Adjustment of visual information for visually impaired people

Filo, Nikola

UNIVERSITY OF ZAGREB
FACULTY OF GRAPHIC ARTS

NIKOLA FILO

ADJUSTMENT OF VISUAL INFORMATION FOR VISUALLY IMPAIRED PEOPLE

MASTER THESIS

Zagreb, 2014

Sveučilište u Zagrebu
Grafički fakultet

NIKOLA FILO

ADJUSTMENT OF VISUAL INFORMATION FOR VISUALLY IMPAIRED PEOPLE

MASTER THESIS

Mentor:
Prof. dr. sc., Lidija Mandić
Co-mentor:
Prof. Guillermo Botella Juan
Prof. Carlos Garcia Sanchez

Student:
Nikola Filo

Zagreb, 2014

Abstract

Visual impairment is a common eye disease that can occurs at any age. There are five main impairments but also a lot of different subtypes of impairments. Visual impairment could happen because some kind of ill in the eye, therefore retina is damaged and eye can't function properly. In the most cases, those impairments can be enhanced with some medical aids and nowadays scientists trying to develop medical software's that will help even better.

Technology is growing fast and it is possible to develop enhancement software that can test some low vision diseases. Software's can show how visual impaired people see which will be shown with foveal algorithm and also tested. With that software it is understandable how to design enhancement software's to enhance vision of visual impaired people. In this master thesis is developed edge detector similar to Canny edge detector which will filter image to binary image with edges. This thesis will also give a convolution algorithm which is primitive operation for image processing.

The aim of this work is to test algorithms for image enhancement with different values in the algorithm and to test those algorithms on the different accelerators that could be used for medical aids. The results of this research can show advantages and disadvantages of equipment that is used and also can indicate how to approach in the future work. This paper would provide general understanding of low vision and an overview of low vision enhancement approaches that could be still researched in the future.

Keywords

Visually impaired, retina, enhancement algorithm, edge detector, hardware implementation

CONTENTS

1. Introduction

1. 1. Vision sight and anatomy of the human eye

There are around 124 million people suffering from some form of low vision disability. The eye is the most important function of human body, which provides us over 90% of visual information from environment during the day. Other senses provide us the remaining 10% of information. The human eye is a steam organ that acts similar to cameras: the transparent front part of the eye refracts light rays projecting them into a less and inverted image on the photosensitive retina. Normal vision is shown in Figure 1. Retina has specialized nerve cells that perform conversion into electrical nerve impulses that allows conscious perception of light, vision which among other things allows us to distinguish colour and depth perception. The human eye has a viewing angle of 200˚ and can recognize 10 million different colors.

Every eye runs three pairs of different eye muscles: two pairs of flat and pair of oblique muscles. The eyeball is movable around all three axes and has three envelopes. The external form whitish - transparent sclera and transparent cornea. The sclera gives the eye a certain firmness and shape. Next to external eye muscles (the second of their grips on the walls of the eye socket) is flat and low pitched the connective ring around the optic nerve in the top of the eye socket, while the upper oblique muscle attaches to the upper wall of the eye socket bone).

Figure 1. Normal vision

The light entering the eye through the iris and the pupil, shown in Figure 2, its focused by the cornea and the crystalline lens onto the retina in the region of the macula, its most sensitive part being the fovea, which is the spot of the sharpest vision. The photon energy of the incoming light is converted by retina into electrical activity and is transferred to the optic disc then along the optic nerve to the brain. Fibres from fovea to the optic disc carry electric signal and it's called Henle fibres in the vicinity of the fovea. The eye gathers light and then transforms that light into visual information also known as „image“. The eye has lenses to focus the incoming light. It is similar to the camera focusing light onto the film to present a picture, in the same way the eye focuses light onto defined layer of cells, which is called retina, to produce an image. [1]

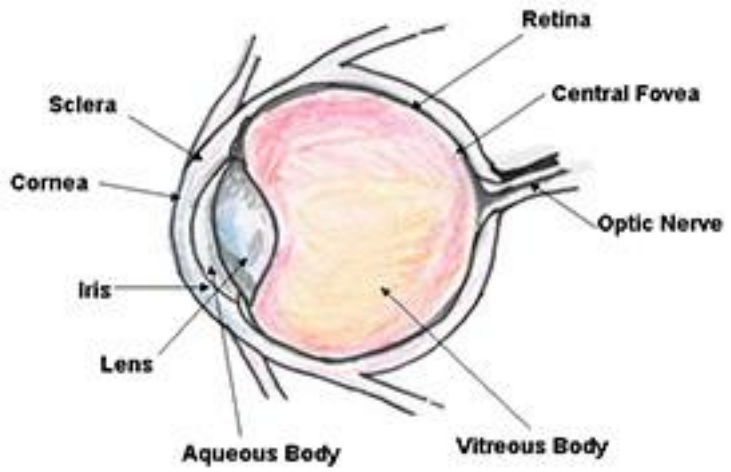

Figure 2. Anatomy of the human eye

2. 1. Retina

The retina is light-sensitive layer of tissue, which is located in the inner surface of the eye. The optics of the eye create an image of the visual world on the retina trough the cornea and lens, which serves much the same function as the film in a camera. When light comes to the retina initiates a cascade of chemical and electrical events that ultimately trigger nerve impulses. These impulses are sent to various visual centres of the brain trough the fibres of the optic nerve. In the embryonic development, the retina and the optic nerve appear as outgrowths of the developing

brain. The retina is considered part of the central nervous system and is a brain tissue. This is a layered structure with couple of layers of neurons that are connected by synapses. The photoreceptor cells are the only neurons that are directly sensitive to the light, the rods and cones. Rods function in dim light and provide black-white vision. Cones are in charge of daytime vision and the perception of colour. A third, rare type of photoreceptor is photosensitive ganglion cell, is important to reflexive responses to bright daylight.

Structure of the retina is divided into ten distinct layers. From closest to the front exterior of head towards the interior and back of the head:

1. Inner limiting membrane
2. Nerve fibre layer
3. Ganglion cell layer
4. Inner plexiform layer
5. Inner nuclear layer
6. Outer plexiform layer
7. Outer plexiform layer
8. Outer nuclear layer
9. External limiting membrane
10. Photoreceptor layer
11. Retinal pigment epithelium

These structure can be simplified into four main processing stages: photoreception, transmission to bipolar cells, transmission to ganglion cells which contain photoreceptors, the photosensitive ganglion cells and last one, transmission along the optic nerve. In every synaptic stage there are laterally connecting horizontal and amacrine cells. The optic nerve is a central tract of many axons of ganglion cells connecting primarily to the lateral geniculate body, a visual change station in the diencephalon (the rear of the forebrain). It also projects to the superior colliculus, the suprachiasmatic nucleus, and the nucleus of the optic tract. It passes through the other layers creating the Optic disk in primates.

Adult humans have the entire retina approximately 72% of a sphere and about 22 mm in diameter. The retina contains about 7 million cones and 75 to 150 million rods. The optic disk or “the blind spot” because of it lacks photoreceptors is located at the optic papilla, zone where the optic nerve fibres leave the eye. That appears as an oval white area of 3 mm2. Next to the optic disk is the macula. In his centre is the fovea, which is in charge of sharp central vision and is less sensitive to light because of its lack of rods. [18]

The retina is around 0.5 mm thick and has three layers of nerve cells and two of synapses, including the unique ribbon synapses. The optic nerve provides the ganglion cell axons to the brain and the blood vessels that open into the retina. The ganglion cells are in the retina while the photoreceptive cells outside. Light must first pass around the ganglion cells and trough the thickness of the retina before reaching the rods and cones. The white blood cells in front of the photoreceptors in the capillaries can be perceived as bright moving dots when looking into blue light. This is known as Scheerer’s phenomenon.

The retina illustrates the information stream in different layers, which effect to different cells. The retina includes horizontal and vertical pathways. In horizontal pathway, horizontal cells and amacrine cells can connect and disconnect the horizontal connection among cells to adjust range of receptive field and intensity of response. The horizontal cells and amacrine cells have many subtypes. Amacrine cells have for example at least 22 different types, their functions are also motion detection, brightness adjustment and texture recognition. In the vertical pathway, the optical signal is tested and converted to electronic signal trough the photoreceptor layer, after is passed to the bipolar cell and further transmitted to the ganglion cell layer in which each layer contains a different subtypes and handle with different information. The retina is also divided into another two categories: on-pathway and off-pathway, which deal with brightness, increase and decrease. [2]

Retina has several layers, photoreceptor layer, horizontal cells, bipolar cells, amacrine cells and ganglion cells.

Photoreceptor layer test optical signals and converts the optical information into electronic signals. Feature of the photoreceptor layer is the unequally distribution for

instance high density in central area and low density in peripheral area. This distribution allow centre of the retina to see details and for peripheral area to see contour information and be able to get to know surroundings. This distribution will not recognize visual accuracy, because muscles of an eyeball are flexible and powerful and they focus on an interesting subject. This characteristic gives the retina a quick look to the interesting information, while reduces processing workload. This example shows how retina can balance between accuracy and real time. Photoreceptor layer have a complex process, there are three steps. First, allocation of various photoreceptors is generated. Secondly, according to the input RGB visual information, wavelength is calculated. Wavelength is the main reason that photoreceptors discharge. In the end, strength of photoreceptors is calculated based on photoreceptor-wavelength sensitivity curve.

Horizontal cells and bipolar cells are related to each other. Photoreceptor directly transfer centre signals to bipolar cells, while horizontal cells collect signals in a large range and get feedback to bipolar to create surrounding. For this field, centre-surrounding, R. W. Rodieck [1965] raised “Difference of Gaussian” (DOG) model. Model can simulate the centre-surrounding receptive field. Model is based on two-dimensional DOG. Experiments show that when given light covers the certain range of a ganglion cell receptive field, the ganglion cell may have weak response, because at the time, input from photoreceptor and inhibitory input from horizontal are equal. The integral of DOG function in continuous region can meet requirements that the central reaction equals to surrounding reaction. In isolated case, each layer cells are generated according to the real physiological data, it is difficult to provide that central response always equals to surrounding response because of random influence. A new weight assignment method for each cell to make the central response and the surrounding response equal to each other is proposed. The weight setting mechanism should have three basic requirements:

- The weight of cell should be same to total number of cells in the region that is concerned. The more the total amount of cells is, the smaller the weight of each cell is.
- The weight of cell has to same as location of cells.

- The weight settings mechanism can always provide that the central area equals to the surrounding area.

Amacrine cells have a significant task in visual signals adjustment. Dark vision, motion detection or non-classical sensitive field formation are fundamental to amacrine cells. AII amacrine cells have largest number and are considered as the most significant type in all amacrine types. There are three main input sources to AII. First of the input is from rod bipolar which accounts for 30% of the all input. Second major input, which accounts for about 20% of input, is from gap hub between cone bipolar cells and amacrine cells. The gap hub is electrical synaptic connections through which cone bipolar and AII cells can swap information. The rest of input information's occur from electrical synapses among AII connections. The more the AII cell connects to each other, the more ganglion cell receptive field will cover. After a different processing in the horizontal level and vertical level, visual information in the end is hand over to the ganglion cell layer. As the final layer in retina, ganglion cell is responsible for gathering all information and transferring information to brain. When test optical signal, there are more than 100 million photoreceptor cells included, but when transferring information to the brain, there are only 1 million ganglion output, in which the productivity of visual information gathering, representation affect, accuracy and real-time. There are four types of ganglion cells that allow on and off taxonomy and P and M taxonomy. P type and M type of ganglion cells are based on morphology. All P type are smaller and have smaller sensitive field than M type ganglion cells in the retina. The on-center P type and on-center M type cells all receive OnConeBipolar output, which means they may have same pretending function, but their receptive field radius is way different. The receptive field radius is set according to physiological data. [2].

3. 1. Low vision diseases

There are many different low vision diseases and damages on the eye. The main preoccupation in this paper will be Macular Degeneration, which will be described in later section because of many characteristics. Here will be described other, also very common diseases. Cataract is an eye disease when normal clear lens converts to cloudy or opaque and causes a decrease in vision in general Figure 3. The lens is focused on the back of the eye, so images appear normal and clear. Clouding effect of this lens is because of cataract disorder. Cataract is a disease that appears during the years, and is common among elderly people, but sometimes it can effect earlier and can be rapid. Sometimes both eyes are affected but that is not always the case. The cause of cataract is not known but some characteristics of this disease are known. Cataract appears when changes are made in the protein structures within the lens that occur over many years. Sometimes it is presented at birth or in early childhood as a result of genes from parents. Serious cause of cataract also can be eye trauma, surgery of the eye or intraocular inflammation. These causes can very rapidly develop this eye disorder. Other effects that can also have negative influence to eye can be ultraviolet light exposure, diabetes, smoking or use of specific medications.

Figure 3. Cataract vision

Glaucoma is a disease of the major nerve of vision. Major nerve is also known as optic nerve. The optic nerve is transmitting light generated nerve impulses from retina to the brain. The brain recognizes those electrical signals as vision. This disease is characterized as progressive damage to optic nerves that are in charge of peripheral vision, as shown in Figure 4. If glaucoma is not treated it can proceed to loss of central vision or blindness. Glaucoma is connected with elevated pressure in the eye but not always. In general, this is raised eye pressure that leads to vision loss. Sometimes glaucoma can appear at normal eye pressure but this could happen because of poor regulation of blood flow into the optic nerve.

Figure 4. Claucoma vision

A retinal detachment is a disease that has separation of the retina from its attachments to the underlying tissue within the eye. In most cases this is the result of retinal breaks, hole or tear. Those breaks may occur when the vitreous gel pulls loose or separates from its attachment to the retina, this usually happens on the peripheral parts of the retina. Two-thirds of the eye is filled with vitreous, which is a clear gel and occupies the space in front of the retina. As the vitreous gel pulls loose, it will sometimes exert traction on the retina, and if the retina is weak, she will tear. Most of the breaks on the retina are not result of injury. Those tears are sometimes accompanied by bleeding if a retinal blood vessel is included in the tears. Lots of people develop separation of the vitreous from the retina as they get older but just a small percentage of separations result in retinal tears. Below is shown how people with this disease see Figure 5.

Figure 5. Retinal detachment vision

Diabetic retinopathy is shown in Figure 6, is a disease that damages retina and it is caused by diabetes, which damages the tiny blood vessels in the back of the eye. In early stages of disease there are no visual symptoms. When the disease progresses it is proliferative stage. New blood vessels, which are fragile, grow along the retina and inside of the eye, gel like vitreous that fills the inside of the eye. If this form is not treated well, those blood vessels will bleed, blurry vision will be performed and it will destroy the retina. Every patient with diabetes type one or two has risk of diabetic retinopathy. Swelling in the portion of the eye called macular edema, it is a part of the eye most sensitive to light, and it makes it hard to do things like read or drive. When new blood vessels form at the back of the eye, they bleed and blur vision. That type of bleeds have tendency to happen often during the sleep. There are no warning signs for diabetic retinopathy. Diagnosis of this disease can be informed during the normal eye exam, which includes visual acuity test, ophthalmoscopy, tonometry, and pupil examination. Treatment for this disease is laser surgery, which stop the edema, and bleeds in the eye. Vitrectomy is another treatment, which removes blood from the back of the eye. For patients with diabetes it is recommended to have eye examinations at least once a year. [3]

Figure 6. Diabetic retinopathy vision

4. 1. OpenCL

In this research OpenCL programming language was used, it is used to implement our algorithms to four different accelerators for that reason in the next section OpenCL programming language is described. OpenCL is free and public standard that everyone can download, all the development tools that programmer may need. This is language standard with which programmers can write applications that will be running on the some sort of hardware. For example, CPU, GPU, etc. Programmers can run the code and don't have concern which company designed the processor or how many cores it contains. The code will compile and execute on Intel's Core processors, Nvidia's Fermi processors, IMB's Cell Broadband Engine or even AMD's Fusion processors. All of these processors have same function in personal computer but not everyone is programmed in the same way. Each of these devices has its own instruction set, and before OpenCL, programmer was needed to know three or more different languages to be able to program those devices. Therefore Apple Inc. which uses different devices from different vendors in their own products and their programmers need to know multiple languages. For easier programming those devices they starting to develop new language standard called OpenCL. In June 2008 Apple and other companies formed OpenCL Working Group in Khronos Group, a consortium of companies whose aim is to advance graphic and graphical media, which had task to develop new language standard. First version called OpenCL 1.0 is released in November 2008. [4]

OpenCL is not so new or different, it is based on C/C++, but improvements are defined like a set of data types, data structures and functions. It also has three main advantages comparison with C or C++ languages. It has portability which is probably main advantage of this language, OpenCL has suitable motto for this, “Write once, run on anything” what will say every vendor that provides hardware it will also provide the tool that compile OpenCL code to run on the hardware. In general this means that programmers can write the code and run it on any compliant device, no matter what device is this, multicore processor or graphic card. This is big advantage over regular high-performance computing, where programmer need to know vendor-specific language to program this specific hardware. This could target multiple devices at once and those devices don’t have to be from same vendor or have to have same architecture, just have to be OpenCL - compliant and everything works perfectly. Those functions are not possible to run in C/C++ programming, where is only available to target one device at the time. For example, if user uses multicore processor from AMD and graphics card from Nvidia, normally programmer wouldn’t be able to target both systems at once because each of those requires a separate compiler and linker, but with OpenCL program programmer can develop executable code for both devices. In that case programmer can unify hardware to perform certain task only with single program and if he wants to add more devices, he doesn’t have to rewrite the code, just need to rebuild the program.

Standardized vector processing is another advantage of OpenCL, the term vector in this case is used in one of three different ways. Physical or geometric vector are used in graphics to identify directions. Mathematical vector are specified as two-dimensional collections of elements, called a matrix. Computational vector is data structure that has multiple elements of same data type. Computational vectors are very important to OpenCL because high-performance processors can operate on multiple values at once. Every processor nowadays are able to process vectors, but C/C++ don’t define basic vector data types because vector instructions are usually vendor-specific. NVidia devices use PTX instructions, IBM devices require AltiVec

instructions and Intel processors use SSE extensions to process vectors and those instructions don’t have anything in common. OpenCL code can unite vector habits and run them on compliant processor. When applications are complied, Nvidia’s compiler will produce PTX instructions, IMB compiler will produce AltiVec instructions and so on. With this kind of programming in OpenCL, high - performance applications will be available on multiple platforms.

Parallel programming is the third big advantage in OpenCL. This advantage is used when programmers compute tasks to multiple processing elements to be performed at the same time. In OpenCL language those tasks are called kernels. Each kernel is specially coded function that will execute one or more compliant devices. Those kernels are sent to device or devices by host application, which is basic C/C++ application that runs on the user development system. For example, host sends kernels to a single device, like the GPU on the computer’s graphic card, but also the same CPU on which the host application is running can execute them. To manage connected devices, host applications use container called context which is shown in Figure 7., this shows how host interact with kernel and device. In general this works pretty easy, from kernel container called a program, host selects a function to create a kernel, and then he associates the kernel with argument data and sends it to a command queue. This is the mechanism through which the host tells devices what to do. Once the kernel is enqueued, the device can execute this particular function. In this way applications can configure different devices to execute different tasks and every task can operate on different data.

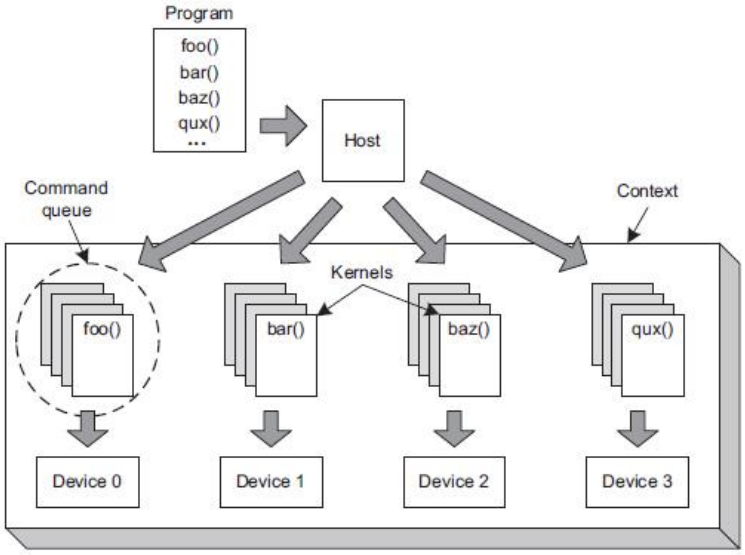

Figure 7. Parallel programming [5]

OpenCL is developed by Apple Inc., which has trademark rights and have initial collaboration with AMD, IBM, Qualcomm, Intel and Nvidia. In June 2008 Apple submitted initial proposal to the Khronos Group when is formed Khronos Compute Working Group with representatives from CPU, GPU, embedded-processor and software companies. The group worked for five months to finish the technical details for OpenCL 1.0. This version of OpenCL is released for public in December 2008. OpenCL 1.0 was released with Mac OS X Snow Leopard, which further extends support for hardware with OpenCL. This software lets any application to use of GPU computing power previously available only to graphics applications. OpenCL is based on the C programming language and it is proposed as an open standard. Later on AMD have decided to support OpenCL instead of developing own language. Nvidia announced full support for the OpenCL 1.0 specification to its GPU Computing Toolkit. In October 2009, IBM released its first OpenCL implementation. [5]
After support of other companies OpenCL 1.1 is developed in June 2010, which contains significant functionality for enhanced parallel programming flexibility, functionality and performance. In this version is included, new data types including three-component vectors and additional image formats, handling commands from multiple host threads and processing buffers across multiple devices, operations on regions of a buffer including, read, write and copy of 1D, 2D or 3D rectangular

regions, enhanced use of events to drive and control command execution. Additional OpenCL built-in C functions such as integer clamp, shuffle and asynchronous strided copies also is improved efficient sharing of images and buffers by linking OpenCL and OpenGL events. In November 2011 the Khronos Group has released Open CL 1.2 specification, which added significant functionality over previous versions in way of performance and features for parallel programming. Features that are included are device partitioning, separate compilation and linking of objects, enhanced image support, built-in kernels which means custom devices that contain specific unique functionality are now integrated more closely into the OpenCL framework. Included features are also DirectX functionality, which allows DX9 media surface sharing for efficient sharing between OpenCL and DX9 or DXVA media surfaces. The Khronos Group announced and release OpenCL 2.0 specification in November 2013. OpenCL 2.0 include updates and additions like shared virtual memory, nested parallelism, generic address space, C11 atomics, pipes, Android Installable Client Driver Extensions, industry support etc.

4. 1. 1. OpenCL specifications

OpenCl specifications are defined in four main parts, called models. First model is Platform model, which specifies how many of processors (the host) are capable of executing OpenCL code (the device). This model defines how are OpenCL functions (called kernels) executed on the devices. Second model is Execution model, which defines environment, which is configured on the host and how kernels are executed on the specific device. This model includes settings that OpenCL provides for host-device interaction. Third model called Memory model, defines memory hierarchy that kernels use regardless of the actual memory architecture. Fourth, Programming model defines how the assembled model is mapped to physical hardware.

4. 1. 2. Platform model

Platform model uses a single host that coordinates execution on devices. Those platforms are specific for implementations on different vendors of the OpenCL application programming interface (API). Thus, devices are targeted to vendors that can interact with them. When programmers are writing an OpenCL code, in platform model they present an abstract device architecture. The platform model defines a

device as an array of compute units where every compute unit is functionally independent from the rest because of scalability of this model. Those compute units are divided into processing element which is shown in Figure 8.

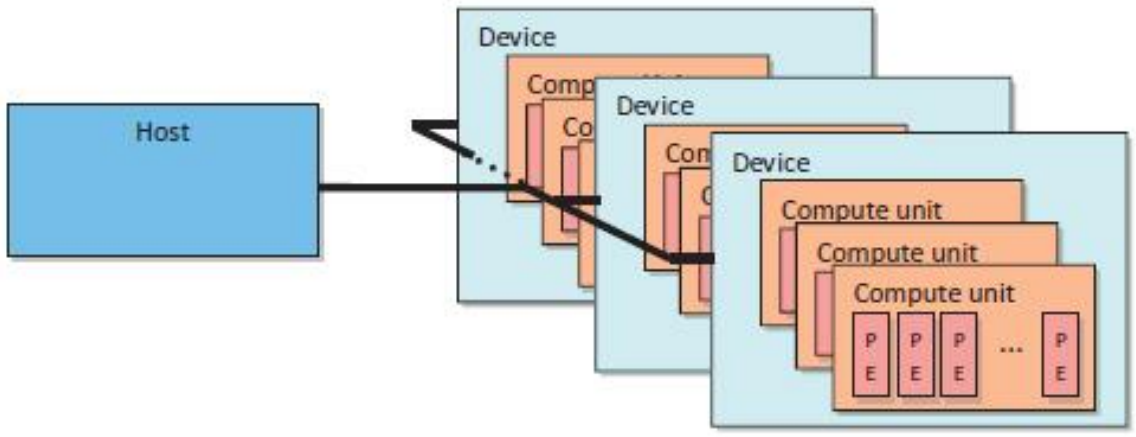

Figure 8. Platform model [4]

4. 1. 3. Execution model

This model has environment that configure the context on the host. Command and data has to be configured to the kernel that is going to be executed on a device. In this language, a context is an container that exists on the host, it coordinates how host-device interaction communicate, manage the memory object that is available to the device, also keep track of the program and kernel that is created for device. In the context, has to be used properties argument to restrict the environment of the context. This may provide a specific platform or enable graphics operations or even enable other parameters in the future. If context is limited, then programmer can use context for multiple platforms and use system that contains several vendors. When creating a context OpenCL allows user callbacks that have to be provided so that can be used to report error information. In this language there is a function that creates a context that automatically includes all devices like CPU, GPU, etc., after creating a context there is a function that present number of devices and structure of devices. All those functions and code writing to set context is very tedious, but after programmer write these steps once, he can use it for almost every project.

4. 1. 4. Memory model

Generally, memory systems are different between computing platforms. Nowadays all CPUs support automatic caching, but many GPUs do not. To support code portability, OpenCL defines summary memory model so that programmers can target when writing code and vendors can map to memory hardware. Those memory spaces are shown in Figure 9 below.

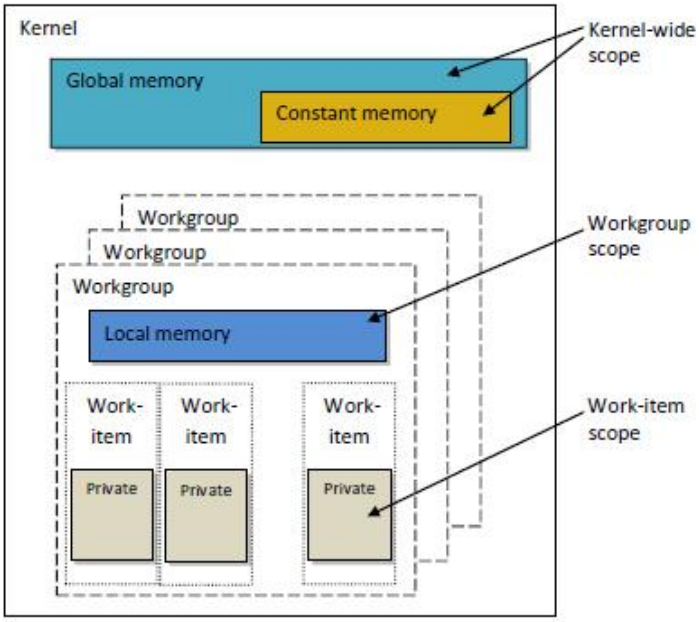

Figure 9. Memory spaces [4]

In OpenCL programs these memory spaces are relevant. Each of those spaces has associated keywords and it is used to specify where a variable should be created or where is the place for certain data. Memory model is divided into four levels of memory. Global memory is like main memory on a CPU-based host, it is visible to all compute units on the device. Every time when data is transferred from host to the device, the data will take place in global memory and conversely. Constant memory is designed for read-only data type but also for data where access to elements is simultaneously by all work-items. If values never change, then those types also get into this category. This model is modelled as part of global memory. When memory object is transferred to global memory can be specified as constant. Local memory is memory which address space is unique for compute device. Local memory is commonly implemented as on-chip memory. It is modelled as being shared by a

workgroup. These memories have much shorter latency and much higher bandwidth than global memory. Private memory is private as the name says, it is unique to an individual work-item. Local variables and nonpointer kernel arguments are private by default. These variables are usually mapped to registers but private arrays and spilled registers are mapped to an off-chip memory. The memory spaces in this language are similar to the models of GPUs nowadays. Detailed relationship between OpenCL memory spaces and AMD 6970 GPU is shown in figure 10. [4]

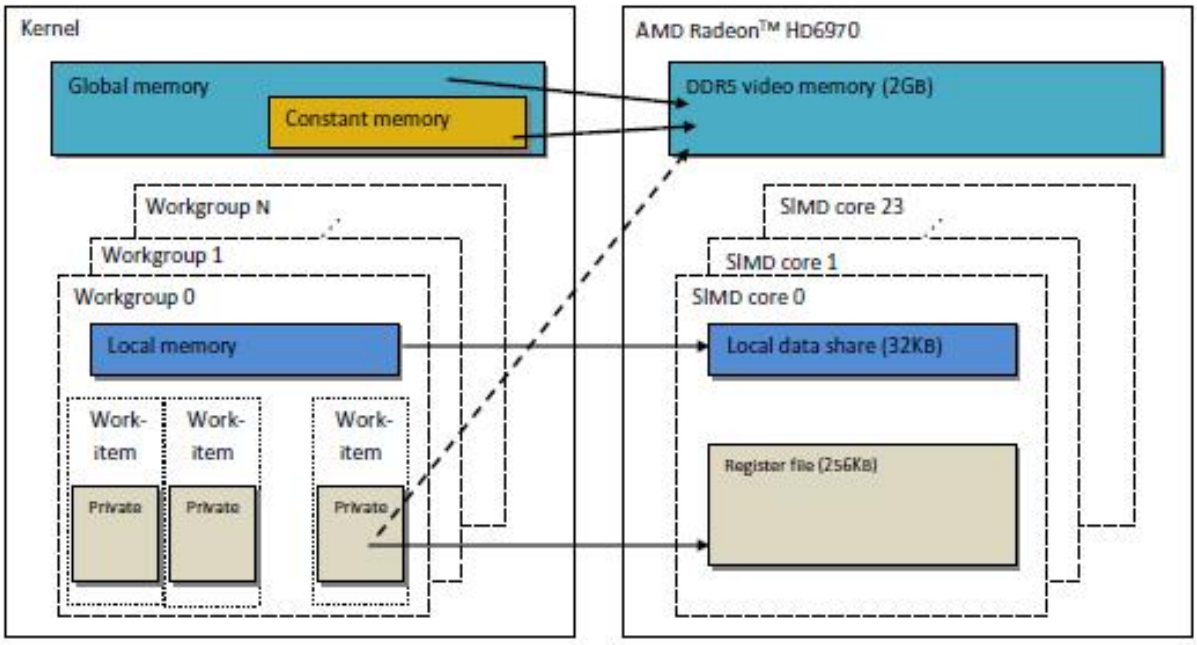

Figure 10. Comparison between OpenCL memory spaces and AMD 6970 GPU [4]

OpenCL is very powerful toolset for writing parallel programs, which can be run on high-performance processors. If programmer uses OpenCL, he doesn't have to know any other programming language because of OpenCL-compliant hardware. Programmers can write the code once and run it on any compliant hardware. For high-performance computing, portability and parallel programming are big advantages which OpenCL language has.

2. State of the art

1. 2. Enhancement algorithms

During the years, many researchers tried to develop some enhancement algorithms for visually impaired people to improve their sight. Further below several of enhancements are described. Every algorithm have same idea, this is detecting edges in the grayscale image. If algorithm read and process coloured image, first task is to convert that image into grayscale colours, then he extracting edges from the image, because edges are relevant information's in the image. In our work, we developed our algorithm for edge detection which is similar to already developed one called „Canny edge detector". Visual impairment is wide range disease for that reason there is no general algorithm for image enhancement. Every form of visual impairment have its own disadvantages because of that, there is a lot of different algorithms for different forms of diseases. All of them have their own purpose for some form of low vision disease. They are described here to better understand our developed algorithm.

1. 2. 1. Canny edge detector

Edge detection is a set of mathematical methods which are finding points in a digital image where the image brightness changes sharply. The points at which brightness changes sharply are organized into a set of curved line segments called edges. This tool is fundamental in image processing and computer vision. In 1986. John F. Canny developed multi-stage algorithm to detect a wide range of edges in images called Canny edge detector. Canny's edge detector has four steps.

- Applying the Gaussian blur to clear the noise from the raw image. Gaussian filter is an 5x5 matrice which have constant $\sigma = 1.4$. After applying Gaussian filter image is slightly blurred version of the original image without single noise.

$$\mathrm{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 5 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \times A \quad (1)$$

• Finding the intensity gradient of the image with four filters to detect horizontal, vertical and diagonal edges in the blurred image.

$$\mathrm{G} = \sqrt{G_{x^2} + G_{y^2}} \quad (2)$$

$$\theta = \operatorname{atan} 2\,(G_y, G_x) \quad (3)$$

• Non-maximum suppression determines if the gradient magnitude assumes a local maximum in the gradient direction. The algorithm categorizes the continuous gradient directions into a small set of discrete directions, and then moves a 3x3 filter over the output of the previous step. Every pixel suppresses the edge strength of the center pixel if its magnitude is not greater than the magnitude of the two neighbors in the gradient direction.

• Thresholding with hysteresis requires high and low thresholds. First is high threshold applied, he marks out the edges which are genuine. From that edges can be traced through the image. While tracing an edge, lower threshold is applied he allow to trace faint sections of edge as long as he find starting point. When this process is done, algorithm give binary image where each pixel is marked as edge pixel or a non-edge pixel. [19]

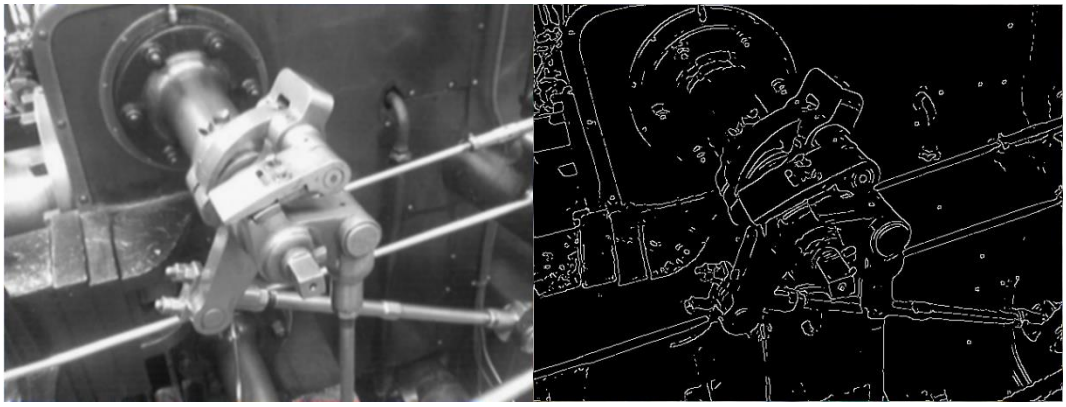

Figure 11. Original image (left side) and filtered through "Canny edge detector" (right side)

1. 2. 2. Sobel operator

Another edge detection algorithm called „Sobel operator“ or sometimes called „Sobel Filter“ also extract edges from the image. Irwin Sobel used „Isotropic 3x3 Image Gradient Operator“ which is convolved with original image to calculate approximations of the derivatives.

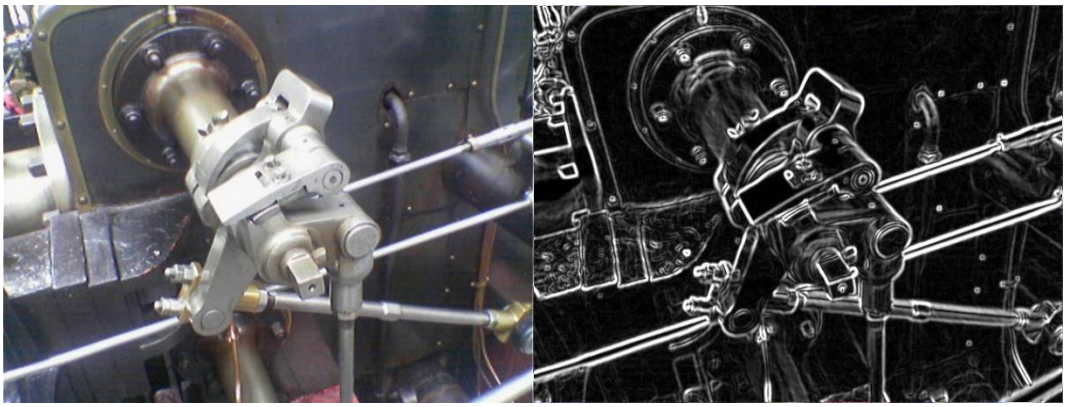

Figure 12. Original image (left side) and filtered through Sobel operator (right side)

1. 2. 3. TRON algorithm

Tinted Reduced Outlined Nature (TRON) is an algorithm that creates an edge-like image and increases the contrast between objects. By this algorithm contrast is increased by highlighting the edges of the moving objects and the edges between objects. Algorithm is performed in three main steps:

- Simplification of the scene, using anisotropic filtering
- Extraction of the significant spatial derivatives, using a hierarchy method
- Boosting the original scene using the simplified spatial derivatives.

Important step before performing edge extraction using spatial derivatives is to simplificate the image and not to extract high frequency noise and textures. For this purpose is used non-linear anisotropic smoothing technique which will eliminate noise and low importance textures and will avoid smoothing object boundaries. This is common process, which smooth the image and maintain the significant edges. When the image is simplificated, next step is to obtain gradient map. It is used algorithm described by Fleck [7]. This algorithm is based on modified Canny filter [8]. Algorithm use simple masks and compute the first derivative in four main directions, horizontal, vertical and two diagonal. The final stage in this algorithm is to rescale the original image thus to a weighting function based on the gradient map in the previous stage. This gradient map is normalized in dynamic range between 0:1 after which is defined threshold value, which will raise all pixels to defined threshold. To get TRON image (Figure 10(c)), the original image (Figure 10(a)) is multiplied with weighting function. In comparison with basic edge detection (Figure 10(b)), advantage of this method is to get more robust against noise and textures, he also keeps some of the chromatic information of the visual scene, with threshold in this algorithm is possible to increase or decrease the color information. [6]

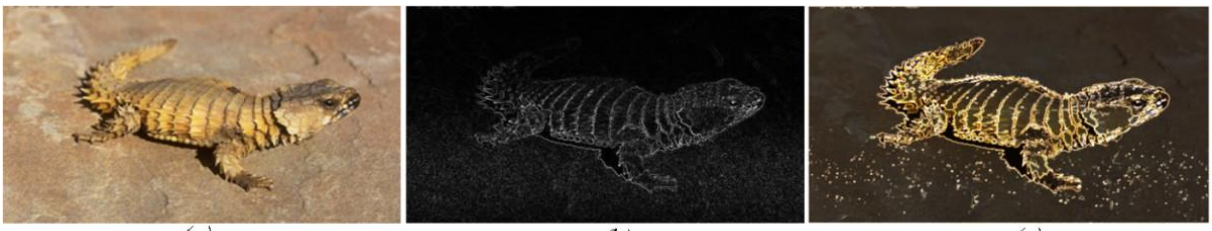

Figure 13. TRON enhancement

1. 2. 4. Cartoonization

Cartoonization algorithm is a technique that creates stylized images. The viewer can recognize the shapes and see them as shadows and texture details. This algorithm is presented by Walid Al-Atabany [6], and main advantage of this technique is that improve the contrast of visually important characteristics using simplification and reducing contrast in low contrast fields, also increasing contrast in high contrast fields. Their version of that algorithm has four main steps:

- Simplification of the image with anisotropic filtering
- Calculating the spatial derivatives of the image
- Quantization of the colours of the simplified image to create cartoon like images
- Combining the quantized image with the negative of the gradient map

The algorithm starts with smoothing the image using the anisotropic diffusion described by Perona and Malik [1991]. The anisotropic diffusion is used for image by converting it to the “YCbCr” colour space. The “Y” channel is for the intensity, that layer is diffused. After diffusion, image is converted back to RGB colour space. The gradient image calculated as given in equations and it is normalized between 0 and 1. Then are defined two threshold values and they set all pixels of normalized gradient image below minimum threshold to 0 and all the pixels above the maximum threshold are set to 1. To make paint-like image effect on the original image they restrict the luminance Y channel of the original image into bins. The full description of this method is previously described by Winnemoller [9]. They increase the visual definition of high contrast fields in the image by combining the negative of the corresponding extracted spatial derivatives. This negative gradient map coat gives a significant edge enhancement as shown in Figure 12(a). Figure 12(b) shows the cartoonized image without colour quantization and Figure 12(c) shows the Cartoonization with colour quantization effect.

Figure 14. Cartoonization enhancement

1. 2. 5. Edge overlaying algorithm

Edge overlaying algorithm also described by Walid Al-Atabany use the same mathematics as Cartoonization enhancement. In this algorithm they recolor and

overlay gradient map onto original image or a simplified interpretation of an original image. In such way contrast should be improved compared to the original. This algorithm is compared with one described by Wolffsohm [8] which he tested on patients while they watching television. Only difference is that Wolffsohn extracted the contour map with and without Gaussian smoothing. With smoothing, the image is little bit blurred compared to anisotropic simplification and without, results in the highlighting of many unwanted gradients as is shown in Figure 13(b-d). In addition to Wolffsohn, developed algorithm use 3 x 3 kernel, which make it hard to highlight the fundamental contours. In Al-Atabany algorithm they apply a simplification processing step to extract only the fundamental spatial derivatives. They use a pyramidal approach to get the spatial derivatives across a range of spatial frequencies. In Figure 13(e-f) are shown results of the edge overlaying on the original image without smoothing and with Gaussian smoothing. Figure 13(g-h) shows the overlaying on the original and cartoon images when using anisotropic diffusion filter. [6]

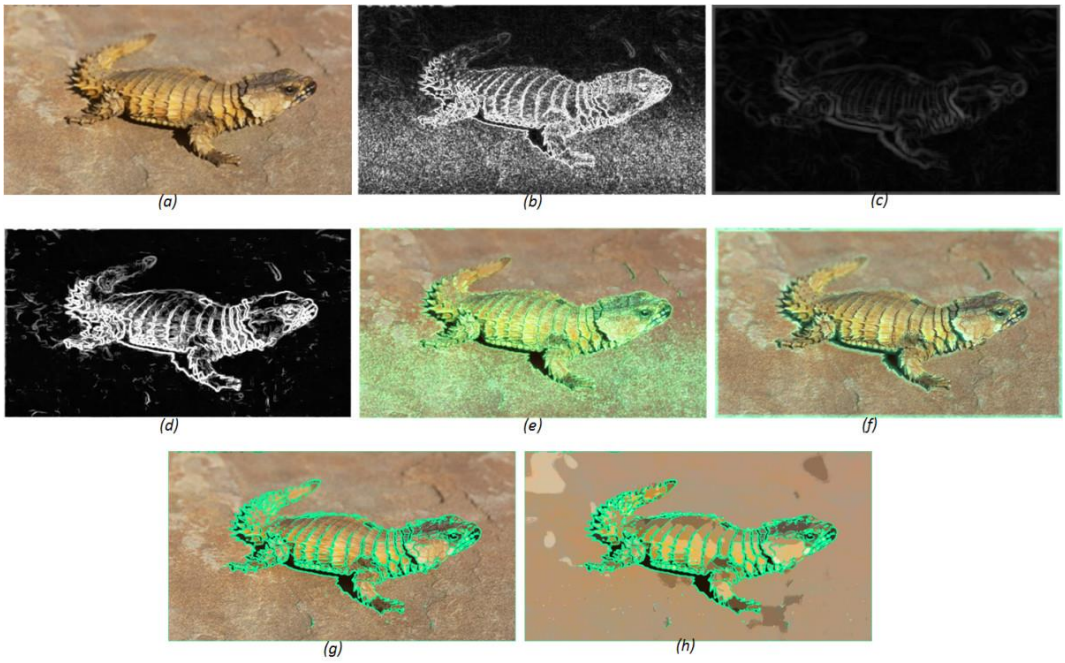

Figure 15. Edge overlaying enhancement

2. 2. Macular degeneration

Main preoccupation in this thesis is macular degeneration, which is painless eye condition in which people have loss of central vision which is shown in Figure 14

because the retina is weakened and does not function adequately. The retina is located in the back of the eye and is the light sensitive tissue, like the film in camera. The retina records the images we see and sends them through optic nerves from eye to the brain. It converts light images into electrical impulses through a chemical reaction. Those signals go to the brain, and then we interpret visual information and relate them to the rest of environment. From the other side, we have macula, which is a small area of the retina located in the central portion. The macula is in charge of central vision and provides the point of focus. The macula provides us vision of fine detail in direct line of sight. Because of macula we have clear vision that allows us to read, drive a bike and recognize colours or even faces. The non-macular areas of the retina provide us side vision and good night vision.

Figure 16. Macular degeneration vision

The most common type of macular degeneration is age-related macular degeneration although there are many types of it. Macular degeneration is related with aging. With time it destroys sharp central vision that we need to see clearly and that we need to do normal daily tasks. In most cases macular degeneration progresses slowly and people notice little change in vision. In other cases, the disease progresses faster and some people may have loss of vision in both eyes. Macular degeneration or better known as age-related macular degeneration (AMD) is more often among older people, 60 years of age and older. AMD usually affects both eyes, although this is not always the case. [10]

2. 2. 1. „Wet“ and „dry“ form of macular degeneration

Age related macular degeneration has two forms shown in figure 3. „Wet“ AMD is aggressive type, it is less common and affects faster. „Dry“ AMD is more often but progress of vision loss is slower. Wet age related macular degeneration is when abnormal blood vessels grow from the choroid under and into the macular area of the retina. These blood vessels are very fragile and often they leak blood and fluid. Those fluids raise position of the macula from normal place and interfere with the retina's main function which causes blur or loss of central vision. When that happens, vision loss may be very fast and severe. Wet macular degeneration does not have stages like dry AMD. The wet form leads to more vision loss than the dry form. People who have dry form of AMD are probably at risk of the disease progressing to the wet form. The dry form can advance and cause vision loss without turning into the wet form. Nowadays there is no way to predict if or when the dry form will progress into wet form. [16]

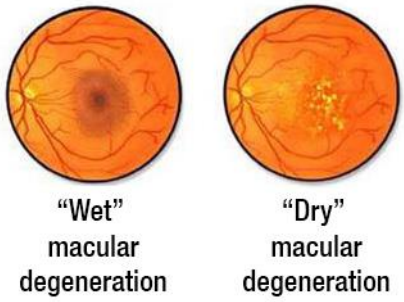

Figure 17. Difference between "Wet" and "Dry" AMD

Age related macular degeneration in dry form has infected the light sensitive cells, which slowly break down. When the macula is less functional central vision diminishes. AMD in dry form is more often only in one eye at the beginning, later it can also affect the other eye. Dry AMD has three stages, which are called early, intermediate and advanced. Every of these stages can occur in one or both eyes. Those with early stage of AMD have several small drusen or a few medium sized drusen and at this point, there are no symptoms and no vision loss. In intermediate stage people have medium sized drusen or larger drusens. In this stage people see a blurred spot in central vision, they need more light for reading and other daily tasks.

Advance stage of AMD is worst of all because of breakdown of light sensitive cells and supporting tissue in retinal area. This loss can cause a very blurred spot in central vision. With time that spot may get bigger and darker and at that point it takes more of central vision. People with this stage have difficulty reading, recognizing faces until they are very close to another person. AMD in the dry form is more common than the wet form. In the case of dry form there are no blood vessel, which have their fluid or blood leakage into the retina. Dry form of AMD can progress and cause vision loss without turning into wet AMD. [17]

2. 2. 2. Diagnosis of AMD

Ophthalmologist can diagnose AMD if he uses eye drops to dilate pupils in eye. Dilating the pupils allows him to see the back of the eye better. He can determine the presence of AMD using various illuminating and magnifying devices. Usual eye exam is also looking at Amsler grid. If a person has AMD, when they are looking at an Amsler grid with one eye they may notice wavy lines instead of straight lines. In every form of AMD, the ophthalmologist can find decreased visual acuity with saving the peripheral vision. The most common macular degeneration early symptom is blurred vision. As less cells are able to function, AMD people will see details less clearly in front of them, like words in a book. This kind of blurred vision will disappear in brighter light. If the loss of these light sensing cells becomes bigger than people may see a small black blind spot in centre of their vision. Dry form of AMD symptoms develops slowly and they do not include blindness. Nevertheless, those symptoms may aggravate daily tasks like reading and driving. Decreased night vision is also one of the symptoms, a decrease in the intensity or brightness of colours and in general reduce the overall vision. Dry AMD can affect one or both eyes, if only one eye is affected, symptoms may not occur because of a healthy eye. These symptoms may also occur in the wet form of age related macular degeneration but the most common are straight lines appearing wavy. The reason why this happens is because of the leaking fluids, which lift the macula above and disturb the vision. At this point larger areas of grey or black dots may appear in central area of vision. There is no written cause of AMD, but there are some certain risk factors of development of age related macular degeneration. Biggest of all is age, although AMD can occur during the middle age. Some studies show that people over age of 60 are at greater risk than other groups of people. Also, smoking, obesity, family history of macular degeneration, high blood pressure, raised blood cholesterol and others can often cause AMD. [10]

2. 2. 3. Treatments for macular degeneration

Treatment for wet form of AMD is laser surgery, photodynamic therapy or injections into the eye. All of those are treatments but none of them are permanent cure for AMD. With all of those treatments AMD may progress despite previous treatments. Laser surgery will destroy leaky blood vessels. Laser light is pointed directly on blood vessels to destroy them with prevention of further vision loss. Laser surgery is not always the best way to treat the eye because sometimes it may also destroy some of healthy tissue and vision. Surgery is effective for slowing visual loss if the blood vessels are developed away from the fovea. Risk of new blood vessels still stands, for that reason the patient may have to treat them more than once. Photodynamic therapy uses a drug that is injected into the vein of the arm, and then the light is directed into the eye to activate the drug process to the blood vessels into the eye. This treatment destroys the blood vessels and decreases progress of vision loss. This type of treatment can slow down vision loss, but it is temporary and the patient may need another therapy. Injections into the eye treatment developed to stop growth of new blood vessels and that revolutionized the treatment of wet AMD. Injections treatment will slow down vision loss and some patients can experience improvement of vision. Patients with advanced AMD on both eyes can get an implant of a telescopic lens into one eye. Telescopic lens is a replacement for natural lens and might reduce the peripheral field of vision but also can improve distance and close – up central vision.

Dry form of macular degeneration progresses slowly and patients with this form of disease are able to live normally, because it often effects one eye more than the other. When a patient has diagnosed advanced stage, there is no treatment that can prevent further loss of vision but treatment can help to delay the possibility of progressing from intermediate to advanced stage of AMD. The National Eye Institute had studies that showed if patients take some dose of formulation antioxidants and zinc, it significantly reduces the risk of advanced stage of macular degeneration. Formulation in antioxidants includes vitamin A, C, E and zinc. This formulation is good for intermediate stage but there is no study that also helps early stage of AMD.

From the other side, patients can prevent their disease through lifestyle changes. These include changing their diet to include more fruits and vegetable, choosing healthy nutrition.

Advance form of AMD can cause loss of central vision in both eyes but those patients can get around in familiar situations. They can use devices, which can improve their vision and allow them to watch television or to read. There is no certain way how to prevent developing of AMD but some of bad habits can be prevention. Eating healthy food, not smoking, maintaining blood pressure, regular weight and regular exercise can help to prevent macular degeneration. [15]

3. Methodology

In this part the methods and strategies used in the experimental part of this master thesis will be described. For better comprehension, methods are divided in two groups: software and hardware methods.

1. 3. Software methodology

This section will present software methods, in this case metrics, used for the experimental part of this master thesis. Furthermore, a theoretical approach to metrics in general and three methods of metrics will be offered. Metrics methods which were used for measuring errors in the experimental part are described in the following sections.

The mean squared error (MSE) [12] measures the average of the squares of the "errors", that is, the difference between the estimator and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss or quadratic loss. The difference occurs because of randomness or because the estimator doesn't account for information that could produce a more accurate estimate. MSE function is calculated in the following way:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\widehat{Y}_i - Y_i)^2 \qquad (4)$$

Second metric used in this master thesis is Peak signal-to-noise ratio (PSNR) [13], which is an engineering term for the ration between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. PSNR is commonly used to measure the quality of reconstruction of lossy compression codecs. For example image compression, which is used in this master thesis, the signal is the original data, and the noise is the error introduced by compression. When comparing compression codecs, PSNR is an approximation to human perception of reconstruction quality. In general, higher PSNR indicates that

the reconstruction is of higher quality. PSNR is most easily defined through the mean squared error, which is defined like this:

$$\mathrm{PSNR} = 10 \times \log_{10}\left(\frac{\mathrm{MAX}_I^2}{\mathrm{MSE}}\right) \quad (5)$$

Third metric presented in this master thesis is the structural similarity (SSIM) [14] index, this is a method for measuring the similarity between two images. SSIM is designed to improve on methods like MSE and PSNR, which have proven to be inconsistent with human eye perception. SSIM considers image degradation as perceived change in structural information. The idea is that structural information pixels have strong inter-dependencies especially when they are spatially close. These dependencies carry important information about the structure of the objects in the visual scene. For that reason, results are decimal values between -1 and 1, and 1 means two identical images. The SSIM metric is calculated through the equation:

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (6)$$

In the designed algorithm for edge extraction first is applied Gaussian blur which will clear the noise from the image and it is not constant, three different filter sizes were applied, it starts from 3x3, 5x5 and in the end is 9x9 Gaussian filter. After applying different sizes different sigmas from 0.1, 0.5, 1, 2 and 5 were applied. Next step in this algorithm was applying of gradient operator for obtaining the gradients intensity and direction. Third step is to determine if the pixel is relative candidate for an edge than its neighbour. For the last step in the algorithm hysteresis thresholding was used, which can find where exactly edges begin and end. Five selected images were filtered through the algorithm and compared with golden standard which is human eye, in this case. Those images are compared through metrics MSE, PSNR and SSIM. All used images are shown below from figure 18 to 22. In those figures are shown original image (A), from the right side is shown golden standard (B) which was compared in the metrics. Left side below is image filtered through Canny edge detector and on the right side below is filtered image through the developed algorithm

(D). Aforementioned metrics are used for measuring errors between two images below.

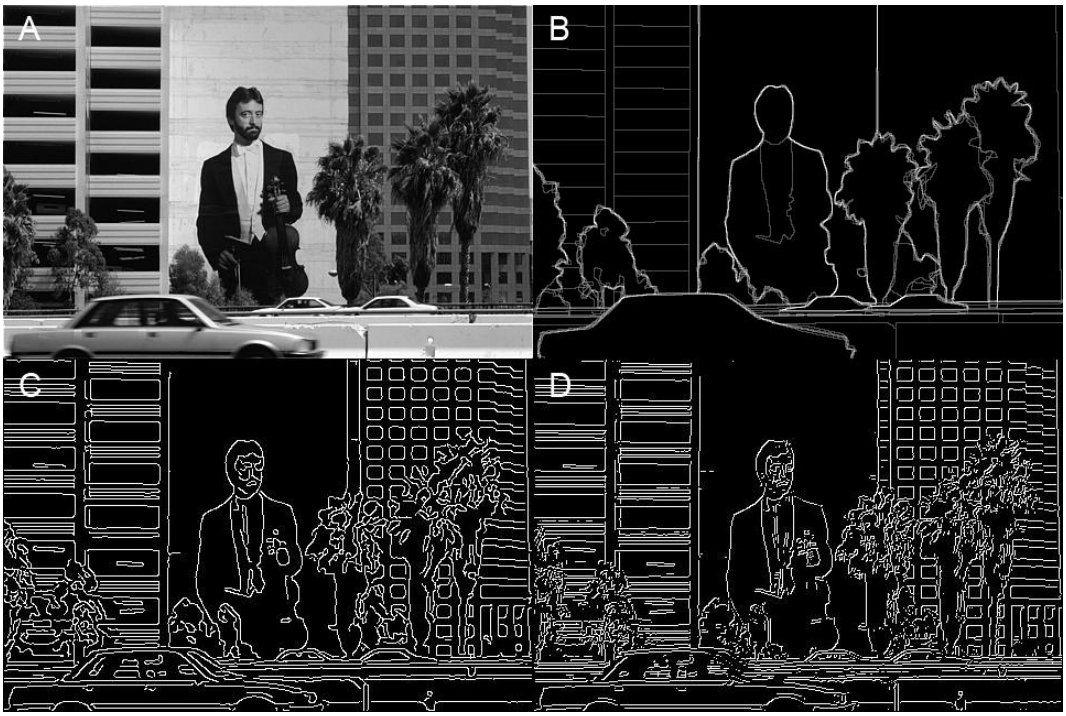

Figure 18. A-Original image, B-Golden standard, C-Canny edge detector, D-Developed edge detector

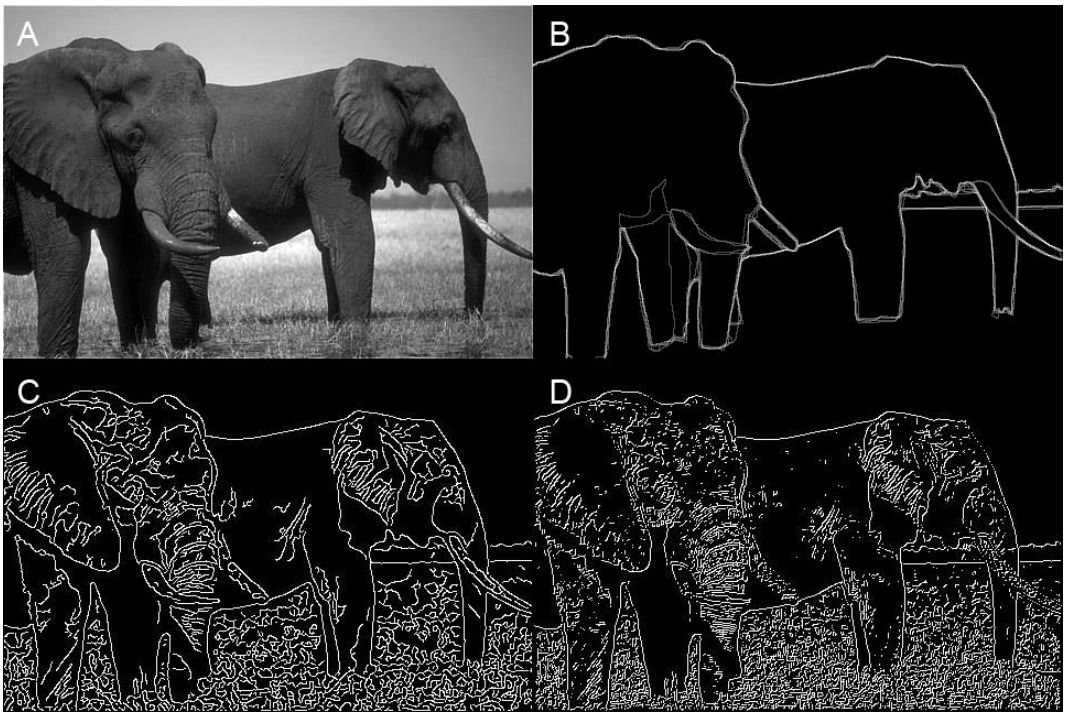

Figure 19. A-Original image, B-Golden standard, C-Canny edge detector, D-Developed edge detector

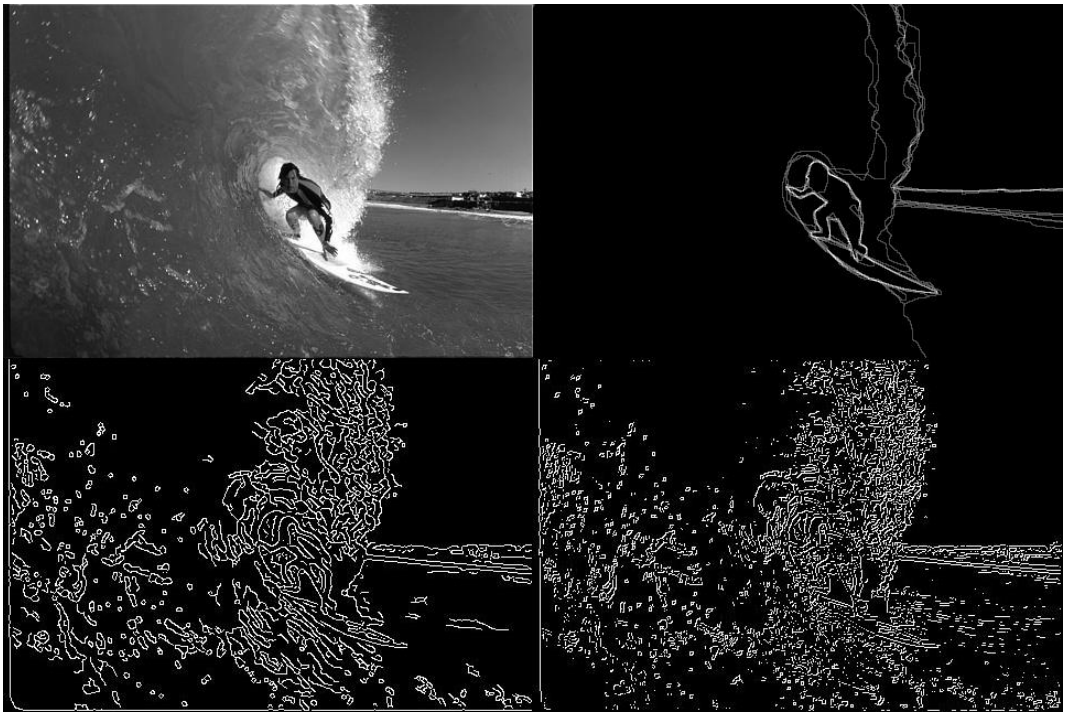

Figure 20. A-Original image, B-Golden standard, C-Canny edge detector, D-Developed edge detector

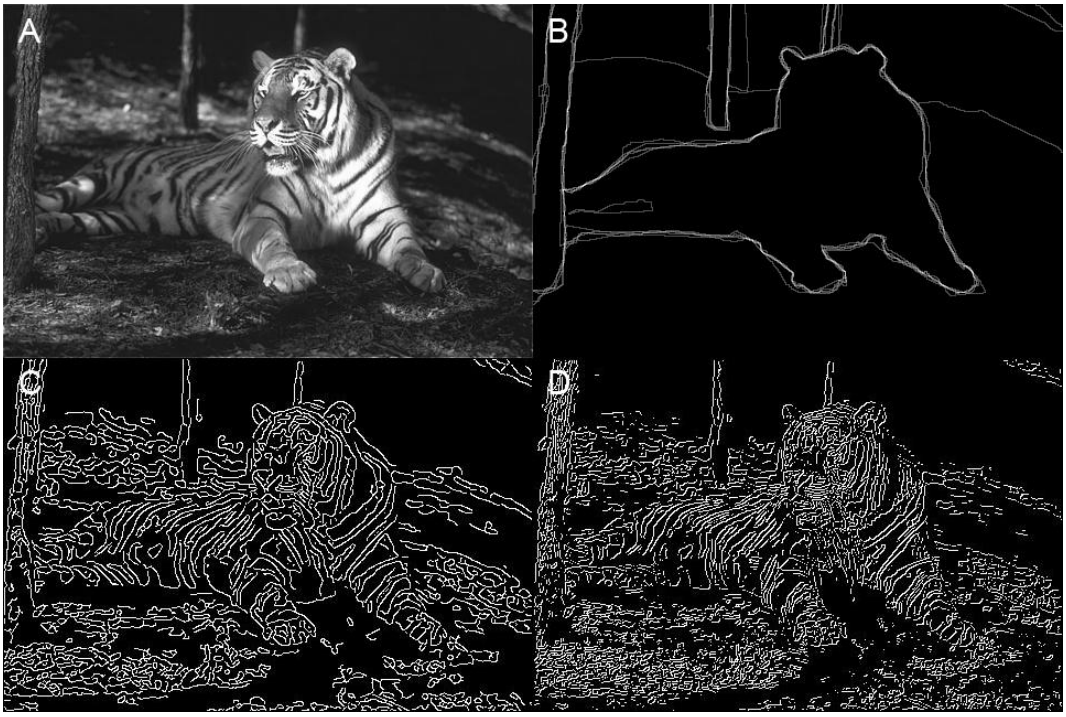

Figure 21. A-Original image, B-Golden standard, C-Canny edge detector, D-Developed edge detector

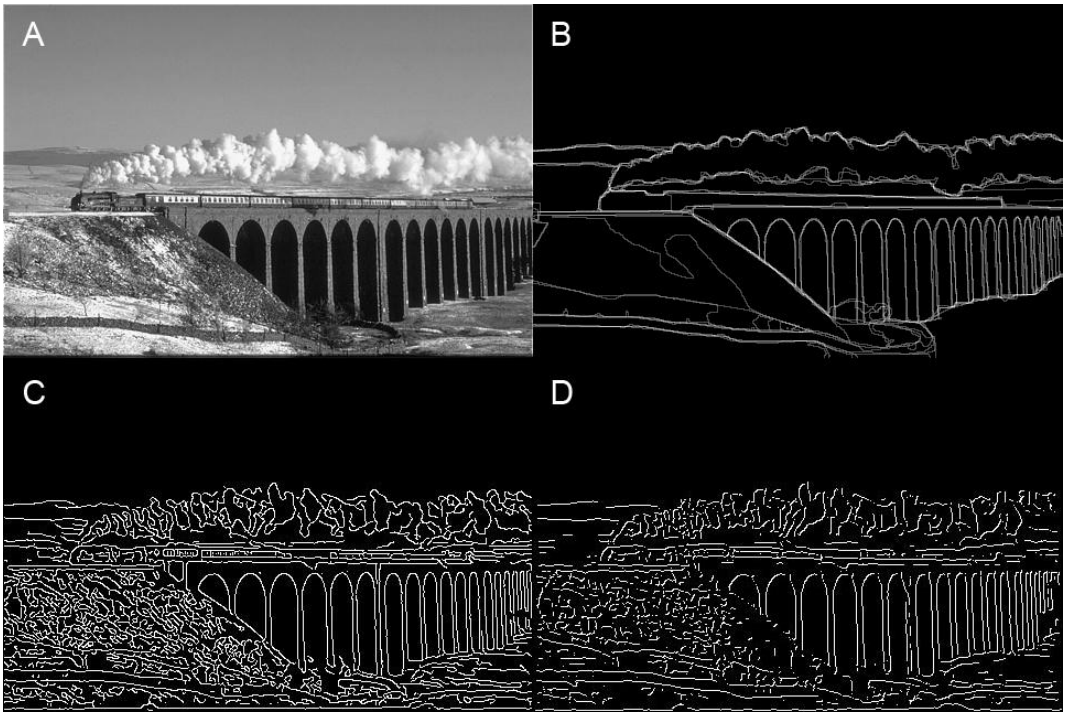

Figure 22. A-Original image, B-Golden standard, C-Canny edge detector, D-Developed edge detector

2. 3. Hardware methodology

In this section hardware methods and strategies performed for the testing of this master thesis are described. The strategy of this master thesis was to develop an edge detection filter that is similar to already developed ones that are implemented into Matlab software. For the purpose of this master thesis the “Canny” edge detection filter was designed, which is described in the section below. Images that were used are greyscale and downloaded from online database [11]. Furthermore, the foveal and convolution algorithms that are based on existing ones were also created for the purpose of this master thesis. The principles of aforementioned algorithms are described in the sections below.

2. 3. 1. Edge detector

Edge detection is a set of mathematical methods that are finding points in a digital image where the image brightness discontinuities, i.e. changes sharply. Those points are organized into a set of curved line segments called edges. In 1986 John F. Canny developed multi-stage algorithm to detect a wide range of edges in images called Canny edge detector.

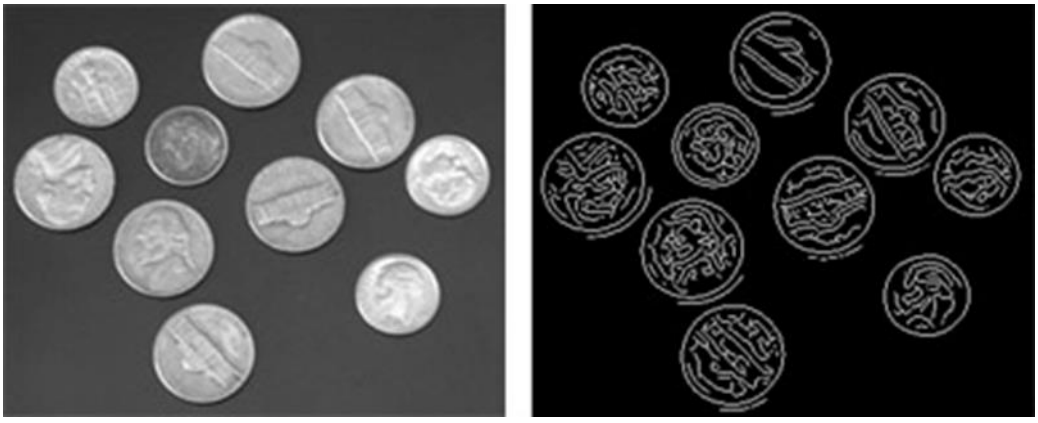

Figure 23. Image segmentation according Canny method

Therefore, Canny edge detector uses the edge-detection method in order to find edges in an image. Furthermore, this edge detector achieves the smoothing process with a linear combination of exponential function after that it detects the strong and weak edges, by means of higher order derivative operators. Moreover, Canny operator is the result of solving an optimization problem using constraints. The criteria for that are local unicity, localization and sensibility. Regarding its power of

edge detection the Canny edge detection algorithm is known as the optimal edge detector. In the next example is shown how Canny edge detector differs from another edge detector (i.e., Sobel edge detector). [21]

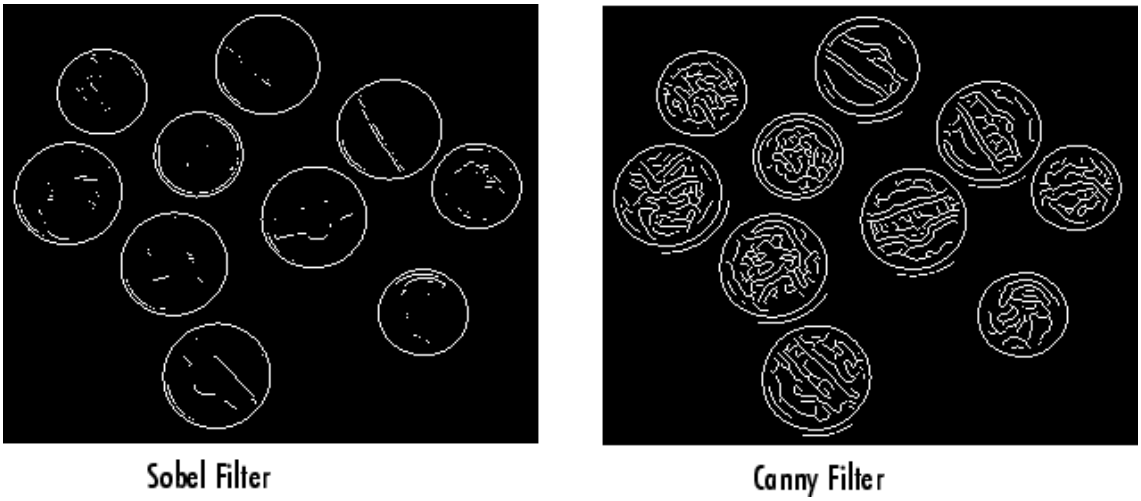

Figure 24. Comparison of a Sobel filter and Canny filter [21]

The main key ideas, methods and principle can be found in a Canny´s paper "A Computational Approach to Edge Detection" [25] that explains how he improved then existing methods of edge detection applying a few principles. First principle was linked with low error rate by marking and getting response only from real edges. Furthermore, the second principle is that the edge points be well localized, the distance between actual edge and edge pixels that are marked and found by detector amounts a minimum. Third, and also the last, principle is based on releasing a response from a single edge; that means that it behaves as last check, avoiding the chance of multiple responses from another edge.

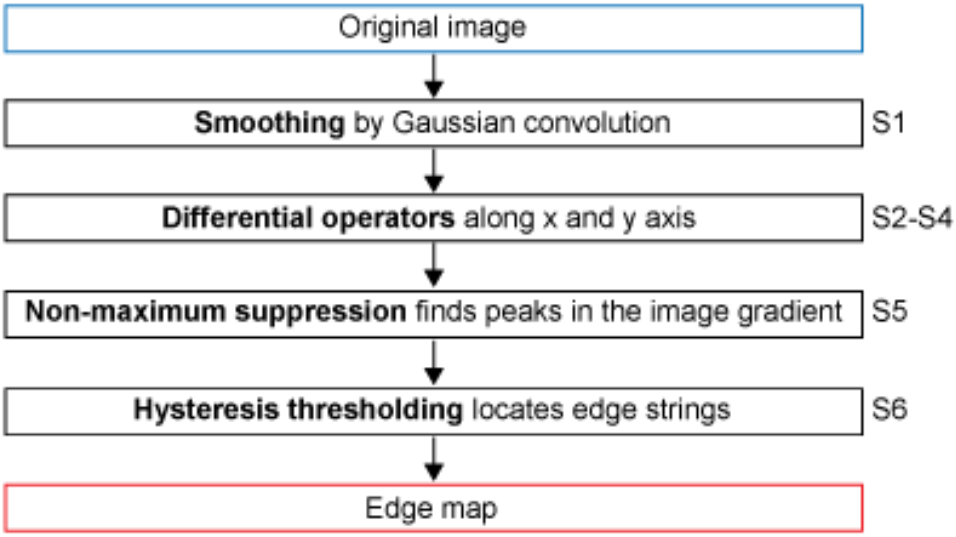

Figure 25. Scheme of the Canny algorithm implemented in hardware

Thus, Canny algorithm first affects an image with the smoothing in order to eliminate the possible noise. Sequential step is finding the image gradient to peak regions with high spatial derivatives. These regions are afterwards tracked along with the algorithm, which suppresses any pixel that is not at its maximum. The gradient array is reduced at this point, i.e., hysteresis. Hysteresis is used therefore in order to track along the remaining pixels, which are not suppressed within previous step. For this proposal, i.e., hysteresis, are used two thresholds. In case when the magnitude is below the first threshold, it is set to zero (made a non-edge). In the other case, when the magnitude is above the high threshold, it is made an edge. Third case is when the magnitude is between the 2 thresholds, when it is set to zero, unless if a path from this pixel to a pixel exists with a gradient above second threshold.
Application of the canny edge detector requires the five stages, which are described in sequential chapters.

First Stage (S1):
First step refers to using a filter in order of removing all the noise from the raw image. Within this step a Gaussian filter mask is applied. Gaussian filter is a 5x5 matrices which have constant $\sigma = 1.4$.

$$\mathrm{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 5 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \times A \tag{7}$$

A convolution mask, with witch a smoothing process is preformed, is usually much smaller than the raw image. As a result, that mask is slide over the image, manipulating a square of pixels at a time. [22]

Second Stage (S2):

Finding the edge strength by taking the gradient of the image is the second step. A Sobel operator does a 2D spatial gradient measurement based on an image. Therefore, the approximate absolute gradient magnitude (i.e., edge strength) at each point could be found. The Sobel operator works with two 3x3 convolution masks, (i.e., convolution kernels). One estimates the gradient in the x-direction (Gx), while another estimates the gradient in the y-direction (Gy). [23]

$$\mathrm{G_x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \mathrm{G_y} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{8}$$

The magnitude of the gradient is then approximated using the formula:

$$|\mathrm{G}| = |\mathrm{G_x}| + |\mathrm{G_y}| \tag{9}$$

Third Stage (S3):

The direction of the edge is calculated using both gradients, in the x and in the y directions. An error will be generated in the case when summation of x components amounts zero. Thus, in the code a restriction exists to set whenever this case occurs. Moreover, when an error happens, the edge direction has to be 90 degrees or 0 degrees, depending on how much the value of the gradient in the y-direction amounts. If Gy is zero, consequently, the edge direction will be 0 degrees.

Otherwise, the edge direction will equal 90 degrees. Edge direction is calculated throughout the following formula: [22]

$$\theta = \tan\left(\frac{G_y}{G_x}\right) \tag{10}$$

Fourth Stage (S4)

After third step the edge direction is known, therefore, the next step is to interconnect the edge direction to a direction that can be traced in an image. In the 5x5 sized image the pixels are aligned as it is shown in the following equation.

$$\begin{matrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & a & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{matrix} \tag{11}$$

According previous image, there are only four possible directions when describing the surrounding pixels of the pixel “a”. They amount: - 0 degrees (horizontal direction), 45 degrees (along the positive diagonal), 90 degrees (vertical direction), or 135 degrees (along the negative diagonal). Therefore, the edge orientation has to be find within one of the mentioned four directions, regarding which direction is the closest one (e.g. if the orientation angle amounts 3 degrees, it will be estimated as zero degrees.

Fifth Stage (S5):

A non-maximum suppression criterion has to be applied after the edge directions are known. It is used to trace along the edge in the edge direction as well as suppress any pixel value that is not considered to be an edge. Fifth stage usually gives a thin line in the output image.

Sixth Stage (S6):

In this last stage, hysteresis is applied in order to eliminate the streaking. Streaking represents the breaking up of an edge contour caused by the operator output fluctuating above and below the threshold. [22] Thresholding with hysteresis requires uses 2 thresholds, high and low thresholds (T1 and T2).

First is high threshold applied, he marks out the edges, which are genuine. From that edges can be traced throughout the image. While tracing an edge, lower threshold is applied which allows tracing faint sections of edge as long as he find starting point. Any pixel in the image which value amounts greater than T1 is considered to be an edge pixel, thus, is marked like it immediately. Moreover, any pixels linked to the edge pixel and that has a value greater than T2 is selected as edge pixels as well. In the conclusion, it is necessary to start with a gradient of T2, and the process will not stop until reaching a gradient below T1.

When this process is done, algorithm gives binary image where each pixel is marked as edge pixel or a non-edge pixel. [19] Process of edge detection processing is shown in figure 26 below.

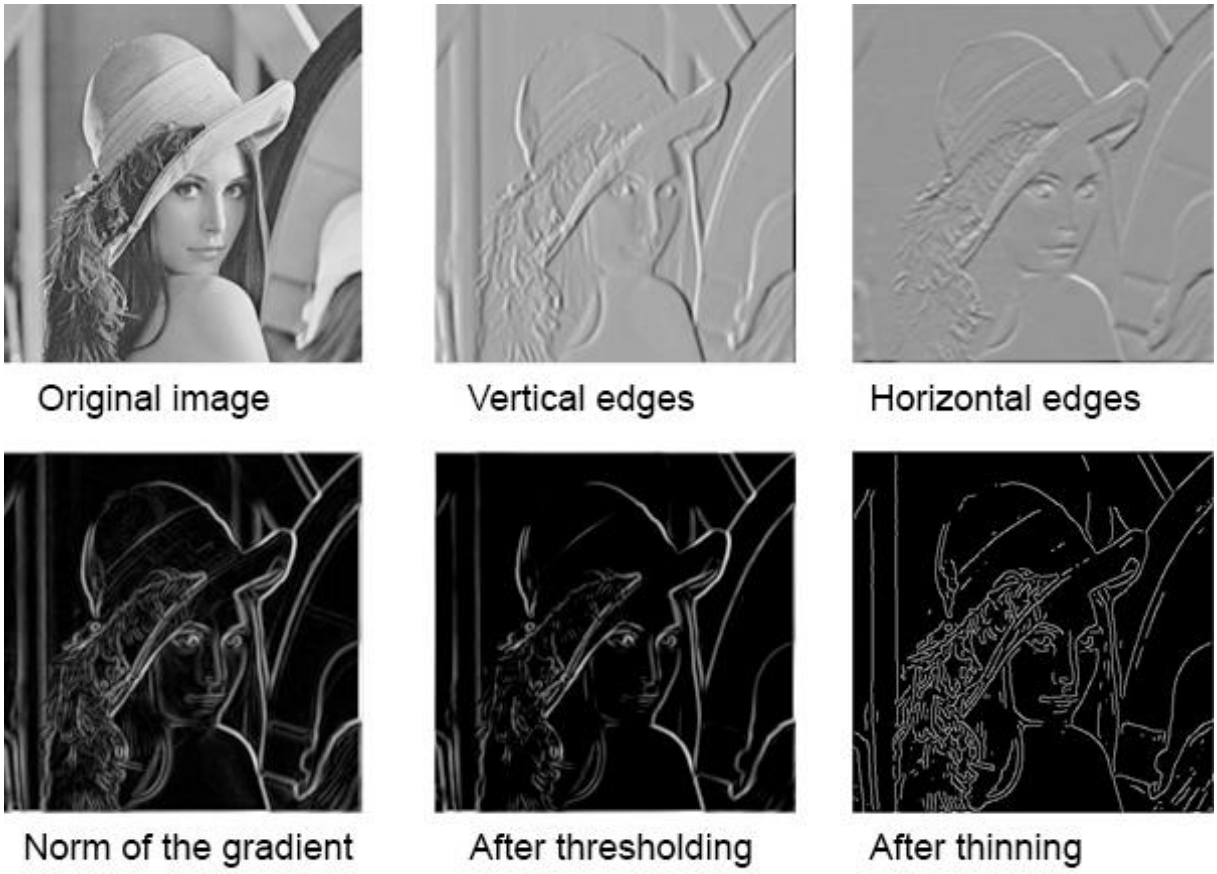

Figure 26. Stages in the algorithm

2. 3. 2. Convolution

Convolution is considered to be one of the most important primitive operations for image processing that changes the intensity of a pixel to reflect the intensities of the surrounding pixels. It is commonly used to create image filters. Furthermore, popular image effects like blur, sharpen, and edge detection can be created using convolution.

Considering the form of mathematical convolution, one input image, a kernel mask (which will act as filter) and the values of a given pixel in the output image are calculated by multiplying each kernel value by the corresponding input image pixel values. The kernel or more specifically, the values held within the kernel, is what determines how to transform the pixels from the original image into the pixels of the processed image. [24]

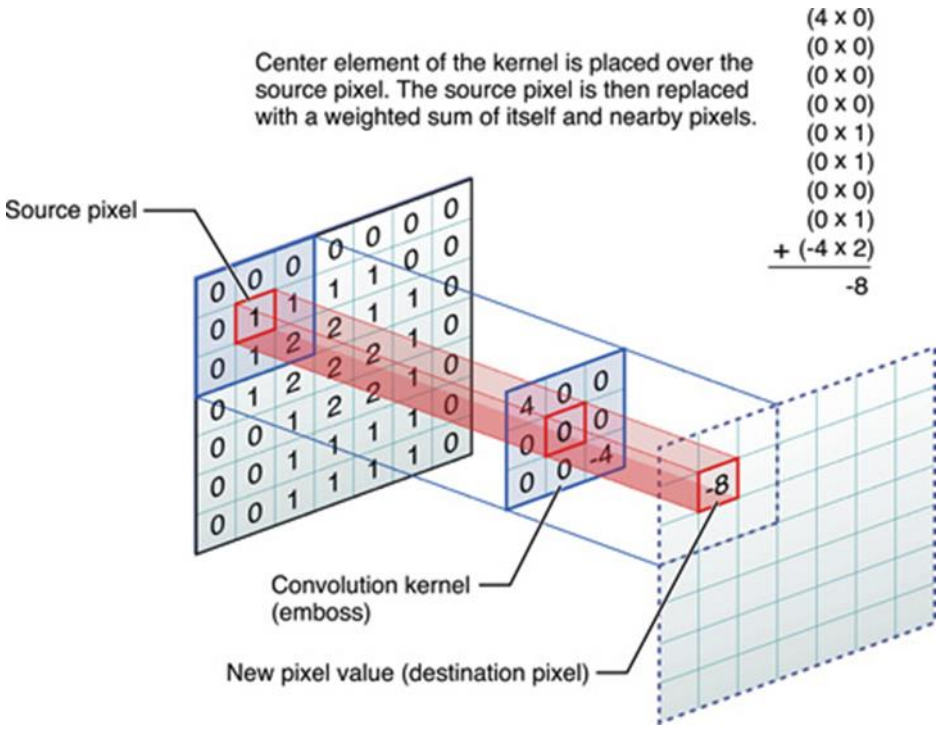

Figure 27. Convolution [24]

A technique for fast convolution is based on the filter mask that is separable. It is possible only if the filter mask can be broken into two one-dimensional signals: a vertical and a horizontal projection:

$$x[r, c] = vert[r] \times hor[c] \quad (12)$$

The value of each pixel in the image is equal to the corresponding point in the horizontal projection multiplied by the corresponding point in the vertical projection. There are, however, an infinite number of separable images. This can be understood by generating discretionary vertical and horizontal projections, and finding the image that matches them.

To perform the convolution process using a separable filter kernel, the process is based on convolving each column in the image with the vertical projection, resulting in an intermediate image. Following, each row of this intermediate image will be convolved with the horizontal projection of the mask. The process is commutative being possible permuting rows with columns. The resulting image is identical to the direct convolution (O(n^2)) of the original image and the filter kernel. In other words, convolving an MxM filter kernel with an NxN image requires a time proportional to N^2M^2, with it, each pixel in the output image depends N^2M^2N^2 M^2 on all the pixels in the filter kernel. In comparison, convolution by separability only requires a time proportional to N^2M. For filter kernels that are hundreds of pixels wide, this technique will reduce the execution time by a factor of hundreds.

2. 3. 3. Simulating foveal vision

Generally, the visual system of primates has a space-variant nature where the resolution is high in the fovea and gradually declines towards the periphery of the visual field. Due to rapid scanning of the eye (saccades), it is possible to accomplish very high resolution via the fovea, preserving a rapid wide field of vision. In order to simulate this sampling behaviour we divide the image into two regions; fovea and periphery, using a multi-scale resolution sampling methods. The model has a 1:1 proportion of pixels in the fovea. Corresponding to one pixel and equal width, the peripheral region is splite into concentric rings. Kernel size grows exponentially with radial distance from the fovea, and every ring is blurred by a Gaussian function.
We presume that the number of pixels in the input image proximate the number of cones sampling the retinal image. Based on the biological size of the fovea with respect to the retina, the number of pixels characterizing the fovea field in the input image is determined. Dimensions are 1mm and 42mm correspondingly. As follows, for an image size of 800 x 800, the number of pixels characterizing the fovea will roughly be 20 x 20. The foveal output image will be:

$$\mathrm{I}_{\mathrm{foveal}}(\mathrm{x},\mathrm{y},\mathrm{r}) = \mathrm{G}_{\sigma}(\mathrm{x},\mathrm{y}) \times \mathrm{I}(\mathrm{x},\mathrm{y},\mathrm{r}) \quad (13)$$

$$\mathrm{G}_{\sigma}(\mathrm{x},\mathrm{y}) = \frac{1}{2\pi\sigma^2}\,\mathrm{e}^{-(\mathrm{x}^2+\mathrm{y}^2)/2\sigma^2} \quad (14)$$

Where Gσ(x, y) is a two dimensional Gaussian averaging filter with a standard deviation σ equalto log(r), and r is the radial distance of the pixel (x, y) from the centre of the input image.

Colour separation

After simulating the foveal-peripheral vision the second stage in the model is to account for the colour separation in the retina.
Input images are basically 2D matrices, with the RGB components. In the contrary, the chromatic information in the human retina is encoded into two color opponent channels; green-red and blue-yellow, and one achromatic channel. For that reason it is converted to a LGRBY colour space.

Achromatic or the L channel represents absolute luminance and extends from 0 to 100, where 0 represents black and 100 white.
The two colour channels, GR and BY, represent the greenness-redness and the blue-ness-yellowness colour opponents respectively.
The negative values of GR indicate green while positive values indicate magenta. In the case of BY colour channel, negative values indicate blue and positive values indicate yellow. Pixels for which GR and BY are equal and are both equal to 0 are achromatic and thus the L channel represents the achromatic scale of gray from black to white.

Horizontal layer

The horizontal cells serve as a negative feedback to gain control on cone cells, adapting the reduction of glutamate release to increase the illumination. Since the 8-bit dynamic range of most jpeg images is small, the variation in illumination is considered to be small, and therefore gaining of control was not considered in this model, although histogram equalization can be used to ensure optimal use of the 8-bit intensity range.
Three types of the horizontal cells exist; HI (achromatic), HII and HIII (chromatic) cells. They have direct electrical synapses with each other and provide inhibitory feedback to the photoreceptors, with receptive field growing towards the periphery. On the other hand, horizontal cells are absent in the fovea. An average Gaussian filter enables obtaining of the output of the horizontal cell by convolving the cone output.

Bipolar layer

Bipolar cells receive their inputs essentially from the cones with inhibitory feedback from the horizontal cells. With decreasing glutamate (increasing photo response), ON bipolar cells depolarize from the connecting photoreceptors, whereas OFF bipolar cells hyperpolarize. The phenomena centre-surround processing is generated from the synapses of surrounding ON and OFF bipolar cells to the retinal ganglion cells. In mammals, the proportion of the centre diameter field to the surround diameter one is range between 1:10. The ratio of the centre diameter field in mammals, to the surround diameter one is between 1:10.

The centre-surround features of the bipolar cells, can be modelled in mathematical form as a difference of two Gaussian low pass filters (DoG). The surround filter, is more low-pass than the centre one. Mathematically DoG output to the retinal ganglion cells can be defined as:

$$\mathrm{DoG}_{\mathrm{Bipolar}}(x, y) = \frac{1}{2\pi\sigma_s^2} e^{-(x^2+y^2)/2\sigma_s^2} - \frac{1}{2\pi\sigma_c^2} e^{-(x^2+y^2)/2\sigma_c^2} \quad (15)$$

σs and σc are the surround and centre standard deviation of the Gaussian filter. Between the surround sigma to the centre one, the ratio is considered to be 1:2, which give a reasonable agreement with the physiologically measured value. Using this ratio value results in a receptive field diameter of the surround larger than the centre diameter by 5 to 6 times. Centre surround processing in the retina, is carried out for Red-centre/Green-surround, Green-centre/Red-surround, Blue-centre/Yellow-surround (parvocellular pathway) and achromatic ON-OFF centre-surround (magnocellular pathway). In this model the five centre-surround signals are calculated as following:

- $DoG_{green\ /\ red} = Horz_{green} - Cone_{red}$
- $DoG_{red\ /\ green} = Horz_{red} - Cone_{green}$
- $DoG_{blue\ /\ yellow} = Horz_{blue} - Cone_{yellow}$
- $DoG_{yellow\ /\ blue} = Horz_{yellow} - Cone_{blue}$
- $DoG_{Luminance} = Horz_{Luminance} - Cone_{Luminance}$

The size of the surround Gaussian kernel is set to 5 times larger than the size of centre kernel in each ON/OFF channel. Although there is no Yellow-centre/Blue-surround processing in the retina, here it was included for purposes of processing symmetry.

Image reconstruction

Reconstruction is accomplished by reversing the processing operations carried out in the three retina layers. The output of DoG process of the bipolar cells is treated as a spatial derivative of the achromatic, R/G and B/Y channels. Given this gradient *G* for each channel, assignment is to reconstruct an image I whose gradient I is very similar to *G*. To achieve this, solution is the equation *I* = *G*. Considering the gradient image is a modified one from the actual gradients of the *L, G_R* and *B_Y* channels of the

LGRBY image, the resulting gradient field G = [G_x, G_y] may not be integrable. To overcome this situation, there is a suitable function *I*, whose gradient should be very close to *G* using the least square error approach by searching the space of all 2D potential functions, that is, to minimize the integral in 2D space.

Macular degeneration simulation

Described model above simulates a human retina. Degeneration is implemented between the foveal simulation (eccentricity simulation) and colour separation. In that way is simulated scotoma similar that is found in AMD patients. This degeneration process is created in the way that first is generated binary mask that simulates lesions on the scotoma region.

The function that generates this mask takes three parameters; the location of the fovea with respect to the whole image (this refers to the area in the image where the person is fixating on it), the size of the degenerated area relative to the macula size, and the degree of degeneration.

For second part of the testing hardware implementation of the algorithms to three different GPU's were developed which will show speed of processing images. Algorithms that were tested are convolution, edge detector and foveal algorithm. GPU's that were used for the experimental part of this master thesis will be described in the following paragraph.

Table 1. Comparison of machines used in the experiments

	Intel Xeon CPU E5-2670	Intel Xeon Phi Accelerator	AMD A10-6800K APU	NVIDIA Tesla K20c
OpenCl version	1,2	1,2	1,2	1,1
Max. Compute Units	32	236	4	13
Local Memory Size	32 KB	32 KB	32 KB	48 KB
Global Memory Size	32074 MB	5773 MB	7197 MB	4799 MB
Max Alloc Size	8018 MB	1924 MB	2048 MB	1199 MB
Max Work-group Size	8192	8192	1024	1024
Max Work-item Dims	(8192 8192 8192)	(8192 8192 8192)	(1024 1024 1024)	(1024 1024 64)

Intel Xeon E5 2670 has clock speed at 2.6 GHz which can run with turbo boost till 3.3 GHz. This machine need power up to 115 W. Xeon E5 have 8 cores and supported with 4 GB of DDR3 memory. It is supported with OpenCL version 1.2.
AMD A10-6800K also known as AMD Richland is Accelerated Parallel Processor (APU) which means that contains CPU and GPU in one. This APU have "Trinity" chip, which have core speed up to 4.1GHz/4.4GHz and GPU clocked at 844MHz. This APU contains GPU called Radeon HD 8670D which can have maximum resolution at 2560x1600 pixels. It contains DDR3 memory connected using a 128-bit memory interface. Core speed is 844 MHz while memory is running on 1067 MHz. Required power is 100 W. [20]
NVIDIA Tesla K20c is high-end professional graphics card. It is built on the 28nm process and based on the GK110 graphics processor. Tesla K20c contains 5120 MB GDDR5 memory which are connected using a 320-bit memory interface. The GPU is operating at frequency of 706 MHz and memory is running at 1300 MHz. Powerful like this it needs power up to 225 W. Also can run in full HD mode.

4. Results

1. 4. Comparison and critical evaluation

1. 4. 1. Software implementation

In the tables below are results calculated from metrics functions. We have 5 different images where first image is golden standard, which is how human eye will detect edges and second is filtered image through our developed algorithm. For testing's we changed two values from algorithm, filter size and sigma. In this case we used three different filter sizes 3x3, 5x5 and 9x9 instead of well-known Gaussian filter with size of 5x5. Instead of using default sigma of 1.4 we used five different values. Our sigmas were 0.1, 0.5, 1, 2 and 5. First table shows results with values of filter size 3x3 and all sigma values.

Table 2. Results with filter size [3x3] and all values of sigma (σ)

Image 1	Filter size/sigma	3/0.1	3/0.5	3/1	3/2	3/5
	SSIM	0,269502	0,273617	0,280152	0,284796	0,286042
	MSE	18,0299	18,24791	18,74172	18,90852	18,92669
	PSNR	35,46808	35,41588	35,29992	35,26143	35,25726
Image 2	Filter size/sigma	3/0.1	3/0.5	3/1	3/2	3/5
	SSIM	0,331886	0,348594	0,391614	0,403953	0,408567
	MSE	10,13387	10,21975	10,31554	10,33041	10,34362
	PSNR	37,97026	37,93361	37,89309	37,88684	37,88129
Image 3	Filter size/sigma	3/0.1	3/0.5	3/1	3/2	3/5
	SSIM	0,369784	0,386647	0,435242	0,448629	0,451769
	MSE	6,758117	6,840694	6,959605	6,98603	6,989333
	PSNR	39,72975	39,67701	39,60216	39,58571	39,58365
Image 4	Filter size/sigma	3/0.1	3/0.5	3/1	3/2	3/5
	SSIM	0,233834	0,241028	0,261189	0,265398	0,268121
	MSE	10,85395	10,90349	11,04718	11,06865	11,07856
	PSNR	37,67213	37,65235	37,5955	37,58707	37,58318
Image 5	Filter size/sigma	3/0.1	3/0.5	3/1	3/2	3/5
	SSIM	0,437025	0,439516	0,445171	0,447201	0,447201
	MSE	20,03488	20,22811	20,67568	20,81441	20,86725
	PSNR	35,01014	34,96846	34,87341	34,84437	34,83336

Regarding to the metric equations this Table 1 show that there is no possible match between two images in the case of SSIM function. All five images are similar but none of them are the same, closest match is in the Image 3 with sigma (σ=5) where the result is 0,451769 which is the highest match. For PSNR metrics, reconstructed image is better if the number is higher, all results are above 30 but not even one is higher over 40, closest is also Image 3 with smallest sigma (σ=0.1). Result is 39,72975.

Table 3. . Results with filter size [5x5] and all values of sigma (σ)

Image 1	Filter size/sigma	5/0.1	5/0.5	5/1	5/2	5/5
	SSIM	0,269502	0,273594	0,292007	0,33177	0,348995
	MSE	18,0299	18,25286	19,09185	19,85816	19,92257
	PSNR	35,46808	35,4147	35,21953	35,04862	35,03456
Image 2	Filter size/sigma	5/0.1	5/0.5	5/1	5/2	5/5
	SSIM	0,331886	0,348607	0,421567	0,498209	0,513701
	MSE	10,13387	10,21975	10,38986	10,55997	10,71687
	PSNR	37,97026	37,93361	37,86191	37,79138	37,72733
Image 3	Filter size/sigma	5/0.1	5/0.5	5/1	5/2	5/5
	SSIM	0,369784	0,386728	0,468242	0,532811	0,542983
	MSE	6,758117	6,840694	7,073562	7,412128	7,448462
	PSNR	39,72975	39,67701	39,53163	39,32858	39,30734
Image 4	Filter size/sigma	5/0.1	5/0.5	5/1	5/2	5/5
	SSIM	0,233834	0,241328	0,275127	0,309892	0,32098
	MSE	10,85395	10,90349	11,10828	11,34941	11,37914
	PSNR	37,67213	37,65235	37,57154	37,47828	37,46692
Image 5	Filter size/sigma	5/0.1	5/0.5	5/1	5/2	5/5
	SSIM	0,437025	0,439492	0,449579	0,454876	0,448874
	MSE	20,03488	20,23141	20,95479	21,96883	22,47256
	PSNR	35,01014	34,96775	34,81518	34,60994	34,51149

In the Table 2 are shown results with filter size 5x5, neither with filter size like in original Canny edge detector there is no match between results. Highest match is in Image 3 with sigma (σ=5). For the PSNR is the same like in the previous table there is no lower result then 30 but either higher then 40, maximum result is in Image 3 with sigma (σ=0.1) and it is 39,72975. For sigmas below value (σ=1) there is no change.

Table 4. . Results with filter size [9x9] and all values of sigma (σ)

Image 1	Filter size/sigma	9/0.1	9/0.5	9/1	9/2	9/5
	SSIM	0,269502	0,273594	0,293464	0,384676	0,467852
	MSE	18,0299	18,25286	19,10341	20,82762	22,14885
	PSNR	35,46808	35,4147	35,2169	34,84161	34,5745
Image 2	Filter size/sigma	9/0.1	9/0.5	9/1	9/2	9/5
	SSIM	0,331886	0,348607	0,42463	0,594423	0,633565
	MSE	10,13387	10,21975	10,39151	10,6954	11,22224
	PSNR	37,97026	37,93361	37,86122	37,73604	37,52722
Image 3	Filter size/sigma	9/0.1	9/0.5	9/1	9/2	9/5
	SSIM	0,369784	0,386728	0,471536	0,607463	0,651232
	MSE	6,758117	6,840694	7,085123	7,552509	7,892727
	PSNR	39,72975	39,67701	39,52454	39,2471	39,05574
Image 4	Filter size/sigma	9/0.1	9/0.5	9/1	9/2	9/5
	SSIM	0,233834	0,241328	0,276201	0,361604	0,391131
	MSE	10,85395	10,90349	11,11654	11,46997	11,61531
	PSNR	37,67213	37,65235	37,56831	37,43239	37,3777
Image 5	Filter size/sigma	9/0.1	9/0.5	9/1	9/2	9/5
	SSIM	0,437025	0,439492	0,450151	0,473444	0,438253
	MSE	20,03488	20,23141	20,96635	22,75167	24,93336
	PSNR	35,01014	34,96775	34,81278	34,45788	34,0602

In the third table results will be the same but numbers will be different, for SSIM Image 3 with sigma (σ=5) is the closest match, and for PSNR is Image 3 with sigma (σ=0.1). For sigmas below value (σ=1) there is almost no change. Therefore sigma values should be higher in the experiments and then results can be more precise. In the figure below is comparison of a best PSNR (A) and SSIM (B) metric. On the right

side where is best result for SSIM metric it is clearly that have best match because filtered image doesn’t have so many edges extracted. Black pixels matches better with golden standard and therefore result is better, closer to the 1.

In the figure 28 are shown best results given by PSNR and SSIM metrics. PSNR result has parameters of filter size 9 and sigma size 0.1. SSIM result is shown with parameters filter size 9 and sigma size 5.

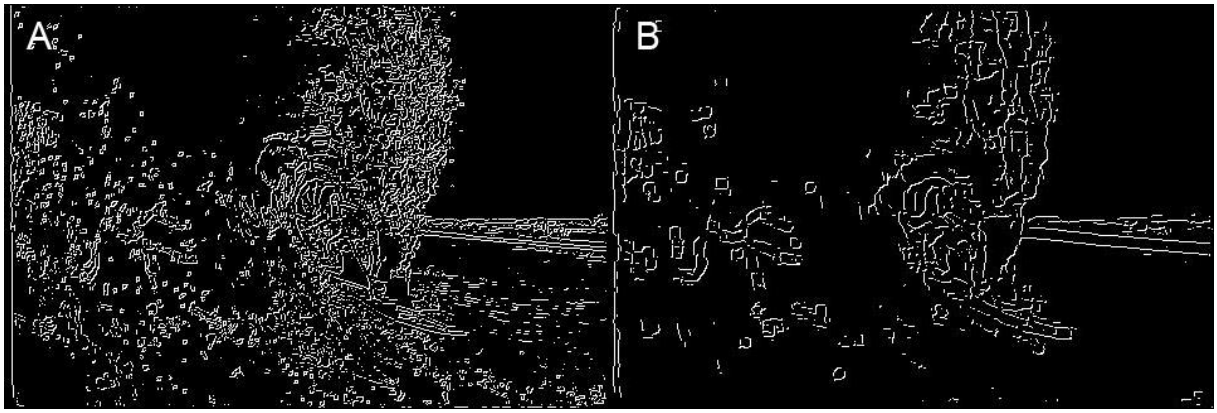

Figure 28. Comparison of best PSNR (A) and SSIM (B) metric

1. 4. 2. Hardware implementation

Table 5 represents results given from three different machines described above. Results are presented in microseconds. For different image sizes there is increasing time for processing of an image. Smaller images are processed faster than larger images, therefore, results increase towards larger images which is seen in the table below. Regarding to the filter size time also increases towards bigger filter size. For the filter size 3, in general NVIDIA machine is fastest for processing this algorithm, only exception is smallest image size where AMD APU has faster time, 63,04 μs.

Table 5. Comparison of a different machines, image sizes and filter size 3

			Intel Xeon CPU E5-2670	Intel Xeon Phi Accelerator	AMD A10-6800K APU	NVIDIA Tesla K20c
		Image Size				
CONVOLUTION	**Filter size**		**3**	**3**	**3**	**3**
	OCL Kernel	**64**	146,944	3147,818	63,04	88,448
	OCL Complete		287559	1836658	194850	2545127
	CPU		80	80	85	92
	OCL Kernel	**128**	700,57	5116,84	449,88	122,88
	OCL Complete		297036	2034795	191474	2628264
	CPU		266	266	477	372
	OCL Kernel	**256**	2099,634	7609,662	2264,32	232,448
	OCL Complete		325619	1831770	211163	2478064
	CPU		2047	2047	1868	937
	OCL Kernel	**512**	16428,492	9307,728	9556,48	740,48
	OCL Complete		288397	1796770	216833	2469529
	CPU		3331	3331	14122	4128
	OCL Kernel	**1024**	27748,854	17630,628	42144,88	2689,856
	OCL Complete		350862	1896893	244293	2463820
	CPU		19183	19183	37657	15783
	OCL Kernel	**2048**	57672,54	89720,192	240172,24	10604,224
	OCL Complete		467289	2135332	464437	2541161
	CPU		48539	48539	137131	68038
	OCL Kernel	**4096**	477178,2	219665,22	1230573,4	95882,944
	OCL Complete		767448	2721544	1557995	2615232
	CPU		196178	196178	522640	243612

Table 6 will show results with filter size 5 which are in general similar to the results in table 5 except that time of processing is here a bit slower that in a previous table. AMD APU is the fastest machine but only with small size images. Every other size of an image is faster processed with NVIDIA Tesla K20c machine.

Table 6. Comparison of a different machines, image sizes and filter size 5

			Intel Xeon CPU E5-2670	Intel Xeon Phi Accelerator	AMD A10-6800K APU	NVIDIA Tesla K20c
		Image Size				
CONVOLUTION	**Filter size**		**5**	**5**	**5**	**5**
	OCL Kernel	**64**	269,85	4008,798	87,72	94,784
	OCL Complete		346375	2024988	195078	2550241
	CPU		117	117	93	84
	OCL Kernel	**128**	709,342	5042,836	1296,72	129,856
	OCL Complete		286460	1825354	195478	2498645
	CPU		426	426	783	329
	OCL Kernel	**256**	2048,596	8208,488	3234,52	348,224
	OCL Complete		323045	1965198	193142	2582997
	CPU		3234	3234	2544	1974
	OCL Kernel	**512**	12404,586	8848,312	14142,76	1151,488
	OCL Complete		320581	1873060	201328	2518359
	CPU		6393	6393	18463	6223
	OCL Kernel	**1024**	33987,284	20107,458	63214,96	4294,976
	OCL Complete		294019	1977857	267134	2525129
	CPU		23292	23292	112823	27286
	OCL Kernel	**2048**	57424,85	82496,034	365715,28	16796,544
	OCL Complete		444607	2062342	593254	2611174
	CPU		100244	100244	399006	111176
	OCL Kernel	**4096**	363172,32	291099,27	1932806	137207,1
	OCL Complete		779095	2854308	2308036	2738857
	CPU		317505	317505	1543496	451460

Table 7 will show results with filter size 7 which are different than in previous tables. NVIDIA machine requires shorter amount of time for processing images comparing to the Intel Xeon Phi Accelerator which needs more time, 379823,15 µs what equals to 0,3799 s.

Table 7. Comparison of a different machines, image sizes and filter size 7

			Intel Xeon CPU E5-2670	Intel Xeon Phi Accelerator	AMD A10-6800K APU	NVIDIA Tesla K20c
		Image Size				
CONVOLUTION	**Filter size**		**7**	**7**	**7**	**7**
	OCL Kernel	**64**	274,842	3741,154	124,36	106,816
	OCL Complete		302204	1806249	196245	2485084
	CPU		125	125	322	190
	OCL Kernel	**128**	730,676	6300,858	908,96	153,344
	OCL Complete		301127	1871626	193949	2436875
	CPU		990	990	603	487
	OCL Kernel	**256**	1397,418	7538,542	4331,28	442,688
	OCL Complete		281078	1964548	209745	2509667
	CPU		4374	4374	7080	3234
	OCL Kernel	**512**	6710,242	10264,496	18641,68	1529,472
	OCL Complete		322589	2041520	219191	2440496
	CPU		8984	8984	19961	9439
	OCL Kernel	**1024**	23517,426	51636,658	85067,68	5808,64
	OCL Complete		388373	1993217	283638	2450144
	CPU		37066	37066	149395	39105
	OCL Kernel	**2048**	68187,18	85946,118	496842,96	23033,024
	OCL Complete		344035	2230334	706443	2488128
	CPU		108681	108681	554159	152478
	OCL Kernel	**4096**	370393,66	379823,15	2629773,1	190445,82
	OCL Complete		753499	2944994	2978992	2743504
	CPU		421318	421318	2106668	625325

Intel's Accelerator has slowest time for processing images with filter size 9 and any other filter therefore times of Intel's Accelerator will be just given in the tables but not commented during comparison. AMD APU machine has the second fastest time for processing images but with this filter size is not fast enough to compete with NVIDIA Tesla. Everything described is presented in the table 8.

Table 8. Comparison of a different machines, image sizes and filter size 9

			Intel Xeon CPU E5-2670	Intel Xeon Phi Accelerator	AMD A10-6800K APU	NVIDIA Tesla K20c
		Image Size				
CONVOLUTION	**Filter size**		**9**	**9**	**9**	**9**
	OCL Kernel	**64**	295,454	4459,806	138,68	113,216
	OCL Complete		257163	1844290	196110	2455718
	CPU		131	131	580	264
	OCL Kernel	**128**	442,736	6758,478	1203,52	198,144
	OCL Complete		301819	1818460	195525	2523688
	CPU		1301	1301	2337	1047
	OCL Kernel	**256**	2822,518	7584,456	6425,24	525,184
	OCL Complete		326067	1991354	200609	2489131
	CPU		5538	5538	8403	2781
	OCL Kernel	**512**	16485,682	10896,386	23268,96	1928,128
	OCL Complete		333288	1786015	214685	2447167
	CPU		9017	9017	44048	11669
	OCL Kernel	**1024**	54145,054	51572,228	107293,28	7532,096
	OCL Complete		337918	1937225	304654	2470805
	CPU		49873	49873	202866	51545
	OCL Kernel	**2048**	61433,904	99203,804	634210,32	29462,912
	OCL Complete		423891	2114776	846387	2579526
	CPU		182320	182320	761001	219107
	OCL Kernel	**4096**	630790,05	400746,77	3316150,5	237340,86
	OCL Complete		847632	2934510	3703683	2882950
	CPU		579406	579406	3091391	823782

In the table 9 are shown results with filter size 15. AMD APU has twice slower time (233,68 µs) than NVIDIA Tesla (131,2 µs) in the processing smallest images. If we compare medium size images like 512, AMD APU (37074,04 µs) has ten times slowest time than NVIDIA Tesla (3118,144 µs).

Table 9. Comparison of a different machines, image sizes and filter size 15

			Intel Xeon CPU E5-2670	Intel Xeon Phi Accelerator	AMD A10-6800K APU	NVIDIA Tesla K20c
		Image Size				
CONVOLUTION	**Filter size**		**15**	**15**	**15**	**15**
	OCL Kernel	**64**	362,002	5243,366	233,68	131,2
	OCL Complete		328759	1878834	205922	2415336
	CPU		512	512	509	416
	OCL Kernel	**128**	541,504	7240,69	1726,76	276,032
	OCL Complete		278203	1836888	199309	2488705
	CPU		2128	2128	4510	1951
	OCL Kernel	**256**	2235,212	7787,182	8375,96	844,608
	OCL Complete		378283	2041176	201371	2541547
	CPU		7101	7101	11192	8131
	OCL Kernel	**512**	12493,33	13115,058	37074,04	3118,144
	OCL Complete		334073	1947213	234067	2517003
	CPU		20188	20188	71016	20873
	OCL Kernel	**1024**	27079,768	72173,08	178981,12	12030,272
	OCL Complete		314212	1841574	373441	2558668
	CPU		71987	71987	328348	90135
	OCL Kernel	**2048**	53299,28	162909,41	1084668,6	47673,216
	OCL Complete		388565	2024182	1245152	2511559
	CPU		300997	300997	1260564	360057
	OCL Kernel	**4096**	396307,69	560541,38	5286147,4	405504,19
	OCL Complete		773734	3108758	5767121	3043347
	CPU		1086395	1086395	4988157	1459177

Comparing Intel Xeon CPU and NVIDIA Tesla, Intel's processing time (706,03 μs) is five times longer than NVIDIA's (142,528 μs) for that reason NVIDIA Tesla machine is best machine to process Convolution algorithm's in the fastest time, shown in table 10.

Table 10. Comparison of a different machines, image sizes and filter size 21

			Intel Xeon CPU E5-2670	Intel Xeon Phi Accelerator	AMD A10-6800K APU	NVIDIA Tesla K20c
		Image Size				
CONVOLUTION	**Filter size**		**21**	**21**	**21**	**21**
	OCL Kernel	**64**	706,03	5469,202	266,68	142,528
	OCL Complete		303831	1759345	192366	2469478
	CPU		481	481	691	421
	OCL Kernel	**128**	946,38	7316,994	2721,08	342,784
	OCL Complete		328150	1714966	196885	2497327
	CPU		2741	2741	6163	2595
	OCL Kernel	**256**	2192,748	10141,146	11397,04	1113,344
	OCL Complete		294772	2002995	207021	2507190
	CPU		5723	5723	23646	7264
	OCL Kernel	**512**	27297,67	15269,058	51168,6	4334,528
	OCL Complete		380174	2029244	238321	2595488
	CPU		32090	32090	101816	31704
	OCL Kernel	**1024**	27462,618	79858,178	257671,88	16676,864
	OCL Complete		352146	2014849	453226	2520625
	CPU		127797	127797	443404	134035
	OCL Kernel	**2048**	82274,776	175819,61	1589814,9	66319,488
	OCL Complete		373142	2291679	1703555	2534732
	CPU		389726	389726	1735709	535571
	OCL Kernel	**4096**	423403,44	730200,29	6939197,5	586577,41
	OCL Complete		857863	3267223	7632370	3102151
	CPU		1601745	1601745	6983262	2404844

Table 11 will show times given for different image sizes processed through edge detector. Shortest time for processing smallest image size (64) have AMD APU (128,36 μs) but after that NVIDIA Tesla is quicker than AMD APU in processing edge detector algorithm's which means that NVIDIA is the best option for processing. Comparison between Intel Xeon CPU and NVIDIA for largest image size is that NVIDIA need four times shorter processing time than Intel.

Table 11. Comparison of a different machines and image sizes

	CANNY EDGE DETECTOR							
	Image size	**64**	**128**	**256**	**512**	**1024**	**2048**	**4096**
Intel Xeon CPU E5-2670	OCL Kernel	485,753	906,028	5571,119	19223,245	56312,218	205288,59	1001529,937
	OCL Complete	556397	491046	464157	557000	602702	695877	1508299
	CPU	655	2615	9693	56410	227117	719790	3096220
Intel Xeon Phi AcceleratoR	OCL Kernel	6003,487	35710,111	41770,596	22558,071	113043,02	198226,35	645568,293
	OCL Complete	2235530	2153419	2477382	2393045	2533114	2701620	3945746
	CPU	542	2158	9645	44671	183638	774580	3030218
AMD A10-6800K APU	OCL Kernel	128,36	621,96	2651,14	11841,18	53676,34	286286,12	1566731,84
	OCL Complete	242407	249498	248729	249023	305810	567324	1945006
	CPU	1322	5328	12474	62127	289346	1160982	4480789
NVIDIA Tesla K20c	OCL Kernel	131,712	285,248	971,616	3641,536	14084,928	56296,928	245075,2
	OCL Complete	2501961	2543469	2616593	2605837	2613994	2633427	2891309
	CPU	602	2506	12710	52045	200553	843081	3185211

For medium sized images NVIDIA Tesla (327,008 µs) need ten times shorter time than AMD APU (7119,58 µs) to process foveal algorithm like shown in table 12. Comparing AMD APU and Intel Xeon CPU, Intel need four times longer time to process same image size. Second largest images has similar case, to process those images NVIDIA Tesla need five times shorter time. From those results it is very easy to conclude that NVIDIA's machine is the best option for processing.

Table 12. Comparison of a different machines and image sizes

	FOVEAL							
	Image size	**64**	**128**	**256**	**512**	**1024**	**2048**	**4096**
Intel Xeon CPU E5-2670	OCL Kernel	391,562	1122,36	7498,012	27667,953	103708,22	419183,62	1790298,885
	OCL Complete	375769	329189	342325	344465	439009	774385	2180628
	CPU	5343	14870	87894	342888	1593705	9060783	39626409
Intel Xeon Phi AcceleratoR	OCL Kernel	2584,316	3838,421	32964,239	79529,992	278726,37	1595469,2	7249933,938
	OCL Complete	2084016	1939519	1807984	2063507	2123019	3578843	9712812
	CPU	2503	12033	86855	361532	1582344	9100305	39558219
AMD A10-6800K APU	OCL Kernel	244,24	937,72	7119,58	30409,92	138845,98	575080,62	3067267,26
	OCL Complete	204948	214037	218927	246631	351805	819656	3435066
	CPU	14371	80993	513971	2531505	11448760	68023317	296427453
NVIDIA Tesla K20c	OCL Kernel	63,808	125,984	627,008	2724,672	12327,104	70313,728	312065,408
	OCL Complete	2553948	2457872	2581091	2525828	2443358	2643430	3014705
	CPU	1908	12824	58179	297544	1254212	7370028	32390288

In the table 13 are given speedUps for the convolution algorithm, where is shown that biggest speedup is at AMD APU machine (4.18) at filter size 9 and smallest speedup has Intel Xeon Phi (0.03) at filter sizes 3, 5, 7 and 9. In every other image size is similar situation but speedups are bit higher with same machine. Highest speedUps vary but they vary in different filter sizes processed with same machine NVIDIA Tesla. All speedUps are also shown graphically in the figure 26.

Table 13. Comparison of a different machines, image sizes and filter sizes

	Image size	**64**					
	Filter size	**3**	**5**	**7**	**9**	**15**	**21**
conv1D	OCL-Intel Multicore	0,54	0,43	0,45	0,44	1,41	0,68
	OCL-Intel MIC	0,03	0,03	0,03	0,03	0,10	0,09
	OCL-AMD APU	1,35	1,06	2,59	4,18	2,18	2,59
	OCL-NVIDIA K20c	1,04	0,89	1,78	2,33	3,17	2,95
	Image size	**128**					
	Filter size	**3**	**5**	**7**	**9**	**15**	**21**
conv1D	OCL-Intel Multicore	0,38	0,60	1,35	2,94	3,93	2,90
	OCL-Intel MIC	0,05	0,08	0,16	0,19	0,29	0,37
	OCL-AMD APU	1,06	0,60	0,66	1,94	2,61	2,26
	OCL-NVIDIA K20c	3,03	2,53	3,18	5,28	7,07	7,57
	Image size	**256**					
	Filter size	**3**	**5**	**7**	**9**	**15**	**21**
conv1D	OCL-Intel Multicore	0,97	1,58	3,13	1,96	3,18	2,61
	OCL-Intel MIC	0,27	0,39	0,58	0,73	0,91	0,56
	OCL-AMD APU	0,82	0,79	1,63	1,31	1,34	2,07
	OCL-NVIDIA K20c	4,03	5,67	7,31	5,30	9,63	6,52

Table 14. Comparison of a different machines, image sizes and filter sizes

	Image size	**512**					
	Filter size	**3**	**5**	**7**	**9**	**15**	**21**
conv1D	OCL-Intel Multicore	0,2	0,52	1,34	0,55	1,616	1,176
	OCL-Intel MIC	0,36	0,72	0,88	0,83	1,54	2,10
	OCL-AMD APU	1,48	1,31	1,07	1,89	1,92	1,99
	OCL-NVIDIA K20c	5,57	5,40	6,17	6,05	6,69	7,31
	Image size	**1024**					
	Filter size	**3**	**5**	**7**	**9**	**15**	**21**
conv1D	OCL-Intel Multicore	0,69	0,69	1,58	0,92	2,66	4,65
	OCL-Intel MIC	1,09	1,16	0,72	0,97	1,00	1,60
	OCL-AMD APU	0,89	1,78	1,76	1,89	1,83	1,72
	OCL-NVIDIA K20c	5,87	6,35	6,73	6,84	7,49	8,04
	Image size	**2048**					
	Filter size	**3**	**5**	**7**	**9**	**15**	**21**
conv1D	OCL-Intel Multicore	0,84	1,75	1,59	2,97	5,65	4,74
	OCL-Intel MIC	0,54	1,22	1,26	1,84	1,85	2,22
	OCL-AMD APU	0,57	1,09	1,12	1,20	1,16	1,09
	OCL-NVIDIA K20c	6,42	6,62	6,62	7,44	7,55	8,08
	Image size	**4096**					
	Filter size	**3**	**5**	**7**	**9**	**15**	**21**
conv1D	OCL-Intel Multicore	0,41	0,87	1,14	0,92	2,74	3,78
	OCL-Intel MIC	0,89	1,09	1,11	1,45	1,94	2,19
	OCL-AMD APU	0,42	0,80	0,80	0,93	0,94	1,01
	OCL-NVIDIA K20c	2,54	3,29	3,28	3,47	3,60	4,10

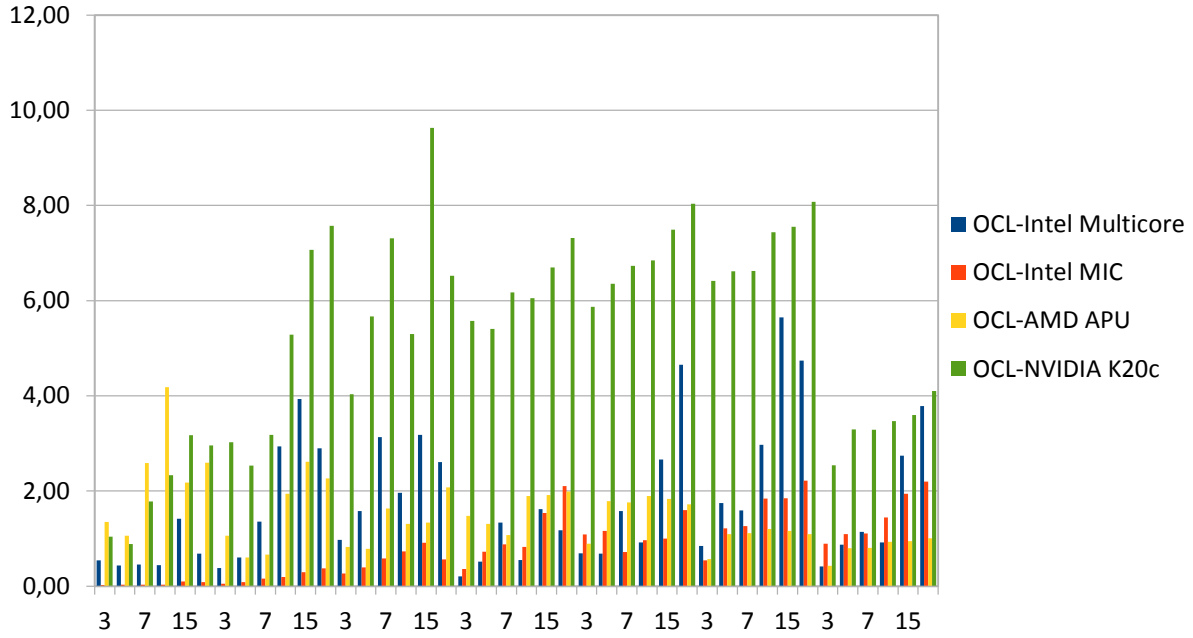

Figure 29. SpeedUp in Convolution

In the table 15 are compared speedUps for edge detector algorithm which has best speedUps at NVIDIA Tesla machine. Higher speedup is at second large image size and is 14.98 what is better shown graphically in figure 27. Smallest speedUps are at Intel Xeon Phi machine and contains 0,1 at the image size 64, 0,06 at image size 128 and 1,98 at image size 256.

Table 15. Comparison of a different machines and filter sizes

	Filter size	64	128	256	512	1024	2048	4096
EDGE DETECTOR	OCL-Intel Multicore	1,3	2,89	1,74	2,93	4,033	3,506	3,091
	OCL-Intel MIC	0,1	0,06	0,23	1,98	1,624	3,908	4,694
	OCL-AMD APU	10	8,57	4,71	5,25	5,391	4,055	2,86
	OCL-NVIDIA K20c	4,6	8,79	13,1	14,3	14,24	14,98	13

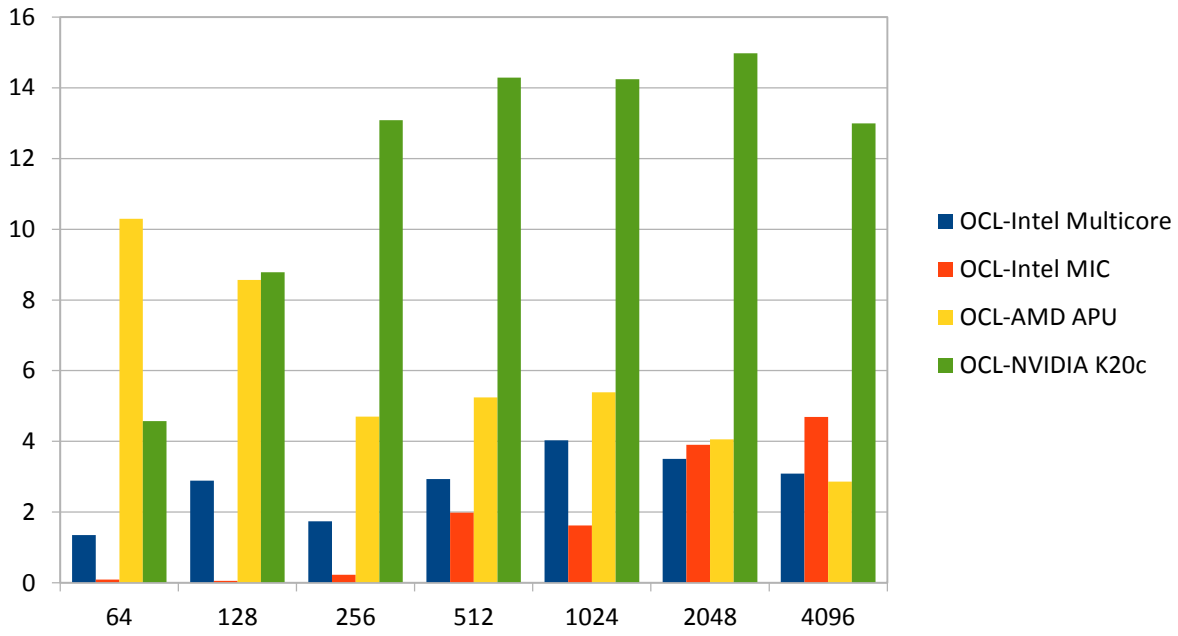

Figure 30. SpeedUp in edge detector algorithm

Compared speedUps for foveal algorithm are given in the table below. For foveal algorithm there is an exception for highest speedup this is at AMD APU which is 118,3. Smallest speedup is at Intel's Xeon Phi machine and it is 0,1 for image size 64, 0,06 for image size 128. All those results are also shown in figure 28 for better visual understanding.

Table 16. Comparison of a different machines and filter sizes

	Filter size	64	128	256	512	1024	2048	4096
FOVEAL	OCL-Intel Multicore	14	13,2	11,7	12,4	15,37	21,62	22,13
	OCL-Intel MIC	0,1	0,06	0,23	1,98	1,624	3,908	4,694
	OCL-AMD APU	59	86,4	72,2	83,2	82,46	118,3	96,64
	OCL-NVIDIA K20c	30	102	92,8	109	101,7	104,8	103,8

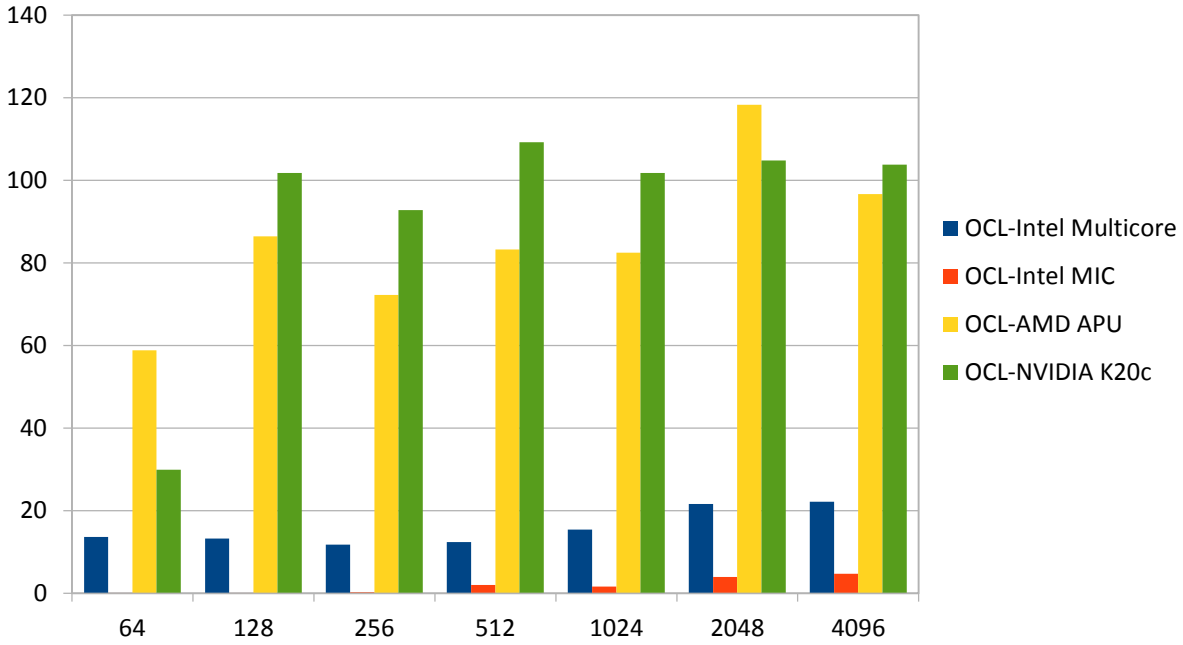

Figure 31. SpeedUp in foveal algorithm

5. Conclusion

Aim of this thesis was to test algorithms on the PC to see if they are processing in the way that could help visual impaired people and to test them onto different CPU and GPU to see if there is an opportunity to implement them onto some low cost medical device. To conclude is that through developed algorithms images are processed for metrics calculation which give us closer look to similarity of the developed algorithms. In the thesis three different metrics are used to get closer look to the filtered image. Not even one metric didn't show exactly the same image. Best results has images with bigger sigma size and bigger filter size than 5. For more accurate results, further testings need to be done to see exact parameters for more similar image.

There are four different machines used in this thesis to test developed algorithms. Every single machine processed every algorithm but neither of them have same time to process for that reason they have some differences. NVIDIA Tesla was the best machine for processing because of his Local memory size which is bigger than in the rest of machines. Therefore NVIDIA is recommend to be used in this type of processing images but this are only experiments. For some kind of medical device which can be used at daily bases, what was the aim of this thesis, that machine is too big. For that reason next experiments and testings need to be done with smaller, more portable machines which could be easily implemented into smaller device like smartphones and tablets.

For final conclusion of this thesis, results are better than expected. Similarity of images is very high and could be even higher with detailed parameters. In the hardware implementation testings showed that NVIDIA has better performance than other machines.

6. Future work

While this thesis has demonstrated the potential of edge detection algorithms and its modifications in it, many opportunities for extending the scope of this thesis remain. First of all, more parameters should be included in the research. Due to certain limitations, expertise in field of computer science and programming, this master thesis is focused on low level of software and hardware implementations.
This master thesis should test smaller machines which could be implemented into smaller portable devices. Smaller machines are way convenient for devices like smartphones or tablets which was the aim of this thesis. To sum up, future work should consider to improve current algorithms and implement them into smaller devices that are used nowadays.

7. Bibliography

1. Martins, J. C. & Sousa, L. A. (2009). Bioelectronic Vision Retina Models, Evaluation Metrcis, and System Design
2. Hui, W., Xu-Dong, G. & Qingsong, Z. (2010). Main Retina Information Processing Pathways Modeling, IEEE (pp. 318-324)
3. http://www.emedicinehealth.com/anatomy_of_the_eye/article_em.htm#eye_anatomy_introduction (last request: 18.05.2014)
4. Gaster, B. R., Howes, L., Kaeli, D., Mistry, P., Schaa, D. (2012). Heterogeneous Computing with OpenCL
5. Scarpino, M. (2012). OpenCL in Action
6. Al-Atabany, W., A Memon, M., Downes, S. M., & Degenaar, P. A. (2010). Designing and testing scene enhancement algorithms for patiens with retina degenerative disorders
7. Fleck, M.M. (1992). Some Defects in Finite-Difference Edge Finders
8. Canny, J. (1986). A Computational Approach to Edge-Detection
9. Holger, W., Olsen, S.C., Gooch, B. (2006) Real-time video abstraction
10. http://www.emedicinehealth.com/age-related_macular_degeneration-health/article_em.htm (last request: 20.06.2014)
11. http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/bench/gray/gm_2/main.html (last request: 20.07.2014)
12. Wackerly, D. & Scheaffer, W., (2008). Mathematical Statistics with Applications
13. Welstead, S. T. (1999). Fractal and wavelet image compression techniques
14. Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P., (2004) Image quality assessment: From error visibility to structural similarity
15. http://www.emedicinehealth.com/age-related_macular_degeneration-health/page6_em.htm#Treatment Overview (last request: 20.06.2014)
16. http://www.emedicinehealth.com/script/main/art.asp?articlekey=128643&ref=128649 (last request: 21.06.2014)
17. http://www.emedicinehealth.com/script/main/art.asp?articlekey=128639&ref=128649 (last request: 21.06.2014)
18. Garaas, T.W. & Pomplun, M. (2007). Retina-Inspired Visual Processing

19. http://homepage.cs.uiowa.edu/~cwyman/classes/spring08-22C251/homework/canny.pdf (last request: 29.08.2014)
20. http://www.pc-specs.com/gpu/AMD/APU_Family/Radeon_HD_8670D/1787 (last request: 1.09.2014)
21. http://www.mathworks.com/help/images/detect-edges-in-images.html (last request: 1.09.2014)
22. http://www.ijcte.org/papers/100-G205-621.pdf (last request 1.09.2014)
23. http://www.generation5.org/content/2002/im01.asp (last request 28.08.2014)
24. https://developer.apple.com/Library/ios/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html (last request: 29.08.2014)
25. http://perso.limsi.fr/vezien/PAPIERS_ACS/canny1986.pdf (last request: 3.09.2014)