models and image processing, 57(3), pp.235-245. Referenced in section: 2.4.2 (page 12).

37. Chang, T. and Kuo, C.C., 1993. Texture analysis and classification with tree-structured wavelet transform. IEEE Transactions on image processing, 2(4), pp.429-441. Referenced in section: 2.4.2 (page 12).

38. Boles, W.W. and Boashash, B., 1998. A human identification technique using images of the iris and wavelet transform. IEEE transactions on signal processing, 46(4), pp.1185-1188. Referenced in section: 2.4.2 (page 12).

39. Chang, S.G., Yu, B. and Vetterli, M., 2000. Adaptive wavelet thresholding for image denoising and compression. IEEE transactions on image processing, 9(9), pp.1532-1546. Referenced in section: 2.4.2 (page 12).

40. Chang, S.G., Yu, B. and Vetterli, M., 2000. Spatially adaptive wavelet thresholding with context modeling for image denoising. IEEE Transactions on image Processing, 9(9), pp.1522-1531. Referenced in section: 2.4.2 (page 12).

41. Claypoole, R.L., Baraniuk, R.G. and Nowak, R.D., 1998, May. Adaptive wavelet transforms via lifting. In Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on (Vol. 3, pp. 1513-1516). IEEE. Referenced in section: 2.4.2 (page 12).

42. Le Pennec, E. and Mallat, S., 2005. Bandelet image approximation and compression. Multiscale Modeling & Simulation, 4(3), pp.992-1039. Referenced in section: 2.4.2 (page 12).

43. Peyré, G. and Mallat, S., 2008. Orthogonal bandelet bases for geometric images approximation. Communications on Pure and Applied Mathematics, 61(9), pp.1173-1212. Referenced in section: 2.4.2 (page 12).

44. Maalouf, A., Carré, P., Augereau, B. and Fernandez-Maloigne, C., 2009. A bandelet-based inpainting technique for clouds removal from remotely sensed images. IEEE Transactions on geoscience and remote sensing, 47(7), pp.2363-2371. Referenced in section: 2.4.2 (page 12).

45. Do, M.N. and Vetterli, M., 2005. The contourlet transform: an efficient directional multiresolution image representation. IEEE Transactions on image processing, 14(12), pp.2091-2106. Referenced in section: 2.4.2 (page 12).

46. Da Cunha, A.L., Zhou, J. and Do, M.N., 2006. The nonsubsampled contourlet transform: theory, design, and applications. IEEE transactions on image processing, 15(10), pp.3089-3101. Referenced in section: 2.4.2 (page 12).

47. Po, D.Y. and Do, M.N., 2006. Directional multiscale modeling of images using the contourlet transform. IEEE Transactions on image processing, 15(6), pp.1610-1620. Referenced in section: 2.4.2 (page 12).

48. Gao, Q., Lu, Y., Sun, D., Sun, Z.L. and Zhang, D., 2013. Directionlet-based denoising of SAR images using a Cauchy model. Signal processing, 93(5), pp.1056-1063. Referenced in section: 2.4.2 (page 12).

49. Anand, S., Kumari, R.S.S., Jeeva, S. and Thivya, T., 2013. Directionlet transform based sharpening and enhancement of mammographic X-ray images. Biomedical Signal Processing and Control, 8(4), pp.391-399. Referenced in section: 2.4.2 (page 12).

50. Lin, Y.C., Liu, Q.H., TAO, L., SONG, L. and ZHAO, M.R., 2010. An image fusion algorithm based on directionlet transform. Nanotechnology and Precision Engineering, 8(6), pp.565-568. Referenced in section: 2.4.2 (page 12).

51. Yao, B. and Fei-Fei, L., 2010, June. Grouplet: A structured image representation for recognizing human and object interactions. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on (pp. 9-16). IEEE. Referenced in section: 2.4.2 (page 12).

52. Mallat, S., 2009. Geometrical grouplets. Applied and Computational Harmonic Analysis, 26(2), pp.161-180. Referenced in section: 2.4.2 (page 12).

53. Ahmed, N., Natarajan, T. and Rao, K.R., 1974. Discrete cosine transform. IEEE transactions on Computers, 100(1), pp.90-93. Referenced in section: 2.4.2 (page 12).

54. Rao, K.R. and Yip, P., 2014. Discrete cosine transform: algorithms, advantages, applications. Academic press. Referenced in section: 2.4.2 (page 12).

55. Chen, W.H., Smith, C.H. and Fralick, S.C., 1977. A fast computational algorithm for the discrete cosine transform. IEEE Transactions on communications, 25(9), pp.1004-1009. Referenced in section: 2.4.2 (page 12).

56. Malvar, H.S., Hallapuro, A., Karczewicz, M. and Kerofsky, L., 2003. Low-complexity transform and quantization in H. 264/AVC. IEEE Transactions on circuits and systems for video technology, 13(7), pp.598-603. Referenced in section: 2.4.2 (page 12).

57. Pearlman, W.A., Islam, A., Nagaraj, N. and 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), pp.1219-1235. Referenced in section: 2.4.2 (page 12).

58. McCanne, S., Vetterli, M. and Jacobson, V., 1997. Low-complexity video coding for receiver-driven layered multicast. IEEE journal on selected areas in communications, 15(6), pp.983-1001. Referenced in section: 2.4.2 (page 12).

59. Liang, J. and Tran, T.D., 2001. Fast multiplierless approximations of the DCT with the lifting scheme. IEEE transactions on signal processing, 49(12), pp.3032-3044. Referenced in sections: 2.4.2 (page 12), 2.4.2 (page 12).

60. Tran, T.D., 2000. The BinDCT: Fast multiplierless approximation of the DCT. IEEE Signal Processing Letters, 7(6), pp.141-144. Referenced in section: 2.4.2 (page 12).

61. Jeong, H., Kim, J. and Cho, W.K., 2004. Low-power multiplierless DCT architecture using image correlation. IEEE Transactions on Consumer Electronics, 50(1), pp.262-267. Referenced in section: 2.4.2 (page 12).

62. Krommweh, J., 2010. Tetrolet transform: A new adaptive Haar wavelet algorithm for sparse image representation. Journal of Visual Communication and Image Representation, 21(4), pp.364-374. Referenced in section: 2.4.2 (page 12).

63. Krommweh, J. and Ma, J., 2010. Tetrolet shrinkage with anisotropic total variation minimization for image approximation. Signal processing, 90(8), pp.2529-2539. Referenced in section: 2.4.2 (page 12).

64. Jain, P. and Tyagi, V., 2015. An adaptive edge-preserving image denoising technique using tetrolet transforms. The Visual Computer, 31(5), pp.657-674. Referenced in section: 2.4.2 (page 12).

65. Easley, G., Labate, D. and Lim, W.Q., 2008. Sparse directional image representations using the discrete shearlet transform. Applied and Computational Harmonic Analysis, 25(1), pp.25-46. Referenced in section: 2.4.2 (page 12).

66. Yi, S., Labate, D., Easley, G.R. and Krim, H., 2009. A shearlet approach to edge analysis and detection. IEEE Transactions on Image Processing, 18(5), pp.929-941. Referenced in sections: 2.4.2 (page 12), 2.4.2 (page 13).

67. Lim, W.Q., 2010. The discrete shearlet transform: A new directional transform and compactly supported shearlet frames. IEEE Transactions on Image Processing, 19(5), pp.1166-1180. Referenced in section: 2.4.2 (page 12).

68. Sweldens, W., 1996. The lifting scheme: A custom-design construction of biorthogonal wavelets. Applied and computational harmonic analysis, 3(2), pp.186-200. Referenced in section: 2.4.2 (page 12).

69. Shiau, Y.H., Jou, J.M. and Liu, C.C., 2004. Efficient architectures for the biorthogonal wavelet transform by filter bank and lifting scheme. IEICE TRANSACTIONS on Information and Systems, 87(7), pp.1867-1877. Referenced in section: 2.4.2 (page 12).

70. Secker, A. and Taubman, D., 2002, September. Highly scalable video compression using a lifting-based 3D wavelet transform with deformable mesh motion compensation. In Image Processing. 2002. Proceedings. 2002 International Conference on (Vol. 3, pp. 749-752). IEEE. Referenced in section: 2.4.2 (page 12).

71. Ding, W., Wu, F. and Li, S., 2004, December. Lifting-based wavelet transform with directionally spatial prediction. In Picture Coding Symposium (Vol. 62, pp. 291-294). Referenced in section: 2.4.2 (page 12).

72. Shen, G. and Ortega, A., 2008, March. Optimized distributed 2D transforms for irregularly sampled sensor network grids using wavelet lifting. In Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on (pp. 2513-2516). IEEE. Referenced in section: 2.4.2 (page 12).

73. Meerwald, P., Norcen, R. and Uhl, A., 2002, January. Cache issues with JPEG2000 wavelet lifting. In Visual Communications and Image Processing 2002 (Vol. 4671, pp. 626-635). International Society for Optics and Photonics. Referenced in section: 2.4.2 (page 12).

74. Chatterjee, S. and Brooks, C.D., 2002. Cache-efficient wavelet lifting in JPEG 2000. In Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on (Vol. 1, pp. 797-800). IEEE. Referenced in section: 2.4.2 (page 12).

75. Kutil, R., 2006, February. A single-loop approach to SIMD parallelization of 2D wavelet lifting. In Parallel, Distributed, and Network-Based Processing, 2006. PDP 2006. 14th Euromicro International Conference on (pp. 8-pp). IEEE. Referenced in section: 2.4.2 (page 12).

76. Półchłopek, W., Maj, W., & Padee, W. (2006, September). Fast integer arithmetic wavelet transform properties and application in FPGA/DSP system. In Signal Processing Conference, 2006 14th European (pp. 1-5). IEEE. Referenced in section: 2.4.2 (page 12).

77. Libjpeg-turbo – JPEG image codec that uses SIMD instructions (MMX, SSE2, AVX2, NEON, AltiVec), https://libjpeg-turbo.org, (access time 2-March-2018). Referenced in section: 2.4.2 (page 12).

78. Tenllado, C., Setoain, J., Prieto, M., Piñuel, L. and Tirado, F., 2008. Parallel implementation of the 2D discrete wavelet transform on graphics processing units: Filter bank versus lifting. IEEE Transactions on Parallel and Distributed Systems, 19(3), pp.299-310. Referenced in section: 2.4.2 (page 12).

79. Franco, J., Bernabé, G., Fernández, J. and Acacio, M.E., 2009, February. A parallel implementation of the 2D wavelet transform using CUDA. In Parallel, Distributed and Network-based Processing, 2009 17th Euromicro International Conference on (pp. 111-118). IEEE. Referenced in section: 2.4.2 (page 12).

80. Lai, Y.K., Chen, L.F. and Shih, Y.C., 2009. A high-performance and memory-efficient VLSI architecture with parallel scanning method for 2-D lifting-based discrete wavelet transform. IEEE Transactions on Consumer Electronics, 55(2), pp.400-407. Referenced in section: 2.4.2 (page 12).

81. Yick, J., Mukherjee, B. and Ghosal, D., 2008. Wireless sensor network survey. Computer networks, 52(12), pp.2292-2330. Referenced in section: 2.4.2 (page 13).

82. Sohrabi, K., Gao, J., Ailawadhi, V. and Pottie, G.J., 2000. Protocols for self-organization of a wireless sensor network. IEEE personal communications, 7(5), pp.16-27. Referenced in section: 2.4.2 (page 13).

83. Mao, G., Fidan, B. and Anderson, B.D., 2007. Wireless sensor network localization techniques. Computer networks, 51(10), pp.2529-2553. Referenced in section: 2.4.2 (page 13).

84. Callahan, S.M., Apple Computer, Inc., 1995. Method and apparatus for real-time lossless compression and decompression of image data. U.S. Patent 5,408,542. Referenced in section: 2.4.2 (page 13).

85. Akter, M., Reaz, M.B.I., Mohd-Yasin, F. and Choong, F., 2008. A modified-set partitioning in hierarchical trees algorithm for real-time image compression. Journal of Communications Technology and Electronics, 53(6), pp.642-650. Referenced in section: 2.4.2 (page 13).

86. Kajiya, J.T., Gabriel, S.A. and Powell III, W.C., Microsoft Corporation, 1999. Image compression to reduce pixel and texture memory requirements in a real-time image generator. U.S. Patent 5,999,189. Referenced in section: 2.4.2 (page 13).

87. Yu, G., Vladimirova, T. and Sweeting, M.N., 2009. Image compression systems on board satellites. Acta Astronautica, 64(9), pp.988-1005. Referenced in section: 2.4.2 (page 13).

88. Yeh, P.S., Armbruster, P., Kiely, A., Masschelein, B., Moury, G., Schaefer, C. and Thiebaut, C., 2005, March. The new CCSDS image compression recommendation. In Aerospace Conference, 2005 IEEE (pp. 4138-4145). IEEE. Referenced in section: 2.4.2 (page 13).

89. Parisot, C., Antonini, M., Barlaud, M., Lambert-Nebout, C., Latry, C. and Moury, G., 2000. On board strip-based wavelet image coding for future space remote sensing missions. In Geoscience and remote sensing symposium, 2000. Proceedings. IGARSS 2000. IEEE 2000 international (Vol. 6, pp. 2651-2653). IEEE. Referenced in section: 2.4.2 (page 13).

90. Kiely, A., & Klimesh, M. (2003). The ICER progressive wavelet image compressor. IPN Progress Report, 42(155), 1-46. Referenced in section: 2.4.2 (page 13).

91. Lin, A. (2012). Hardware implementation of a real-time image data compression for satellite remote sensing. In Remote Sensing-Advanced Techniques and Platforms. InTech. Referenced in section: 2.4.2 (page 13).

92. Altürk, A. and Keinert, F., 2012. Regularity of boundary wavelets. Applied and Computational Harmonic Analysis, 32(1), pp.65-85. Referenced in section: 2.4.2 (page 13).

93. Su HA, Liu QU, Li JI. Boundary effects reduction in wavelet transform for time-frequency analysis. WSEAS Transactions on Signal Processing. 2012 Oct;8(4):169-79. Referenced in section: 2.4.2 (page 13).

94. Černá, D., Finěk, V., Gottfried, M., Hübnerová, P. and Paulusová, S., 2009. Boundary artifact reduction in wavelet image compression. Referenced in section: 2.4.2 (page 13).

95. Strutz, T. and Rennert, I., 2012. Two-dimensional integer wavelet transform with reduced influence of rounding operations. EURASIP Journal on Advances in Signal Processing, 2012(1), p.75. Referenced in section: 2.4.2 (page 13).

96. Zavadsky, V., 2004. Image compression by rectangular wavelet transform. arXiv preprint cs/0406008. Referenced in section: 2.4.2 (page 13).

97. Kong, F. (2016, August). Research of edge detection algorithm based on wavelet transformation. In Eighth International Conference on Digital Image Processing (ICDIP 2016) (Vol. 10033, p. 100331H). International Society for Optics and Photonics. Referenced in section: 2.4.2 (page 13).

98. Bai, J. and Zhou, H., 2011, July. Edge detection approach based on directionlet transform. In Multimedia Technology (ICMT), 2011 International Conference on (pp. 3512-3515). IEEE. Referenced in section: 2.4.2 (page 13).

99. Farid, M. M., Kurugollu, F., & Murtagh, F. D. (2003, March). Adaptive wavelet eye-gaze-based video compression. In Opto-Ireland 2002: Optical Metrology, Imaging, and Machine Vision (Vol. 4877, pp. 255-264). International Society for Optics and Photonics. Referenced in section: 2.4.2 (page 13).

100. Somayajula, S. P. K., Dhatrika, S. C., & Puvvula, D. Wave Atoms Decomposition based Eye Iris Image Compression. Referenced in section: 2.4.2 (page 13).

101. Said, A. and Pearlman, W.A., 1996. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. IEEE Transactions on circuits and systems for video technology, 6(3), pp.243-250. Referenced in section: 2.4.2 (page 13).

102. Lei, S.M. and Sun, M.T., 1991. An entropy coding system for digital HDTV applications. IEEE transactions on circuits and systems for video technology, 1(1), pp.147-155. Referenced in section: 2.4.2 (page 13).

103. Meany, J.J. and Martens, C.J., McDonnell Douglas Corporation, 1998. Error resilient method and apparatus for entropy coding. U.S. Patent 5,850,482. Referenced in section: 2.4.2 (page 13).

104. Sullivan, G.J. and Wiegand, T., 1998. Rate-distortion optimization for video compression. IEEE signal processing magazine, 15(6), pp.74-90. Referenced in sections: 2.4.2 (page 13), 2.4.5 (page 16).

105. Knuth, D.E., 1985. Dynamic huffman coding. Journal of algorithms, 6(2), pp.163-180. Referenced in sections: 2.4.3 (page 13), 2.4.5 (page 16).

106. Advanced video coding for generic audiovisual services, ITU-T Recommendation H.264 and ISO/IEC 14496-10 (AVC), 2009. Referenced in section: 2.4.4 (page 14).

107. Richardson, I.E., 2011. The H. 264 advanced video compression standard. John Wiley & Sons. Referenced in section: 2.4.4 (page 14).

108. Budagavi, M., 2016. High Efficiency Video Coding (hevc): Algorithms and Architectures. Springer. Referenced in section: 2.4.4 (page 14).

109. Kaddachi, M.L., Soudani, A., Lecuire, V., Makkaoui, L., Moureaux, J.M. and Torki, K., 2012. Design and performance analysis of a zonal DCT-based image encoder for Wireless Camera Sensor Networks. Microelectronics Journal, 43(11), pp.809-817. Referenced in section: 2.4.5 (page 15).

110. Pennebaker, W.B., Mitchell, J.L., Langdon, G.G. and 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), pp.717-726. Referenced in sections: 2.4.5 (page 15), 2.4.5 (page 16).

111. Chandra, A. and 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), pp.355-368. Referenced in sections: 2.4.5 (page 15), 2.4.5 (page 16).

112. Amutha, R., 2013. Low complexity energy efficient very low bit-rate image compression scheme for wireless sensor network. Information Processing Letters, 113(18), pp.672-676. Referenced in section: 2.4.5 (page 16).

113. Bayer, F.M. and Cintra, R.J., 2012. DCT-like transform for image compression requires 14 additions only. Electronics Letters, 48(15), pp.919-921. Referenced in section: 2.4.5 (page 16).

114. da Silveira, T.L., Oliveira, R.S., Bayer, F.M., Cintra, R.J. and Madanayake, A., 2017. Multiplierless 16-point DCT approximation for low-complexity image and video coding. Signal, Image and Video Processing, 11(2), pp.227-233. Referenced in section: 2.4.5 (page 16).

115. Gersho, A. and Gray, R.M., 2012. Vector quantization and signal compression (Vol. 159). Springer Science & Business Media. Referenced in section: 2.4.5 (page 16).

116. Hauck, E.L., INTELLIGENT STORAGE Inc, 1986. Data compression using run length encoding and statistical encoding. U.S. Patent 4,626,829. Referenced in section: 2.4.5 (page 16).

117. Witten, I.H., Neal, R.M. and Cleary, J.G., 1987. Arithmetic coding for data compression. Communications of the ACM, 30(6), pp.520-540. Referenced in section: 2.4.5 (page 16).

118. Masmoudi, A., Puech, W., & Bouhlel, M. S. (2010). Efficient adaptive arithmetic coding based on updated probability distribution for lossless image compression. Journal of Electronic Imaging, 19(2), 023014. Referenced in section: 2.4.5 (page 16).

119. Finite State Entropy - A new breed of entropy coder, http://fastcompression.blogspot.fr/2013/12/finite-state-entropy-new-breed-of.html (access time 31-Jan-2018), https://github.com/Cyan4973/FiniteStateEntropy (access time 31-Jan-2018). Referenced in sections: 2.4.5 (page 16), 4.6.2 (page 91).

120. Duda, J., 2009. Asymmetric numeral systems. arXiv preprint arXiv:0902.0271. Referenced in section: 2.4.5 (page 16).

121. Duda, J., 2013. Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding. arXiv preprint arXiv:1311.2540. Referenced in section: 2.4.5 (page 16).

122. Duda, J., Tahboub, K., Gadgil, N.J. and Delp, E.J., 2015, May. The use of asymmetric numeral systems as an accurate replacement for Huffman coding. In Picture Coding Symposium (PCS), 2015 (pp. 65-69). IEEE. Referenced in section: 2.4.5 (page 16).

123. Najmabadi, S.M., Wang, Z., Baroud, Y. and Simon, S., 2015, September. High throughput hardware architectures for asymmetric numeral systems entropy coding. In Image and Signal Processing and Analysis (ISPA), 2015 9th International Symposium on (pp. 256-259). IEEE. Referenced in section: 2.4.5 (page 16).

124. Zstandard – lossless data compression algorithm, http://facebook.github.io/zstd (access time 2-March-2018). Referenced in sections: 2.4.5 (page 16), 2.4.5 (page 17).

125. Girod, B., 1992. Psychovisual aspects of image communication. Signal Processing, 28(3), pp.239-251. Referenced in section: 2.4.5 (page 16).

126. Alakuijala, J., Obryk, R., Stoliarchuk, O., Szabadka, Z., Vandevenne, L. and Wassenberg, J., 2017. Guetzli: Perceptually Guided JPEG Encoder. arXiv preprint arXiv:1703.04421. Referenced in sections: 2.4.5 (page 16), 2.4.5 (page 16).

127. Alakuijala, J., Obryk, R., Szabadka, Z. and Wassenberg, J., 2017. Users prefer Guetzli JPEG over same-sized libjpeg. arXiv preprint arXiv:1703.04416. Referenced in sections: 2.4.5 (page 16), 2.4.5 (page 16).

128. Google, Inc., https://www.google.com (access time 31-Jan-2018). Referenced in section: 2.4.5 (page 16).

129. https://github.com/google/guetzli/blob/master/guetzli/order.inc (access time 31-Jan-2018), https://github.com/google/butteraugli/blob/master/butteraugli/butteraugli.cc (access time 31-Jan-2018). Referenced in section: 2.4.5 (page 16).

130. Brotli compression format, https://github.com/google/brotli (access time 2-March-2018). Referenced in section: 2.4.5 (page 16).

131. Brotli Compressed Data Format, RFC 7932, https://tools.ietf.org/html/rfc7932 (access time 2-March-2018). Referenced in section: 2.4.5 (page 16).

132. Sneyers, J. and Wuille, P., 2016, September. FLIF: Free lossless image format based on MANIAC compression. In Image Processing (ICIP), 2016 IEEE International Conference on (pp. 66-70). IEEE. Referenced in section: 2.4.5 (page 16).

133. FLIF (Free Lossless Image Format), www.flif.info (access time 31-Jan-2018). Referenced in sections: 2.4.5 (page 16), 4.5.2 (page 78).

134. Presentation FREE LOSSLESS IMAGE FORMAT, http://flif.info/slides/FLIF_ICIP16.pdf (access time 31-Jan-2018). Referenced in section: 2.4.5 (page 16).

135. Sneyers J. and Wuille P. JiF: Image Compression With an Auto-Indexing and Context-Learning MANIAC, https://drive.google.com/file/d/0BwMTfsYj-_l6eWZWRHg3RGtwQW8/view?pli=1 (access time 31-Jan-2018). Referenced in section: 2.4.5 (page 16).

136. Sze, V. and Budagavi, M., 2012. High throughput CABAC entropy coding in HEVC. IEEE Transactions on Circuits and Systems for Video Technology, 22(12), pp.1778-1791. Referenced in section: 2.4.5 (page 16).

137. Fisher, Y., 2012. Fractal image compression: theory and application. Springer Science & Business Media. Referenced in section: 2.4.5 (page 16).

138. Eldar, Y.C. and Kutyniok, G. eds., 2012. Compressed sensing: theory and applications. Cambridge University Press. Referenced in section: 2.4.5 (page 16).

139. Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., & Edwards, D. D. (2003). Artificial intelligence: a modern approach (Vol. 2, No. 9). Upper Saddle River: Prentice hall. Referenced in section: 2.4.5 (page 16).

140. Nasrabadi, N.M., 2007. Pattern recognition and machine learning. Journal of electronic imaging, 16(4), p.049901. Referenced in section: 2.4.5 (page 16).

141. Jiang, F., Tao, W., Liu, S., Ren, J., Guo, X. and Zhao, D., 2017. An end-to-end compression framework based on convolutional neural networks. IEEE Transactions on Circuits and Systems for Video Technology. Referenced in section: 2.4.5 (page 16).

142. Toderici, G., Vincent, D., Johnston, N., Hwang, S.J., Minnen, D., Shor, J. and Covell, M., 2016. Full resolution image compression with recurrent neural networks. arXiv preprint. Referenced in section: 2.4.5 (page 16).

143. Toderici, G., O'Malley, S.M., Hwang, S.J., Vincent, D., Minnen, D., Baluja, S., Covell, M. and Sukthankar, R., 2015. Variable rate image compression with recurrent neural networks. arXiv preprint arXiv:1511.06085. Referenced in section: 2.4.5 (page 16).

144. WaveOne – Context-adaptive compression of digital media, http://www.wave.one (access time 31-Jan-2018). Referenced in section: 2.4.5 (page 16).

145. Rippel, O. and Bourdev, L., 2017. Real-time adaptive image compression. arXiv preprint arXiv:1705.05823. Referenced in section: 2.4.5 (page 16).

146. https://www.dropbox.com (access time 31-Jan-2018). Referenced in section: 2.4.5 (page 17).

147. Lepton image compression: saving 22% losslessly from images at 15MB/s, https://blogs.dropbox.com/tech/2016/07/lepton-image-compression-saving-22-losslessly-from-images-at-15mbs (access time 31-Jan-2018), https://github.com/dropbox/lepton (access time 31-Jan-2018). Referenced in sections: 2.4.5 (page 17), 2.4.5 (page 17).

148. Samuel, A.L., 1959. Some studies in machine learning using the game of checkers. IBM Journal of research and development, 3(3), pp.210-229. Referenced in section: 2.5 (page 17).

149. Cortes, C. and Vapnik, V., 1995. Support-vector networks. Machine learning, 20(3), pp.273-297. Referenced in section: 2.5 (page 17).

150. Rasmussen, C.E., 2004. Gaussian processes in machine learning. In Advanced lectures on machine learning (pp. 63-71). Springer, Berlin, Heidelberg. Referenced in sections: 2.5 (page 17), 2.6.1 (page 20).

151. Møller, M.F., 1993. A scaled conjugate gradient algorithm for fast supervised learning. Neural networks, 6(4), pp.525-533. Referenced in section: 2.5 (page 17).

152. Hastie, T., Tibshirani, R. and Friedman, J., 2009. Unsupervised learning. In The elements of statistical learning (pp. 485-585). Springer, New York, NY. Referenced in section: 2.5 (page 17).

153. Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J. and Scholkopf, B., 1998. Support vector machines. IEEE Intelligent Systems and their applications, 13(4), pp.18-28. Referenced in section: 2.5 (page 17).

154. Li, M. and Yuan, B., 2005. 2D-LDA: A statistical linear discriminant analysis for image matrix. Pattern Recognition Letters, 26(5), pp.527-532. Referenced in section: 2.5 (page 17).

155. Mika, S., Ratsch, G., Weston, J., Scholkopf, B. and Mullers, K.R., 1999, August. Fisher discriminant analysis with kernels. In Neural networks for signal processing IX, 1999. Proceedings of the 1999 IEEE signal processing society workshop. (pp. 41-48). Ieee. Referenced in section: 2.5 (page 17).

156. Murphy, K.P., 2006. Naive bayes classifiers. University of British Columbia, 18. Referenced in section: 2.5 (page 17).

157. Larose, D.T., 2005. k-nearest neighbor algorithm. Discovering Knowledge in Data: An Introduction to Data Mining, pp.90-106. Referenced in section: 2.5 (page 18).

158. Crawley, M.J., 2007. Generalized linear models. The R book, pp.511-526. Referenced in section: 2.5 (page 18).

159. Basak, D., Pal, S. and Patranabis, D.C., 2007. Support vector regression. Neural Information Processing-Letters and Reviews, 11(10), pp.203-224. Referenced in section: 2.5 (page 18).

160. Seo, S., Wallat, M., Graepel, T. and Obermayer, K., 2000. Gaussian process regression: Active data selection and test point rejection. In Mustererkennung 2000 (pp. 27-34). Springer, Berlin, Heidelberg. Referenced in section: 2.5 (page 18).

161. Dietterich, T.G., 2000, June. Ensemble methods in machine learning. In International workshop on multiple classifier systems (pp. 1-15). Springer, Berlin, Heidelberg. Referenced in section: 2.5 (page 18).

162. Quinlan, J.R., 1986. Induction of decision trees. Machine learning, 1(1), pp.81-106. Referenced in section: 2.5 (page 18).

163. Hartigan, J.A. and Wong, M.A., 1979. Algorithm AS 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics), 28(1), pp.100-108. Referenced in sections: 2.5 (page 18), 3.5.3 (page 60).

164. Xu, P., 2006. Voronoi cells, probabilistic bounds, and hypothesis testing in mixed integer linear models. IEEE Transactions on information theory, 52(7), pp.3122-3138. Referenced in section: 2.5 (page 18).

165. Jin, X. and Han, J., 2016. K-medoids clustering. In Encyclopedia of Machine Learning and Data Mining (pp. 1-3). Springer US. Referenced in section: 2.5 (page 18).

166. Bezdek, J.C., Ehrlich, R. and Full, W., 1984. FCM: The fuzzy c-means clustering algorithm. Computers and Geosciences, 10(2-3), pp.191-203. Referenced in section: 2.5 (page 18).

167. Pal, N.R. and Bezdek, J.C., 1995. On cluster validity for the fuzzy c-means model. IEEE Transactions on Fuzzy systems, 3(3), pp.370-379. Referenced in section: 2.5 (page 18).

168. Chuang, K.S., Tzeng, H.L., Chen, S., Wu, J. and Chen, T.J., 2006. Fuzzy c-means clustering with spatial information for image segmentation. computerized medical imaging and graphics, 30(1), pp.9-15. Referenced in section: 2.5 (page 18).

169. Pal, N.R., Pal, K., Keller, J.M. and Bezdek, J.C., 2005. A possibilistic fuzzy c-means clustering algorithm. IEEE transactions on fuzzy systems, 13(4), pp.517-530. Referenced in section: 2.5 (page 18).

170. Cai, W., Chen, S. and Zhang, D., 2007. Fast and robust fuzzy c-means clustering algorithms incorporating local information for image segmentation. Pattern recognition, 40(3), pp.825-838. Referenced in section: 2.5 (page 18).

171. Cannon, R.L., Dave, J.V. and Bezdek, J.C., 1986. Efficient implementation of the fuzzy c-means clustering algorithms. IEEE transactions on pattern analysis and machine intelligence, (2), pp.248-255. Referenced in section: 2.5 (page 18).

172. Zhang, S., Wang, R.S. and Zhang, X.S., 2007. Identification of overlapping community structure in complex networks using fuzzy c-means clustering. Physica A: Statistical Mechanics and its Applications, 374(1), pp.483-490. Referenced in section: 2.5 (page 18).

173. Hathaway, R.J. and Bezdek, J.C., 2001. Fuzzy c-means clustering of incomplete data. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 31(5), pp.735-744. Referenced in section: 2.5 (page 18).

174. Hathaway, R.J., Bezdek, J.C. and Hu, Y., 2000. Generalized fuzzy c-means clustering strategies using L/sub p/norm distances. IEEE transactions on Fuzzy Systems, 8(5), pp.576-582. Referenced in section: 2.5 (page 18).

175. Karayiannis, N.B. and Bezdek, J.C., 1997. An integrated approach to fuzzy learning vector quantization and fuzzy c-means clustering. IEEE Transactions on Fuzzy Systems, 5(4), pp.622-628. Referenced in section: 2.5 (page 18).

176. Navarro, J.F., Frenk, C.S. and White, S.D., 1997. A universal density profile from hierarchical clustering. The Astrophysical Journal, 490(2), p.493. Referenced in section: 2.5 (page 18).

177. Corpet, F., 1988. Multiple sequence alignment with hierarchical clustering. Nucleic acids research, 16(22), pp.10881-10890. Referenced in section: 2.5 (page 18).

178. Johnson, S.C., 1967. Hierarchical clustering schemes. Psychometrika, 32(3), pp.241-254. Referenced in section: 2.5 (page 18).

179. Steinbach, M., Karypis, G. and Kumar, V., 2000, August. A comparison of document clustering techniques. In KDD workshop on text mining (Vol. 400, No. 1, pp. 525-526). Referenced in section: 2.5 (page 18).

180. Karypis, G., Han, E.H. and Kumar, V., 1999. Chameleon: Hierarchical clustering using dynamic modeling. Computer, 32(8), pp.68-75. Referenced in section: 2.5 (page 18).

181. Bandyopadhyay, S. and Coyle, E.J., 2003, April. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies (Vol. 3, pp. 1713-1723). IEEE. Referenced in section: 2.5 (page 18).

182. White, S.D. and Frenk, C.S., 1991. Galaxy formation through hierarchical clustering. The Astrophysical Journal, 379, pp.52-79. Referenced in section: 2.5 (page 18).

183. Suzuki, R. and Shimodaira, H., 2006. Pvclust: an R package for assessing the uncertainty in hierarchical clustering. Bioinformatics, 22(12), pp.1540-1542. Referenced in section: 2.5 (page 18).

184. Murtagh, F., 1983. A survey of recent advances in hierarchical clustering algorithms. The Computer Journal, 26(4), pp.354-359. Referenced in section: 2.5 (page 18).

185. Anderberg, M.R., 1973. Cluster analysis for applications (No. OAS-TR-73-9). Office of the Assistant for Study Support Kirtland AFB N MEX. Referenced in section: 2.5 (page 18).

186. Rousseeuw, P.J., 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics, 20, pp.53-65. Referenced in section: 2.5 (page 18).

187. Milligan, G.W. and Cooper, M.C., 1986. A study of the comparability of external criteria for hierarchical cluster analysis. Multivariate Behavioral Research, 21(4), pp.441-458. Referenced in section: 2.5 (page 18).

188. Revelle, W., 1979. Hierarchical cluster analysis and the internal structure of tests. Multivariate Behavioral Research, 14(1), pp.57-74. Referenced in section: 2.5 (page 18).

189. Baker, F.B. and Hubert, L.J., 1975. Measuring the power of hierarchical cluster analysis. Journal of the American Statistical Association, 70(349), pp.31-38. Referenced in section: 2.5 (page 18).

190. Peeters, J.P. and Martinelli, J.A., 1989. Hierarchical cluster analysis as a tool to manage variation in germplasm collections. Theoretical and applied genetics, 78(1), pp.42-48. Referenced in section: 2.5 (page 18).

191. Arifin, A.Z. and Asano, A., 2006. Image segmentation by histogram thresholding using hierarchical cluster analysis. Pattern recognition letters, 27(13), pp.1515-1521. Referenced in section: 2.5 (page 18).

192. Gruvaeus, G. and Wainer, H., 1972. Two additions to hierarchical cluster analysis. British Journal of Mathematical and Statistical Psychology, 25(2), pp.200-206. Referenced in section: 2.5 (page 18).

193. Bridges Jr, C.C., 1966. Hierarchical cluster analysis. Psychological reports, 18(3), pp.851-854. Referenced in section: 2.5 (page 18).

194. Köhn, H.F. and Hubert, L.J., 2006. Hierarchical cluster analysis. Wiley StatsRef: Statistics Reference Online. Referenced in section: 2.5 (page 18).

195. Rabiner, L. and Juang, B., 1986. An introduction to hidden Markov models. ieee assp magazine, 3(1), pp.4-16. Referenced in section: 2.5 (page 18).

196. Yamato, J., Ohya, J. and Ishii, K., 1992, June. Recognizing human action in time-sequential images using hidden markov model. In Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on (pp. 379-385). IEEE. Referenced in section: 2.5 (page 18).

197. Eddy, S.R., 1998. Profile hidden Markov models. Bioinformatics (Oxford, England), 14(9), pp.755-763. Referenced in section: 2.5 (page 18).

198. Bahl, L., Brown, P., De Souza, P. and Mercer, R., 1986, April. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86. (Vol. 11, pp. 49-52). IEEE. Referenced in section: 2.5 (page 18).

199. Varga, A. and Moore, R.K., 1990, April. Hidden Markov model decomposition of speech and noise. In Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on (pp. 845-848). IEEE. Referenced in section: 2.5 (page 18).

200. Blunsom, P., 2004. Hidden markov models. Lecture notes, August, 15, pp.18-19. Referenced in section: 2.5 (page 18).

201. Morgan, N. and Bourlard, H., 1990, April. Continuous speech recognition using multilayer perceptrons with hidden Markov models. In Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on (pp. 413-416). IEEE. Referenced in section: 2.5 (page 18).

202. Reynolds, D.A., Quatieri, T.F. and Dunn, R.B., 2000. Speaker verification using adapted Gaussian mixture models. Digital signal processing, 10(1-3), pp.19-41. Referenced in section: 2.5 (page 18).

203. Reynolds, D.A. and Rose, R.C., 1995. Robust text-independent speaker identification using Gaussian mixture speaker models. IEEE transactions on speech and audio processing, 3(1), pp.72-83. Referenced in section: 2.5 (page 18).

204. Zivkovic, Z., 2004, August. Improved adaptive Gaussian mixture model for background subtraction. In Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on (Vol. 2, pp. 28-31). IEEE. Referenced in section: 2.5 (page 18).

205. Rasmussen, C.E., 2000. The infinite Gaussian mixture model. In Advances in neural information processing systems (pp. 554-560). Referenced in section: 2.5 (page 18).

206. Huang, Y., Englehart, K.B., Hudgins, B. and Chan, A.D., 2005. A Gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses. IEEE Transactions on Biomedical Engineering, 52(11), pp.1801-1811. Referenced in section: 2.5 (page 18).

207. Yang, M.H. and Ahuja, N., 1998, December. Gaussian mixture model for human skin color and its applications in image and video databases. In Storage and Retrieval for Image and Video Databases VII (Vol. 3656, pp. 458-467). International Society for Optics and Photonics. Referenced in section: 2.5 (page 18).

208. Greenspan, H., Ruf, A. and Goldberger, J., 2006. Constrained Gaussian mixture model framework for automatic segmentation of MR brain images. IEEE transactions on medical imaging, 25(9), pp.1233-1245. Referenced in section: 2.5 (page 18).

209. Povey, D., Burget, L., Agarwal, M., Akyazi, P., Kai, F., Ghoshal, A., Glembek, O., Goel, N., Karafiát, M., Rastrow, A. and Rose, R.C., 2011. The subspace Gaussian mixture model—A structured model for speech recognition. Computer Speech and Language, 25(2), pp.404-439. Referenced in section: 2.5 (page 18).

210. Torres-Carrasquillo, P.A., Reynolds, D.A. and Deller, J.R., 2002, May. Language identification using Gaussian mixture model tokenization. In Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on (Vol. 1, pp. I-757). IEEE. Referenced in section: 2.5 (page 18).

211. Xuan, G., Zhang, W. and Chai, P., 2001. EM algorithms of Gaussian mixture model and hidden Markov model. In Image Processing, 2001. Proceedings. 2001 International Conference on (Vol. 1, pp. 145-148). IEEE. Referenced in section: 2.5 (page 18).

212. Hassoun, M.H., 1995. Fundamentals of artificial neural networks. MIT press. Referenced in section: 2.6.1 (page 19).

213. 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. Referenced in section: 2.6.1 (page 19).

214. McClelland, J. L., Rumelhart, D. E., & PDP Research Group. (1987). Parallel distributed processing (Vol. 2). Cambridge, MA:: MIT press. Referenced in section: 2.6.1 (page 19).

215. Wiatowski, T. and Bölcskei, H., 2017. A mathematical theory of deep convolutional neural networks for feature extraction. IEEE Transactions on Information Theory. Referenced in section: 2.6.1 (page 19).

216. Nesterov, Y. (2007). Gradient methods for minimizing composite objective function. Referenced in section: 2.6.1 (page 19).

217. Karlik, B., & Olgac, A. V. (2011). Performance analysis of various activation functions in generalized MLP architectures of neural networks. International Journal of Artificial Intelligence and Expert Systems, 1(4), 111-122. Referenced in section: 2.6.1 (page 19).

218. Baba, N. (1989). A new approach for finding the global minimum of error function of neural networks. Neural networks, 2(5), 367-373. Referenced in section: 2.6.1 (page 19).

219. Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In Neural networks for perception (pp. 65-93). Referenced in sections: 2.6.1 (page 19), 2.6.1 (page 19), 2.6.1 (page 19), 2.6.1 (page 24).

220. Datta, B. N. (2010). Numerical linear algebra and applications (Vol. 116). Siam. Referenced in section: 2.6.1 (page 19).

221. Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747. Referenced in section: 2.6.1 (page 19).

222. Pavelka, A., & Procházka, A. (2004). Algorithms for initialization of neural network weights. In In Proceedings of the 12th Annual Conference, MATLAB (pp. 453-459). Referenced in section: 2.6.1 (page 20).

223. Tetko, I. V., Livingstone, D. J., & Luik, A. I. (1995). Neural network studies. 1. Comparison of overfitting and overtraining. Journal of chemical information and computer sciences, 35(5), 826-833. Referenced in sections: 2.6.1 (page 20), 2.6.1 (page 21).

224. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), pp.1929-1958. Referenced in sections: 2.6.1 (page 20), 2.6.1 (page 21), 2.6.3 (page 33), 3.5.2 (page 55).

225. Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(Feb), 281-305. Referenced in sections: 2.6.1 (page 21), 2.6.1 (page 28), 3.5.1 (page 51).

226. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199. Referenced in section: 2.6.1 (page 21).

227. Manessi, F., & Rozza, A. (2018). Learning Combinations of Activation Functions. arXiv preprint arXiv:1801.09403. Referenced in section: 2.6.1 (page 24).

228. LeCun, Y., Touresky, D., Hinton, G., & Sejnowski, T. (1988, June). A theoretical framework for back-propagation. In Proceedings of the 1988 connectionist models summer school (pp. 21-28). CMU, Pittsburgh, Pa: Morgan Kaufmann. Referenced in section: 2.6.1 (page 24).

229. MacKay, D. J. (1992). A practical Bayesian framework for backpropagation networks. Neural computation, 4(3), 448-472. Referenced in section: 2.6.1 (page 24).

230. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation (No. ICS-8506). California Univ San Diego La Jolla Inst for Cognitive Science. Referenced in section: 2.6.1 (page 24).

231. Le Cun, Y. (1986). Learning process in an asymmetric threshold network. In Disordered systems and biological organization (pp. 233-240). Springer, Berlin, Heidelberg. Referenced in section: 2.6.1 (page 24).

232. Cun, Y. L. (1985). A learning scheme for asymmetric threshold networks. Proceedings of Cognitiva, 85, 599-604. Referenced in section: 2.6.1 (page 24).

233. Yann, L. (1987). Modèles connexionnistes de l'apprentissage (Doctoral dissertation, PhD thesis, These de Doctorat, Universite Paris 6). Referenced in section: 2.6.1 (page 24).

234. Parker, D. B., & Learning Logic, T. (1985). Report TR-47. MIT. Center for Computational Research in Economics and Management Science, MIT, Cambridge, MA. Referenced in section: 2.6.1 (page 24).

235. Werbos, P. (1974). Beyond regression: new fools for prediction and analysis in the behavioral sciences. PhD thesis, Harvard University. Referenced in section: 2.6.1 (page 24).

236. Hinton, G.E. and Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. science, 313(5786), pp.504-507. Referenced in section: 2.6.1 (page 25).

237. Povey, D., Zhang, X., & Khudanpur, S. (2014). Parallel training of DNNs with natural gradient and parameter averaging. arXiv preprint arXiv:1410.7455. Referenced in section: 2.6.1 (page 27).

238. McKay, M. D. (1992, December). Latin hypercube sampling as a tool in uncertainty analysis of computer models. In Proceedings of the 24th conference on Winter simulation (pp. 557-564). ACM. Referenced in section: 2.6.1 (page 29).

239. LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. R. (1998). Efficient backprop. In Neural networks: Tricks of the trade (pp. 9-50). Springer, Berlin, Heidelberg. Referenced in section: 2.6.1 (page 29).

240. Specht, D.F., 1990. Probabilistic neural networks. Neural networks, 3(1), pp.109-118. Referenced in section: 2.6.2 (page 30).

241. Schmidhuber, J., 2015. Deep learning in neural networks: An overview. Neural networks, 61, pp.85-117. Referenced in section: 2.6.2 (page 30).

242. Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105). Referenced in sections: 2.6.2 (page 31), 3.5.2 (page 55).

243. CIFAR-10, collection of images that are commonly used to train machine learning and computer vision algorithms, https://www.cs.toronto.edu/~kriz/cifar.html (access time 31-Jan-2018). Referenced in section: 2.6.2 (page 31).

244. Cheng, Y., Wang, D., Zhou, P., & Zhang, T. (2017). A Survey of Model Compression and Acceleration for Deep Neural Networks. arXiv preprint arXiv:1710.09282. Referenced in section: 2.6.3 (page 32).

245. Ioffe, S. and Szegedy, C., 2015, June. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). Referenced in sections: 2.6.3 (page 33), 3.5.1 (page 51), 3.5.2 (page 55).

246. Sola, J., & Sevilla, J. (1997). Importance of input data normalization for the application of neural networks to complex industrial problems. IEEE Transactions on nuclear science, 44(3), 1464-1468. Referenced in section: 2.6.3 (page 33).

247. Ng, A. Y. (2004, July). Feature selection, L 1 vs. L 2 regularization, and rotational invariance. In Proceedings of the twenty-first international conference on Machine learning (p. 78). ACM. Referenced in section: 2.6.3 (page 33).

248. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R.R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580. Referenced in section: 2.6.3 (page 33).

249. Oh, K. S., & Jung, K. (2004). GPU implementation of neural networks. Pattern Recognition, 37(6), 1311-1314. Referenced in section: 2.6.3 (page 33).

250. Omondi, A. R., & Rajapakse, J. C. (Eds.). (2006). FPGA implementations of neural networks (Vol. 365). New York, NY, USA:: Springer. Referenced in section: 2.6.3 (page 33).

251. Farabet, C., Martini, B., Akselrod, P., Talay, S., LeCun, Y., & Culurciello, E. (2010, May). Hardware accelerated convolutional neural networks for synthetic vision systems. In Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on (pp. 257-260). IEEE. Referenced in section: 2.6.3 (page 33).

252. Cotton, N. J., & Wilamowski, B. M. (2011). Compensation of nonlinearities using neural networks implemented on inexpensive microcontrollers. IEEE Transactions on Industrial Electronics, 58(3), 733-740. Referenced in section: 2.6.3 (page 33).

253. Patra, J. C., Kot, A. C., & Panda, G. (2000). An intelligent pressure sensor using neural networks. IEEE transactions on instrumentation and measurement, 49(4), 829-834. Referenced in section: 2.6.3 (page 33).

254. Potter, M. C., Wyble, B., Hagmann, C. E., & McCourt, E. S. (2014). Detecting meaning in RSVP at 13 ms per picture. Attention, Perception, & Psychophysics, 76(2), 270-279. Referenced in section: 2.7 (page 33).

255. Bastani, V., Helfroush, M.S. and Kasiri, K., 2010. Image compression based on spatial redundancy removal and image inpainting. Journal of Zhejiang University SCIENCE C, 11(2), pp.92-100. Referenced in section: 2.7 (page 33).

256. Wang, W., Stuijk, S. and De Haan, G., 2015. Exploiting spatial redundancy of image sensor for motion robust rPPG. IEEE transactions on Biomedical Engineering, 62(2), pp.415-425. Referenced in section: 2.7 (page 33).

257. Huynh-Thu, Q. and Ghanbari, M., 2008. Scope of validity of PSNR in image/video quality assessment. Electronics letters, 44(13), pp.800-801. Referenced in section: 2.7 (page 34).

258. Wang, Z., & Bovik, A. C. (2009). Mean squared error: Love it or leave it? A new look at signal fidelity measures. IEEE signal processing magazine, 26(1), 98-117. Referenced in section: 2.7 (page 34).

259. Wang, Z., Simoncelli, E. P., & Bovik, A. C. (2003, November). Multiscale structural similarity for image quality assessment. In Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on (Vol. 2, pp. 1398-1402). Ieee. Referenced in section: 2.7 (page 34).

260. Papadimitriou, C.H., 2003. Computational complexity (pp. 260-265). John Wiley and Sons Ltd. Referenced in section: 2.7 (page 35).

261. Quinn, M.J. and Quinn, M.J., 1994. Parallel computing: theory and practice (Vol. 2). New York: McGraw-Hill. Referenced in section: 2.7 (page 35).

262. Shah, S. (2014). Real-time image processing on low cost embedded computers. Techincal report No. UCB/EECS-2014–117. Referenced in section: 2.8 (page 36).

263. Jiang, H., & Su, X. (2013). The embedded image acquisition system based on the ARM. Journal of Convergence Information Technology, 8(9), 845. Referenced in section: 2.8 (page 36).

264. Requirements of Low Power VLSI Design and Analysis of Flip-flops: Sources of Power ConsumptionFeb 1, 2017. Referenced in section: 2.8 (page 37).

265. An Introduction to Classical Electromagnetic RadiationOct 30, 1997. Referenced in section: 2.8 (page 37).

266. Encyclopedia of Parallel Computing (Springer Reference)Sep 8, 2011. Referenced in section: 2.8 (page 37).

267. Olive, M. M., & Abel, M. G. (2003). Research methods: Quantitative and Qualitative approaches. Analia Manriquiz (2011), Citizen. Referenced in section: 3.1 (page 39).

268. Roelofs, G., & Koman, R. (1999). PNG: the definitive guide. O'Reilly & Associates, Inc.. Referenced in section: 3.2.2 (page 40).

269. Boutell, T. (1997). Png (portable network graphics) specification version 1.0. Referenced in section: 3.2.2 (page 40).

270. OpenJPEG, open-source JPEG 2000 codec, http://www.openjpeg.org/ (access time 20-Feb-2018), https://github.com/uclouvain/openjpeg (access time 20-Feb-2018). Referenced in sections: 3.2.3.3 (page 43), 4.5.1 (page 76).

271. The JPEG 2000 Suite Sep 21, 2009. Referenced in section: 3.2.3.3 (page 43).

272. Zhang, Q., & Li, B. (2015). Dictionary learning in visual computing. Synthesis Lectures on Image, Video, & Multimedia Processing, 8(2), 1-151. Referenced in section: 3.3.2 (page 46).

273. Xiong, Z., Liveris, A. D., & Cheng, S. (2004). Distributed source coding for sensor networks. IEEE signal processing magazine, 21(5), 80-94. Referenced in section: 3.3.4 (page 50).

274. Slowack, J., Skorupa, J., Deligiannis, N., Lambert, P., Munteanu, A., & Van de Walle, R. (2012). Distributed video coding with feedback channel constraints. IEEE Transactions on Circuits and Systems for Video Technology, 22(7), 1014-1026. Referenced in section: 3.3.4 (page 50).

275. Lennox, B., Montague, G. A., Frith, A. M., Gent, C., & Bevan, V. (2001). Industrial application of neural networks—an investigation. Journal of Process Control, 11(5), 497-507. Referenced in section: 3.5.1 (page 51).

276. Lakhmi J, Rao V (1999) Industrial applications of neural networks. CRC, Boca Raton, FL. Referenced in section: 3.5.1 (page 51).

277. Rovithakis, G. A., & Christodoulou, M. A. (2012). Adaptive control with recurrent high-order neural networks: theory and industrial applications. Springer Science & Business Media. Referenced in section: 3.5.1 (page 51).

278. Thwin, M. M. T., & Quah, T. S. (2005). Application of neural networks for software quality prediction using object-oriented metrics. Journal of systems and software, 76(2), 147-156. Referenced in section: 3.5.1 (page 51).

279. Chou, W. K., Yun, D. Y., & Tseng, C. C. (1993). A constant-time neural network for multiple selection of extreme values. J. Inf. Sci. Eng., 9(3), 445-459. Referenced in section: 3.5.1 (page 51).

280. Jung, S., & su Kim, S. (2007). Hardware implementation of a real-time neural network controller with a DSP and an FPGA for nonlinear systems. IEEE Transactions on Industrial Electronics, 54(1), 265-271. Referenced in section: 3.5.1 (page 51).

281. Botros, N. M., & Abdul-Aziz, M. (1994). Hardware implementation of an artificial neural network using field programmable gate arrays (FPGA's). IEEE Transactions on Industrial Electronics, 41(6), 665-667. Referenced in section: 3.5.1 (page 51).

282. Gorr, W. L., Nagin, D., & Szczypula, J. (1994). Comparative study of artificial neural network and statistical models for predicting student grade point averages. International Journal of Forecasting, 10(1), 17-34. Referenced in section: 3.5.1 (page 51).

283. Harezlak, K., Kasprowski, P., & Stasch, M. (2014). Towards accurate eye tracker calibration–methods and procedures. Procedia Computer Science, 35, 1073-1081. Referenced in section: 3.5.1 (page 52).

284. Torch framework, www.torch.ch (access time 9-Oct-2017). Referenced in sections: 3.5.2 (page 54), 3.5.3 (page 56).

285. Ng, A.Y., 2004. Feature selection, L1 vs. L2 regularization, and rotational invariance. Proceedings of the twenty-first international conference on Machine learning, ACM, p. 78. Referenced in section: 3.5.2 (page 54).

286. He, K., Zhang, X., Ren, S. and Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. Proceedings of the IEEE international conference on computer vision, pp. 1026-1034. Referenced in section: 3.5.2 (page 55).

287. Xu, B., Wang, N., Chen, T. and Li, M., 2015. Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853. Referenced in section: 3.5.2 (page 55).

288. Lin, H. W., Tegmark, M., & Rolnick, D. (2017). Why does deep and cheap learning work so well?. Journal of Statistical Physics, 168(6), 1223-1247. Referenced in section: 3.5.3 (page 58).

289. Jain, A. K. (2010). Data clustering: 50 years beyond K-means. Pattern recognition letters, 31(8), 651-666. Referenced in section: 3.5.3 (page 60).

290. Klir, G., & Yuan, B. (1995). Fuzzy sets and fuzzy logic (Vol. 4). New Jersey: Prentice hall. Referenced in section: 3.5.3 (page 60).

291. Hornof, A. J., & Halverson, T. (2002). Cleaning up systematic error in eye-tracking data by using required fixation locations. Behavior Research Methods, Instruments, & Computers, 34(4), 592-604. Referenced in section: 4.4.3 (page 69).

292. Kakadu, closed-source JPEG 2000 codec, http://kakadusoftware.com/ (access time 20-Feb-2018). Referenced in section: 4.5.1 (page 76).

293. JasPer, open-source JPEG 2000 codec, https://www.ece.uvic.ca/~frodo/jasper/ (access time 20-Feb-2018). Referenced in section: 4.5.1 (page 76).

294. JJ2000, open-source JPEG 2000 codec, http://www.dclunie.com/jj2000/JPEG%202000%20implementation%20in%20Java.html (access time 20-Feb-2018). Referenced in section: 4.5.1 (page 76).

295. x264, open-source H.264/MPEG-4 AVC codec, https://www.videolan.org/developers/x264.html (access time 20-Feb-2018). Referenced in section: 4.5.1 (page 76).

296. x265, open-source H.265/MPEG-H HEVC codec, https://www.videolan.org/developers/x265.html (access time 20-Feb-2018). Referenced in section: 4.5.1 (page 76).

297. Xvid, open-source MPEG-4 codec, https://www.xvid.com/ (access time 20-Feb-2018). Referenced in section: 4.5.1 (page 76).

298. Klapetek, P., Necas, D., & Anderson, C. (2004). Gwyddion user guide. Czech Metrology Institute, 2007, 2009. Referenced in section: 4.5.3 (page 78).

299. Mallat, S. G. (1989). Multifrequency channel decompositions of images and wavelet models. IEEE Transactions on Acoustics, Speech, and Signal Processing, 37(12), 2091-2110. Referenced in section: 4.5.3 (page 80).

300. Uytterhoeven, G., Van Wulpen, F., Jansen, M., Roose, D., & Bultheel, A. (1997). Waili: Wavelets with integer lifting. Referenced in section: 4.5.3 (page 80).

301. Goh, S.S., Jiang, Q. and Xia, T., 2000. Construction of biorthogonal multiwavelets using the lifting scheme. Referenced in section: 4.5.3 (page 80).

302. Guangjun, Z., Lizhi, C., & Huowang, C. (2001, October). A simple 9/7-tap wavelet filter based on lifting scheme. In Image Processing, 2001. Proceedings. 2001 International Conference on (Vol. 2, pp. 249-252). IEEE. Referenced in section: 4.5.3 (page 80).

303. Le Gall, D., & Tabatabai, A. (1988, April). Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques. In Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on (pp. 761-764). IEEE. Referenced in section: 4.5.3 (page 80).

304. Zandi, A., Allen, J. D., Schwartz, E. L., & Boliek, M. (1995, March). CREW: Compression with reversible embedded wavelets. In Data Compression Conference, 1995. DCC'95. Proceedings (pp. 212-221). IEEE. Referenced in section: 4.5.3 (page 80).

305. Schwartz, E. L., Zandi, A., & Boliek, M. P. (1995, August). Implementation of compression with reversible embedded wavelets. In Applications of Digital Image Processing XVIII (Vol. 2564, pp. 32-44). International Society for Optics and Photonics. Referenced in section: 4.5.3 (page 80).

306. JPEG2000 filters supported in JPEG2000 presentation, Ericsson Research, Media Lab, http://www.autex.spb.su/download/wavelet/jpeg2000/jpeg2000_christ2.pdf (access time 2-March-2018). Referenced in section: 4.5.3 (page 80).

307. Daubechies, I., 1992. Ten lectures on wavelets. Society for industrial and applied mathematics. Referenced in section: 4.5.3 (page 81).

308. Nguyen, T., Marpe, D., Schwarz, H., & Wiegand, T. (2011, September). Reduced-complexity entropy coding of transform coefficient levels using truncated golomb-rice codes in video compression. In Image Processing (ICIP), 2011 18th IEEE International Conference on (pp. 753-756). IEEE. Referenced in section: 4.6.2 (page 91).

309. Brailovsky, I., Kravtsunov, E., & Plotkin, D. (2004). A New Low-Complexity Entropy Coding Method. In 14th Intenational Conference of Computer Graphics and Vision. Moscow State University. Referenced in section: 4.6.2 (page 91).