Open Platform to Detect and Monitor Macular Disorders

Navid Mohaghegh

A Dissertation Submitted to the Faculty of Graduate Studies in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Graduate Program In Electrical Engineering and Computer Science,

York University, Toronto, Ontario

June 2019

ABSTRACT

Macular disorders (MDs) such as Age-related Macular Degeneration (AMD) and Central Serous Retinopathy (CSR) cause Visual Distortions (VDs) while affecting human vision and the quality of life. Home-monitoring helps with disorder early detection and possibly slow down or even progress prevention while reducing the risk of vision loss and medical management costs.

We addressed the challenge of developing accurate, rapid, and low-cost home monitoring technology for the detection and progress assessment of MDs. The proposed methods allow the detection of small VDs using a novel approach called NGRID. The proposed NGRID platform is a unified software and hardware system that assist eye-care professionals in running the visual tests from hospitals or remotely at patients' home. Advanced programming techniques such as Standard Vector Graphic (SVG) and voice recognition were used to develop the required software. The high security, capacity, and availability of the computer cluster running NGRID enable the access of millions of people to run the test and assess the progress of their MDs at home. NGRID sends the results to the medical practitioner to better manage the patients.

We tested CSR patients using NGRID. The patients were asked to answer if they see the VD test frames wavy or with missing parts. Patient's voice is processed to extract the answers and detect metamorphopsia or scotoma, and results displayed in a graph called heatmap, which visually shows how the visual field is affected. Furthermore, we successfully verified the heatmaps with patients' Optical Coherence Tomography (OCT) images, which is the golden standard methodology for MD diagnostic. We confirmed the location of the detected VDs with the patients once they gain normal vision.

The proposed NGRID research platform can offer significant advantages for home monitoring and subsequently, control of MDs. NGRID opened new avenues towards the generation of first MD big data suitable for medical industries. Finally, NGRID aims to offer medical practitioners better ways to monitor patients at home, where using OCT is not possible. Clinical trials for NGRID on other MDs such as AMD may allow medical practitioners for faster intervention when Anti-Vascular Endothelial Growth Factor (Anti-VEGF) is needed.

DEDICATION

I like to thank my wife. Without her patience and support, this work would not have been possible. I also want to thank my parents for their unconditional love and continued support during this process.

ACKNOWLEDGMENTS

I want to express my gratitude to all who have helped me to complete this dissertation. This thesis would not be possible without the encouragement and assistance of my mentors. I am especially indebted to my supervisor (Dr. Ebrahim Ghafar-Zadeh) for his untiring effort, support, and constructive feedback which motivated me to complete my thesis and publish my papers. I sincerely appreciate him for letting me do my Ph.D. research in Biologically Inspired Sensors and Actuator (BioSA) laboratory that made me discover the beauty of merging computer science and biomedical engineering. I cannot thank him enough for all his constant efforts and valuable comments in all theoretical and experimental aspects of my thesis. I am also deeply indebted to my co-supervisor (Dr. Sebastian Magierowski) for his valuable inputs and critiques during the committee meetings, presentations, and writing and submitting my conference and journal papers derived from this thesis.

I like to thank Dr. Peter Kertes, Chief of Ophthalmology at Sunnybrook hospital for his valuable comments. I also like to thank my supervisory committee members (Dr. Mokhtar Aboelaze and Dr. Jia Xu) for their valuable inputs and critiques during the committee meetings. Finally, I like to thank my defense committee members (Dr. Regina Lee, Dr. Benoit Gosselin, Dr. Ali Sadeghi-Naini, Dr. Mokhtar Aboelaze, Dr. Jia Xu, Dr. Sebastian Magierowski, and Dr. Ebrahim Ghafar-Zadeh) for accepting to evaluate this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

3D-CTAG	Three-dimensional Computer-automated Threshold Amsler Grid
AAG	Accelerated Amsler Grid
AES	Advanced Encryption Standard
AFFS	Advanced Fringe Field Switching
AG	Amsler Grid
AMD	Age-related Macular Degeneration
API	Application Programming Interface
ASYNC	Asynchronous
AntiVEGF	Anti-Vascular Endothelial Growth Factor
CPU	Central Processing Unit
CSC	Central Serous Chorioretinopathy, also known as CSCR and CSR
CSCR	Central Serous Chorioretinopathy, also known as CSC and CSR
CSR	Central Serous Retinopathy, also known as CSCR and CSC
CSS	Cascading Style Sheets
DME	Diabetic Macular Edema
EHR	Electronic Health Record
ETDRS	Early Treatment Diabetic Retinopathy Study
FPGA	Field Programmable Gate Array
FS	File System
GCL	Ganglion cells Layer in the retina
GIMP	GNU Image Manipulation Program
GMIS	Graphical Macular Interface System
GNU	GNU's Not UNIX
Gbps	Giga-bit per seconds
HBA	Host Bus Adapter
HCA	Host Controller Adapter
HDD	Hard Disk Drive
HFFS	High-Transmittance Fringe Field Switching
HMD	Head Mounted Display
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
INL	Inner Nuclear Layer in the retina
IPL	Inner Plexiform Layer in the retina
IoT	Internet of Things
JS	JavaScript
LB	Load Balancer
LCD	Liquid Crystal Display
LMS	Long-Medium-Short Waves
LogMAR	The Logarithm of the Minimum Angle of Resolution

LTE	Long Term Evolution
LVS	Linux Virtual Server
LXC	Linux Containers
LZ	Lempel and Ziv compression
MCPT	Macular Computerized Psychophysical Test
MCU	Micro Controller Unit
NFL	Nerve Fiber Layer in the retina
NoSQL	Not only SQL
OCT	Optical Coherence Tomography
ONL	Outer Nuclear Layer in the retina
OPL	Outer Plexiform Layer in the retina
OS	Operating System
OVS	Oscillating Visual Stimuli
OpenCV	Open Computer Vision
OpenZFS	Open Zettabyte File System
PCM	Pulse-Code Modulation
PHP	Preferential Hyperacuity Perimeter
REB	Research Ethical Board
RPE	Retinal Pigment Epithelium in the retina
RSA	Rivest, Shamir, and Adelman algorithm
SHA	Secure Hash Algorithm
SQL	Structured Query Language
SSD	Solid State Drive
SSL	Secure Socket Layer
SSO	Single Sign-On
SVG	Standard Vector Graphic
TAG	Threshold Amsler Grid
TLS	Transport Layer Security
TOD	Task-Oriented Fixation
UI	User Interface
VD	Visual Distortion
VDT	Visual Distortion Test
VR	Virtual Reality
WWW	World Wide Web
XML	Extensible Markup Language
ZIL	ZFS Intent Log

1 INTRODUCTION

Macular disorders such as Myopic Maculopathy, Macular Holes, Diabetic Macular Edema, Age-Related Macular Degeneration (AMD) and Central Retinopathy (CSR) cause visual distortion (VD) in their early stages [1], [2]. In their advanced phases, macular disorders cause central vision loss. Among these retinal conditions, AMD is the leading cause of blindness and will affect the central vision of 196 million people over the age of 65 worldwide by 2020 [3]. The central vision effects are very diverse and can impact patients' basic mobility, road crossing, driving, scene viewing, reading, working with computers and electronics and even cause depression [4], [5]. Patients with macular disorders may require low vision rehabilitation assistance to achieve some simple goals like news reading, leisure and entertainment, computer/mobile use for personal communication, and correspondence. Therefore, the macular disorders severely affect the quality of life and raise the direct and indirect medical management cost of macular disorders [6]. Early detection of macular disorders is crucial as close monitoring allows for intervention before irreversible damage occurs. In other words, the earlier the VD is detected, the better the treatment of the macular disorder can be managed [7]. Currently, various imaging methods including Optical Coherence Tomography (OCT) are used as the standard assessment tools to monitor the progress of the macular disorders [8]. Although OCT is a high-resolution technique to show any deformation in the cross-section of the retina, OCT is not a low cost and easy to use method and only a medical expert can use it. Therefore, it cannot be a choice for home-monitoring purposes. Despite significant advances in biomedical technologies, still, the early detection and home-monitoring of VD associated with various macular disorders is a significant challenge. Computerized Graphical Interface (CGI) methods have recently attracted eye care professionals' attention as a low cost, home-use method for rapid detection of damage in the visual field as well as monitoring the progress of macular disorders.

1.1 VD CAUSED BY MACULAR DISORDERS

As seen in Figure 1a, light passes through the cornea and the lens thus allowing an image to be focused on the retina. The retina changes this light energy into a signal that can be transmitted to the brain via the optic nerve on the top layer of the retina. The macula is a small area of the retina approximately 4 mm from the optic disk. This area contains the fovea and provides the sharpest central vision. When the gaze is fixed on an object, the macula, the lens, and the object are in a straight line. The fovea is a shallow, capillary-free, depression in the center of the macula approximately 1.5 mm in diameter and contains more than 199,000 to 300,000 cones per square millimeter [9], [10]. As seen in Figure 1c, the deflection of the retina near the fovea might be due to a macular disorder such as Central Serous Chorioretinopathy (CSR) or Age-related Macular Degeneration (AMD) resulting in the distortion of the created image on the retina. Figure 1 shows the main components of the retina when a deflection has occurred due to a macular disorder. It is noteworthy that this deformation is the source of VD in the visual sharpness, color and even the distortion of straight lines [11], [12].

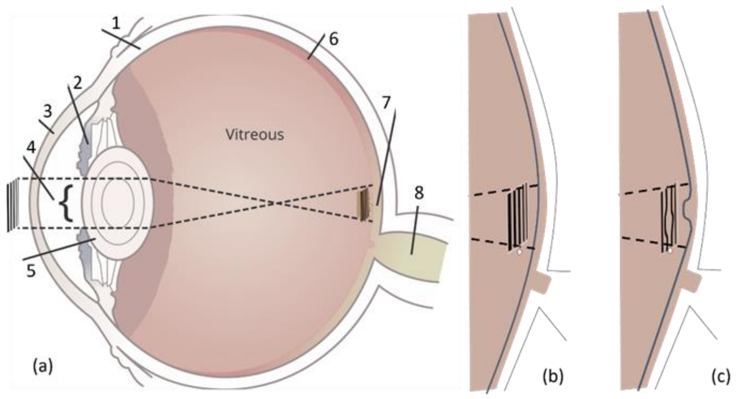


Figure 1 - (a) Simplified schematic of the Eye Structure (b) healthy macula (c) unhealthy macula (the deformed retinal basement due to a macular disorder such as CSR). Various parts of the eye including (1) sclera, (2) iris, (3) cornea, (4) pupil, (5) lens, (6) entire retina, (7) macula and in its center fovea, (8) optic nerve.

1.2 RELATED MACULAR DISORDERS

Macular Edema, CSR, and AMD are three well known macular disorders that cause deformation in the retina. This section outlines these macular disorders [13]–[18]. Macular Edema occurs when the fluid collects within the macula leading to swelling of the retina's tissue and subsequently results in visual distortion [10], [19]. Diabetic Macular Edema (DME), Cystoid Macular Edema (CME) are two subtypes of macular edema. DME or swelling of the retina occurs in diabetes from the leakage of fluid from blood vessels. In CME, fluid accumulates in cyst-like spaces within the macula. The collection of fluid results in the deformation of the retina with a significant effects on vision. On the other hand, CSR is a condition that occurs because of leakage of fluid at the level of the retinal pigment epithelium (RPE) [16], [17]. It is noteworthy that the epithelium is one of the four main types of tissues in the body and the RPE is the pigmented cell layer that nourishes retinal visual cells. As seen in Figure 2c, this leakage results in deformation of the retina similar to that imposed by Macular Edema. There is usually no underlying cause, however, in most cases the condition is preceded by work-related mental stress. In AMD, drusen – yellow colored pigments made up of lipids – builds under the basement membrane of the retinal pigment epithelium causing significant vision loss [13], [15]. As the macula is the central part of the retina consisting of the highest concentration of photoreceptors, the emergence, and growth of drusen over the years causes loss of vision. AMD can be classified into two categories, dry and wet. Dry drusen are small, round and discrete whereas in wet AMD, drusen are larger with indistinct margins. As shown in Figure 2d, the presence of drusen can displace the mosaic retina causing patients to complain about the observation of wavy lines [20], [21]. In addition to AMD, CSR, and Macular Edema, other types of macular disorders may result in the deformation of the retina and thus VD in the early stages of the development of the disease [10], [12].

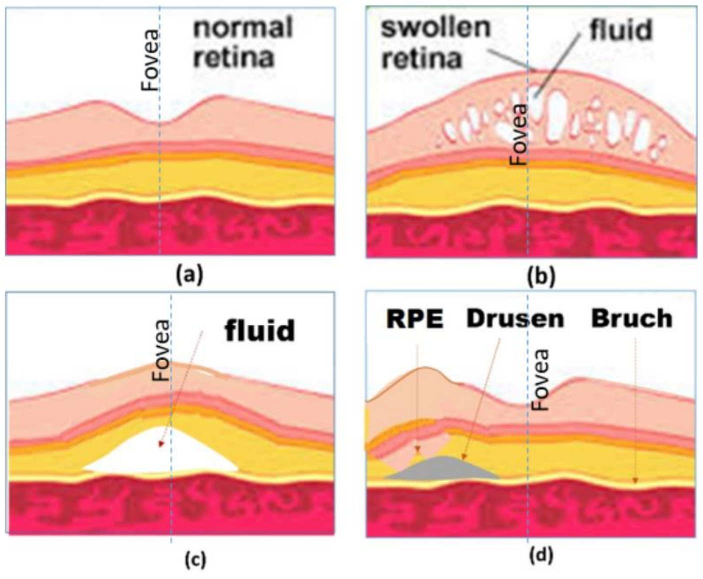
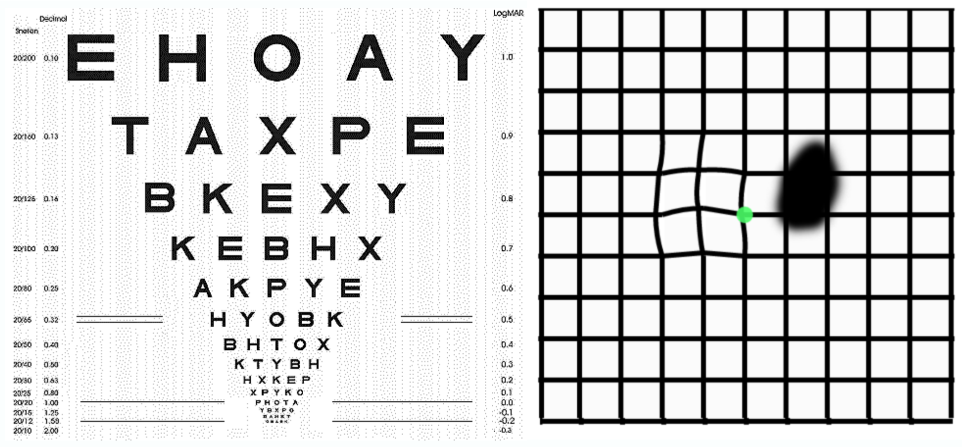


Figure 2 - Illustration of retina section of (a) normal eye and the eye suffers from (b) Edema, (c) Central Serous Retinopathy (CSR) and (d) AMD. In this figure, RPE is the retinal pigment epithelium.

1.3 VD ASSESSMENT IN PATIENTS WITH MACULAR DISORDERS

Among various visual assessment methods, the visual acuity test [22] is widely used to evaluate the smallest letters that a patient can read on a standardized Snellen chart or the more accurate Logarithm of the Minimum Angle of Resolution (LogMAR) chart as displayed in Figure 3a. The LogMAR chart called Early Treatment of Diabetic Retinopathy Study Chart (ETDRS) was created as a more accurate means of measuring visual acuity. Despite the great advantages of ETDRS, the studies show that visual acuity is a poor indicator of macular disorders including AMD [23]–[25]. This is because the acuity test can only evaluate the presence of AMD, but it does not provide any information about the location or the progression of AMD

In 1947 Marc Amsler created a printed grid for the detection of AMD [10], [26], which he called the Amsler Grid [27]. Currently, the Amsler Grid (AG) is the gold standard of home monitoring for patients suffering from macular disorders. The grid is used in a monocular fashion with their best near correction (see Figure 3b) while the participants are asked to fixate at the center. An AG sheet approximately 8x8 inches in size should be held 20cm from one eye when the other eye is covered. In this test, first, the patients should wear their eyeglasses and look at the dot in the center of the AG sheet. Then, they should detect any missing corners or any lines that are wavy or missing. The AG test can successfully monitor metamorphopsia and scotomas as seen in Figure 3b [25]. Although it is widely used, the Amsler Grid shows low accuracy for detecting VD changes smaller than the distance between the horizontal or vertical lines in patients suffering from AMD from one time to another. Moreover, AG has limitations such as cortical completion (the brain completes a partially formed image) [26], lack of accurate central fixation (looking at an object involves our macula and is called central fixation), crowding of the lines (confusion due to the presence of multiple lines), and poor patient compliance (there is no way to monitor if they actually did the test every day).



(a) (b)

Figure 3 - Visual Chart Tests: (a) ETDRS Visual Acuity Chart and (b) Amsler grid with Scotoma (right) and Metamorphopsia (left) seen on Amsler grid in AMD patients.

OCT is a noninvasive medical imaging technique used to provide an optical cross-section of the retina [29], [30]. This technique is widely used for obtaining sub-surface images of opaque materials at a resolution equivalent to a low-power microscope. By employing near-infrared light, it can capture micrometer-resolution features [10], [31]. OCT can play a key role in evaluating patients with macular disorders and those undergoing treatment [32].

Proper analysis of OCT images requires training and extensive practice, for this reason, it is not suitable for home diagnostics. Figure 2 shows the OCT illustrations of normal, AMD, and Macular Edema afflicted retinas.

1.4 SUMMARY AND ORGANIZATION OF THE DISSERTATION

Computerized graphical VD assessment methods have recently emerged to overcome the problems mentioned earlier. Despite significant advances, there are many challenges in developing and using computerized VD assessment techniques. These methods can be divided into two main groups including static and dynamic approaches that are put forward in chapter 2 along with the challenges of computerized methods. We follow up the discussion with our mathematical model in chapter 3 to better simulate the effect of deformations. The mathematical models helps us to design better VD tests. Based on the discussion and mathematical models, we introduce our approach to VD tests, namely the NGRID platform in chapter 4. We review test results of NGRID platform in chapter 5 followed up with conclusion and our future plans in chapter 6.

2 RELATED WORKS

As we reviewed in chapter 1, the Amsler Grid is used as a standard method to detect scotoma and metamorphopsia in macular disorders. In this method, the grid pattern is used as a still image while the patient fixates at the center of the display (central fixation) and tries to detect any deformation or curvy lines in the grid. In this chapter we discuss the related works around various static and dynamic methods for detection and monitoring visual deformation in the visual field.

2.1 STATIC VD ASSESSMENT METHODS

In this section we review static VD assessment methods. These methods are static in nature and does not require the graphical patterns to be displayed to the patients with a specific timing requirement.

2.1.1 ENHANCED PORTABLE AG TEST

Many efforts have been made to convert the paper-based AG test to a computerized AG test [33], [34]. Among these efforts, as shown in Figure 4a, Hirji [35] presented a portable near-eye ophthalmic device using a portable tablet computer incorporated with a phoropter rod to limit the movements of the head and allow a better view of the Amsler Grid. Despite the fact that this device allows for better fixation at the center of the grid by reducing the head movements, as with the AG charts' test, it still suffers from the completion effect. As described in [36], the brain fills or completes the gaps in the visual field of each eye. Therefore, this completion effect may cause wrong answers in this test. As shown in Figure 4b, a finger-touch tablet or smartphone can be used for directly marking any area of VDs such as metamorphopsia. By offering advantages of faster interaction with patients, this technology not only can be used in eye-care clinics but also, a portable smart tablet is suitable for self-monitoring of macular disorders at home. Despite the advantages mentioned above, this method, in addition to its problems with the completion effect, also suffers from a lack of eye-tracking to assure fixation compliance during testing.

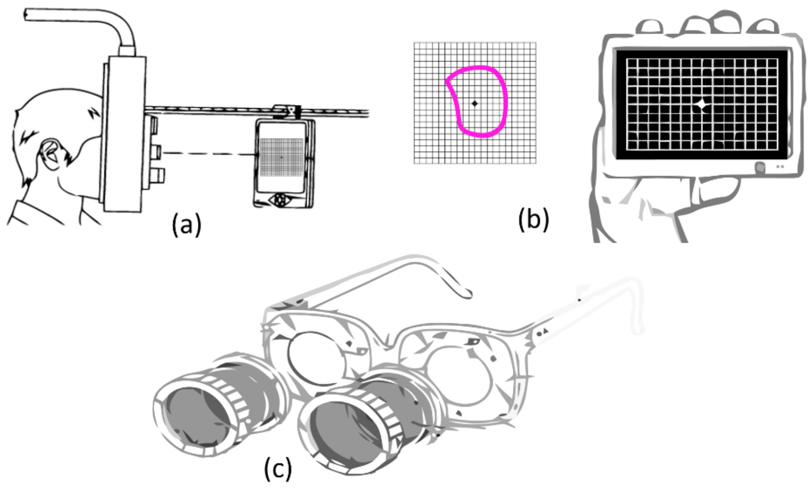


Figure 4 - (a) Near Eye Ophthalmic Device utilized with a phoropter rod to better display Amsler Grid, (b) Portable Smart Tablet Used for Evaluation of Metamorphopsia and (c) Eyeglasses Used in Threshold Amsler Grid Test Equipped with Mountable Polarized Filters.

2.1.2 THRESHOLD AMSLER GRID TEST

As described in subsection 2.1.1, the AG is directly displayed in front of the eye.

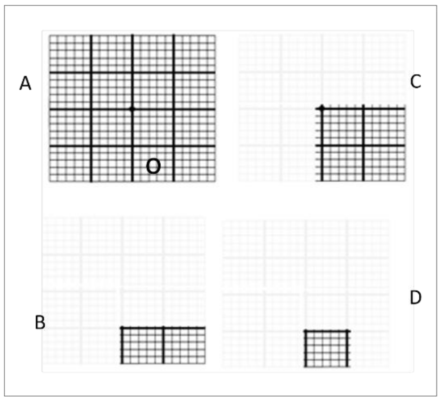
Therefore, the accuracy of detecting scotoma and metamorphopsia is limited by overall visual acuity and the performance of the eye is functioning properly or not. The Threshold Amsler Grid (TAG) test has been designed to increase the accuracy of the traditional AG tests. A TAG test can be performed through cross-polarizing filters that cause low luminance conditions in which the grid becomes barely perceptible as seen in Figure 4c [37]. It is noteworthy that the cross-polarizing filter is used for enhancing vision by filtering the reflected lights. However, this filter reduces luminance conditions. Such eyeglasses, reported by Sadun et al. [38], can vary the amount of luminance of the observed AG by the patient. Therefore, this will greatly increase the sensitivity of the patient's eyes to perceive the presence of scotomas. Although the threshold Amsler grid shows better performance in comparison with traditional AG method, 50% of all scotomas remain undetected [39].

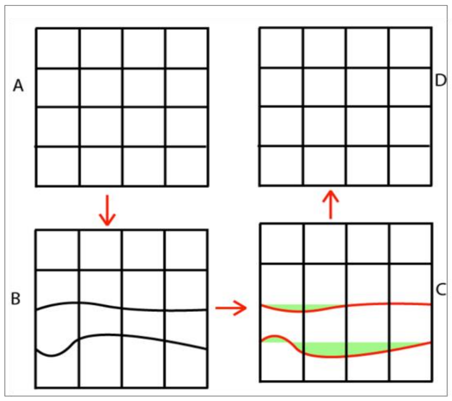
2.1.3 ACCELERATED AMSLER GRID TEST

In an AG test, the patients should explain their VD experience in order to approximately find the locations of VD in their visual fields, however, with AG test, patients cannot be provided with a quantitative analysis of their VD. One of the first steps towards quantifying these answers was taken by proposing the Accelerated AG test (AAG).

In AAG test, sub AG blocks ¼, ⅟₁₆ and ⅟₃₂ the size of the largest AG block (with 32 small blocks such as block C in Figure 5a) are displayed for covering the entire display (A, B and C in Figure 5a). Patients provide ‘yes’ or ‘no’ answers in response to viewing these blocks. The observation of a normal block with straight-lines or abnormal blocks with curvy-lines can represent the presence or absence of VDs respectively (Figure 5a). In this method, proposed by Palanker [40], at the first step, AG block A is displayed in the top-right, bottom-right, top-left, and bottom-left in order to detect the location of VDs. In the next steps, the AG blocks C and D are scanned in the detected location with VD in order to accurately detect the location of the VDs.

The accuracy of AAG method is limited to the distance between the grid lines that create the sub AG blocks. Figure 5b shows that VD can be observed in two adjunct blocks or other combinations of the smallest AG blocks. In this method, the smallest detectable metamorphopsia lesions in the visual field of the patient can be equal to the smallest AG block. By decreasing the size of the smallest block, for instance to $1/64$ or lower, the accuracy of this technique is increased, however, it will require more time to scan all blocks thus causing eye fatigue. Consequently, it results in higher difficulty to fixate on the center of the screen.





(a) (b)

Figure 5 - AG Based VD detection methods (a) Binary Amsler Interactions (b) Deformable Amsler Grid (DAG). DAG has Ability to Apply Correction Vectors to Improve Patient Vision (A) denotes the normal Grid (B) is the Grid as seen by the patient (C) is the correction vectors that are highlighted in red and correction values that are measured in green which denotes the amount of deflection from the straight grid lines (D) is the improved final grid as viewed by the patient after applying the correction vectors

By assuming a VD happens in sub AG block D (denoted with “o” in Figure 5a), after answering seven “yes/no” questions; the VD block can be identified. One may suggest a pure binary search tree approach to accelerate the speed even further. Despite the advantages of this method, this method still may suffer from the completion effect and a lack of controlled fixation which are inherited from the Amsler Grid as already explained.

2.1.4 **DEFORMABLE AMSLER GRID**

This method relies on the fact that when an AG with all straight lines (Figure 5b-A) is projected on the patient’s retina, the patient partially sees a deformed (Figure 5b-B) line. Assuming that the patient can spot the metamorphopsia displayed in Figure 5b-B, then it could be possible to project an AG with the same deformed lines (Figure 5b-C) but in the opposite direction to ease the metamorphopsia experienced by the patient as shown in Figure 5b-D. As shown in Figure 5b-C, the alternate grid has the same deformed lines, but in the opposite direction. This is achieved by applying a correction vector (highlighted in red in Figure 5b-C). Figure 5b-D is the result of the correction vector is applied.

Based on this method, Kohn et al. [41] proposed a portable device including a tablet with a deformable AG program that can interactively measure the metamorphopsia via measuring the correction vector. Based on this technique, the patient can change the location of the vertices of the grid until the grid appears substantially rectilinear to the patient. The deformations of the pattern are recorded by the program and can be retrieved for evaluation of deformation and the progress of the disorder. The inherited difficulties of the AG test such as the completion effect apply to this technique as well. Additionally, the fixation at the center is another important problem; this is because the patient’s focus will move toward the deformation location while interactively changing the deformations.

In the next section we review the VD assessment methods that have more dynamic requirements on how the patterns should be projected.

2.2 DYNAMIC VD ASSESSMENT METHODS

The second group of VD assessment methods relies on projecting the stimulation pattern over a short time and indirectly measuring the visual performance based on the ability of the eye in detecting the stimuli. These methods are dynamic in nature. It is noteworthy that the projection time of graphics to patients' retina in dynamic methods is much shorter the static methods.

2.2.1 PHP TEST

Preferential Hyperacuity Perimeter (PHP) is another computerized VD assessment method that utilizes hyperacuity to detect changes and distortions in the central retina. Hyperacuity or Vernier Acuity is the ability to detect small misalignments between two lines. This acuity based measurement method is found to be much more accurate than Snellen type acuity (see chapter 1) VD detection in patients suffering from macular disorders [39], [42], [43]. Hyperacuity can also be defined as the capability of the visual sensor to transcend sampling limits set by discrete receiving elements. As seen in Figure 6a, if a dotted line with a small bump and a large bump (one bump is very small in comparison to the other one) is presented to an individual for 500 ms, the individual may not able to perceive the smaller bump due to the presence of the larger bump. PHP relies on this concept and simply flashes a dotted line across the screen for a short period (Figure 6b). This dotted line contains a small visual distortion (e.g., a bump in the line that can be as small as 0.3 degrees). Patients with a macular disorder might ignore this small bump due to the presence of a bigger VD caused by the macular disorder. Patients should record any perceived distortions by clicking or drawing the "bump" they have observed on the screen. A macular map, also known as a heatmap, (Figure 6c) is obtained with quantitative values of the area and intensity of the metamorphopsia. The higher the number of clicks around the areas that no artificial bump is presented, the higher the chance of metamorphopsia presence in those regions, and the darker that area is rendered in heatmap [44].

2.2.2 **3D-CTAG TEST**

Three-dimensional Computer-automated Threshold Amsler Grid (3D-CTAG) is another visual field test [33], [45]. Patients, with one eye covered, are positioned in front of a touch-sensitive computer screen on a head-chinrest. They are asked to finger-trace the areas of an Amsler grid that are missing from their field of vision (Figure 7a). Various degrees of contrasts of the Amsler grid are presented by repeating the test at different grayscale levels. It is noteworthy that the number of vertical and horizontal lines and their width can be changed to achieve various contrasts.

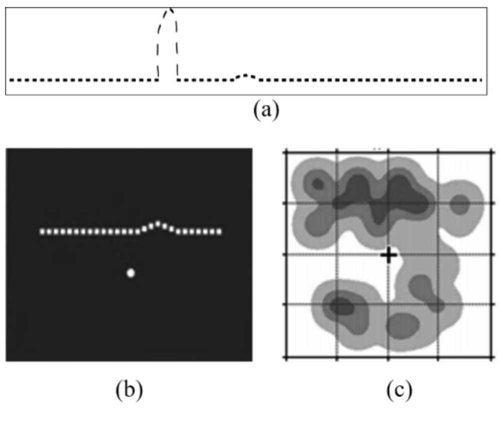
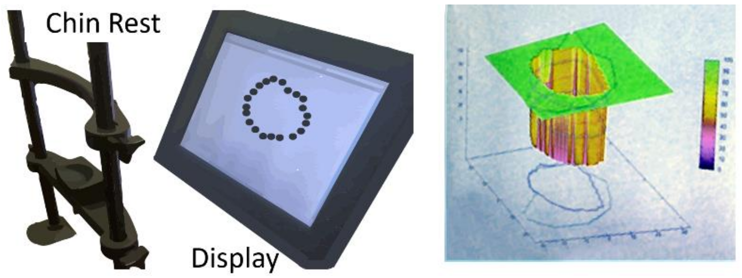


Figure 6- (a) Presenting a dotted line with the presence of a small bump and another large bump for 500ms to healthy individuals. (b) PHP Test Apparatus – a dotted line with a very small artificial bump shortly flashed on the screen

and patients should record where they see a bump using stylus pen (c) is the generated PHP heatmap that shows the location and severity of visual distortions in patient's visual field.



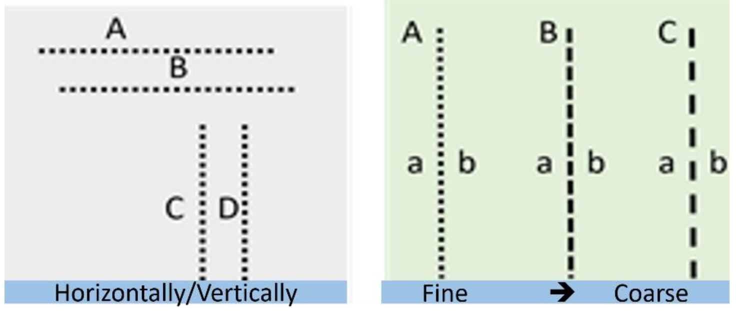
(a) (b)

Figure 7 - 3D-CTAG Apparatus – (a) shows the overall apparatus with chin rest to limit the head movements (b) final result of the test which highlights the severity of affected visual distortion areas

3D-CTAG is used to test the 25 degrees of the central visual fields at different contrast levels. Among the efforts made in this direction, Robison *et al.* reported a 3D-CTAG test with five different contrast levels [45] (5%, 10%, 20%, 40% and 100% contrast). The detected VD area with each contrast was mapped to one of five different levels in the z-axis direction denoted with colors in Figure 7b. This method can reflect the effect of contrast on the accuracy with which a VD area is detected; however, the visual fixation at the center is still a problem. This is because the attention of the patient jumps from the center of the display to its surrounding area as shown in Figure 7a.

2.2.3 MACULAR COMPUTERIZED PSYCHOPHYSICAL TEST

Macular Computerized Psychophysical Test (MCPT) is another visual field test that acts based on hyperacuity [39], [46]. The patient has to bring the mouse cursor to the center point on the screen to view the next flashed dotted line. This task initiates a stimulus and simultaneously a forced fixation. This task is called Task Oriented Fixation (TOF) and should be repeated in order to view the next flashed dotted-line. This TOF based test ensures that the patient is going to re-fixate at the center of the screen even in the event that they momentarily lose their central fixation. A dotted line (white dots on a black background with maximal contrast) flashed at a random order across different perifoveal locations (see Figure 8a). By using a dotted line (e.g., line A in Figure 8a), the test is similar to the aforementioned hyperacuity assessment methods in accurately detecting the VDs. In MCPT the patient uses a mouse to click on a central dot, after which a new dotted line is presented at a different location (e.g., line B in Figure 8a) on the screen. The patient marks any scotoma or metamorphopsia by marking the corresponding locations on the screen with a mouse [39].



(a) (b)

Figure 8 - Stimulating patterns used in (a) MCPT and (b) M-Chart test techniques

2.2.4 **M-CHART SCORE TEST**

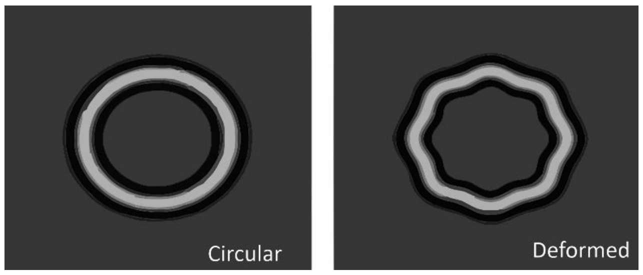
A metamorphopsia chart or so-called M-CHART is used to quantify the metamorphopsia in patients' visual field [47], [48]. The test contains 19 kinds of horizontal and vertical dotted lines with different widths and lengths that are displayed in different visual angles ranging from 0.2 to 2.0 degrees [49]. The lines are presented at 30 cm, monocularly, while patients are allowed to have their eye-glasses to achieve best near corrected vision. As shown in Figure 8b, the patient starts by seeing a vertical or horizontal dotted line in between two letters (as seen in line A, B, and C in Figure 8b) that is seen by the patient as curved or misaligned. The patient is then presented with changes in the dots from fine to coarse until the metamorphopsia disappears. The minimum visual angle needed to detect the metamorphopsia is recorded [50]. In this technique, the fixation at the center of the visual field is not trivial. Additionally, it is challenging to obtain metamorphopsia scores from those patients with visual acuities 20/100 or less or the ones suffering from very large central scotomas.

2.2.5 **SHAPE DISCRIMINATION TEST**

Wang et al. [51] proposed a method to test visual hyperacuity using shape discrimination. In this test as seen in Figure 9, one circular pattern is shown to the patient along with another similar shape that is deformed slightly.

Patients should identify the deformed pattern. The test showed that AMD patients have significantly worse performance in detecting radial deformation of the patterns when compared with normal control eyes [51].

The primary goal of this test is the early detection of a macular disorder. Very small circles can be screened in the patient's retina in order to quantify the progress of the macular disorder. However, the smaller the circles, the lower the accuracy in distinguishing normal from deformed circles.



(a) (b)

Figure 9 - Shape Discrimination (a) circular pattern and (b) another similar pattern with radial deformation of 8/2π (right).

2.2.6 POSITIONING TECHNIQUES

There were many moveable tasks introduced over the years to assess the visual field. For instance, Enoch *et al.* [52] reported a method for measuring metamorphopsia. As shown in Figure 10a, while one of the blocks is fixed, the other block is movable, and the patient should be able to align it vertically or horizontally with another one. In another effort, Schmid et al. presented the Vernier Hyperacuity Test (see Figure 10b), with a black background screen and a few white objects that are slightly misaligned and should be aligned by the patient. A more complex aligning task was proposed by Weicek *et al.* [53] who reported the square completion task with movable and immovable dots. The patient should be able to move the dots in order to form a rectangular shape as shown in Figure 10. The patient should be able to detect the dots and accurately align them horizontally and vertically. In all the above-mentioned tests with movable patterns, still, it is difficult for patients to maintain fixation at the center of the screen. Stewart [54] addressed this problem by proposing oscillating visual stimuli (OVS) to detect metamorphopsia. The test pattern (Figure 11a) is positioned in front of the patient to cover at least 40° or more of the visual field.

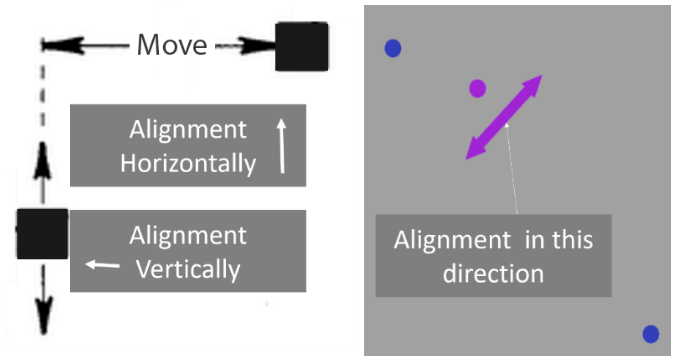


Figure 10 - Aligning Method: (a) Basic Aligning Task and (b) Vernier Hyperacuity Test.

Similarly, in the real-time retinal tracking method [55], a random flash point is projected in a different location of the patient's retina (see Figure 12b). The patient should immediately press a button once a projected flashed light point is seen. The method is widely used for eye diseases such as Glaucoma [56]. It is noteworthy that the aforementioned methods are used to detect the locations of VDs in the visual field and to consequently detect metamorphopsia.

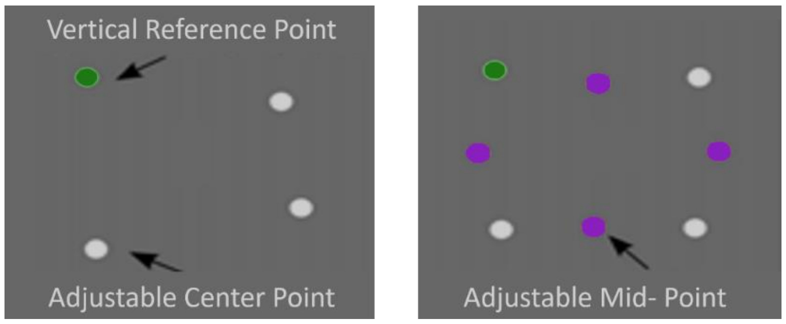


Figure 11 - Square completion task: (a) a fixed green dot and three adjustable white dots and (b) four movable orange-colored bisecting dots.

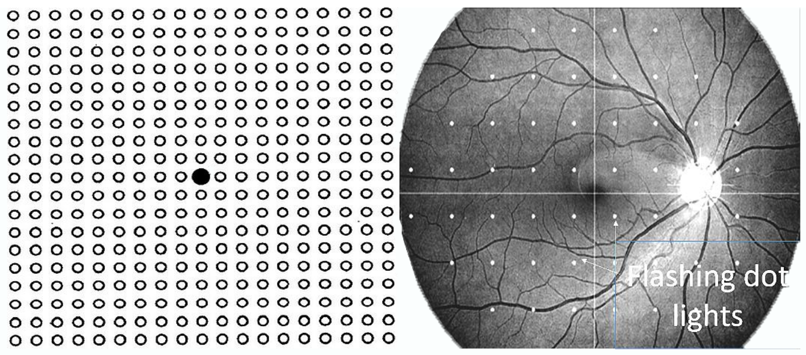


Figure 12 - Vibrating Method: (a) Oscillating Visual Stimuli and (b) real-time retinal tracking.

2.2.7 **DISCUSSION**

Computerized static VD assessment methods including portable threshold, accelerated, and deformable AG methods, in general, are prone to the filling-in effect and may not detect small metamorphopsia and scotoma regions. This is due to the predictive shape of the grid and the fact that the brain will guess and fill-in the gaps. As reported by Fink et al. [38] 3D-CTAG like other AG based methods suffer from the filling in effect due to the predictable static grid-based nature of the test. This effect was addressed in the NGRID method [57] by offering a series of different patterns projected in a different location of the retina. Additionally, AG methods slightly suffer from the inaccurate fixation at the center of the screen if they are used as home-monitoring tools, however, if the collection of responses are managed by an assistant, the fixation at the center can be better performed. This is because the attention of the patient will be placed at the center during the test.

In comparison with the aforementioned AG-based methods, by taking the advantages of hyperacuity, PHP allows higher accuracy so that this method enables the detection of small metamorphopsia and scotomas in the early stages of development of these disorders. Although PHP demonstrates high accuracy in comparison with AG methods, it allows very weak fixation at the center of the screen. This is because the location of the bump shape patterns are randomly changed and the patient should detect them using a mouse or finger-touch screen. Using dynamic methods such as PHP, the entire central field should be covered. In other words, although PHP, M-CHART, and Shape Discrimination all quantify metamorphopsia, none of these tests track eye movement and therefore cannot ensure proper fixation during the test [2], [58]–[60]. Weicek et al. [53] used a computer-based Amsler grid test, square completion, and dioptric pointing task with eye-tracking to quantify metamorphopsia in patients with maculopathy.

Similar to PHP, M-Chart lines also have to scan of view to be able to detect metamorphopsia and scotomas accurately. However, this increases the period of the test and subsequently decreases the accuracy. Furthermore, each line in the M-Chart should be repeated multiple times to cover from fine to coarse dotted lines. For MCPT with a flash duration of greater than 180 ms, the dots located on a retinal lesion theoretically should be perceived as being either misaligned or missing. This is because, for a short duration, the fovea requires re-fixation on the displayed line and this shift in fixation should cause the apparent movement of these dots from a misaligned position to an aligned one, thus giving the patient the perception of movement. For flash durations less than 180 ms, such apparent movement could be extinguished and the sensitivity of the test reduced [39]. One may argue the static tests such as AG based methods not only lower diagnostic accuracy due to poor compliance in fixation and inherit crowding and filling-in effects [61], they cannot be quantified to show the progress of disorders. On the other hand, dynamic tests such as PHP and MCPT, by offering the higher accuracy, can be considered as better choices to quantify the progression of macular disorders. It is noteworthy that moveable methods, such as the Square completion task and Oscillating Visual Stimuli can be considered in the group of dynamic methods with higher accuracy.

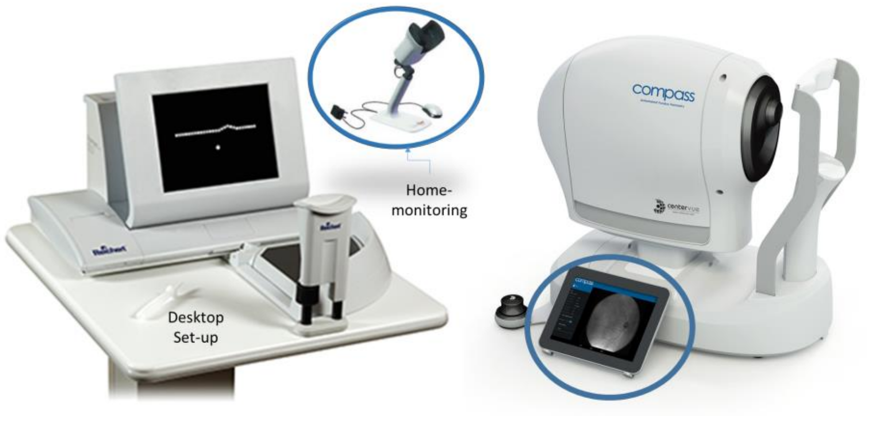
Table 1 presents a summary of GI methods, their advantages and disadvantages, and other details based on our study in this dissertation. Table 1 compares various factors including the (1) accuracy, (2) complexity or easy-to-use, (3) required time for a complete test, (4) portability. As seen in this table, “High”, “Low”, and “Medium” have been used to compare factors 1-4. As also seen in this table, in all methods except AG, threshold AG, and PHP home, PC should be employed. Therefore, they are considered as portable methods. However, by developing the smart-phone version of these methods, their portability can improve. Another factor is complexity. A comparison between the complexity of patterns projected in the patient’s retina shows that in AG, Accelerated AG, and Shape Discrimination, simple patterns are used and accordingly detect the location of VD through a low complexity procedure. This complexity can increase the required time to run a test. The required time for running a simple test such as AG or threshold AG is low. However, these methods cannot be used for quantifying the progress of macular disorders, and they can not necessarily be used to accurately detect them. Among various factors, PHP method (Figure 13a) has demonstrated higher accuracy [44], [62]–[64].

All aforementioned visual assessment methods have great potential for commercialization by offering new software applications to run in the PC, laptop or smartphone. However, for highly professional visual distortion assessment with high accuracy, the design and implementation of a specific device is required. In this direction, the PHP method was commercialized by Foresee Inc. [44], [62]–[64] for monitoring the macular disorder (see Figure 13a). Another company called Centervue [55] developed a device for monitoring the visual field using various methods including real-time retinal tracking (see Figure 13b). Despite significant advantages, these devices are expensive and can be suitable for professional eye-care settings. It is worth mentioning that while the fixation problem could be mitigated in the future by using an eyetracker, most commercially or clinically available tests have not included the eyetracker system in their instruments. Additionally, there are many psychological or physiological factors that can affect the performance of proposed computerized methods. Therefore, the success of these methods is dependent on the clinical trial. To date, a few efforts have been made to systematically study the performance of graphical interface techniques for various macular disorder. Among various techniques discussed in this paper, only AG and PHP have been systematically studied and their statistics widely reported based on various clinical trials. As reported in [62], [65], the sensitivities of AG and PHP for the detection of AMD are 78% and 87% respectively.

It is noteworthy that the focus of this dissertation was placed on the graphical interface methods from a computer science perspective. This dissertation is the first to review these methods. Most of the proposed methods discussed here need to be used by experts and trained personnel. Therefore, in their current form, they are not suitable for home diagnostics. However, these methods potentially can be used at home in the future using, for instance, a smartphone and tablet. The advantage of these methods for home diagnostic purposes has been reported in other review papers [62]. The focus of this dissertation is placed on the graphical computerized methods used for the detection of the macular disorders.

Table 1 - A quick Comparison of Visual Distortion Methods

Test	Accuracy	Complexity	Measurement Time	Portability	
Amsler Grid	Low	Low	Low	Good	[26], [33]
Threshold Amsler Grid	Medium	Low	Low	Medium	[39]
PHP	High	Medium	High	Poor	[39], [42], [43]
PHP Home	High	Medium	High	Good	[44], [62]–[64]
3D-CTAG	Medium	Medium	High	Poor	[33], [45]
MCPT	Medium	High	High	Poor	[39], [46]
M-CHART	Medium	Low	Low	Poor	[47], [48]
Shape Discrimination	Medium	Low	Low	Poor	[51]
Positioning	High	High	High	Medium	[52]



(a) (b)

Figure 13 - Commercially available visual field monitoring systems using (a) PHP method with two versions of desktop and portable devices and (b) real-time retinal tracking method.

2.3 SUMMARY

This chapter reviewed the dominant dynamic and static VD assessment methods. Despite significant advances, these methods suffer from low accuracy, high duration test’s time and difficulty in central fixation point. Also, these techniques can only provide a qualified assessment of VD. This thesis addresses the challenges above by proposing a new VD method called NGRID. In this method, a new unified hardware/software system is developed for generating and projecting various patterns in the short period. NGRID platform allows for collection and processing of the patient responses to create a specific heatmap to quantitively evaluate the damages in the retina for macular disorders. Before we start explaining the NGRID platform, we put forward simple mathematical models that helps us to better design VD tests in chapter 3.

3 MATHEMATICAL MODELING OF A SIMPLE VD

A number of attempts have been made to model the optical schematic of the human eye [59], [66]–[68]. The main goal of these models is to explain various optical phenomena in our vision. They also allow various optical refraction and aberrations to be computationally studied and compared to better understand our optical vision limitations. In this chapter, we focus on a simplified mathematical model specific to macular visual distortions (VDs) that are introduced by the presence of drusen, edema or even neovascularization and omit the optical aberrations that happen outside the macula. We discuss our simplified mathematical model for macular VDs and present the outcome of our algorithm to simulate VDs on a sample textual scene similar to what patients may experience due to their macular disorder.

As mentioned in chapter 1, many patients that suffer from macular disorders may experience metamorphopsia when viewing an Amsler grid as shown to Figure 3b. The physical changes in the retina due to macular disorders can significantly affect visual performance. The deformation of the retina due to macular disorders exerts forces in cellular layers. These forces can displace the cells that form the visual field. Therefore the straight lines might be seen as curvy (metamorphopsia) [69], on the other hand, deformation of the retina might slightly deviate it from the focal point of the lens. This may change sharpness, contrast, and brightness of patients' vision. Due to the complexity of the visual effects of these disorders, it is not possible to precisely quantify the effect of VDs and consequently predict the observed images by the patient. In this section, we propose a low complexity model of VD in the visual field of the patient suffering from CSR by taking into account the above mention effects. Given the fact that the accuracy of the computerized VD graphical methods relies on the accurate detection of distorted lines, it is crucial to have an idea of the distorted images seen by the patient. Our mathematical model can be utilized to design better VD test patterns.

3.1 EFFECT OF CSR IN THE VISUAL FIELD

Central Serous Retinopathy (CSR) or Central Serous Chorioretinopathy (CSCR) is a condition that occurs because of leakage of fluid at the level of retinal pigment epithelium [2]. This results in the creation of micrometer scale structures like a cavity. Figure 17a shows the OCT image of the retina affected by CSR. The structure can be modeled with cylindrical or semi-spherical geometries. One may argue that the deformation of the retina can affect the mosaic of cones and rods and other components between the photoreceptor layer and the basement of the retina as seen in Figure 1c. The deformation of the retina due to CSR can slightly stretch the retinal layers and consequently displace the visual sensing points on the top layer. These small displacements can cause significant visual distortion in the visual field of a patient. In the next sections, we derive the mathematical relationships for modeling a simple cylindrical shape as well as a more general extension of this model through semi-spherical shapes' deformation. Even though the CSR structure does not precisely assume these shapes, the models can be used to estimate the visual distortion caused by CSR. We start our models with visual distortions that are introduced by cylindrical shape drusen or edema and expand the model to more generic spherical shapes subsequently.

3.2 VDS DUE TO CYLINDRICAL SHAPE MACULAR DEFORMATIONS

We begin our modeling with cylindrical shape macular changes. As shown in Figure 14, P2 presents a small subsection within the macula that has a cylindrical shape deformation. The deformation in the LMS cone mosaic tiles can be caused due to various macular disorders by a presence of drusen, edema or even near fovea neovascularization. We are going to study the effect of this cylindrical shape and compare what is projected in a healthy macula sub-sectional plane (shown in Figure 14, P1) versus what is projected in a macula sub-sectional plane with the presence of cylindrical shape deformation (shown in Figure 14, P3).

As shown in Figure 14 sub-sectional macular plane P1, lines L1 through L7 (i.e., the sample lines from the outside eye scenery viewed by the human eye) are projected to the macula without any changes. However as soon as we project the same set of lines for an unhealthy macula (e.g., the macula with a deformation), we will notice some changes in the location of projections in the LMS cones mosaic tiles. As shown in Figure 14 P3, the location of the lines L1, L2 and L3 are not changed. These lines are projected normally as they happen to be outside of the cylindrical deformation showed in in Figure 14 P2.

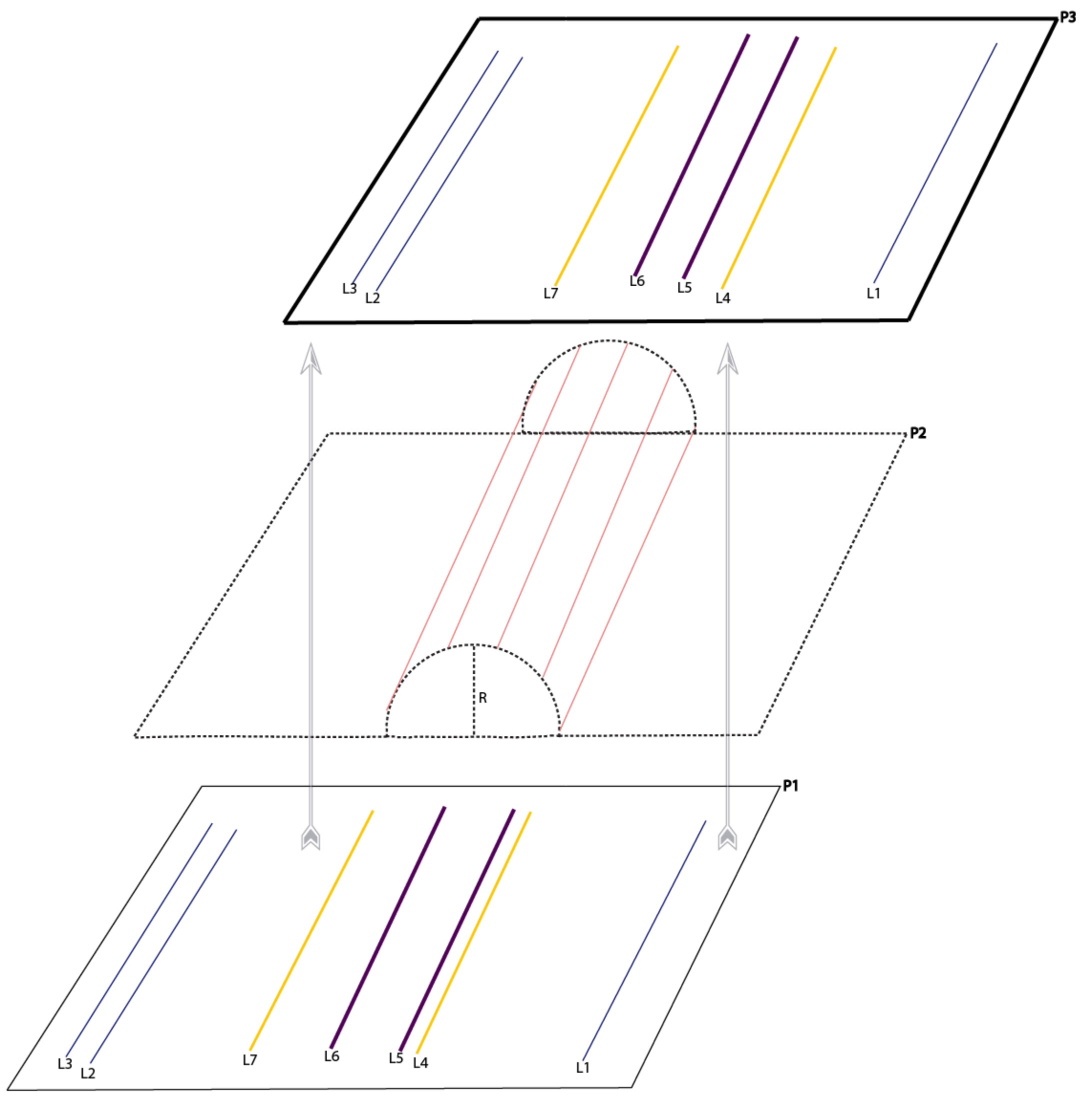


Figure 14 - Effect of the presence of cylindrical shape deformation in a sub-sectional macular plane (P2). P1 presents what is projected in the corresponding healthy sub-sectional macular plane. P3 presents the final changes and what is projected in the corresponding unhealthy sub-sectional macular plane. In P3, line L5 is displayed due to the cylindrical shape deformation presented on P2.

The location of projection of the line L5 is slightly shifted in the LMS cones mosaic tiles due to a bump introduced by the presence of the cylindrical deformation in the corresponding sub-sectional macular region. Please note the location of L4, L6, and L7 lines are not changed drastically even though that is projected within the boundaries of the cylindrical displayed in Figure 14 P2. These changes can be explained by mathematical relation shown in Figure 15.

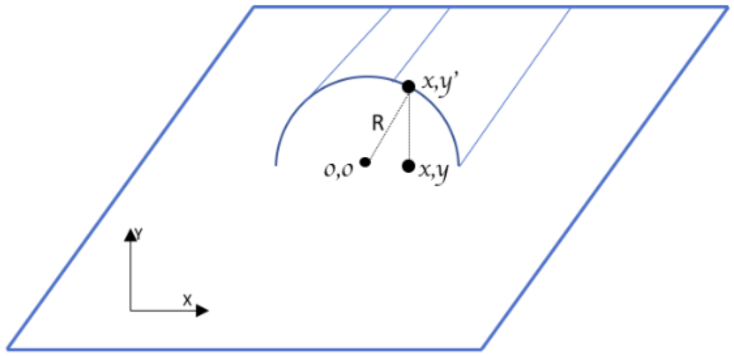
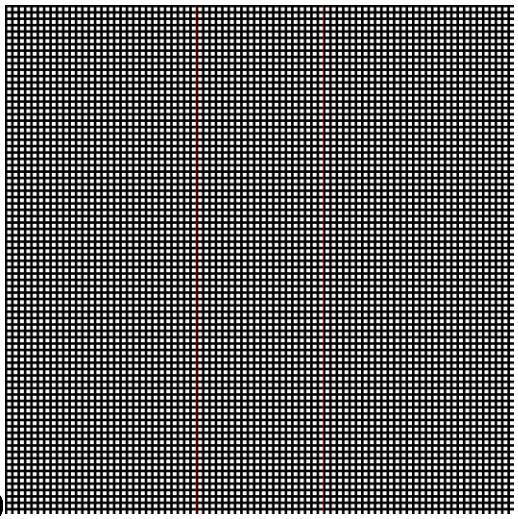
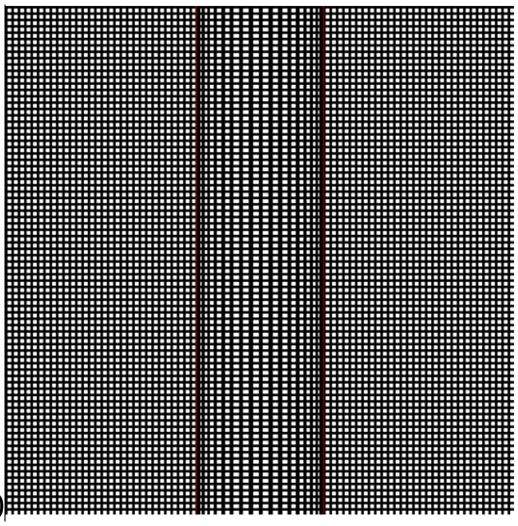


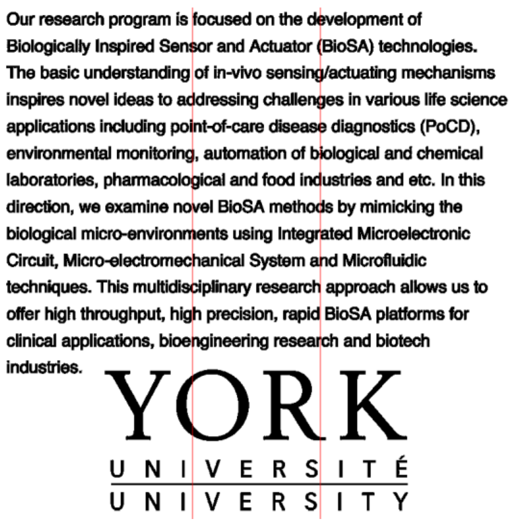
Figure 15 - Deformation Relation of the VDs Due to Cylindrical Shape Macular Deformations

We now apply the deformation relation shown in Figure 15 to a simple Amsler grid (Figure 16a) to better visualize see the changes. As shown in Figure 16b, we have used black and white canvas with size of 800x800 pixels at 96 DPI. The radius of the cylinder is 100 pixels and the boundaries of the cylinder are marked with helper guider lines highlighted in red. We also applied the same cylindrical VDs to a normal text and showed the results in Figure 16c and Figure 16d respectively.

It is noteworthy to mention that, as is evident in Figure 16b, we only considered VDs in vertical lines. In the next section (spherical model) we discuss changes that happen vertically and horizontally (e.g. the full Cartesian coordinates in a sphere shape VD).

(a) 

(b) 

(c) 

Our research program is focused on the development of Biologically Inspired Sensor and Actuator (BioSA) technologies. The basic understanding of in-vivo sensing/actuating mechanisms inspires novel ideas to addressing challenges in various life science applications including point-of-care disease diagnostics (PoCD), environmental monitoring, automation of biological and chemical laboratories, pharmacological and food industries and etc. In this direction, we examine novel BioSA methods by mimicking the biological micro-environments using Integrated Microelectronic Circuit, Micro-electromechanical System and Microfluidic techniques. This multidisciplinary research approach allows us to offer high throughput, high precision, rapid BioSA platforms for clinical applications, bioengineering research and biotech industries.

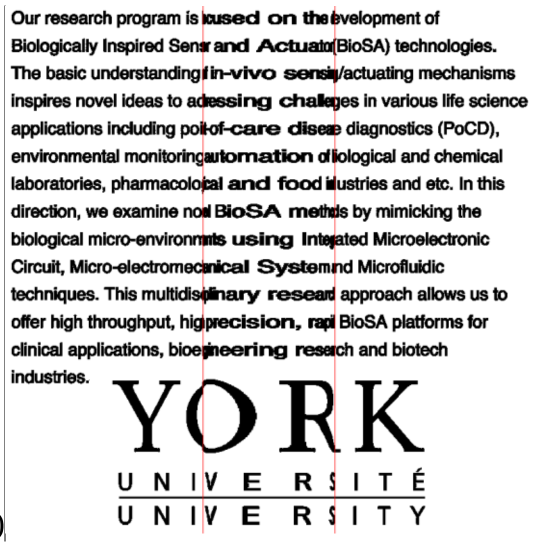
(d) 

Figure 16 - Cylindrical VD Simulation. (a) is the normal Amsler grid (b) is the outcome of applying cylindrical shape VDs to the normal Amsler grid (c) is the normal text that healthy macula would see (d) is the outcome of applying cylindrical shape VDs to a normal test shown in c.

3.3 VDS DUE TO SPHERICAL SHAPE MACULAR DEFORMATIONS

The creation of CSR is due to the collection of fluid under the basement of the retina. This results in the creation of micrometer scale structures like a cavity as seen in Figure 17a. This figure shows OCT images of the retina affected by CSR. The structure can be modeled with spherical or semi-spherical geometries. One may argue that the deformation of the retina can affect the cones, rods, and other components between the photoreceptors and the basement of the retina including the entire retinal pigment epithelium layers.

As demonstrated in Figure 17b, the deformation of the retina due to CSR can slightly stretch the retina layers and consequently displace the visual sensing points on the top layer. These small displacements can cause significant VD. Here we derive the mathematical relationships for modeling a half-spherical shape's deformation. Even though the CSR structure does not exactly assume this shape, our simple model can be extended to estimate the visual distortion caused by CSR.

A CSR deformation can be modeled with a sphere as illustrated in Figure 17b-d. Naturally, the cross section of this deformation assumes a half-circular locus. Given the fact that the number of sensing points or 'pixels' in the retina is the same before (orange points in Figure 17c) and after deformation (red points in Figure 17c), we can assume these sensing points are distributed uniformly before and after retinal stretching. For this reason, as a result of this deformation, the observed pattern in the retina differs from the pattern stimulated or observed in the normal retina. In other words, the distance between the sensing points in the deformed retina will be different from the one in the normal retina. Therefore, the stimuli pattern will be different from the pattern observed by the patient (see Figure 17c). In other words, the light from each point of the stimuli pattern that reaches the deflected retina at point x_1 where the light will be observed by the patient in point x_2 (see Figure 17d).

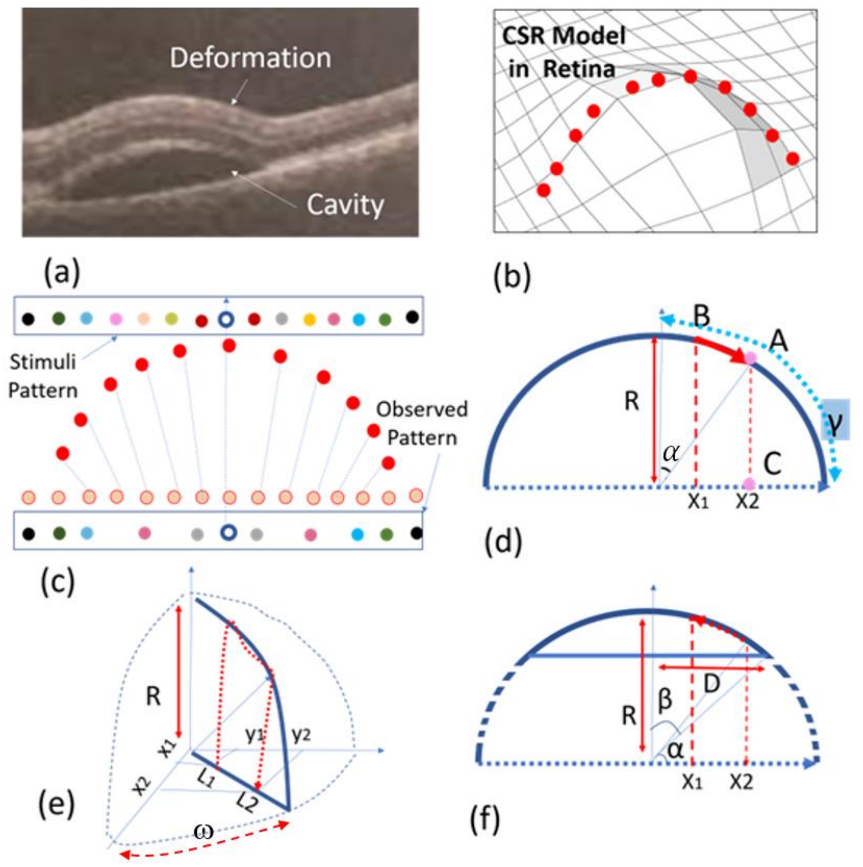


Figure 17 - Proposed Model: a) OCT image of a CSR cavity, b) a semi-spherical model, c)-d) half-spherical shape activity using discrete and continuous presentation, e) 3D half-spherical shape activity with partially spherical shape cavity model, f) semi-spherical-cylindrical shape CSR cavity mode

With reference to Figure 17d, x_1 can be obtained from the following equation.

$$\frac{x_1}{R} = \frac{\alpha R}{(\gamma)R} \tag{1}$$

Where R is the sphere's radius, $\gamma=\pi/2$ and α are the angels between the sensing point and horizontal axis that become equal to $\pi x_2/2R$ using equation 1. On the other hand x_2, is equal to $R\sin(\alpha)$, thus x_2 becomes equal to $R\sin(\pi x_2/2R)$ that is a one-dimensional model of displacement of the visual point from x_1 to x_2. However, the real visual point should be modeled in two dimensions as seen in Figure 17e. In this figure L_1 and L_2 replce x_1 and x_2 so that $L_1 = \sqrt{x_1^2 + y_1^2}$ and $L_2 = \sqrt{x_2^2 + y_2^2}$ where (x_1 , y_1) are the cartesian cordinates of the object point seen in a normal retina. This point is displaced to (x_2 , y_2) so that $L_1 =$

$R\sin(\pi L_2/2R)$. In the other hand, tan(ω)=$y_1/x_1=y_2/x_2$ where ω is an angle shown in Figure 17e. By combining above information, the equation 2 and 3 are obtained.

$$x_2 - x_0 = \frac{(x_1-x_0)R}{\sqrt{(x_1-x_0)^2+(y_1-y_0)^2}} . \sin\left(\frac{\gamma\sqrt{(x_1-x_0)^2+(y_1-y_0)^2}}{R}\right) \quad (2)$$

$$y_2 - y_0 = \frac{(x_1-x_0)R}{\sqrt{(x_1-x_0)^2+(y_1-y_0)^2}} . \sin\left(\frac{\gamma\sqrt{(x_1-x_0)^2+(y_1-y_0)^2}}{R}\right) \quad (3)$$

Where (x_0, y_0) is the center of spherical VD area. Equations (2)-(3) are used to obtain the displacement of the stimulation pattern where $(x_1-x_0)^2 + (y_1-y_0)^2 < R^2$. The actual CSR deformation shape as seen in OCT image in Figure 17a, can be modeled with a semi-sphere shape structures. With reference to x_1 in Figure 17e-f, $\gamma=\beta=\sin^{-1}(D/2R)$, therefore equations (2) and (3) with this having γ can be re-written to obtain the displacement of (x_1,y_1) to (x_2,y_2).

In the aforementioned spherical shape CSR modeling, we discussed a simple model that has been extracted from the OCT images of human subjects in order to create the required patterns for the detection of macular disorders. We use this modeling technique to estimate the distortion of a pattern seen by the patient. This can help to design patterns with maximum observable distortion by patients.

The effect of CSR models including half-spherical and semi-spherical models on three predefined patterns are shown in Figure 18. In these simulations, R=250 and D=60 and 125 pixels for modeling purposes. By assuming the patterns (Figure 18a, d, and g) are projected upon the retina, Figure 18b, f, and h estimate the distorted images seen by a patient suffering from mild CSR (D=60). On the other hand, Figure 18c, g and i estimate the distorted images are seen by a patient suffering from severe CSR using (D=125).

The simulation VD patterns are shown in Figure 18g that are shown to patients we will discuss the details on how these patterns are created in chapter 4. The estimated distortions in spherical shape model are shown in Figure 18h-i. Based on these results, the straight lines are distorted similarly that a CSR patient may experience VDs. This is why a group of straight lines can be used to detect the VDs in the visual field. Also based on the simulation results shown in Figure 18, the severity of the deformation of the VDs experienced in the test lines are dependent on how large the CSR cavities are (D=60 vs. D=125). As can be seen in Figure 18h, in early stages the distortion can hardly be observed.

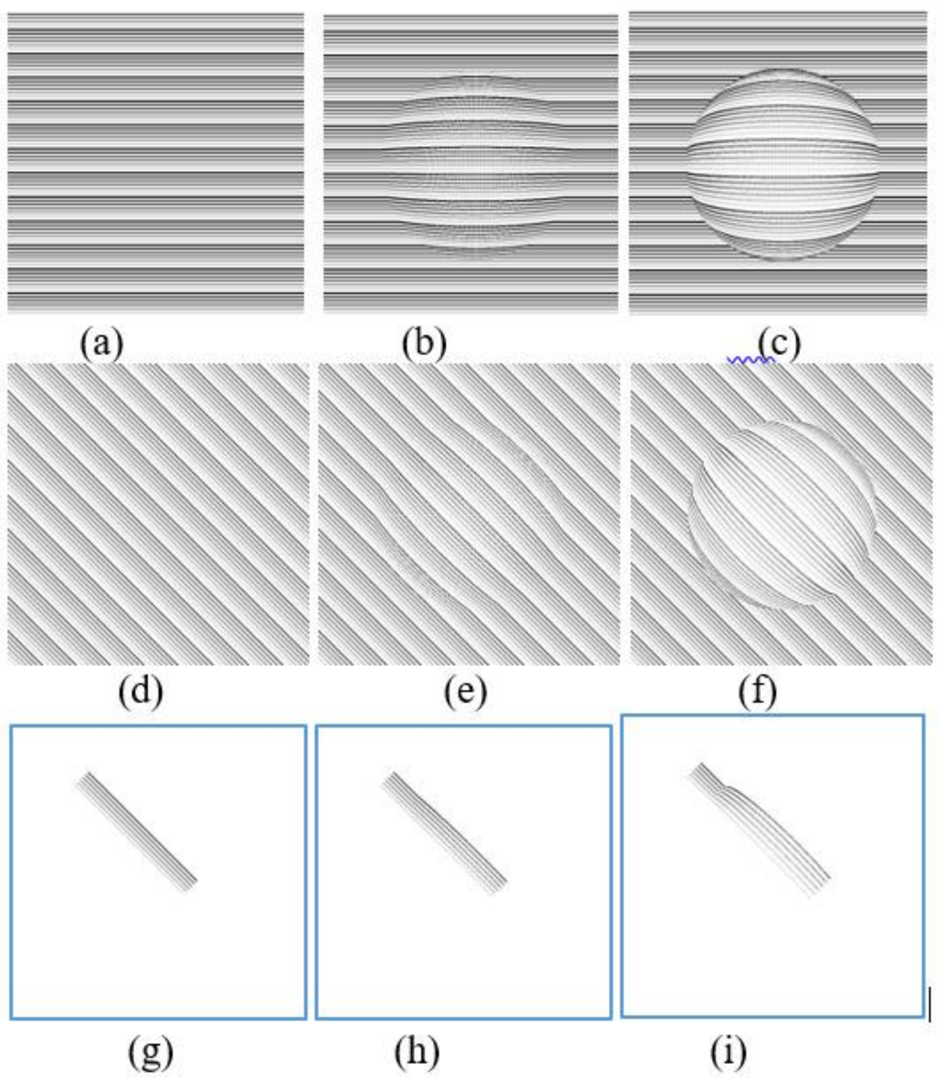


Figure 18 - Simulation Results of semi-spherical (b, e, and h) and spherical (c, f and i) models on three different patterns (a, d and g).

Figure 19 aims to display the same simulation on more natural scenery that patients experience in their day to day life. Figure 19a is presenting a sample text while Figure 19d is showing a sample photo that is seen by the normal eye. As seen in Figure 19b and e, the distortions are lower (D=60) than the distortions seen in Figure 19c and f (D=125). Another interesting point is that the distortion can be better recognized in the text (Figure 19b and c) than in the photo (Figure 19e and f). Overall the simulation results are shown in Figure 18, and Figure 19can confirm the advantage of simple patterns like straight lines or text to be projected in the patient's retina for the VD tests. Please refer to Appendix F for sample code demonstration on how we programmed the VD Simulator.

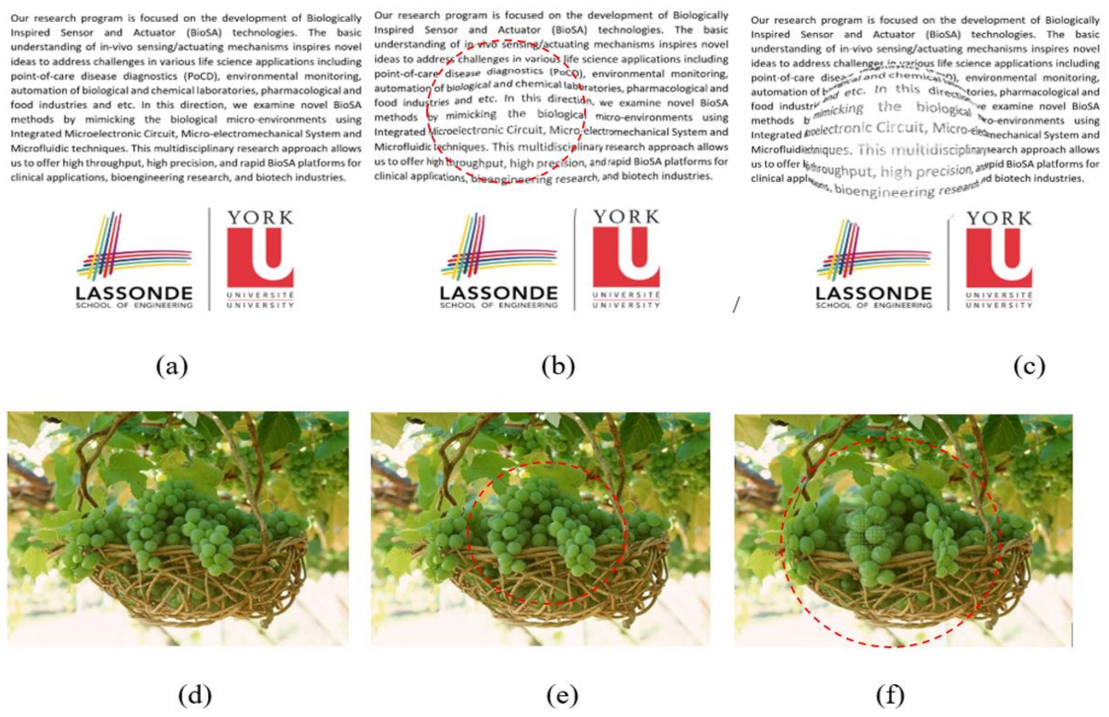

Our research program is focused on the development of Biologically Inspired Sensor and Actuator (BioSA) technologies. The basic understanding of in-vivo sensing/actuating mechanisms inspires novel ideas to address challenges in various life science applications including point-of-care disease diagnostics (PoCD), environmental monitoring, automation of biological and chemical laboratories, pharmacological and food industries and etc. In this direction, we examine novel BioSA methods by mimicking the biological micro-environments using Integrated Microelectronic Circuit, Micro-electromechanical System and Microfluidic techniques. This multidisciplinary research approach allows us to offer high throughput, high precision, and rapid BioSA platforms for clinical applications, bioengineering research, and biotech industries.

Figure 19 - Simulation Results of semi-spherical (b, and e) and spherical (c, and f) models on three different real-world scenery (a, and d).

3.4 **SUMMARY**

In this chapter, we discussed simple mathematical models for simplified VDs. We saw how small the effect of VDs could be on distorting the real world sceneries. In the next chapter, we introduce NGRID platform which allows various VD tests to be designed and performed for patients. We discuss how NGRID allows the collection and processing of the VD test results to help detection and progress monitoring of macular disorders. We use our mathematical model to simulate the distortion effect on our designed VD test patterns.

4 PROPOSED NGRID PLATFORM

In this chapter, we discuss a new Graphical Macular Interface System (GMIS) for accurate, rapid and quantitative measurement of visual distortion (VDs) in the central vision of patients suffering from macular disorders. In this system, a series of predefined graphical patterns are randomly selected from a library of patterns and visualized on the screen, then the identified VDs by the patient are recorded as binary codes using various control methods including speech recognition. Scalable Vector Graphics (SVG) is used to generate the patterns and save them into a central library. Based on the projected patterns and the patients' responses, a VD graph or so-called heatmap is generated for eye-care professionals.

We also demonstrate and discuss the functionality of the proposed system for the detection and progress assessment of a macular condition in patients suffering from Central Serous Chorioretinopathy (CSR) and compare the results with standard Optical Coherence Tomography (OCT) images. Also, we characterize the proposed technique to evaluate the systematic error and response time on healthy human subjects with normal/corrected vision. Below we discuss our approach towards a novel wearable visual distortion diagnostic system called NGRID. As shown in Figure 20, this embedded system is composed of various software and hardware components for macular disorder detection and monitoring. We will discuss the details for software implementation in section 4.4 and review the hardware details in section 4.8. The proposed platform can be used as the best alternative for home monitoring of various macular disorders, and the responses can be stored in secured cloud server for the future big data analysis.

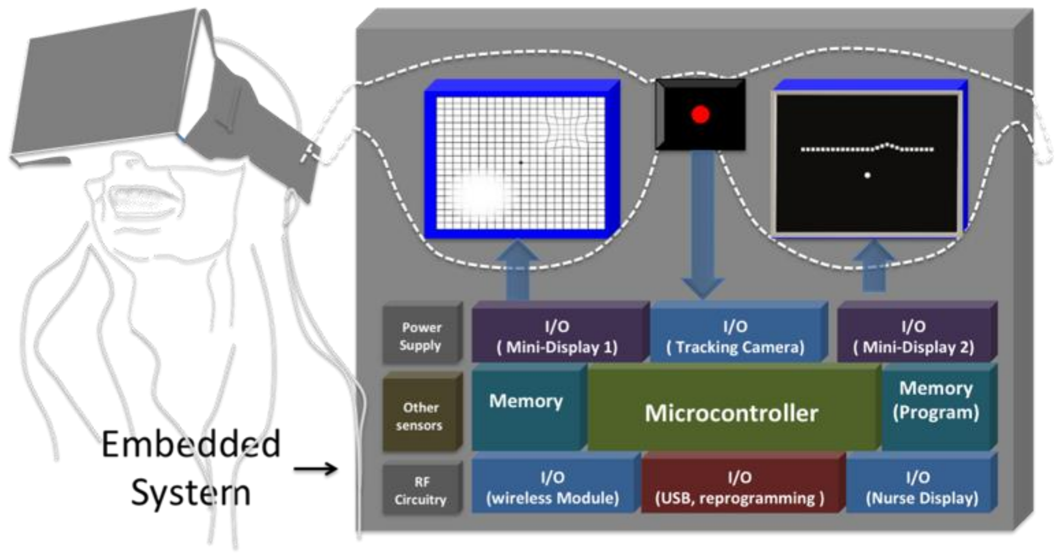


Figure 20 - Hardware stack used in wearable embedded NGRID system.

4.1 HIGH-LEVEL VIEW OF THE PROPOSED GMIS PLATFORM

As mentioned in Chapter 1, macular disorders such as Myopic Maculopathy, Macular Holes, Diabetic Macular Oedema, Age-Related Macular Degeneration (AMD) and Central Serous Retinopathy (CSR) affects central vision. Early detection of macular disorders is crucial as close monitoring allows for intervention before irreversible damage occurs.

It is not possible for patients to visit the hospital or doctor's office very frequently to have OCT to assess the progress of macular disorders such as AMD. However, the use of an affordable at home monitoring device will help them to monitor the progress of the disease. Such a device needs to accurately and quickly perform graphical tests to detect and monitor the VDs' progress. Such a device can update the doctor's office to help to contact the patient to visit the medical office if an urgent treatment such as Anti-VEGFs is required. This enables better patient care.

We propose a new approach towards the development of a head-mounted point-of-care diagnostic system for the detection and continuous monitoring of macular disorders at home. This system is an open-source platform that allows administration of various graphical macular tests by utilizing advance standard set of graphical interface eye tests (e.g., Amsler Grid, Threshold Amsler Grid, Macular Computerized Psychophysical Test, PHP and our own suite of test to excel macular VD detection and monitoring). This multi-Grid or so-called NGRID system aims to address challenges that are discussed in the previous chapter in regards to macular disorder VD detection and monitoring. NGRID has the following characteristics:

- Allow for a series of graphical patterns to be shown to the patients via commodity computers (e.g., tablets, laptops or smartphones)
- Provide simple ways to collect the test answers from the patients (e.g., simply do speech recognition on the patient's voice to understand and extract the patient's answers).

- Allows for quick and on-the-fly validation of answers to make sure patients are following and conforming to the test procedures properly.

NGRID’s affordability allows it to be used at home very frequently to ease the detection and monitoring of macular disorders as an alternative to paper-based AMSLER Grid. Once adopted by various hospitals, it can also provide means of comparing various graphical tests and the effectiveness of experimental preventive drugs. Also, big data analysis on the collected test results can answer various demographic questions and lifestyle effects on different macular tests such as AMD and CSR. We start with providing essential background information on software implantation of NGRID.

4.2 SOFTWARE BACKGROUND IN ADVANCE COMPUTER VECTOR GRAPHICS

In computer graphics, a raster image is a dot matrix data structure representing a rectangular grid of pixels (colorful point) [70]. In contrast to the raster graphic, vector graphics is the use of geometrical primitives such as points, lines, curves, shapes or polygons. Vector graphics use mathematical expressions and vectors to represent images. The vectors (also called paths) lead through locations called control points or nodes. Each of these points has a position on the X and Y axes of the Cartesian work plane and determines the direction of the path. Further, each path may be assigned a stroke color, shape, thickness, and fill. These properties don't increase the size of vector graphic files in a substantial manner. Vector graphics can be magnified infinitely without loss of quality, while raster graphics are resolution dependent, meaning they cannot be scaled up to an arbitrary resolution without loss of apparent quality.

Raster image editors (e.g., Adobe Photoshop [71] and GIMP [72]) revolve around editing pixels. Vector-based image editors (e.g., Adobe Illustrator [73] and InkScape [74]) revolve around editing lines, shapes, and paths (e.g., vectors in general).

One of the first uses of vector graphic displays was the US SAGE Air Defense System in 1957 [75]. In 2001, the World Wide Web Consortium (W3C) [76] put together a standard for vector graphics, which is called Scalable Vector Graphics (SVG) [77]. SVG standards are complex, but widely deployed and are royalty-free. Many web browsers (including the ones in mobile phones) now have basic support for rendering SVG data.

SVG is completely independent of the resolution of the rendering device. SVG files are essentially Extensible Markup Language (XML) printable texts that describe both straight and curved paths along with their attributes (e.g., colors, thickness, and transparency). Vector graphics shapes (e.g. straight lines or curved paths, images, and text) can be easily crafted using XML SVG files. Graphical objects can be grouped, styled, transformed and composited into unified objects to compose more sophisticated graphics.

SVG drawings can be interactive and dynamic. For example, animations can be defined and triggered by using programming or scripting languages. Web browsers support various sophisticated SVG shapes and animations via JavaScript. Basically, a series of programming scripts access the SVG Document Object Model (DOM) and provide complete access to all elements, attributes, and properties. A rich set of event handlers such as 'onmouseover' and 'onclick' can be assigned to any SVG graphical object to further allow human-computer interaction with the SVG [78].

The proposed hardware for NGRID (will be discussed in section 4.8) is powerful enough to run and render XML-based SVGs that can be manipulated via XML DOM. NGRID Graphics can also be created using SVG editor programs (e.g., InkScape [74], which have a similar look and feel to Adobe Illustrator [73]).

4.3 NGRID – THE PROPOSED GMIS PLATFORM

The proposed GMIS relies on displaying predefined patterns and collecting patient responses using control devices. Based on the patterns and patient's responses, a heatmap is generated for the detection and progress assessment of a macular condition. Below we discuss the details for each component comprising our proposed system including an overview of the patterns, use of control input devices to provide ways to answer the test, as well as different techniques to create the heatmap of visual distortions for visual and quantitative assessment of macular disorders.

4.3.1 NGRID VD TEST PATTERNS

A GMIS includes a pattern that is used as stimuli projected onto the retina. The patient's response is recorded while the patient has concentrated their attention gaze onto the center of the screen. In order to avoid the problem of changing the fixation during the examination, in this work, a series of patterns are shown, and the patient is asked to react only if the predefined pattern is observed as being distorted. In other words, the proposed method does not require capturing the position of each distorted pattern via mouse or stylus. Let us assume a VD test is composed of a series of N frames that are visualized, and the patients' responses are recorded correspondingly. In each frame displayed to patients, P_{ϖ}

represents a pattern that is displayed in that particular frame (ϖ). A pattern can be a single or a group of straight-lines with predefined Cartesian coordinates as seen in Figure 21.

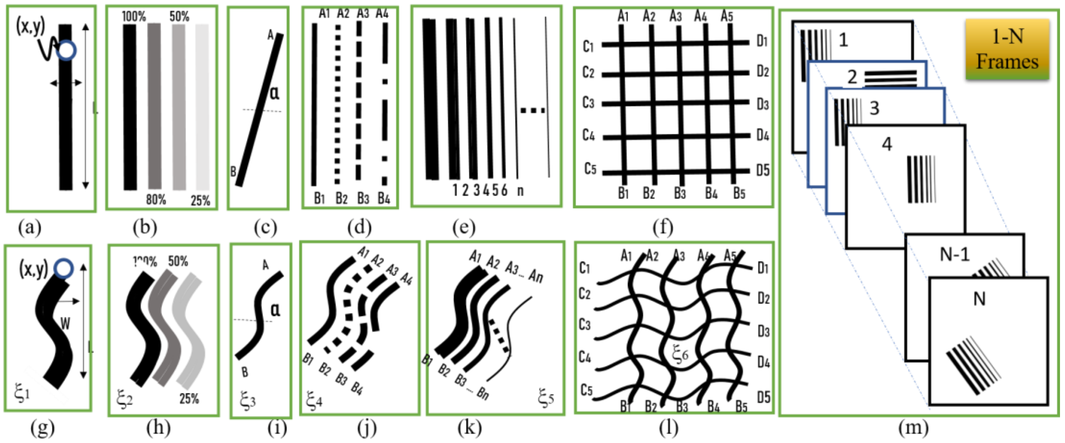


Figure 21 - Illustration of various patterns created with (a)-(f) straight lines and (g)-(l) distorted lines along with (m) a series of frames. The graphical pattern in each figure (g-l) are called ξ_1 –ξ_6 respectively.

There are several parameters that can be selected to generate various patterns using straight lines as seen in Figure 21a-f. These parameters include width (W), length (L), grayscale (G), color (C), angle (α) and form (e.g. solid, dash, dash-dot, etc.) of lines. A number (n) of parallel lines can form a pattern (Figure 21e) or a network of patterns (Figure 21f). The number of frames N and the time of projection (t_p) can be controlled by knowing the permitted time of examination. The optimum values of parameters are obtained for a high accuracy assessment of VDs. For instance, as the default, black lines on a white canvas are chosen to achieve the highest contrast. However, for patients who have difficulty recognizing such patterns, the system can further adjust colors, thickness, contrast, and brightness appropriately for each individual patient. Additionally, the length and number of parallel lines are two parameters that can be chosen to maximize the misalignment with the remaining part of each line or other lines. Figure 21g-l illustrate simplified models of distorted patterns namely ξ_1, ξ_2, ξ_3, ξ_4, ξ_5 and ξ_6 respectively. One may argue ξ_2 can be identified better than ξ_3 and ξ_1. Also ξ_5 can be detected easier than ξ_4 and $\xi_6$with a healthy eye. Figure 21c and 3i show patterns that can be repeated with different angles (α) in different positions. A series of such patterns are shown to the patient (see Figure 21m).

Most VD test includes a group of frames with straight lines, however we have also used distorted lines to run a VD test on healthy participants to evaluate systematic errors and response times using various control devices. For unhealthy participants we validated that they can see distorted patterns. it is noteworthy that in each frame with a positive response, all Cartesian Coordinates points of a pattern are recorded. We take advantage of SVG for the generation of pattern, frames and the collection of responses [18], [19].

4.3.2 NGRID TEST CONTROL INPUT DEVICES

As mentioned, the response of a patient is recorded using a control input device including keyboard, joystick and speech recognition (Figure 22a-c). The response of a patient for each one of N frames can simply be ‘0’ or ‘1’. At the end of an examination, N bits should be collected if a response is registered for all frames. All patterns associated with ‘1’ bits will be used to create the heatmap. For the selection of an appropriate control device for a patient, the response time and error are two metrics that should be minimized. A characterization study has been made in the last section of this chapter for the comparison of response time and errors in healthy participants. Among the control devices, speech recognition has the advantage of collecting the responses of patients for various patterns. We also can record the patient's voice (with patient consent) through the duration of the test to allow further verification of the patient’s answers if needed.

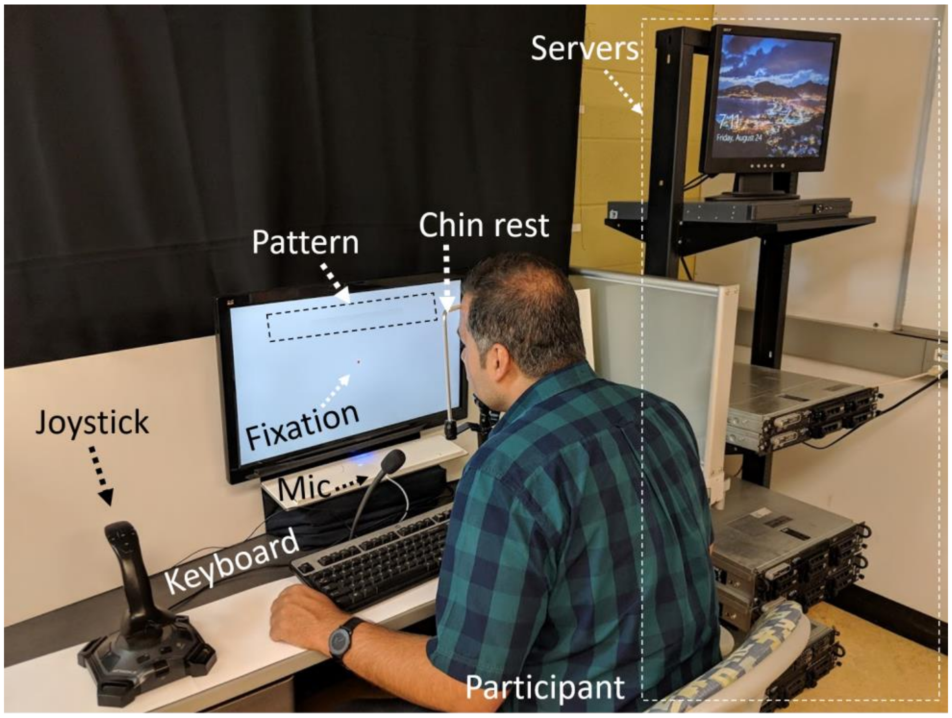


Figure 22 - A photograph of the NGRID platform including servers, control input devices (joystick, keyboard, microphone for speech recognition), monitor and a human participant who has fixated at the center of the screen (fixed his head in front of the screen using a chinrest

4.3.3 **HEATMAP**

As already mentioned, the recorded responses include a series 0's and 1's that are used for VD assessment. This assessment can be performed using different methods to approximately calculate the damaged area in the macula. Herein we introduce three different methods (seen Figure 23a-c), however, in this thesis, we will only use the heatmap method shown in Figure 23c.

Threshold method: In this method, the number of 0's, regardless of the positions of associated patterns projected in the retina, can be counted and compared with a threshold number (Figure 23a). This number should be greater than the systematic error. It is noteworthy that the systematic error is defined as the average number of errors made by the participants with healthy eyes. This method can be used to detect the onset or to measure the progress, of macular conditions. However, it doesn't give any information about the location of the lesion regions in the eye.

Interpolation method: The set of 0's can be used to create a boundary function. This boundary can be created by connecting the center of patterns (Figure 23b). This method can give an estimate of the retinal lesions by calculating the surface of the boundary. However, this method suffers from a lack of accuracy since the center of each pattern is not necessarily placed in the lesion region.

Heatmap method: In this method, all patterns related to the positive responses, which identified as distorted by the patient, are collapsed in a single frame. In this frame, each pixel may meet γ distorted patterns where $0 \leq \gamma$. The heatmap is generated by giving color to each pixel with transparency proportional to γ (Figure 23c). Therefore, the higher the γ, the higher the probability of identifying the damaged area in the retina. In this work, we use this method and a program, namely the NGRID Heatmap Generator (as shown in Figure 23). This program and its algorithm are discussed in section 4.7.

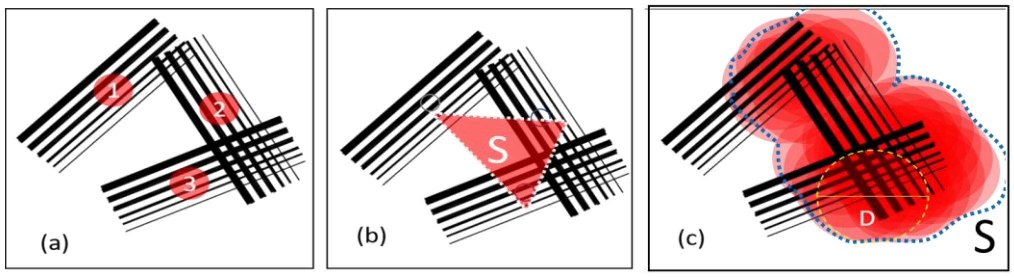


Figure 23 - Data Analysis strategies for creating the heatmap of the affected visual field; (a) threshold method, (b) interpolating method and (c) heatmap method. For simplicity, only the transparent circles (denoted with yellow borders and labeled as D) associated with pixels possessing two crossing patterns are shown in (c).

4.4 SOFTWARE IMPLEMENTATION OF NGRID PLATFORM

In this section, we discuss the essential elements needed to create the NGRID platform to facilitate NGRID VD tests and also securely store the massive amount of data potentially collected from a broad population of patients using the systems outside the clinics and hospitals. Our main goal is to offer a low cost, easy-to-use home-based VD assessment method for patients suffering from macular conditions. Therefore, we develop a hardware platform (will be discussed in details in the next chapter) to allow patients with an internet connection to perform their VD test daily. This platform includes a so-called NGRID Data Center that can also be used as a Big Data processing facility for many applications including testing the efficiency of applied treatments and drugs.

We also discuss the details of NGRID software for generating the test patterns, projecting the patterns in the form of VD tests for healthy and unhealthy participants, collecting the responses, user interfaces for entering the participants' information and modifying the specification of NGRID programs and generating heatmaps for the attention of medical practitioners and ophthalmologists.

As seen in Figure 24, the NGRID platform allows users, including patients or ophthalmology assistants or supervisors, access to the platform. At the first step, the users are identified as (a) supervisor, (b) patient/assistant or physician and (c) software designers. A designer can develop new patterns and generate new VD tests using the proposed web-based tool (will be introduced subsequently in this chapter). The generated data are saved in the Data Center and are collected in the Database (DB). NGRID data center has many other components which are responsible for processing the results and generating heatmaps as well as a secure, load-balanced, web portal to access NGRID admin and patient components. As shown in Figure 24, the supervisor can reconfigure the system parameters including retention of the stored log files as well as adding new designers and physician login credentials to the system.

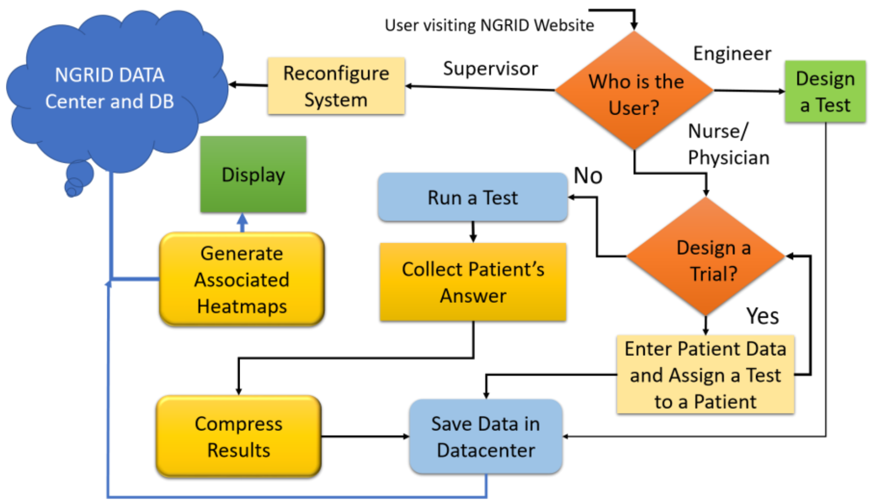


Figure 24 - The flowchart diagram of the proposed NGRID software platform. The patient will be given a login ID to download NGRID App to perform the VD tests from home.

An important part is the access of eye-care professionals to design a trial as well as patients to run a test. A trial is designed in order to enter the patient's information and set a specific VD test. A patient can be allocated to different trials with different ID numbers. The trials are saved in the database of the NGRID data center. Once the trial is designed and the test is set to a particular patient, the patient can '*run the test*'. In a typical NGRID VD test, there exists a series of frames that are displayed and the patient's responses are collected, compressed, and saved in the data center. The VD test results can be displayed in the form of a heatmap using the data saved in the data center. In this platform, several programs were developed using different programming techniques. A summary of these programs is shown in Table 2 and details are discussed subsequently.

Table 2 - A summary of programs developed in the NGRID Software Platform

Name of Program	Language	Input/Process/Output
NGRID Test App	C++ and C#	Consumes stored tests patterns and produces raw test results in NGRID Datacenter
NGRID Database	SQL and NoSQL	The highly available database within NGRID datacenter that keeps all test data, credentials and test results
NGRID SVG Editor	HTML5, CSS3 and JavaScript	Facilitates creating various SVGs that compose patterns used in NGRID Tests. The outputs are store in NGRID Database
NGRID Admin	Scala, Java, HTML5/CSS3 and JavaScript	This website allows for physicians and admins to login and produces and control various NGRID Tests
NGRID Heatmap Generator	R, C#, C++, OpenCV, HTML5, CSS3 and JavaScript	Consumes raw test results and after various rasterization and image processing, produces heatmap results for a test that a patient has completed.

4.4.1 DESIGN OF VD TEST VIA NGRID SVG EDITOR

Scalable Vector Graphics (SVG) [20] is used to create complex graphical patterns. SVG is a royalty-free and widely used standard that is supported by many web browsers including the ones in mobile phones and tablets. This allows the custom made NGRID SVG patterns to be displayable in commodity hardware without the need to pay royalty fees. This is an important factor in developing low-cost software for home diagnostics purposes. Furthermore, SVG is completely independent of the resolution of the underlying rendering device which allows for VD tests to be easily scaled for any screen sizes, pixel densities or for different Head Mounted Displays (HMD) that can be afforded by hospitals and doctors' offices. Table 3 shows a summary of the benefits of using SVG as the main standard for pattern creation in NGRID.

Table 3 - Comparison of Raster and Vector Images

Feature	Raster	SVG
Unanimously supported in a variety of web browsers and operating systems	yes	yes
Very small compressed file size	no	yes
Easy and fast manipulation through mathematical expressions	no	yes
Resolution independent	no	yes
Very small memory footprint to be used in embedded devices and small displays	no	yes

As an example, a simple SVG program (See SVG Program 1) is shown below for drawing a dashed line or so-called line001 from Cartesian coordinates (x_1=100, y_1=100) to (x_2=400, y_2=200) in a canvas with width and height of 500 pixels at a scale of 1.0 without any magnification.

SVG Program 1: *Drawing simple patterns*

```
<svg width="800" height="800" xmlns="http://www.w3.org/2000/svg">

      <line id="line001" stroke-dasharray="2,2" stroke-width="1.0"
      stroke="#000000"

            x1="100" y1="100"

            x2="400" y2="200"

      />

</svg>
```

Since SVGs are based on Extensible Markup Language (XML), the arrangements of each pattern can be programmatically scripted and changed via the manipulation of the SVG Document Object Model (DOM) [80]. DOM manipulation provides a very powerful feature to NGRID that allows sophisticated graphical shapes to be designed, animated and scripted mathematically. NGRID creates extended APIs that allow easy manipulation of SVG DOM. This allows NGRID to create shapes that are purely stated in terms of mathematical expressions. NGRID DOM manipulation also allows for the creations of sophisticated VD graphic tests via a program that creates and controls various SVG shapes and their attributes (e.g., location, scale and various style actors such as color, thickness, and transparency). This means that mathematical expressions and the shapes of the patterns they describe can be further modified through the NGRID API. A sample NGRID custom API for DOM manipulation is shown below (See SVG Program 2) to change line001 coordinates from (x_1=0, y_1=0) to new (x_1=50, y_1=50) as well as the color of the line from black to red. As seen in the above program, once the first part of the code is executed, it results in the transformation of the original SVG. Indeed, SVGs can be created simply by writing text files that conform to the SVG standard. Since SVG creation can be time-consuming and prone to human coding errors, NGRID has addressed this issue by providing an enhanced customized online SVG Editor that allows ways to create and modify the pattern of each frame in a fully graphical way without the need to write SVG code as seen in Figure 25.

***SVG Program 2:** Manipulating SVG DOM via custom APIs*

Original SVG:

```
<svg width="500" height="500"
xmlns="http://www.w3.org/2000/svg">

<line id="line001" x1="0" y1="0" x2="400" y2="200" stroke-
dasharray="2,2" stroke-width="1.0" stroke="#000000"/></svg>
```

Custom API Calls:

```
var newX1 = 50;
var newY1 = 50;
var redColour = "#FF0000";
NGRIDsvg.select("line001")
            .attr("x1", newX1)
            .attr("y1", newY1);
            .attr("stroke", redColour);
```

Final transformed SVG:

```
<svg width="500" height="500"
xmlns="http://www.w3.org/2000/svg">
<line id="line001" x1="50" y1="50" x2="400" y2="200" stroke-
dasharray="2,2" stroke-width="1.0" stroke="#FF0000"/></svg>
```

This will ease the process of pattern generation for NGRID VD tests. The NGRID SVG Editor is a web-based system that allows for easy and rapid creation as well as saving the tests online in the NGRID data center.

As mentioned in Table 2, the NGRID Test App renders the patterns for the patient to do the VD tests. NGRID uses an embedded web browser in headless mode. This allows the same tests to be seamlessly used in any commodity hardware as-is and without any modification. SVG patterns are embedded inside an HTML5 canvas which is controlled using the static test JavaScript APIs. These APIs allow for manipulation of the test graphics (i.e., showing the next pattern upon receiving the patient's answer) and for automatically uploading the test results, answers and UI related events back to the NGRID database in the data center.

The NGRID Editor can be used to create a series of frames to create or modify a VD test and save it in the NGRID library. Figure 25, shows the NGRID customized SVG Editor which allows NGRID designers to easily create various graphics for a frame or add additional frames to a test. Once the design process is done, by pressing the save button, all the frames will be stored in the NGRID database for later use. It is noteworthy to mention that SVG editing can be very complex as the graphics are expressed in mathematical vector notation. As noted earlier the NGRID SVG Editor simplifies the SVG editing task by introducing very simplified features similar to Microsoft Paint.

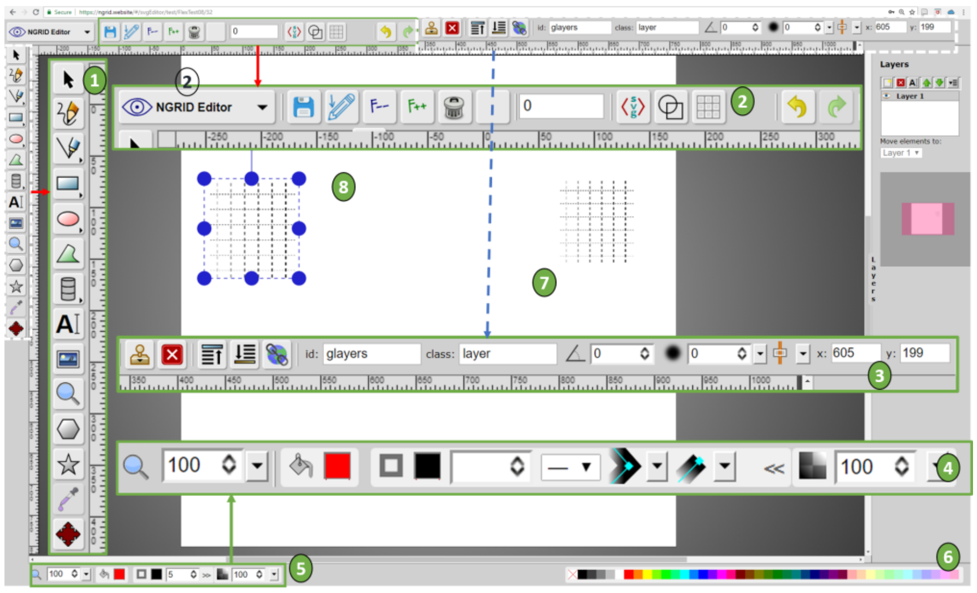


Figure 25 - NGRID Editor is a customized online SVG Editor for the easy creation of VD test patterns. (1) identifies the general editing buttons that allow for SVG pattern creation and modification (2) identifies the customized buttons that allow the automatic retrieval of the patterns from the NGRID data center. This allows for test data to be automatically mounted in the editor for editing purposes (3 and 4) are more detailed SVG editing tools that further allows the editor to

rotate, group in layers or even change the style of the SVG patterns (5 and 6) are tools that are used to introduce various colors and opacities (7 and 8) are sample VD Test patterns that are designed.

4.4.2 SCHEMATIC OF NGRID DATABASE FOR STORING TESTS AND TRIALS DATA

Figure 26 shows the Test and SVG data structure that is stored inside the NGRID database. Each user of the system is given specific permission. The permission determines the role of the user in the system (e.g., if they are a system admin and can add a patient to the system or if they are a medical practitioner that can assign different AMD tests to a patient). Each test is composed of a series of SVGs. A medical trial is composed of various patients that are assigned one or more tests to do.

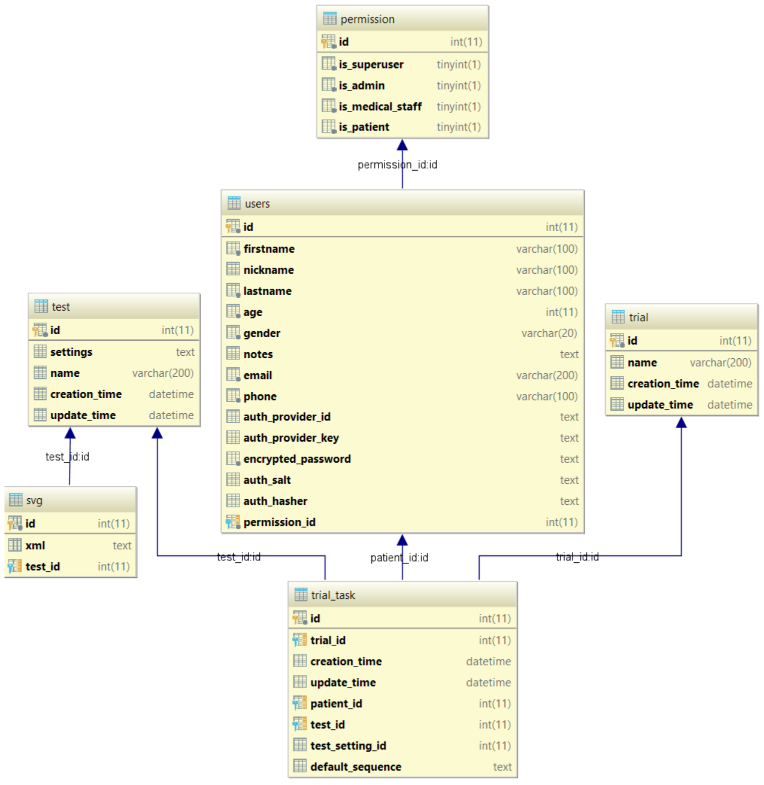
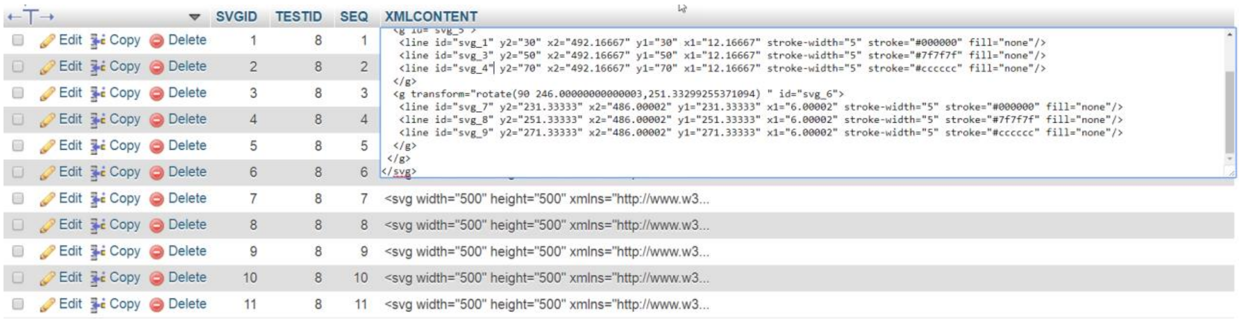


Figure 26 -Schema Details on NGRID SVGs, NGRID Tests and AMD Trials Database

NGRID Technical Admin staff also have access to the Test SVG database Tables that allow them to perform various low level archival and backup tasks. As shown in Figure 27, NGRID also provides a custom interface for admin technical staff to run various low-level queries against the Test SVG tables.



	SVGID	TESTID	SEQ	XMLCONTENT
Edit Copy Delete	1	8	1	
Edit Copy Delete	2	8	2	
Edit Copy Delete	3	8	3	
Edit Copy Delete	4	8	4	
Edit Copy Delete	5	8	5	
Edit Copy Delete	6	8	6	
Edit Copy Delete	7	8	7	<svg width="500" height="500" xmlns="http://www.w3...
Edit Copy Delete	8	8	8	<svg width="500" height="500" xmlns="http://www.w3...
Edit Copy Delete	9	8	9	<svg width="500" height="500" xmlns="http://www.w3...
Edit Copy Delete	10	8	10	<svg width="500" height="500" xmlns="http://www.w3...
Edit Copy Delete	11	8	11	<svg width="500" height="500" xmlns="http://www.w3...

```
   <line id="svg_1" y2="30" x2="492.16667" y1="30" x1="12.16667" stroke-width="5" stroke="#000000" fill="none"/>
   <line id="svg_3" y2="50" x2="492.16667" y1="50" x1="12.16667" stroke-width="5" stroke="#7f7f7f" fill="none"/>
   <line id="svg_4" y2="70" x2="492.16667" y1="70" x1="12.16667" stroke-width="5" stroke="#cccccc" fill="none"/>
  </g>
  <g transform="rotate(90 246.00000000000003,251.33299255371094) " id="svg_6">
   <line id="svg_7" y2="231.33333" x2="486.00002" y1="231.33333" x1="6.00002" stroke-width="5" stroke="#000000" fill="none"/>
   <line id="svg_8" y2="251.33333" x2="486.00002" y1="251.33333" x1="6.00002" stroke-width="5" stroke="#7f7f7f" fill="none"/>
   <line id="svg_9" y2="271.33333" x2="486.00002" y1="271.33333" x1="6.00002" stroke-width="5" stroke="#cccccc" fill="none"/>
  </g>
 </g>
</svg>
```

Figure 27 - Performing Admin Low Level Queries Against the Test SVG Tables

NGRID admin interface is written in Scala and benefits from advanced hardware and software load-balancing. We will discuss the details in the hardware section and explain how the system can serve millions of concurrent connections allowing it to serve many hospitals and patients via various advanced techniques such as caching the most frequently used queries as well as AMD Test SVG Graphics to enhance the overall system speed.

4.5 **DATA COLLECTION FROM THE SENSORY SYSTEM**

It is vital to quickly and accurately collect the test answers from the patients. In traditional graphical macular tests (e.g., PHP, MCPT and computerized AMSLER) patients were asked to perform a pointing task (e.g., to point to an area that patient experiences the visual distortion and difficulty). Pointing tasks inherently carry a level of difficulty that is explained by Fitts' Law [81] and described with an Index of Difficulty (pointing tasks become harder as the target area to point becomes smaller. The pointing task becomes even harder when multiple targets with greater distance are targeted for pointing). Of course, not having any pointing task involved in the test would eliminate this inherent difficulty.

To avoid any pointing tasks and associated overheads as well as inherited systematic errors, NGRID performs real-time voice recognition while recording the patient's voice for future audits. This will allow the patient to simply *say* what is in their mind without having to use a keyboard, mouse or other pointing devices. This will also help patients to not lose fixation on the center of the screen as a result of moving the mouse and fixating on the cursor/pointer. If desired, NGRID also allows designing tests that can use both traditional input devices as well as voice recognition (to assure complete coverage for all possible macular test types and styles).

The voice recognition is carried out via the NGRID voice recognition engine which communicates the recognized voice phrases back to the visual distortion test through a secure web socket. Also, this engine logs in the detected words and/or commands. This communication between the sensory system and data collection is performed at a high-level. Basically, the NGRID speech recognition is first configured by setting up the locale culture, the lexicon of detectable grammar words, amount of noise/silence required after each spoken word (here 200 ms) along with links to already established secure web-socket and database connections. Once the NGRID speech engine detects a word and/or command, it will store the results in a temporarily NoSQL DB Speech Log Table and also notify the client's VD test code, via a web-socket, about the recognized word. The client code will later act based on the state of the test and the recognized speech.

Each test has a specific lexicon that is accepted by the NGRID speech recognition engine via a file called *ngrid.speech*. The content of ngrid.speech can be modified by test designers. In this work, we used the following words ‘Yes’, ‘No’, ‘Bad’, ‘Brighter’, ‘Darker’, ‘Finish’, ‘Good’, ‘Next’, ‘Pause’, ‘Previous’, ‘Stop’, ‘Thicker’, ‘Thinner’ and ‘Wait’ to control various aspect of our NGRID VD tests. The program snippet below provides a high-level instantiation of the NGRID’s speech engine.

SVG Program 3: *Voice Recognition’s Communication*

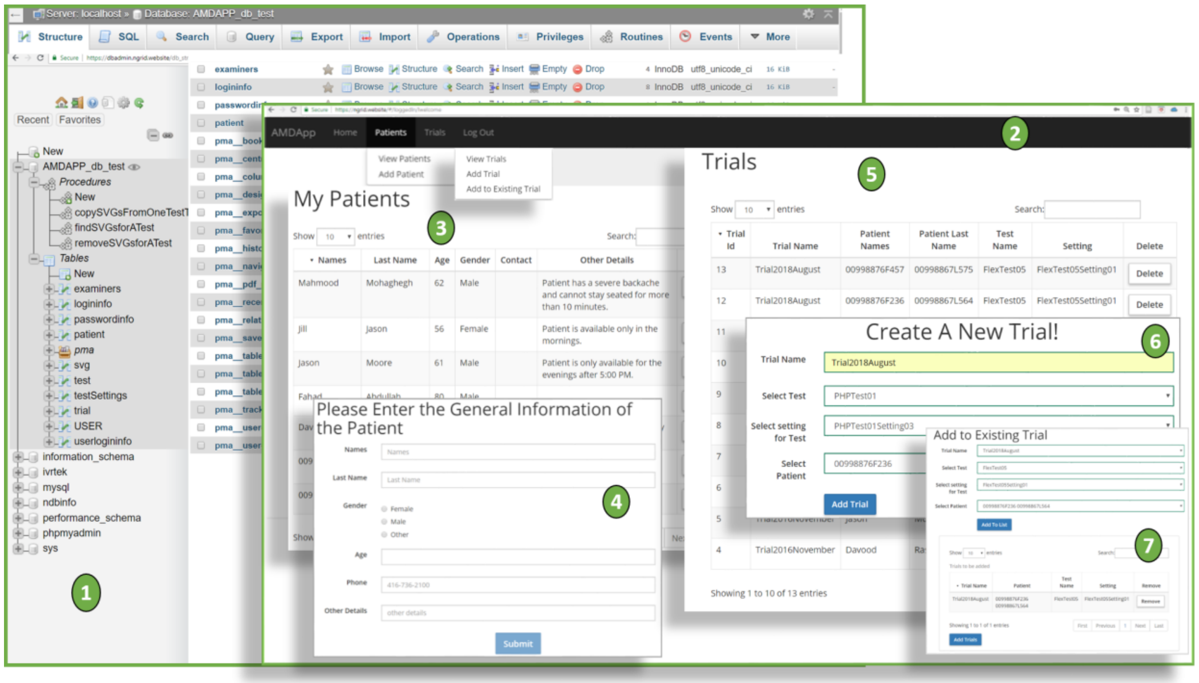
```
public void  SpeechEngineInit() {

        NgridSpeechRecognitionEngine speech = new NgridSpeechRecognitionEngine("en-
        US");

        speech.loadGrammar("ngrid.speech");

        speech.endOfWord = 200; //ms

        speech.speechRecognized += new
        AsyncEventHandler<EventArgs>(SpeechRecognized);

        speech.storageDB = ngridNoSQLSpeechStorageDB;

        speech.webSocketServerPipe = ws;

}


public async void SpeechRecognized(EventArgs e){

        storageDB.log(e.recognized.timestamp, paitentId, trialId, e.recognized.text);

    if (ws) {

        ws.notify(e.recognized.timestamp, paitentId, trialId, e.recognized.text);

    }

}
```

4.6 **NGRID ADMINISTRATIVE USER INTERFACE**

The NGRID Admin portal is a series of online administrative user interfaces (UIs) which were designed for patient data entry and for test control purposes. The general admin UI was designed for adding patients’ information, and the control admin UI was designed for controlling the VD tests. The NGRID admin interfaces are written in Scala in order to benefit from the load balancing done at the hardware level to serve many of concurrent connections allowing it to serve many hospitals and patients. It also uses a caching layer to save the most frequently used queries or VD tests to further enhance the speed of serving patients and medical practitioners.

General Admin: To perform a VD test for a patient, first the patient should be added to the system. Figure 28, shows the process of adding a patient to the system. Importantly, the system does not retain any Electronic Health Records (EHR) of the patients. NGRID can provide a cross-reference ID to correlate, patient test results to the hospital EHR. This approach assures one-way access of EHR record to be initiated by hospitals authorized personnel and further guarantee that any patient information is used or saved in the NGRID database. If sufficient authorization and consent are provided by both hospitals and patients (mainly for the patient's first name, last name, age and gender), NGRID can also retain this limited information. Once the patients are added to the system and the desired tests are created, an ID is generated and used instead of the actual personal information of the patient. The ID is unique and helps to keep the patient anonymous and also hides the associated technical clinical test details.

NGRID technical admin staff (or a designer, see Figure 24) also have direct access to the Test SVG Database Admin Interface (Figure 28-1) that allows them to perform various low-level DB Table archival and backups for the database. Figure 28-2 shows the NGRID General Admin interface (intended to be used by medical staff tasked with creating trials, adding patients to the system and assigning them to the trials). Figure 29 shows the NGRID Control Test Admin interface (intended to be used by technical staff who implement the tests and run various low-level tasks and queries against the Test SVG tables).



Names	Last Name	Age	Gender	Contact	Other Details
Mahmood	Mohaghegh	62	Male		Patient has a severe backache and cannot stay seated for more than 10 minutes.
Jill	Jason	56	Female		Patient is available only in the mornings.
Jason	Moore	61	Male		Patient is only available for the evenings after 5:00 PM.
Fahad	Abdullah	80	Male		

Trial Id	Trial Name	Patient Names	Patient Last Name	Test Name	Setting	Delete
13	Trial2018August	00998876F457	00998867L575	FlexTest05	FlexTest05Setting01	Delete
12	Trial2018August	00998876F236	00998867L564	FlexTest05	FlexTest05Setting01	Delete
4	Trial2016November	Davood				

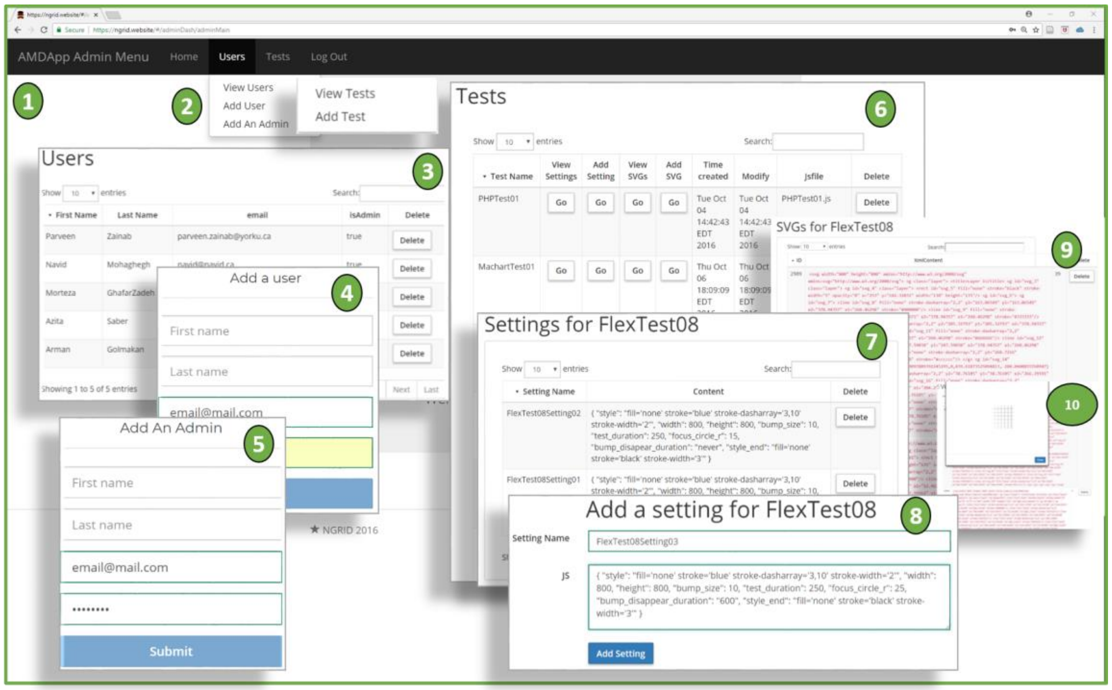
Showing 1 to 10 of 13 entries

Trial Name	Trial2018August
Select Test	PHPTest01
Select setting for Test	PHPTest01Setting03
Select Patient	00998876F236

Trial Name	Patient	Test Name	Setting	Remove
Trial2018August	00998876F236 00998867L564	FlexTest05	FlexTest05Setting01	Remove

Figure 28- (1) Specialized Database Admin Portal that allows for low-level archival and backup tasks to be done on the NGRID DB as well optimization on DB indices (2) is the overview of the NGRID Admin interface that allows adding patients and assigning them to trials (3 and 4) refer to how patients are added to the system (5 and 6) refer to how a new trial is added to the system (7) is how patients are assigned to trials.

Control Test Admin: This interface provides an easy way of manipulation and low-level manual editing of SVGs for a VD test (see Figure 29). The creation and modification of tests are mainly targeted for use by NGRID technical staff that are supervised by a participating doctor’s office to create different VD test styles as per request of the medical practitioner and ophthalmologists. Figure 29, shows the Control Test Admin interface that allows for adding new staff to the system as well as creating and customizing new NGRID VD Tests. It is noteworthy that patients can benefit from the NGRID’s settings with a preset of built-in tests as well as more specialized customized tests that are created based on the ophthalmologists' instructions. New tests can be added and edited via the online SVG Editor as explained in Figure 25.



Users

Show 10 entries Search:

First Name	Last Name	email	isAdmin	Delete
Parveen	Zainab	parveen.zainab@yorku.ca	true	Delete
Navid	Mohaghegh	navid@navid.ca	true	Delete
Morteza	GhafarZadeh			Delete
Azita	Saber			Delete
Arman	Golmakan			Delete

Showing 1 to 5 of 5 entries Next Last

Add a user

First name

Last name

email@mail.com

Add An Admin

First name

Last name

email@mail.com

••••••••

Submit

Tests

Show 10 entries Search:

Test Name	View Settings	Add Setting	View SVGs	Add SVG	Time created	Modify	Jsfile	Delete
PHPTest01	Go	Go	Go	Go	Tue Oct 04 14:42:43 EDT 2016	Tue Oct 04 14:42:43 EDT 2016	PHPTest01.js	Delete
MachartTest01	Go	Go	Go	Go	Thu Oct 06 18:09:09 EDT	Thu Oct 06 18:09:09 EDT		

SVGs for FlexTest08

Settings for FlexTest08

Show 10 entries Search:

Setting Name	Content	Delete
FlexTest08Setting02	{ "style": "fill='none' stroke='blue' stroke-dasharray='3,10' stroke-width='2'", "width": 800, "height": 800, "bump_size": 10, "test_duration": 250, "focus_circle_r": 15, "bump_disapear_duration": "never", "style_end": "fill='none' stroke='black' stroke-width='3'" }	Delete
FlexTest08Setting01	{ "style": "fill='none' stroke='blue' stroke-dasharray='3,10' stroke-width='2'", "width": 800, "height": 800, "bump_size": 10,	Delete

Add a setting for FlexTest08

Setting Name: FlexTest08Setting03

JS: { "style": "fill='none' stroke='blue' stroke-dasharray='3,10' stroke-width='2'", "width": 800, "height": 800, "bump_size": 10, "test_duration": 250, "focus_circle_r": 25, "bump_disappear_duration": "600", "style_end": "fill='none' stroke='black' stroke-width='3'" }

Add Setting

Figure 29 - Control Test Admin UI: (1) the overall online interface that allows staff and admins to be added to the system as well as creating and editing new NGRID VD Tests (2,3,4 and 5) allows for adding new staff (medical practitioner) and admins (NGRID test designers and technical staff) to the system. Only names, the email address is required (6) view the existing library of VD tests in the system. The medical practitioner can administer the test once the VD test is designed and stored in the NGRID library (7 and 8) adding a new test with customized settings. A test needs a name, series of settings such as duration of a test and for advance animated test associated scripts (9 and 10) low-level modification to SVG codes associated with a test. This is where the NGRID SVG Editor is launched as shown in Figure 25.

4.7 **NGRID HEATMAP**

As mentioned in Table 2, the NGRID Heatmap Generator program is developed to create a diagram with quantitative measures of the damaged VD area. It aims to both visually and quantitively measure the damaged area in the retina. As aforementioned, NGRID VD tests show a series of graphical patterns to the patients and collect their answers. The answers are post-processed in the NGRID datacenter that is later used by the NGRID Heatmap Generator program to render a visualized heatmap along with the Heat-index score.

The NGRID heatmap generator program finds the location of the scotoma and metamorphopsia by going through patient's answers frame-by-frame to create PerfectMatrix (PM) and HeatMatrix (HM). PM uses rasterized graphics from SVGs by considering the display resolution and pixel density at which the test is performed. HM will be similar while it distinguishes the frames that are seen as 'Bad' or '1s' (e.g., due to the presence of metamorphopsia or scotoma).

The HeatMap Matrix (HMM) is calculated by using HMM = PM – HM. This matrix provides the location of VDs along with the severity of the VDs at each location. Let us assume, F(i) is a matrix representing a frame from a series of N frames that are sequentially projected on the retina where $0<i<N+1$ and F(i) is an $m \times m$ matrix that represents an $m \times m$ image. N and m are natural numbers.

In the following example, $N=16$ and $m=5$, however in the actual VD tests, m and N can be much higher (e.g., $m=800$ pixels and $N=167$). In this example, the sixteen frames were defined with different patterns as seen in Figure 30.

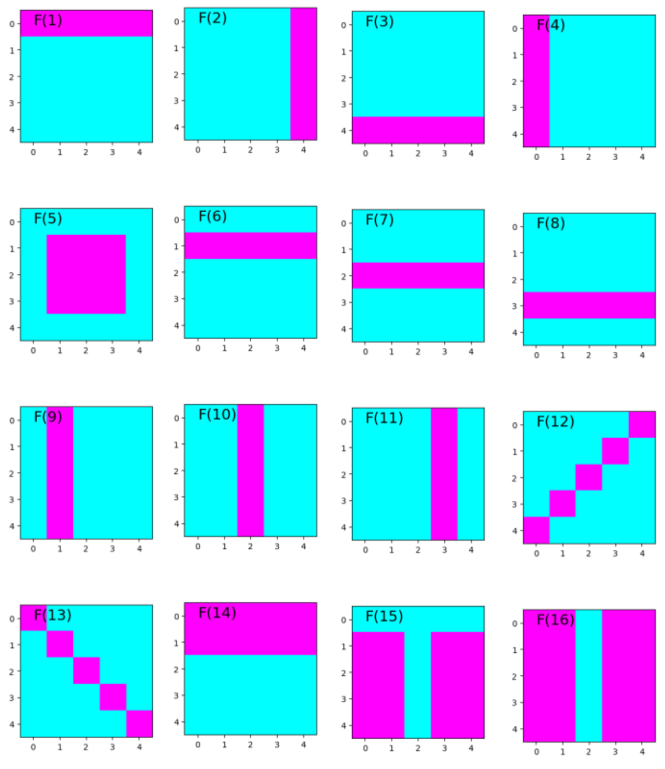


Figure 30 - An example of 16 frames in a simplified demonstration of heatmap algorithm

Each frame can be represented with one of the following matrices F(1)-F(16).

$$F(1)=\begin{bmatrix}1&1&1&1&1\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix},\quad F(2)=\begin{bmatrix}0&0&0&0&1\\0&0&0&0&1\\0&0&0&0&1\\0&0&0&0&1\\0&0&0&0&1\end{bmatrix},\quad F(3)=\begin{bmatrix}0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\1&1&1&1&1\end{bmatrix},\quad F(4)=\begin{bmatrix}1&0&0&0&0\\1&0&0&0&0\\1&0&0&0&0\\1&0&0&0&0\\1&0&0&0&0\end{bmatrix},$$

$$F(5)=\begin{bmatrix}0&0&0&0&0\\0&1&1&1&0\\0&1&1&1&0\\0&1&1&1&0\\0&0&0&0&0\end{bmatrix},\quad F(6)=\begin{bmatrix}0&0&0&0&0\\1&1&1&1&1\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix},\quad F(7)=\begin{bmatrix}0&0&0&0&0\\0&0&0&0&0\\1&1&1&1&1\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix},\quad F(8)=\begin{bmatrix}0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\\1&1&1&1&1\\0&0&0&0&0\end{bmatrix},$$

$$F(9)=\begin{bmatrix}0&1&0&0&0\\0&1&0&0&0\\0&1&0&0&0\\0&1&0&0&0\\0&1&0&0&0\end{bmatrix}, F(10)=\begin{bmatrix}0&0&1&0&0\\0&0&1&0&0\\0&0&1&0&0\\0&0&1&0&0\\0&0&1&0&0\end{bmatrix}, F(11)=\begin{bmatrix}0&0&0&1&0\\0&0&0&1&0\\0&0&0&1&0\\0&0&0&1&0\\0&0&0&1&0\end{bmatrix}, F(12)=\begin{bmatrix}0&0&0&0&1\\0&0&0&1&0\\0&0&1&0&0\\0&1&0&0&0\\1&0&0&0&0\end{bmatrix},$$

$$F(13)=\begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\end{bmatrix}, F(14)=\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix}, F(15)=\begin{bmatrix}0&0&0&0&0\\1&1&0&1&1\\1&1&0&1&1\\1&1&0&1&1\\1&1&0&1&1\end{bmatrix}, F(16)=\begin{bmatrix}1&1&0&1&1\\1&1&0&1&1\\1&1&0&1&1\\1&1&0&1&1\\1&1&0&1&1\end{bmatrix}$$

Let’s assume the answer of a patient to each frame F(i) can be represented with AN(i) where 0<i<17. More precisely A(i)=0 when the answer is ’good’ and AN(i)= 1 when the answer is ’bad’. In this example, the AN matrix is equal to [0 0 1 0 1 0 1 1 0 1 0 1 1 0 0 0]. The combination of all frames results in a PM can be obtained from the following equation. For our sample frames (F(1)-F(16)) the value of PM is shown in Equation 2. It is noteworthy to mention that PM acts as a reference comparison matrix that demonstrates a perfect scenario which the patient experiences no VDs.

$$PM = \sum_{i=1}^{i=16} F(i) \tag{1}$$

$$\text{PM} = \begin{bmatrix} 5. & 4. & 3. & 4. & 5. \\ 5. & 7. & 4. & 7. & 5. \\ 4. & 5. & 5. & 5. & 1. \\ 5. & 6. & 3. & 6. & 4. \\ 5. & 4. & 2. & 4. & 5. \end{bmatrix} \tag{2}$$

PM matrix is also used as the initial value of the heat matrix (HM). We now start going through the patient’s answers. In this example, for frame 0<i<17, if AN(i) is 'good', then equation 3 is obtained and if AN(i) is 'bad', then equation 4 will be obtained. It is noteworthy to mention that in a regular test, a patient may have extra time to visit some of the test frames more than once. If the patient is having doubts about seeing VDs on some of the repeated frames, we cannot deterministically understand if the patient indeed saw the frame with VD or made a mistake on answering a particular frame. We treat these cases with more suspicious on patient indeed observed VDs in the frame, and hence we apply double penalty (e.g., "-2" heat factor in Equation 4). Since the patient is not able to visit any of the frames more than two times (due to the restriction on test duration), we do not use more than double penalty on VD frames.

$$HM = HM + F(i) \tag{3}$$

$$HM = HM - 2 * F(i) \tag{4}$$

Therefore, HM is obtained as follows.

$$HM = \begin{bmatrix} 7. & 8. & 3. & 0. & 7. \\ 10. & 8. & 2. & 8. & 10. \\ 5. & 4. & -5. & 4. & 5. \\ 5. & 3. & -3. & 3. & 5. \\ 4. & 5. & -2. & 5. & 4. \end{bmatrix} \tag{5}$$

By differentiating PM and HM, HMM is obtained as follows (HMM = PM – HM):

$$HMM = \begin{bmatrix} -2. & -4. & 0. & -4. & -2. \\ -5. & -1. & 2. & -1. & -5. \\ -1. & 1. & 10. & 1. & -1. \\ -1. & 3. & 6. & 3. & -1. \\ 1. & -1. & 4. & -1. & 1. \end{bmatrix} \tag{6}$$

All values lower than mean values of $\overline{\overline{HMM}}=\sum_{i=1}^{i=5}\sum_{j=1}^{j=5}\frac{HMM(i,j)}{25}$ that is zero in this example will be ignored (this is mainly done to avoid graphical noise in later stages when we render the VD areas on a visualized heatmap graph). The new HMM is obtained as follows.

$$HMM = \begin{bmatrix} 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 2. & 0. & 0. \\ 0. & 1. & 10. & 1. & 0. \\ 0. & 3. & 6. & 3. & 0. \\ 1. & 0. & 4. & 0. & 1. \end{bmatrix} \tag{7}$$

The z-score of the HMM values are obtained as seen below using the $z = (X - \mu) / \sigma$ relationship where z is the z-score, X is the value of the element, μ is the population mean, and σ is the standard deviation (SD). This can further differentiate the values with SD lower than μ. It is noteworthy, using the Z-score and other functions below, that the pixels involved in the VD area are differentiated from other pixels. This process will result in increasing the visibility of the NGRID heatmap even if the frames processed are minimal (in real VD tests a much higher number of frames are displayed).

$$HMM_Z=\begin{bmatrix} -0.54. & -0.54. & -0.54. & -0.54. & -0.54. \\ -0.54. & -0.54. & 0.30. & -0.54. & -0.54. \\ -0.54. & -0.11. & 3.72. & -0.11. & -0.54. \\ -0.54. & 0.73. & 2.01. & 0.73. & -0.54. \\ -0.11. & -0.54. & 1.16. & -0.54. & -0.11. \end{bmatrix} \quad (8)$$

By clamping on heatmap z-score values with higher than 75% SD the following matrix clamped Z-score HMM_{ZC} is obtained. It is noteworthy to mention that we used z-score to be able to compare VD heat results for different tests that a patient will do over time. The tests may have different normal distributions for heat values, and z-score allows a unified way to compare the scores over time. Also, we clamped the value on 75% of the population of heat values to further reduce the rendering graphical noise on VD heatmap graphs.

$$HMM_{ZC} = \begin{bmatrix} 0 & 0 & 0.00 & 0 & 0 \\ 0 & 0 & 0.00 & 0 & 0 \\ 0 & 0 & 3.72 & 0 & 0 \\ 0 & 0 & 2.01 & 0 & 0 \\ 0 & 0 & 1.16 & 0 & 0 \end{bmatrix} \quad (9)$$

This matrix can be converted to a heatmap pattern as seen in the heatmap (Figure 31b). The heatmap figure indicates the visual field with visual distortion. As noted above, in each pixel, a circle with a certain opacity is drawn. The opacity (OP) and Radius (RA) matrices are obtained as discussed below.

$OP = 100*HMM_{ZC} / Max(HMM_{ZC})$ where each member of OP is between 0 to 100 and "Max" is a function that finds the maximum value in the matrix. The lower the value in

OP, the higher the transparency will be at the heatmap circle drawn at that location. The color of each circle in each pixel is obtained from the following matrix.

$$OP = \begin{bmatrix} 0 & 0 & 0.00 & 0 & 0 \\ 0 & 0 & 0.00 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 54.1 & 0 & 0 \\ 0 & 0 & 31.2 & 0 & 0 \end{bmatrix} \quad (10)$$

Similarly, the radius matrix (RA) is obtained so that RA = k*HMM_{ZC} / Max(HMM_{ZC}) where each member of RA is between 0 to k (k is defined in each VD test and can be from 1-20 depending on the sparse area of collapsed frames combined). It is noteworthy that k tries to factor in the area that no graphical coverage is done in the VD test. Herein we select k=1. The lower the value in RA, the smaller the radius of the circle will be. The radius of the corresponding heat location will be:

$$RA = \begin{bmatrix} 0 & 0 & 0.00 & 0 & 0 \\ 0 & 0 & 0.00 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0.54 & 0 & 0 \\ 0 & 0 & 0.31 & 0 & 0 \end{bmatrix} \quad (11)$$

It is also noteworthy that the values in the RA matrix cannot be directly rendered in the heatmap graphical canvas without scaling up the values according to the canvas resolution. To explain this better, please consider the following RA_r matrix which demonstrates the rounded-up values of RA to the nearest integer numbers:

$$RA_r = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (12)$$

As seen in Equation 12, RA_r cannot help to render the heatmap graph as the rounding operation will wrongly eliminates many of the VD locations (the diagram is shown in Figure 31a). Instead, the correct approach is to scale up the RA values based on the size of the drawing heatmap canvas. As demonstrated in Figure 31b, the drawing heatmap canvas has a higher resolution with three heatmap circles.

RA and OP matrices define the radius and opacity of each heatmap circles. For instance, considering the above RA matrix, for $RA_{(3,4)} = 0.54$ to be drawn in a heatmap canvas of size 400x400 pixels, will result in a scaled-up factor of 80 (e.g. 5x5 to 400x400). This means values for drawing a circle at pixel location (3,4) are mapped to (240, 320) with a radius of 43 pixels (location of $3 \times 80 = 240$ and $4 \times 80 = 320$ as well as radius size of $0.54 \times 80 = 43.2$ pixels which rounded to 43 pixels).

As seen in Figure 31b, the heatmap area is highlighted that shows where the patient experiences VD. The percentage of VD area in the visual field (η) and the center of this area (χ) can be calculated as follows. In order to obtain η, a program was developed to count the total number of pixels under the heatmap area. In this example (Figure 31b), η is calculated to be 11.25%.

To calculate coordinates of χ, for each pixel in the heatmap area, RA_{ij} and OP_{ij} represent the opacity and relative radius in *i* and *j* Cartesian coordinates. By knowing that OP_{22} is 100% in (2, 2), using the following equation 13 and equation 14, the center of the heatmap area will be obtained (and scaled up in the drawing heat canvas accordingly).

$$x_\chi = Av\,(i * OP_{ij}) \qquad (13)$$

$$y_\chi = Av\,(j * OP_{ij}) \tag{14}$$

where Av() represents a function that obtains the mean value of *i* and *j* for all available non-zero OP_{ij}. In this example, by substituting, (2, 2) and the related opacity in the above equations, χ will be equal to (2, 2) that will be scaled up to pixel coordinates of (160,160) in the drawing canvas size of 400x400 pixels. It is noteworthy that we can use a larger drawing canvas in order to show the heatmap with higher accuracy for medical purposes. Here we demonstrated a scale-up factor of 80 (e.g. 400x400 pixels from 5x5 pixels) while in our real VD tests we start with a minimum of 800x800 as well as 1200x1200. Please refer to Appendix E for sample code demonstration on how we programmed the results shown in this chapter.

We discuss the results and present the heatmap of VDs that are generated using the explained NGRID Heatmap Generator program in chapter 5.

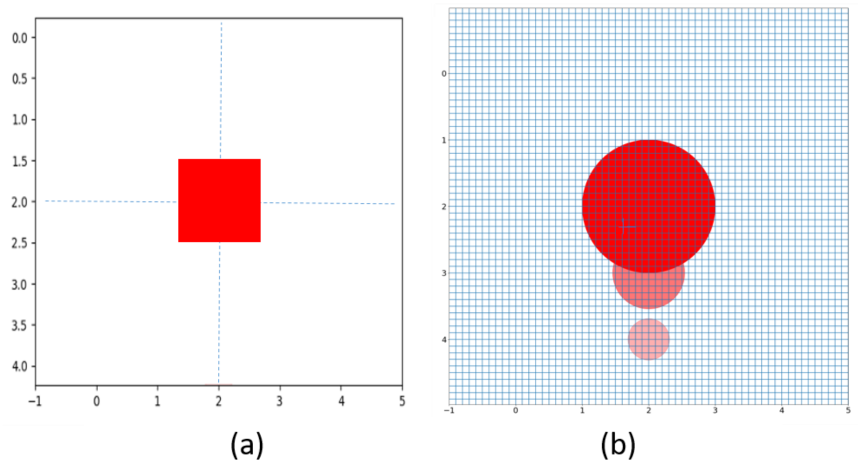


Figure 31 - An example of creating heatmap: (a) two overlapped features and (b) single, larger feature covering the second part.

4.8 HARDWARE IMPLEMENTATION

In this section, we discuss the essential elements needed to create a hardware platform to facilitate NGRID tests and also securely store the massive data collected across the globe for Big Data analytics.

To truly study macular disorders, we need large datasets to compare different test methodologies or even future treatments or preventative drugs. Since NGRID AMD tests need to be conducted across the globe, we designed our tests to be able to run from commodity hardware such as tablets, personal computers, and even smart mobile phones. As we explained in the software section, we can run the tests not only in commodity hardware but also in specialized HMDs that are equipped with advanced sensors such as eye-trackers.

NGRID software collects a large number of events based on patient interactions. As an example, we can collect what part of the screen the patient is looking at any given moment by using eye-tracker, or what the patient said at any given time through NGRID speech recognition. We can also collect very granular NGRID UI events to enhance post-processing tasks. This data will be compressed, de-duplicated and stored locally. This allows the individual patient device nodes to be able to work in offline mode (e.g. operate in areas with no internet connectivity). Once connected to the internet, the compressed data can be streamed in a real-time mode in the NGRID Data Centre. We can also add LTE and 5G data connectivity to truly bring NGRID into IoT era.

The NGRID Data Centre (Figure 33) is composed of a set of physical servers to create the cloud infrastructure that allows for linear expansion of NGRID when expansion needed for serving more patients across the globe. Currently, a total of 7 servers are utilized to conduct high availability and sizing for future tests conducted across the glob. Each server has 64 GB of RAM and two physical Intel Xeon 5460 CPUs. The servers are also equipped with 4 Gbps Host Bus Adapter (HBA) card to access the NGRID Storage Area Network (SAN). Each server runs on Linux Kernel 4.4 with the support of Linux Containers (LXC) for OS-Level Virtualization and also allows for Linux Virtual Server (LVS) to perform load detecting, fault tolerance and load balancing of services. Each server is equipped with TCP offload and Crypto Accelerator card (Cavium Nitrox) to accelerate SSL/TLS process for secure communication. Servers are also connected through a backplane of Host Channel Adapter (HCA) using Mellanox InfiniBand (IB) with Remote Direct Memory Access for in cluster storage access and IP over IB for fast 10 Gbps IP communication within the cluster. Figure 32 shows the server setup.

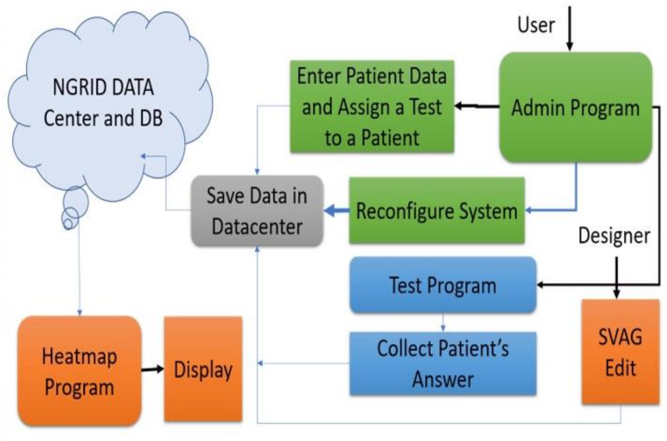


Figure 32 - The proposed hardware platform with the software process flow

To further utilize each server, we run multiple services using LXC. The services are composed of SQL Database Data Nodes along with SQL workers and DB cluster managers, NGINX Web Server acting as a static server while performing reverse proxy on the dynamic web requests to the NGRID Application Server written in Scala. We also use LXC to host Apache Cassandra and Spark nodes/workers for big data crawling across the cluster to run our various analytics. Our LXC also hosts the Apache Shiro security platform, which provides full SSL/TLS hardening and Single-Sign-On (SSO). Redis is utilized as an in-memory distributed cache for any small to medium size IO intensive scripts, using NoSQL. With the help of LVS (through persistent hashing, port health check and monitoring the least established connections), we perform load-balancing and provide high availability. Through Apache Spark and LVS/LXC containers, the cluster can support a near linear growth in servers to handle more NGRID AMD traffic/load.

In our current prototype, to provide a highly available storage for our micro-scaled cluster, we are connecting 20 x 300 GB SAS 6.0 Gbps 15k RPM hard disk drives (with 4x 128 GB solid-state drives acting as a dedicated read and write cache) to a cluster of active/passive OpenZFS links (via HBA 12 Gbps). To expose the LUNs to the NGRID cluster (hosted on OpenZFS), we use 24 port SAN Surfer switch (connected to each server in the cluster via 4 Gbps fiber channel link). Each OpenZFS pool has a dedicated SSD cache along with ZFS Intent Log (ZIL) to accelerate synchronous write transactions. LZ4 compression and deduplication are enabled for the archives.

Any project that deals with Big Data requires proper measures for Data Integrity and Storage. In the NGRID platform, each test can produce as much as a 10 MB of very detailed log files and around 8 MB of raw voice recording. For 2,000,000 patients across the world to simply perform the test only once, we will need around 35 TB of storage. And if they repeat the test daily around 12 PB of raw storage would be needed to retain yearly data which will quickly become cost prohibitive. Here are the details for 12 minutes of voice recording and recognition:

- Audio recording is done in raw mode (as we do real-time voice recognition on a stream of PCM u8 mono channel 11025 samples per second with 8 bits per sample).
- File size estimate of 12 minutes per recording and voice recognition = 11025 samples per second * 8 bits per sample * 12 minutes * 60 seconds per minute = 11025*8*12*60 = 7.57 MB

To avoid dealing with high and costly storage, we enabled GZIP compression on our collected data (since the patient is not speaking continuously we achieved around 400-500% reduction in file sizes for our recordings). Also, the same was done for the stream of events and SQLite logs (since only raw ASCII logs with fairly repetitive keywords are used, a compression ratio of 700-800% is possible). This means our storage size can be reduced 5.5 times less the actual raw size which lands us in the range of 2 PB for yearly usage.

Figure 33 shows the NGRID platform high-level architecture, is set to target at least 100,000 unique patients across hospitals in Canada, India, and Iran, with plans to increase to a range of 2,000,000 patients to cover all AMD patients across the globe.

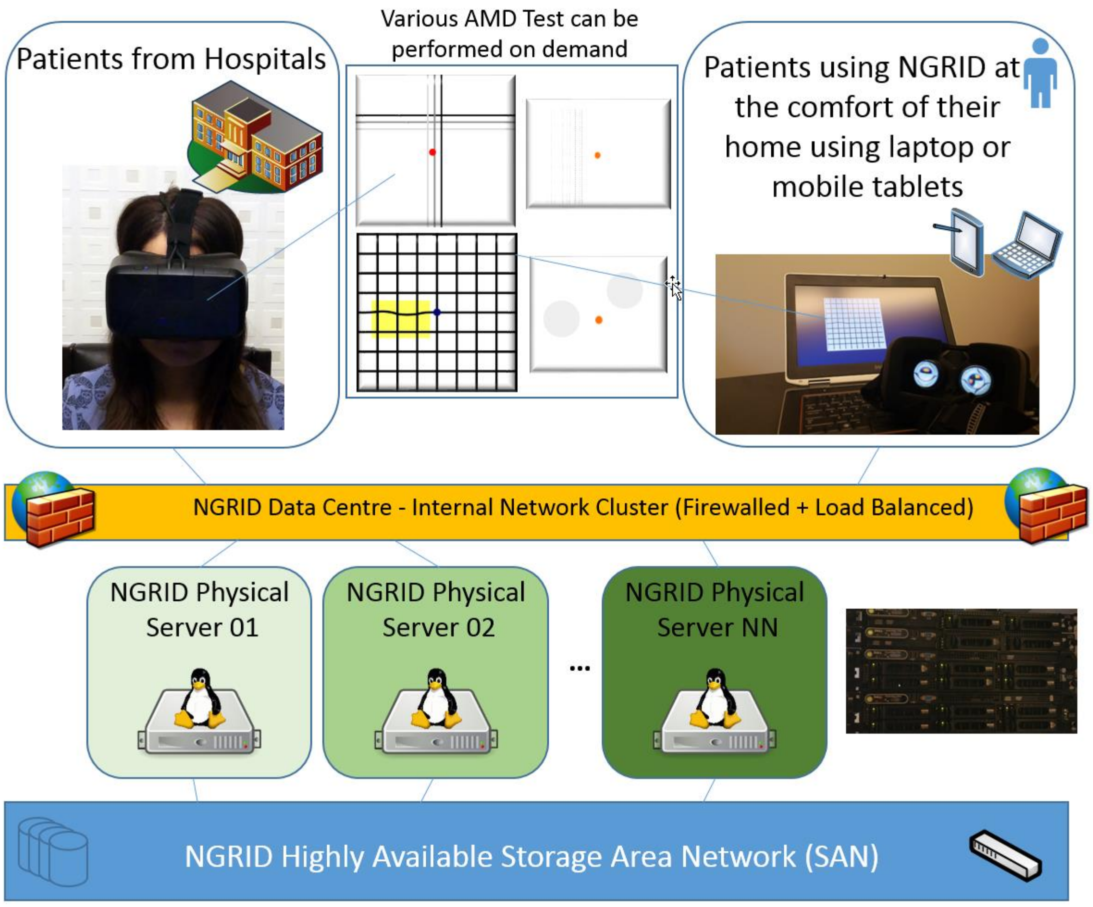

Patients from Hospitals
Various AMD Test can be performed on demand
Patients using NGRID at the comfort of their home using laptop or mobile tablets
NGRID Data Centre - Internal Network Cluster (Firewalled + Load Balanced)
NGRID Physical Server 01
NGRID Physical Server 02
...
NGRID Physical Server NN
NGRID Highly Available Storage Area Network (SAN)

Figure 33 - High-level Overview on NGRID Datacenter

4.9 NGRID SECURITY

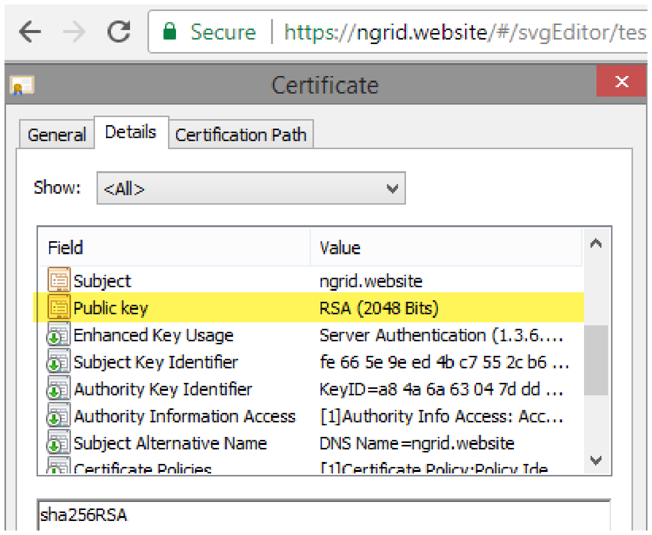
In this section, we discuss the details around security and access levels to the NGRID platform.

The Transport Layer Security (TLS) and Secure Socket Layer (SSL) are cryptographic protocols that are widely used for secure communication over insecure data links. TLS provides privacy and data integrity needed for the Internet including financial institutions. TLS/SSL can provide a highly secure connection through the use of public/private key infrastructure. Breaking the symmetric cryptography with a high key length (more than 4096 bits) is almost impossible using current supercomputers (the time required to break the code far exceeds the span of a communication).

Once the NGRID test is completed, the patient's answers are transferred to the NGRID data center. This communication is done via TLS/SSL. The NGRID platform does not save or use Extended Electronic Health Records (EHR). The only saved information that is obtained under patient consent is related to the patient's name, age, gender, and NGRID Internal Globally Unique ID (GUID or simply ID). Even this information is encrypted using Advanced Encryption Standard (AES 256) to avoid privacy leaks.

NGRID database access is guarded through an advance control list, firewall, and IP level filtration. NGRID patient ID allows the hospital to cross-reference patients in the NGRID database with the hospital EHR records. As shown in Figure 34, all database access along with the online admin user interface is encrypted using SHA256 hashing with RSA 2048 bit length keys which provide the same level of encryption as banks and financial institutions.

As we will explain in the hardware section, NGRID Data Centre can serve many patients across the globe and store a massive amount of test results and information (in the range of 12 Peta Bytes). The data cannot be analyzed by traditional Structured Query Language (SQL). We are using a cluster of Apache Spark and Kafka along with Scala and R Languages to perform data analytics on the Big Data (mainly around the test results). Figure 35 shows various sensors and how they interact to collect and verify the answers from patients.


Secure | https://ngrid.website/#/svgEditor/tes

Certificate

General | Details | Certification Path

Show: <All>

Field	Value
Subject	ngrid.website
Public key	RSA (2048 Bits)
Enhanced Key Usage	Server Authentication (1.3.6.…
Subject Key Identifier	fe 66 5e 9e ed 4b c7 55 2c b6 …
Authority Key Identifier	KeyID=a8 4a 6a 63 04 7d dd …
Authority Information Access	[1]Authority Info Access: Acc…
Subject Alternative Name	DNS Name=ngrid.website
Certificate Policies	[1]Certificate Policy:Policy Ide

sha256RSA

Figure 34 - TLS/SSL Certificate for NGRID Site

Figure 36 illustrates the high-level interaction between software components. The end-user NGRID Device is mainly used by patients. It uses TLS/SSL/HTTPS to securely load various tests and patterns from the NGRID Data Centre. This is done via the onboard wireless capabilities (both 802.11 and LTE). Various input devices ranging from basic Keyboard, Mouse and Touch to more advance Joystick and Voice Recognition is enabled. Also, feedback sensors like head and eye-tracking provide measures to make sure the patient is fixating at the center of the screen. All the patient input is stored in local storage which at the end of the test will be uploaded to the NGRID Data Centre for further processing.

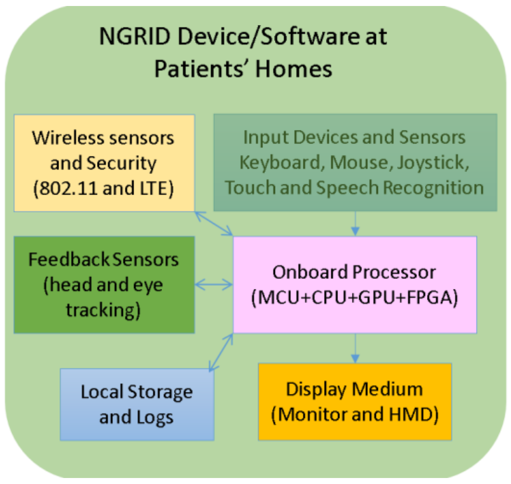

NGRID Device/Software at Patients' Homes
Wireless sensors and Security (802.11 and LTE)
Input Devices and Sensors Keyboard, Mouse, Joystick, Touch and Speech Recognition
Feedback Sensors (head and eye tracking)
Onboard Processor (MCU+CPU+GPU+FPGA)
Local Storage and Logs
Display Medium (Monitor and HMD)

Figure 35 - Hardware sensory overview of various components used in a typical NGRID VD test

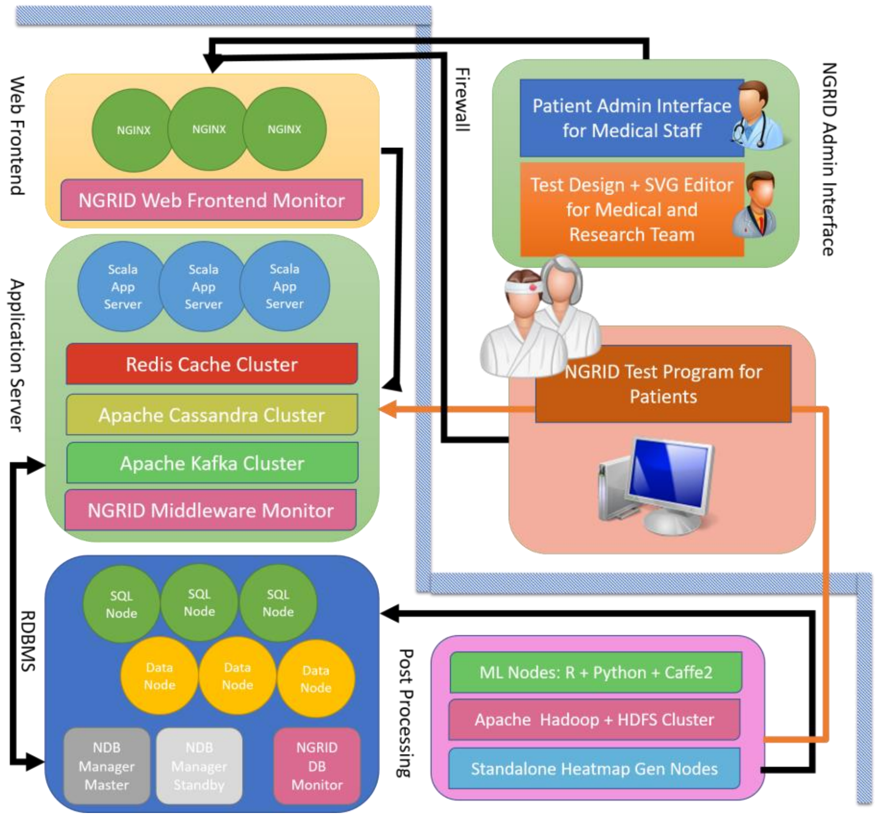


Figure 36 - High-level illustration of proposed software and hardware architecture

4.10 **SUMMARY**

In this chapter, we discussed the details around the implementation of the NGRID platform. We explained how tests are designed, assigned to patients along with internals of how heatmaps are generated to assess the progress of a macular disorder. We also discussed how the NGRID cluster is designed and scaled up along with details on various hardware components used in the platform. In the next chapter, we discuss the results of NGRID tests along with the corresponding generated heatmap with comparison to patients' OCT images.

5 RESULTS

In this section, we demonstrate and discuss the experimental results using healthy and non-healthy participants. We compare the heatmap results with patients' OCT images to show how heatmap results conform to OCT images and can serve as an indicator to detect and monitor the progress of the macular disorders.

5.1 NGRID TEST APPARATUS

The test apparatus is shown in Figure 22. A chinrest was used to assure that the participants could comfortably fixate at the center of the screen and provide their responses to test frames via keyboard, joystick or voice recognition input devices. An eye tracker is used to record the fixation location during the test. As shown in Figure 37, three different Visual Distortion Tests (VDT1, VDT2, and VDT3) from the NGRID default library were selected. Three different control input devices (keyboard, joystick and speech recognition) for each test were also selected. Each participant was trained for 10 minutes and instructed on each VD test with all the control input devices using counterbalanced perfect square of the above tests and input devices (3 VD Tests × 3 Input Devices).

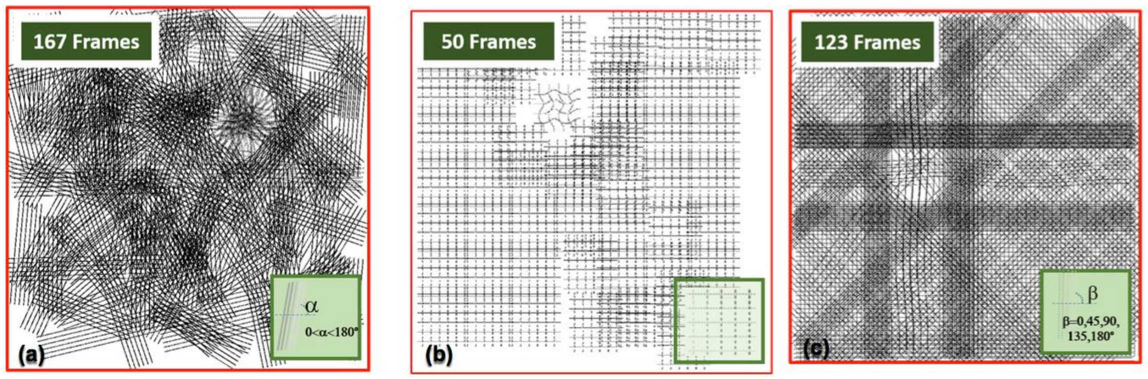


Figure 37 - Illustration of (a) VDT1, (b) VDT2 and (c) VDT3 including all frames collapsed in a single frame to better illustrate the entire test coverage.

5.2 VD TEST FRAMES AND DESIGNS

The VD test frames are generated via NGRID SVGEditor. VDT1 is composed of a total of 167 frames that were individually presented to the participants in a random order. Similarly, we have 50 frames for VDT2 and 123 frames for VDT3. For the purpose of illustration and to better explain the procedure, in Figure 37, we intentionally collapsed all the frames into a single frame for better visualization.

We conduct the tests on healthy and non-healthy participants. Healthy participants helped us to determine the systematic error and response time of the NGRID platform. Since we were measuring the systematic errors of the system (by having healthy participants), we deliberately added a few distorted patterns in the tests to assure participants carefully go through the test without being able to predict answers (otherwise they could say they saw all the frames as ‘Good’).

As seen in Figure 37, the projected frames cover the entire screen which covers the central 20 degrees of the visual field of view both in horizontal and vertical nasally. As seen in Figure 37b, VDT2 is very similar to VDT1, with the difference in grouping and the rotation angle (β) being 45°κ (where κ is an integer number between 0 to 8). VDT2 also includes a single distorted pattern similar to distorted patterns in VDT1. It is noteworthy to mention that we deliberately used only one distorted frame to simulate minor VD cases that affect only one small location. We wanted to assure participants can accurately detect single minor distortions as well. VDT3 is composed of parallel lines that are displayed with a random rotation angle (α) ranging from 0° to 360°.

5.3 NGRID TEST ON HEALTHY PARTICIPANT

To better characterize the NGRID platform and obtain possible systematic errors and response times of participants for various input devices, we tested the system on healthy participants. As shown in Figure 37, three different VD tests (VDT1, VDT2, and VDT3) from the NGRID default library were selected. Three different control input devices (keyboard, joystick and speech recognition) for each test was also selected. Twenty healthy participants (12 females, 8 males) with healthy corrected vision were recruited from the local university to test the NGRID platform. The mean age was 23.8 years (SD = 6.1). The healthy participants were selected to identify inherited systematic errors, any accuracy issues of the platform. The apparatus is the same as Figure 22. The VD tests were performed by all healthy participants. Table 4 provides a summary of each test for the number of errors and response times. It is noteworthy that information related to errors and response times were extracted from the detailed log of captured responses during the test. The log provides detailed recording of speech, keyboard and joystick actions as well as the timestamp of each event in milliseconds.

Table 4 - General Tests Specification

Test	Duration (seconds)	Number of Frames	Average and Standard Deviation of Errors	Average and Standard Deviation of Response Times (seconds)
VDT1	480	167	8.3% ± 4.2%	1.16 ± 0.45 sec
VDT2	180	50	15.6% ± 20.7%	0.97 ± 0.48 sec
VDT3	360	123	9.1% ± 5.5%	1.03 ± 0.42 sec

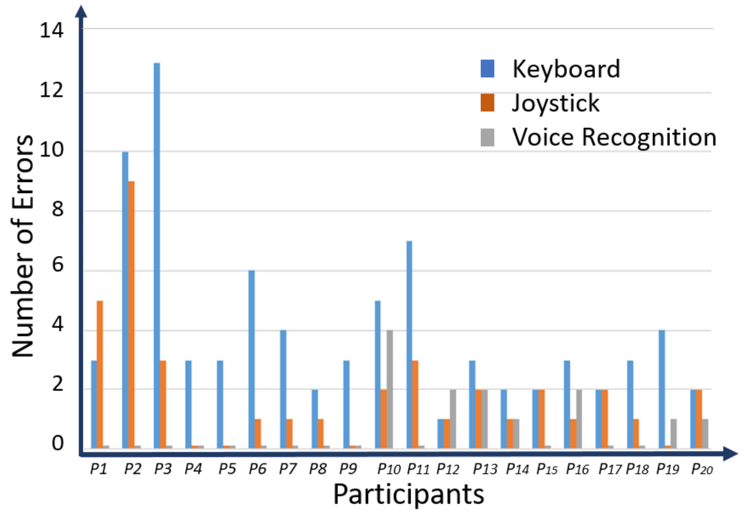


Figure 38 - Systematic error in VD tests performed by healthy participants based on a control device

As seen in Figure 38, the errors made by each one of the twenty healthy participants combined in all three VDT1, VDT2 and VDT3 tests separated by the control device (joystick, keyboard, and voice) are presented. For this purpose, we extracted and analyzed the responses associated with each one of the control devices separately from the logs. Speech recognition had the least number of errors as a participant could effortlessly say what they think about a frame seen, and no pointing task was involved. Participants made the maximum number of errors when they used the keyboard and made the minimum number of wrong responses when they used voice recognition. One may argue that participants have to pay attention to press the right keys in the keyboard and carefully fixate at the center of the display. In agreement with the experimental results, this increases the number of systematic errors. The joystick with less need for visual attention can be used which is in conformance with the presented experimental results. As expected, the participants, while looking at the center of the display, could speak their responses without losing their attention or fixation point. This is why the errors made by participants when they use voice recognition device were lower than the errors made using the other two control input devices.

It is noteworthy that VDT2 had only one artificially induced VD across all the frames while participants did not know how many artificially distorted frames exist in a test. We noticed a high number of errors (as high as 35%) on this particular test. We also noticed that many participants expressed frustration for missing the one artificially induced VD while doing the test. This can be considered as a systematic error that can be easily avoided by changing the color or shape of the patterns to make it less similar to other patterns. Another unlikely, but possible hypothesis was that some of the participants could suffer from a minor macular disorder. However, these participants were able to successfully identify the distorted pattern in VDT1 and VDT3. As aforementioned, we think that since the patterns in VDT2 were very similar to each other, and that there was only one artificially induced VD created, coupled with the fact that participants wanted to finish the test as fast as possible, caused this systematic error in VDT2. It is noteworthy that we decided to offer voice recording for the participants to allow post-processing of the answers in case a participant makes a mistake answering a frame to enable better error correction.

We did not generate any heatmap for healthy participants as we artificially created the VD frames and counting the errors would be equivalent of generating a heatmap (we know which frames contain VDs).

5.3.1 **CHARACTERIZATION AND SYSTEMATIC RESPONSE TIME**

Figure 41a-c show the response times for each frame done for every twenty participants using the keyboard, joysticks and voice input devices. In this result, the combination of three different VD tests performed by all twenty participants were used, where each participant is distinguished with a certain color in the result charts.

Participants had a fixed time to do the test and, depending on their speed, could go through all the frames a few times (all participants were able to go through all test frames and even repeat most of the frames one more time).

Voice recognition, in comparison to the joystick and keyboard, demonstrates a slower control device. One may argue that this depends on the speed of the voice recognition system, therefore, using a real-time digital system, it may be possible to reduce the response time. However, we also need to remember that speaking a simple single word like 'Good' or 'Bad' will take more time compared to a simple binary pointing task such as pressing a button [21]. As expected, the response time when the participants used the keyboard or joystick was almost the same.

It is noteworthy that the response time in each frame has three components. These components are (a) the required time that a participant needs to make a decision about a frame just seen and (b) the required time that a participant needs to react and take action using the control input device and (c) the required time that the computer needs to process the response received from the participant. We assumed the first component is almost the same when different control devices are used.

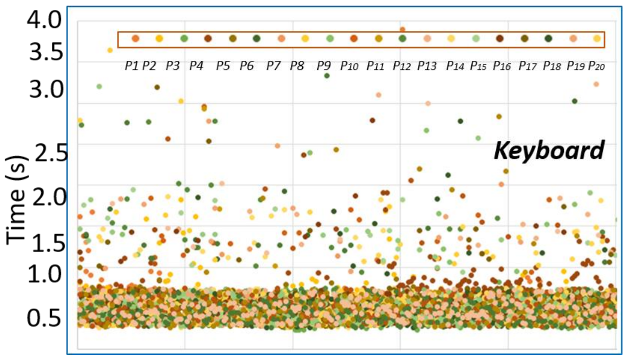
5.4 NGRID TEST ON NON-HEALTHY PARTICIPANT

In this section, we demonstrate and discuss the functionality and applicability of the proposed NGRID platform for non-healthy participants suffering from a macular condition called Central Serous Chorioretinopathy (CSCR or CSR) [2]. We selected CSR as CSR patients can recover their normal vision after a few months, which allows verifying our heatmap foundings directly with the patients in addition to their OCT images. CSR is an idiopathic macular condition and usually happens in men aged between 20 and 50 (10 annual cases in 100,000 males) [21], but is also associated with high levels of stress and usage of inhaled steroids [17], [83]. Studies show disturbing psychological events and high levels of stress can trigger CSCR in more than 75% of the patients [23]. Also, there are studies that link sleep disturbances, hypertension and autoimmune diseases to CSR [21], [22].

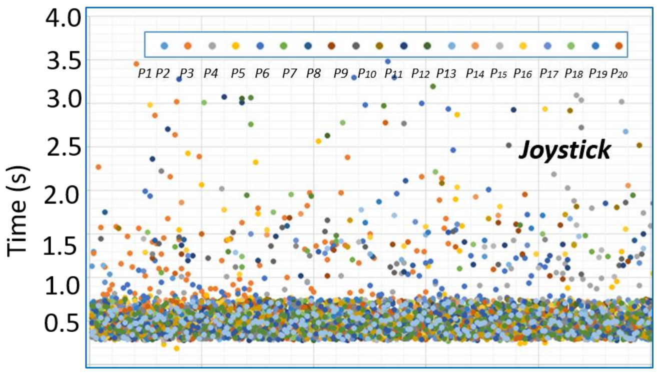
For diagnosis purposes, an eye-care professional starts examining a dilated eye and performs optical coherence tomography (OCT) and fluorescein angiography. This may reveal localized serous detachment of the neurosensory retina at the level of retinal pigment epithelium (RPE).

The Amsler Grid is used for documenting the affected areas of the visual field. Most of the patients are expected to have a full recovery between 1 and 6 months. In rare chronic cases, laser treatment, photodynamic therapy or even Ranibizumab Anti-VEGF are utilized (reduced visual acuity may still persist) [2], [24]. CSR can become a recurrent problem which makes follow-ups necessary [84].

(a)



(b)



(c)

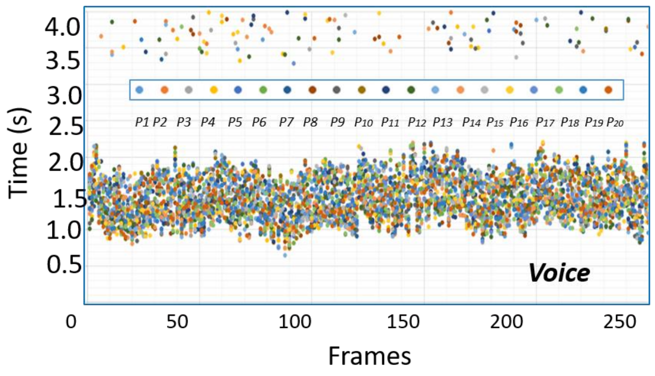


Figure 39 - The response time diagram related to control input devices (a) Keyboard, (b) Joystick and (c) Voice recognition used for performing VD tests. The voice recognition has the longest response times whereas the keyboard and joystick produced the shortest response times.

5.4.1 NGRID TEST AND VERIFICATION OF THE RESULTS AGAINST OCT IMAGES

Here the results of the NGRID test and corresponding OCT images are demonstrated and discussed. We used the OCT images of a participant as the control of

NGRID results. The subject is a 48-year-old male suffering from CSR. The rapid change of his macular condition allowed us to repeat the NGRID tests and compare with OCT follow up for observation of changes. The subject performed the VD tests using VDT1, VDT2, and VDT3, but without any distorted patterns. The responses for NGRID VD tests were collected, post-processed and heatmaps were generated through the NGRID Heatmap Generator program. Figure 40, Figure 41, Figure 42, Figure 43, Figure 44, and Figure 45 show both the NGRID heatmap and the corresponding OCT images. The experiments were performed on the right and left eyes. NGRID detected VD on each eye which fully conforms to OCT images. The heatmaps were generated using the NGRID Heatmap Generator program with the algorithm that was explained in Chapter 4. It is It is noteworthy that OCT images show how severely retina is impacted while heatmap results show how severely the visual field is impacted. Heatmap cannot replace OCT, however, the higher and more severe retinal changes in OCT, the more severe distortion exist in the visual field. Based on the results, the proposed NGRID platform can possibly be used as an accurate method to detect and assess the progress of CSR. We are conducting additional clinical trials to extend the same NGRID platform for other macular disorders such as AMD.

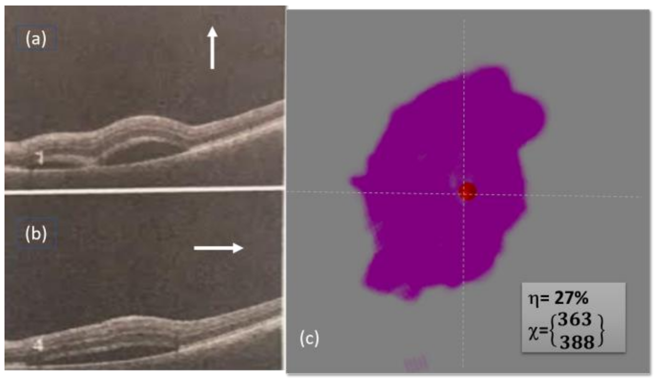


Figure 40 - First Macular Disorder Measurement Results of left Eye: (a)-(b) OCT vertical and horizontal OCT images and (c) NGRID heatmap results with $\eta = 27\%$ and χ=(363, 388)

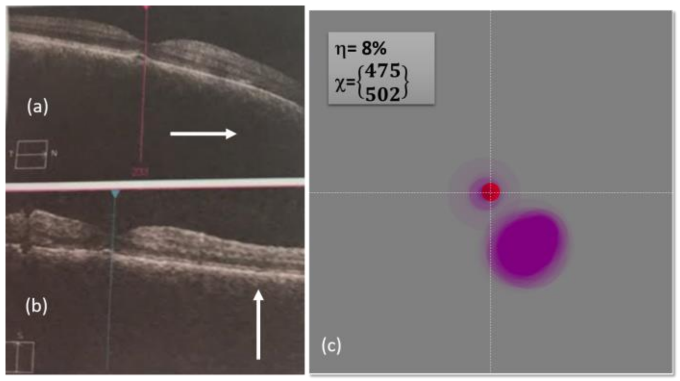


Figure 41 - First Macular Disorder Measurement Results of right Eye: (a)-(b) OCT vertical and horizontal OCT images and (c) NGRID heatmap results with $\eta = 8\%$ and χ=(475, 502)

The subject visited the Sunnybrook Hospital Emergency Room (ER) while observing a large yellow circle in the center of his vision. The subject also complained about metamorphopsia upon seeing the paper-based Amsler grid. AMD and CSR were the initial diagnosis based on the symptoms. However, OCT images from his left eye revealed a CSR lesion as in Figure 40 and related heatmap result. As per this NGRID result, the VD left eye of the patient is about 27%. The damage is almost placed in the center (363, 388) of the visual field where the exact center of the display is located at Cartesian coordinates x=400 and y=400. Figure 41, shows the OCT images along with their corresponding NGRID heatmap images related to the right eye in the first visit. As seen in this figure $\eta = 8\%$ and χ=(475, 502). Based on this result, the right eye also shows the CSR condition, however the patient did not complain because the effect of CSR was negligible in his vision. Figure 42, and Figure 43 show the OCT and heatmap NGRID results related to left and right eye respectively after a couple of weeks. As seen in these results, η has decreased in both eyes which shows that the CSR was reduced. However, in the third visit, the CSR had not decreased with almost the same η in the left eye as seen in Figure 44. For the final visit, as seen in Figure 45, the effect of CSR in the right eye decreasing as seen in both OCT and NGRID images. It is noteworthy that the location of the CSR disorder in the retina can move from one spot to another. This is because the main cause of CSR is the influence of fluid under the basement of the retina. This fluid can move over time due to the eye movement and other activities.

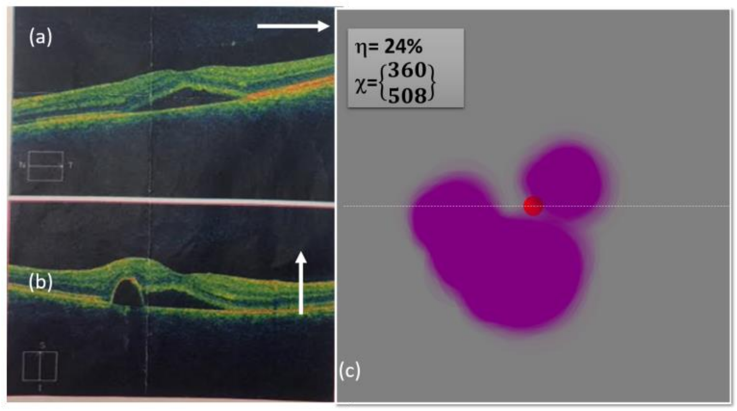


Figure 42 - Second Macular Disorder Measurement Results of left Eye: (a)-(b) OCT vertical and horizontal OCT images and (c) NGRID heatmap results with $\eta = 24\%$ and χ=(360, 508).

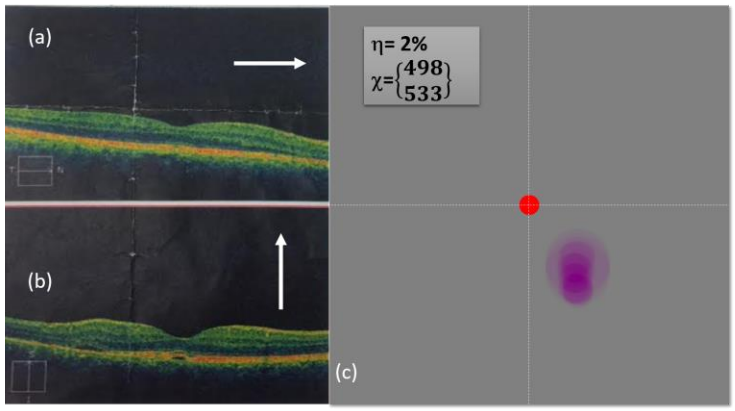


Figure 43 - Second Macular Disorder Measurement Results of right Eye: (a)-(b) OCT vertical and horizontal OCT images and (c) NGRID heatmap results with $\eta = 2\%$ and χ=(498, 533).

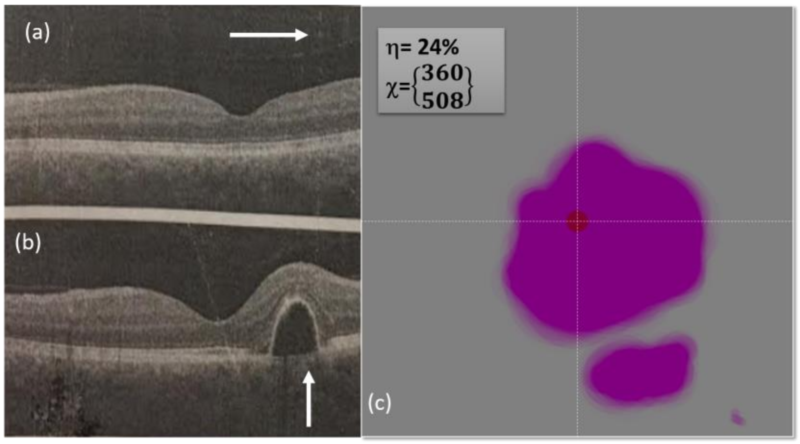


Figure 44 - Third Macular Disorder Measurement Results of left Eye: (a)-(b) OCT vertical and horizontal OCT images and (c) NGRID heatmap results with $\eta = 24\%$ and χ=(360, 508).

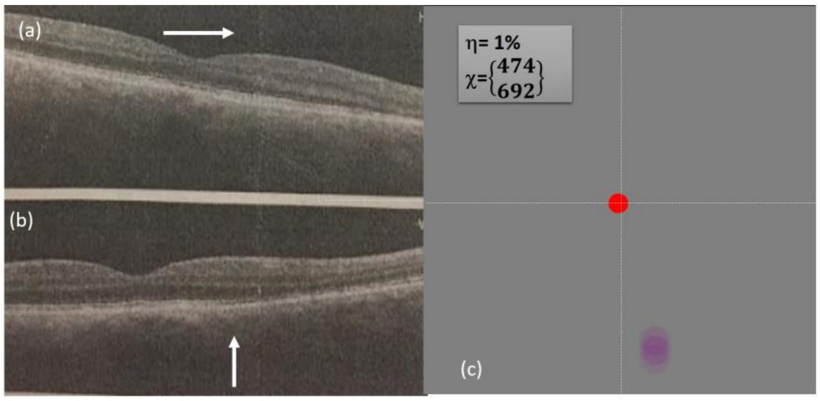


Figure 45 - Third Macular Disorder Measurement Results of right Eye: (a)-(b) OCT vertical and horizontal OCT images and (c) NGRID heatmap results with $\eta = 1\%$ and χ=(474, 692).

As aforementioned, the subject performed the VD tests and his responses were collected, post-processed, and heatmaps were generated using the Heatmap Generator program. We used our semi-spherical shape simulator program to get a better sense of how the patient may view the test patterns during his CSR progress. The results are illustrated in Figure 46b, Figure 47b, and Figure 48b. These results estimate the visual distortion of the patient during different stages of CSR.

We used the heatmap results to simulate VD effects on a normal visual field. Depend on the heatmap results, we manually selected up to three VD centers on the heatmap and applied low, medium, and high R values for the visually selected VD locations (the values are selected from the Radius matrix in the heatmap results). Furthermore, we used our semi-spherical model (discussed in chapter 3) to simulate the effect of the VDs on the visual field based on the manually chosen R values. The goal here was to show the simulated results to the CSR patients to verify that they experienced the VDs around the same VD areas that we detected through the heatmap results. We successfully confirmed that the patient experienced VDs in the locations and severity that we discovered through the heatmap.

5.4.2 DISCUSSION ON THE NGRID HEATMAP RESULTS

As seen in Figure 46a, Figure 47a, and Figure 48a, the vertical (AB) and horizontal (CD) cross section of OCT images are in agreement with the corresponding heatmap results. It is noteworthy to mention that it is expected that VD simulation and heatmaps conform to each other as we derived the visual field simulations directly from heatmap results, as explained in the previous section. However, we can also visually confirm that the length and severity of the VDs conform to what is seen in the heatmap.

In Figure 47a, by changing the vertical cross-section from AB, to EF, likely only one cavity related to the upper distorted area could be seen. Similarly, the OCT images in the direction of AC in Figure 46a, could still show two cavities. Please note the OCT images, heatmap graph and estimated VDs in the visual field of the right eye in the second visit is seen in Figure 47a-b. As seen in these images, the VD is very small. This is why the patient did not complain about his right eye in the first or even second visit. However, as seen in OCT images, test and modeling results, the presence of this distortion is observed. This can prove the importance of the proposed method for detecting the VDs.

5.4.3 VERIFICATION OF THE ESTIMATION OF VISUAL DISTORTION

Based on the results shown in Figure 40, and Figure 48, we can estimate the visual distortion in the left and/or right in the two visits. We applied the same model on a picture as seen in Figure 49. These images show the variation of the observed image by the patients in different tests. This allows us to have an idea on how the patient may see real-world scenery. As seen in these figures, the visual distortion is changing from one test to another due to the change of the cavity or the movement of fluid between the retinal upper layer and its basement.

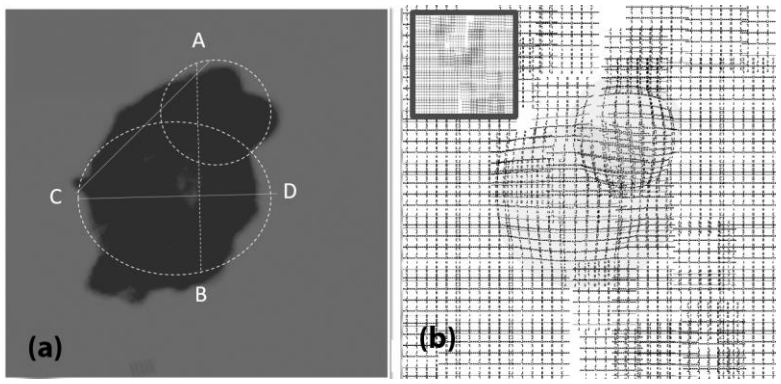


Figure 46 - Experimental results (Left eye, First visit): a) heatmap graph and b) estimated VD using the CSR model.

The observed distortion in each Figure 49a-c, can be calculated using the total area covered by the circle to the total area. However, this percentage can be more accurately calculated using the heatmap generated in each test. For this purpose, we can count the number of pink pixels (N) in each heatmaps as demonstrated via the distortion value (η) in Figure 40, and Figure 45. Therefore, the distortion (η) on Figure 49a-c is equal to 100xN/M which is 27%, 24%, and 2% respectively.

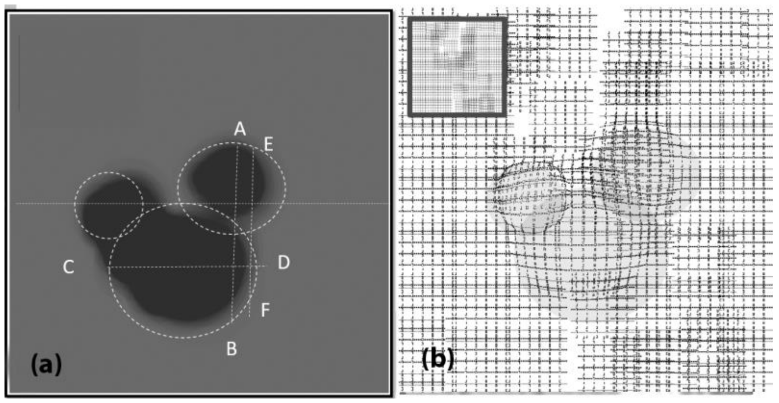


Figure 47 - Experimental results (Left eye, second visit): a) heatmap graph and b) estimated VD using the CSR model.

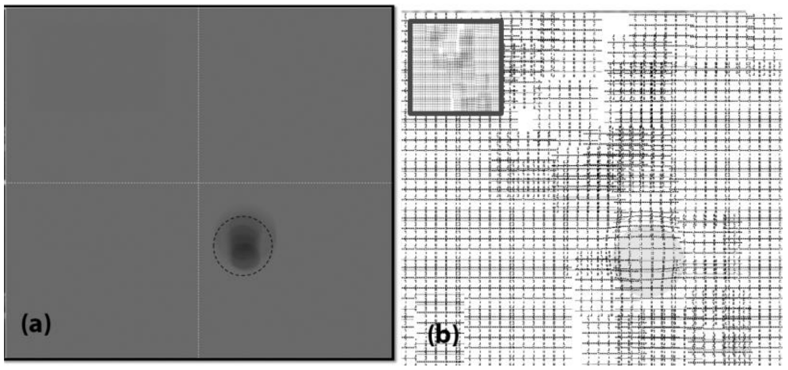
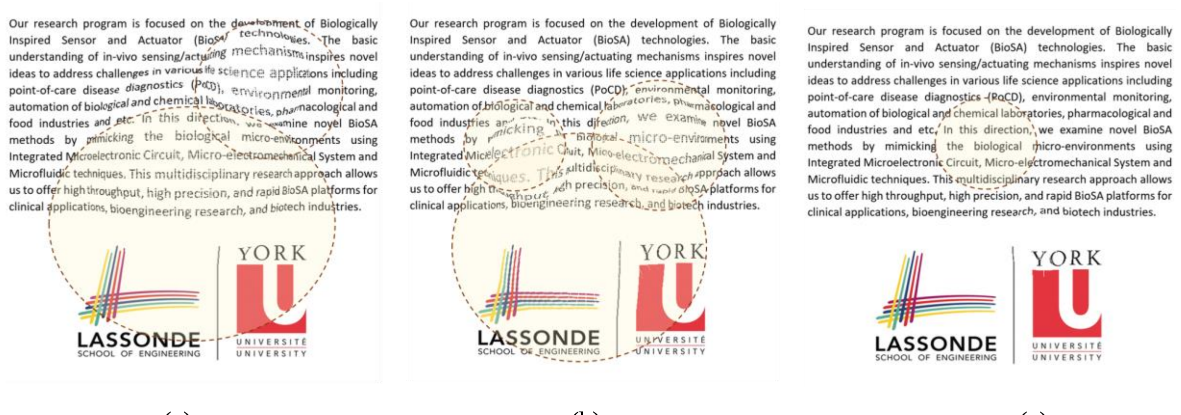


Figure 48 - Experimental results (Right eye, second visit): a) heatmap graph and b) estimated VD using the CSR model.


Our research program is focused on the development of Biologically Inspired Sensor and Actuator (BioSA) technologies. The basic understanding of in-vivo sensing/actuating mechanisms inspires novel ideas to address challenges in various life science applications including point-of-care disease diagnostics (PoCD), environmental monitoring, automation of biological and chemical laboratories, pharmacological and food industries and etc. In this direction, we examine novel BioSA methods by mimicking the biological micro-environments using Integrated Microelectronic Circuit, Micro-electromechanical System and Microfluidic techniques. This multidisciplinary research approach allows us to offer high throughput, high precision, and rapid BioSA platforms for clinical applications, bioengineering research, and biotech industries.

(a) (b) (c)

Figure 49 - Observed images by the patients suffering from CSR. In the (a) First visit, left eye, (b) in the second visit left eye and (c) in the second visit right eye.

5.5 SUMMARY

In this chapter, we reviewed the systematic errors and response time characterization of the NGRID platform through controlled tests done on healthy participants. Once we characterized the platform and had a clear understanding of response time and systematic errors, we conducted NGRID tests on unhealthy CSR participants. By comparing the OCT images and generated NGRID heatmap results, we confirmed that we can possibly use NGRID for detection and monitoring the progress of CSR. We are conducting additional clinical trials to confirm the same for other macular disorders such as AMD.

6 CONCLUSION AND FUTURE WORK

This chapter briefly describes the research achievements in this Ph.D. thesis followed by future works. We proposed a novel platform for detecting and monitoring of visual distortions (VD) in macular disorders. We developed the required hardware and software for generating the graphical patterns, displaying the patterns, conducting VD tests, collecting the patients' responses, and creating the associated heatmaps. A visual heatmap of VDs with quantitative measures can provide better indicators compared to traditional VD tests, such as the Amsler Grid or other related computerized methods, for eye-care physicians. The implementation and preliminary measurements and results were also reported. The system was also successfully tested and verified on healthy and unhealthy patients. This work has recently opened an avenue to obtain the required approval from Research Ethical Board (REB) in Sunnybrook Hospital, Toronto, Canada for running a clinical trial on a large number of patients.

6.1 RESEARCH ACHIEVEMENTS

In this thesis, we proposed a novel platform for evaluating the VDs caused by macular disorders such as CSR. As described in chapters 4-5, a unified software/hardware platform (NGRID) is used to generate custom-made VD Test patterns which are to be presented to patients. The patterns are projected to the patients' retina, and their answers are collected to create the assessment graph called heatmap. The software part of this platform includes numerous programs consisting of SVG Editor for generating the graphical test patterns, NGRID Test program to display the VD Test patterns to allow conducting VD tests and collection of the patients' responses, NGRID Heatmap Generator that analyzes and the patients' answers and create a heatmap to quantify VDs for further processing. We also created VD simulator programs to better visualize the effect of simplified VDs on patient visual field. These programs allow users including patients, ophthalmologist assistants, engineering administrators, and the chief research supervisor to directly or remotely (through the internet) access the NGRID cluster servers, run VD's test, modify the program's settings, collect various data and finally perform the detailed VD analysis and heatmap generation. The developed hardware system including secure simple board computer to run the NGRID VD Test programs as well as various sensory systems such as touch screen, customized joystick, eye-trackers, and speech analyzers enabled securely running programs by a large number of users. This Ph.D. research has resulted in the following contributions:

1. Proposed a novel VD model of macular disorder such as CSR (see Chapter 3).
2. Proposed a novel VD detection and assessment method called NGRID (see Chapter 4).
3. Performed VD test on healthy subjects using three different control device including keyboard, voice recognition system and joystick (see Chapter 5).

4. Demonstrated the VD tests using healthy subjects and proved that VR method reveals less error but higher test' duration time (see Chapter 5).
5. Performed VD test using unhealthy subject suffering from CSR (see Chapter 5).
6. Demonstrated the results of VD test on unhealthy subject and successfully compared with OCT results of the same patient (see Chapter 5).
7. Developed database layers and user interface programs related to NGRID platform (See Appendix A and Appendix B).
8. Designed and developed a custom made SVG editor dedicated to NGRID platform (see Appendix C).
9. Developed a hardware/software platform dedicated to NGRID platform (see Appendix D).
10. Developed heatmap method and dedicated to NGRID platform (see Appendix E).
11. Developed a CSR model program for evaluating the deformation (see Appendix F).
12. Incorporated voice recognition system as an input method (see Appendix A – Sensory Systems).
13. Developed data center program dedicated to NGRID platform (see Chapter 4).

As the outcome of this thesis, so far the following conference and journal papers have been submitted or accepted for the publications:

1. N. Mohaghegh, E. Ghafar-Zadeh, S. Magierowski, NGRID: A Novel Platform for Detection and Progress Assessment of Visual Distortion Caused by Macular Diseases, Elsevier Journal - Computers in Biology and Medicine, *Accepted*, June 2019.

2. N. Mohaghegh, E. Ghafar-Zadeh, and S. Magierowski, “Recent Advances of Computerized Graphical Methods for the Detection and Progress Assessment of Visual Distortion Caused by Macular Disorders,” Vision, vol. 3, no. 2, p. 25, Jun. 2019.
3. A Novel Method for Detection and Progress Assessment of Visual Distortion Caused by Macular Disorder: A Central Serous Retinopathy (CSR) Case Study, Journal of Medical and Biological Engineering Computing, *Submitted* 2019.
4. N. Mohaghegh, E. Ghafar-Zadeh, S. Magierowski, A Novel Macular Visual Distortion Assessment Method: A Chorioretinopathy (CSR) Case study, accepted IEEE ENBENG 2019 conference.
5. N. Mohaghegh, S. Munidasa, Q. Owen, Z. Zahio, E. Ghafar-Zadeh, S. Magierowski, Age-Related Macular Degeneration Diagnostic Tools: Hardware and Software Development, IEEE MWSCAS, Windsor, August 2018.
6. N. Mohaghegh, E. Ghafar-Zadeh, S. Munidasa, S. Magierowski, Toward Age-related Macular Degeneration (AMD) Big Data: Hardware and Software Design and Implementation, IEEE CCECE 2017, April 29th-May 3rd, Windsor, Canada.
7. N. Mohaghegh, E. Ghafar-Zadeh, S. Magierowski, A Wearable Diagnostic System for Age-Related Macular Degeneration Conference: 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2016.

The research work in this thesis has opened new avenues for developing novel VD technologies and created a novel device that can be further modified and used for ophthalmology studies. In the next section, some of the unmet challenges are put forward for future works.

6.2 FUTURE WORKS

The research described in this thesis can be continued in the following directions as seen in Figure 50 and described below.

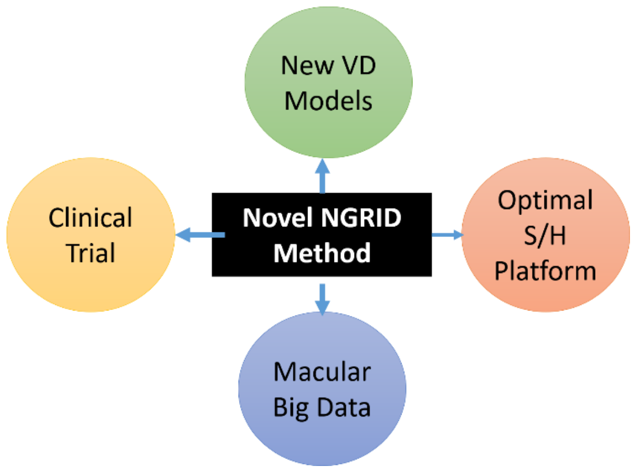


Figure 50 - Future works plans

6.2.1 DEVELOPING ACCURATE VD MODEL FOR VARIOUS MACULAR DISORDERS

In this thesis, we introduced a new approach by developing a method to explain the deformation of a graphic pattern observed by the patients suffering from macular disorders. Based on this model, it is possible to approximately estimate the images observed by the patient. A macular disorder's model, not only can be used to accurately measure the VD changes, but also it inspires the development of new technologies for the correction of macular disorders if they are in the early stages of their development. In this thesis, we proposed low-complexity spherical shape models for the comparison of NGRID heatmap and the OCT results. In the future, more realistic models should be developed. These models can be proposed for various macular disorders including AMD, Macular Holes, and Macular Edema.

6.2.2 CLINICAL TRIALS ON PATIENTS SUFFERING FROM MACULAR DISORDERS

The proposed NGRID platform in this Ph.D. thesis has successfully been tested, and its applicability was proved by using human healthy and unhealthy participants. However, complete clinical trials using a large number of patients suffering from various macular disorders such as CSR and AMD should be performed to fully characterize the functionality, accuracy, and speed of VD tests. These clinical trials can help to optimize the proposed methods and take another step for commercialization. The proposed platform allows us to perform the VD tests and collect the test data.

6.2.3 DEVELOPMENT OF OPTIMAL NGRID PLATFORM

This platform can be further enhanced by upgrading the hardware and related software system in the future. For instance, by adding a faster eye-tracker (currently we are using a 120 Hz tracker), the errors due to the fixation issues can be reduced. Also, the voice recognition system can be performed on a new application-specific hardware board FPGA or similar technologies (such as new Intel-Compute-Stick-II) for real-time voice recognition and consequently reduce the test’s duration time.

Using SVG allows us not only to be able to present the VD tests to patients via low-cost commodity LCD monitors, but also to be able to show the same tests in smaller mobile as well as head-mounted displays (e.g. Oculus Rift). Due to budgetary constraints we could not use any medical grade HFFS (High-Transmittance Fringe Field Switching) or AFFS+ (Advanced Fringe Field Switching) monitors which have enhanced color calibration, near optimal color and luminosity reproduction as well as minimum color distortion. For proper commercialization, more advanced medical-grade hardware will be needed.

6.2.4 MACULAR BIG DATA USING NGRID PLATFORM FOR OTHER APPLICATIONS

To date, a few papers reported the development of medical big-data[86]–[88] for various biomedical applications but the development of a big data repository for macular disorders or retinal disorders had not been met yet. Assuming that the proposed NGRID system can be used for a large number of patients around the world, the proposed platform can be used to collect a large amount of data that can be used for ophthalmology study or pharmaceutical purposes. For instance, the effect of the drug in a patient suffering from the macular disorder can be tracked by running the VD tests over the days, and the related progression of disorder can be evaluated. Therefore, this thesis has opened a novel approach to develop new Big Data.

APPENDIX A – NGRID TEST APPLICATION

NGRID Test Application is a thick graphical Microsoft Windows-based desktop application that can run on commodity personal computers and laptops. This application allows for various NGRID Tests to be done for patients at hospitals or at comfort of their homes. Furthermore, the application helps collect test results and send it to NGRID Datacenter for post-processing and archival. In this section we explain the technical details around implementation of NGRID Desktop Application.

The desktop application has the following main tasks:

- **Fetch the test data**: securely connect to NGRID Datacenter and fetch NGRID Test graphics and settings
- **Display the test**: Renders the graphics for the patient
- **Sensory system**: Allows for variety of sensory devices to be used during the test. Devices such as keyboard, mouse, voice and eye-tracker, joystick are some examples.
- **Collection and validation**: Collect the answers that patient provides and also perform basic validation before storing the test answers.
- **Archival:** Securely send the test results to NGRID Datacenter to be archived and post-processed for medical practitioner.

The desktop application can run in Online mode or Offline mode. While working in online mode, constant connection to NGRID Datacenter is established (through Secure

WebSocket) which allows sending push notification from hospital to the application (this is useful to instantiate an update on the application to install newer versions). In offline mode, all test results are stored locally until online connection establishes to send the results to datacenter. In the following sections we discuss the details around each aforementioned main tasks.

FETCH THE TEST DATA

As shown in sample code below, we contact https://ngrid.website and fetch all the associated data that is needed for a given trial id that patient is enrolled in. The data includes participant ID and associated test data (mainly test SVGs and settings). Upon successful fetch and validation the test will be started.

```
//fetch a test data for a given trial id for the patient
var trial_id = $("#trial_id").val();
console.log("Trial id is: " + trial_id);
$.ajax({
        url: "https://ngrid.website:9000/app/getAllTrialData/" + trial_id,
        async: false,
        type: "GET",
        beforeSend: function(xhr){xhr.setRequestHeader('X-NGRID-DESKTOP-APP', 'true');},
        success: function(data) {
                console.log("Trial Data is fetched.");
        }
}).done(function(data) {
        data.svgs.sort(SVGSortBySeqComprator);
        trial_fetched_data = data;
        console.log("Trial Data is fetched and svgs are sorted by sequence id.");

        //remove xmlns parent tag as we use our own schema
        $.each(trial_fetched_data.svgs, function(i, item) {
                var xml = $.parseXML(item.xmlContent);
                $xml = $( xml );
                var temp_svg_parsed = $xml.find('svg');
                item.xmlContent =
temp_svg_parsed[0].innerHTML.replace('xmlns="http://www.w3.org/2000/svg"',"").trim();
        });

        //set the gloabl values to be used during the test for logging
        global_partcipant_id = trial_fetched_partcipant_id;
        global_trial_id = trial_fetched_data.trials.trial_id;
        global_test_name = trial_fetched_data.tests.test_name;
        $.each(trial_fetched_data.settings, function(i, item) {
                if ( item.id == trial_fetched_data.trials.settingId)
                {
                        global_setting_details =item.settingContent;
                        global_setting_id = item.settingName;
                }
        });
        console.log("global_partcipant_id: " + global_partcipant_id);
        console.log("global_trial_id: " + global_trial_id);
        console.log("global_test_name: " + global_test_name);
        console.log("global_setting_id: " + global_setting_id);
        console.log("global_setting_details: " + global_setting_details);


        $("#div_trial_id_container").hide(750);
        $("#div_trial_id_container").remove();

        start_the_exam();
}
```

DISPLAY THE TEST

One of the main objective of the NGRID platform is to allow macular graphical tests to be done seamlessly in desktop, mobile and tablet computers. To do this Standard Vector Graphics (SVG) are used as the main protocol to create test data. However the SVG needs to be rendered. We use HTML5, CSS3 and Javascript along with SVG to create an NGRID VD Test and use Chromium Embedded Framework (CEF) to render and display the test. This allows the test to be seamlessly rendered in desktop, tablet and mobile devices without any change. We heavily customize CEF to allow asynchronous script execution from C# and browser side to create a fully dynamic and intractable NGRID test. We only show important CEF modifications below (complete code is over 2000 lines):

```
public partial class NGRIDDesktopApp : Form
{
        public CefSettings cef_settings = new CefSettings();
        public ChromiumWebBrowser chromeBrowser;

        public NGRIDDesktopApp()
        {
                InitializeComponent();

                // Specify Global Settings
                this.cef_settings = new CefSettings();
                //cef_settings.CachePath = "cache";
                cef_settings.CefCommandLineArgs.Add("enable-media-stream", "1");
                cef_settings.LogSeverity = LogSeverity.Verbose;
                cef_settings.CefCommandLineArgs.Add("disable-gpu", "1");
                cef_settings.CefCommandLineArgs.Add("no-proxy-server", "1");

                //Add  NGRID protocol:  i.e. ngrid://commands
                cef_settings.RegisterScheme(new CefCustomScheme
                {
                        IsStandard = true,
                        SchemeName = "ngrid",
                        SchemeHandlerFactory = new CefSharpSchemeHandlerFactory(),
                        DomainName = "",
                IsSecure = true
            });

            Cef.Initialize(cef_settings);

            while (!Cef.IsInitialized)
            {
                Thread.Sleep(100);
            }

            chromeBrowser = new ChromiumWebBrowser("http://localhost:8080/ar/index.html")
            {
                BrowserSettings =
                {
                   DefaultEncoding = "UTF-8",
                        WebGl = CefState.Disabled
                },
                RequestHandler = new NavidRequestHandler()
            };


            //Register objects to be called from JS (both sync and async are exposed).
            chromeBrowser.JavascriptObjectRepository.Register("bound", new BoundObject(),
isAsync: false, options: BindingOptions.DefaultBinder);
            chromeBrowser.JavascriptObjectRepository.ResolveObject += (sender, e) =>
            {
                var repo = e.ObjectRepository;
                if (e.ObjectName == "boundAsync")
                {
                        repo.Register("boundAsync", new AsyncBoundObject(), isAsync:
true, options: BindingOptions.DefaultBinder);
                }
            };

            SingletonBrowsers.Instance.browsers.Add("tab1", new NavidWebBrowserWithSignal {
browser = chromeBrowser });

        SingletonBrowsers.Instance.browsers["tab1"].setBrowserLoadingStateChangeNotificationEvent(
);
        }

        private void OnFormClosing(object sender, FormClosingEventArgs e)
        {
            Cef.Shutdown();
        }

        private void OnButtonShowDevToolsChrome_Click(object sender, EventArgs e)
        {
            this.chromeBrowser.ShowDevTools();
        }


        //login
        private void button1_Click(object sender, EventArgs e)
        {
            chromeBrowser.ExecuteScriptAsync("login.js", "show_login();");
            chromeBrowser.Invalidate();
        }

        //...
}



public class NavidWebBrowserWithSignal
{
        public ChromiumWebBrowser browser { set; get; }
        public bool isLoading = true;
        public void setBrowserLoadingStateChangeNotificationEvent()
        {
                browser.LoadingStateChanged += Browser_LoadingStateChanged;
        }

        private void Browser_LoadingStateChanged(object sender, LoadingStateChangedEventArgs e)
        {
                this.isLoading = e.IsLoading;
        }

        public void EvaluateScript(string script, string method)
        {
                // only call via UI thread to get correct scheduler
                var scheduler = System.Threading.Tasks.TaskScheduler.Default;

                //1 minute timeout
                var task = new System.Threading.Tasks.Task( () =>  EvaluateScript(script, method,
TimeSpan.FromMinutes(1)) );
                task.Start(scheduler);
        }

        public object EvaluateScript(string script, string method, TimeSpan timeout)
        {
                //make sure browser is ideal
                int count = 600;
                while (isLoading && count > 0)
                {
                        Thread.Sleep(100);
                        count--;
                }

                object result = null;

                if (browser.IsBrowserInitialized && !browser.IsDisposed && !browser.Disposing)
                {
                        var task = browser.EvaluateScriptAsync(script, method, timeout);
                        var complete = task.ContinueWith(t =>
                        {
                                if (!t.IsFaulted)
                                {
                                        var response = t.Result;
                                        result = response.Success ? (response.Result ?? "null") :
response.Message;
                                }
                        }, TaskScheduler.Default);
                        complete.Wait();

                }
                return result;
        }

        public event EventHandler TriggerJSExec;
        protected virtual void OnThresholdReached(EventArgs e)
        {
                EventHandler handler = TriggerJSExec;
                if (handler != null)
                {
                        handler(this, e);
                }
        }

}

//to customize headers sent
public class NavidRequestHandler :  CefSharp.Handler.DefaultRequestHandler
{
        public override bool OnBeforeBrowse(IWebBrowser browserControl, IBrowser browser, IFrame
frame, IRequest request, bool isRedirect)
```

```
        {
                Console.WriteLine("request.Url is: " + request.Url);
                return base.OnBeforeBrowse(browserControl, browser, frame, request, isRedirect);
        }


        public override bool OnResourceResponse(IWebBrowser browserControl, IBrowser browser,
IFrame frame, IRequest request, IResponse response)
        {
                Console.WriteLine("request.Url is: " + request.Url);
                string[] header_keys = response.ResponseHeaders.AllKeys;
                foreach (string key in header_keys)
                {
                        string value = response.ResponseHeaders[key];
                        Console.WriteLine(key + " ---> " + value);
                }

                return base.OnResourceResponse( browserControl,  browser,  frame,  request,
response);
        }
}




public class CefSharpSchemeHandlerFactory : ISchemeHandlerFactory
{
        public const string SchemeName = "http";

        private static readonly IDictionary<string, string> ResourceDictionary;

        static CefSharpSchemeHandlerFactory()
        {
                ResourceDictionary = new Dictionary<string, string>
                {
                        { "/home.html", "home.html" }
                };
        }

        public IResourceHandler Create(IBrowser browser, IFrame frame, string schemeName, IRequest
request)
        {
                var uri = new Uri(request.Url);
                var fileName = uri.AbsolutePath;
                string resource;

                if (uri.Host == "ngrid.website" && schemeName == "ngrid")
                {
                        return new CefSharpSchemeHandler();
                }
                else if (string.Contains(fileName, ".html", StringComparison.OrdinalIgnoreCase))
                {
                        return ResourceHandler.FromString("", ".html");
                }
                else if (ResourceDictionary.TryGetValue(fileName, out resource) &&
!string.IsNullOrEmpty(resource))
                {
                        var fileExtension = Path.GetExtension(fileName);
                        return ResourceHandler.FromString(resource, includePreamble: true,
mimeType: ResourceHandler.GetMimeType(fileExtension));
                }

                return null;
        }
}



internal class CefSharpSchemeHandler : ResourceHandler
```

```
{
        public override Stream GetResponse(IResponse response, out long responseLength, out string
redirectUrl)
        {
                string[] header_keys = response.ResponseHeaders.AllKeys;
                foreach (string key in header_keys)
                {
                        string value = response.ResponseHeaders[key];
                        Console.WriteLine(key + " ---> " + value);
                }

                redirectUrl = null;
                responseLength = -1;

                response.MimeType = MimeType;
                response.StatusCode = StatusCode;
                response.StatusText = StatusText;
                response.ResponseHeaders = Headers;

                if (ResponseLength.HasValue)
                {
                        responseLength = ResponseLength.Value;
                }
                else
                {
                        //If no ResponseLength provided then attempt to infer the length
                        if (Stream != null && Stream.CanSeek)
                        {
                                responseLength = Stream.Length;
                        }
                }

                return Stream;
        }

        public override bool ProcessRequestAsync(IRequest request, ICallback callback)
        {
                callback.Continue();
                return true;
        }
}
```

As mentioned, we heavily customize CEF and run the browser in headless mode (without any borders or window decorations). This means the NGRID App will be in full screen and the topmost application. Also we clip the cursor and limit the keyboard events to make sure wrong events are ignored and no other application can interfere the test. This requires fundamental changes in the core windows applications in an unmanaged (known as unsafe mode). Below we demonstrate a few of these advanced techniques in order to have the CEF to remain the topmost application and also clip the mouse cursor within the CEF boundaries.

```
internal static class NGRIDNativeMethods
```

```
{
    // See http://msdn.microsoft.com/en-us/library/ms649021%28v=vs.85%29.aspx
    public const int WM_CLIPBOARDUPDATE = 0x031D;
    public static IntPtr HWND_MESSAGE = new IntPtr(-3);

    // See http://msdn.microsoft.com/en-us/library/ms632599%28VS.85%29.aspx#message_only
    [DllImport("user32.dll", SetLastError = true)]
    [return: MarshalAs(UnmanagedType.Bool)]
    public static extern bool AddClipboardFormatListener(IntPtr hwnd);

    // See http://msdn.microsoft.com/en-us/library/ms633541%28v=vs.85%29.aspx
    // See http://msdn.microsoft.com/en-us/library/ms649033%28VS.85%29.aspx
    [DllImport("user32.dll", SetLastError = true)]
    public static extern IntPtr SetParent(IntPtr hWndChild, IntPtr hWndNewParent);

    [DllImport("user32.dll")]
    private static extern void ClipCursor(ref Rectangle rect);
}
```

SENSORY SYSTEM

There are wide range of supported sensors and input devices for NGRID desktop application. We allow keyboard, mouse, joystick, voice recognition, eye tracking right of the box. Here is a sample code for voice recognition that not only detects a preset of defined vocabulary, but also record the entire speeches during the test and mark each test answer within the recording. This will allow to verify the speech recognition accuracy as the entire spoken test answers can be manually verified.

```
namespace NGRIDDesktopApp
{
    /// <summary>
    /// We use simple words in our speech engine
    /// Each word has a label text and asscoiated text verb
    /// e.g. in start-start: start is a the label that speech
    /// engine will detect and upon detection will output the
    /// verb which is start.
    /// </summary>
    public class Word
    {
        public string Text { get; set; }
        public string Verb { get; set; }
    }

    public class NGRIDSpeechToTextEngine
    {
        public SQLiteConnection db_connection;

        public DateTime test_start_time = DateTime.Now;
        public string partcipant_id;
        private SpeechRecognitionEngine speech_recognition_engine;
        public string test_type;
        public string trial_id;

        public WebSocketServer websocket = new WebSocketServer();
        private readonly List<Word> words = new List<Word>();

        public NGRIDSpeechToTextEngine(WebSocketServer websocket, SQLiteConnection db_connection,
string trial_id, string test_type, string partcipant_id, DateTime t_now)
        {
            this.websocket = websocket;
            this.db_connection = db_connection;
            this.trial_id = trial_id;
            this.test_type = test_type;
            this.partcipant_id = partcipant_id;

            try
            {
                // create the engine
                speech_recognition_engine = create_speech_engine("en-US");

                // hook to events
                speech_recognition_engine.AudioLevelUpdated += engine_audio_level_updated;
                speech_recognition_engine.SpeechRecognized += engine_speech_recognized;

                speech_recognition_engine.InitialSilenceTimeout = TimeSpan.FromSeconds(3600);
                speech_recognition_engine.EndSilenceTimeoutAmbiguous =
TimeSpan.FromMilliseconds(80);
                speech_recognition_engine.BabbleTimeout = TimeSpan.FromMilliseconds(80);
                Console.WriteLine("============================================");
                Console.WriteLine("BabbleTimeout: {0}", speech_recognition_engine.BabbleTimeout);
                Console.WriteLine("InitialSilenceTimeout: {0}",
speech_recognition_engine.InitialSilenceTimeout);
                Console.WriteLine("EndSilenceTimeout: {0}",
speech_recognition_engine.EndSilenceTimeout);
                Console.WriteLine("EndSilenceTimeoutAmbiguous: {0}",
                speech_recognition_engine.EndSilenceTimeoutAmbiguous);
                Console.WriteLine("============================================");

                // load dictionary
                load_engine_grammer();

                // use the system's default microphone
                speech_recognition_engine.SetInputToDefaultAudioDevice();

                // start listening
                speech_recognition_engine.RecognizeAsync(RecognizeMode.Multiple);

                //record("open new Type waveaudio Alias recsound", "", 0, 0);
                //record("record recsound", "", 0, 0);

                test_start_time = t_now;
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message, "Voice recognition failed");
            }
        }

        private void db_run_statment(string sql)
        {
            var command = new SQLiteCommand(sql, db_connection);
            command.ExecuteNonQuery();
        }

        public void websocket_send(string message)
        {
            foreach (var session in websocket.GetAllSessions()) session.Send(message);
        }


        [DllImport("winmm.dll", EntryPoint = "mciSendStringA", ExactSpelling = true, CharSet =
CharSet.Ansi, SetLastError = true)]
```

```
        private static extern int record(string lpstrCommand, string lpstrReturnString, int 
uReturnLength, int hwndCallback);

        private SpeechRecognitionEngine create_speech_engine(string preferredCulture)
        {
            foreach (var config in SpeechRecognitionEngine.InstalledRecognizers())
                if (config.Culture.ToString() == preferredCulture)
                {
                    speech_recognition_engine = new SpeechRecognitionEngine(config);
                    break;
                }

            // if the desired culture is not found, then load default
            if (speech_recognition_engine == null)
            {
                Console.WriteLine(
                    "The  lang is not installed on this machine, the speech-engine will continue 
using "

                    + SpeechRecognitionEngine.InstalledRecognizers()[0].Culture + " as the default 

culture.",
                    "lang " + preferredCulture + " not found!");

                speech_recognition_engine = new 
SpeechRecognitionEngine(SpeechRecognitionEngine.InstalledRecognizers()[0]);
            }

            return speech_recognition_engine;
        }

        private void load_engine_grammer()
        {
            try
            {
                var texts = new Choices();
                var lines = File.ReadAllLines(Environment.CurrentDirectory + "\\ngrid.speech");
                foreach (var line in lines)
                {
                    // skip commentblocks and empty lines..
                    if (line.StartsWith("--") || line.StartsWith("#") || line == string.Empty) 
continue;

                    // split the line
                    var parts = line.Split('|');

                    // add commandItem to the list for later lookup or execution
                    words.Add(new Word {Text = parts[0], Verb = parts[1]});

                    // add the text to the known choices of speechengine
                    texts.Add(parts[0]);
                }

                var wordsList = new Grammar(new GrammarBuilder(texts));
                speech_recognition_engine.LoadGrammar(wordsList);
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        //get the commmand payload
        private string get_command_text(string command)
        {
            try
            {
               var cmd = words.Where(c => c.Text == command).First();
               return cmd.Verb;
            }
            catch (Exception)
            {
                return command;
            }
        }


        //Handles the SpeechRecognized event of the engine control.
        private void engine_speech_recognized(object sender, SpeechRecognizedEventArgs e)
        {
            var t_now = DateTime.Now;

            if ((t_now - test_start_time).TotalSeconds < 2) return;

            var log_object = new LogsObject();
            log_object.participant_id = partcipant_id;
            log_object.trial_id = trial_id;
            log_object.test_type = test_type;
            var user_said = get_command_text(e.Result.Text);
            //log_object.message = "At offset " + (t_now - test_start_time) + " user said " + 
user_said;
            log_object.message = "At offset " + 
speech_recognition_engine.AudioPosition.ToString("c") + " user said " + 
                                 user_said;
            log_object.timestamp = t_now.ToString("yyyy-MM-dd HH:mm:ss.fff");
            log_object.action = "Voice";

            db_run_statment(log_object.sqlite_create_insert_statment()); //store to db
            websocket_send("voice: " + user_said);
        }

        // Handles the AudioLevelUpdated event of the engine control for the progress bar and 
audio volume level
        private void engine_audio_level_updated(object sender, AudioLevelUpdatedEventArgs e)
        {
            //prgLevel.Value = e.AudioLevel;
        }


        ///call once we want to Close and stop the recording
        public void Close()
        {
            //for debug
            record("save recsound " + this.file_name, "", 0, 0);
            record("Close recsound", "", 0, 0);

            // unhook events
            speech_recognition_engine.RecognizeAsyncStop();
        }

        public void Dispose()
        {
            //clean references to speech engine
            speech_recognition_engine.Dispose();
        }
    }
}
```

All NGRID sensors are given an instance of a WebSocket that they can have a direct two-way communication of events and commands to and from the desktop app. This allows interfacing a wide range of sensors to the system.

We also provide Serial Communication Interface (SCI), Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I^2C) protocols in our hospital version of the desktop which is accompanied by ARM embedded board to allow integration of even more customized sensors. The ARM embedded board also is given an instance of WebSocket that allows it to act as hub for SCI, SPI and I^2C to aggregate and broadcast events. Furthermore, we also allow low-level micro-controller integration from the PC or ARM board to enable control of advanced analog and PWM sensors that may require hard real-time signaling. Below is a sample code that demonstrate enablement of analog readings and pulse with modulating. The code will be compiled into a firmware to run on the microcontroller interfaced with the ARM board:

```
/************************************ ATD STARTS ***********************************/

/**
 * power up ATD and enable PAD04 pin
 */
static void atd_init(void)
{
        ATD0CTL2 = 0x80;     /* Power up A/D, no interrupts */
        ATD0CTL3 = 0x00;     /* Doe eight conversions */
        //ATD0CTL4 = 0x85;   /* 8-bit mode */
        ATD0CTL4 = 0x05;     /* 10-bit mode */
        ATD0CTL5 = 0xA4;     /* 1 0 1 0 0 1 0 0
                                | | | |   \___/
                                | | | |     |
                                | | | |     \__ Bit 4 of Port AD --> PAD04
                                | | | \________ MULT = 0 => one channel only
                                | | \__________ Scan = 1 => continuous conversion
                                | \____________ DSGN = 0 => unsigned
                                \______________ DJM = 1 => right justified */
}

/************************************ PWM STARTS ***********************************/

#define PORTIO_8        *(volatile unsigned char *)
#define PORTIO_16               *(volatile unsigned short int *)

#define IO_BASE  0

#define     PWMCNT01_16BIT      PORTIO_16(IO_BASE + 0xac)     /* pwm channel 0,1 counter, 16bit */
#define     PWMCNT23_16BIT      PORTIO_16(IO_BASE + 0xae)     /* pwm channel 2,3 counter, 16bit */
#define     PWMCNT45_16BIT      PORTIO_16(IO_BASE + 0xb0)     /* pwm channel 4,5 counter, 16bit */
#define     PWMCNT67_16BIT      PORTIO_16(IO_BASE + 0xb2)     /* pwm channel 6,7 counter, 16bit */
#define     PWMPER01_16BIT      PORTIO_16(IO_BASE + 0xb4)     /* pwm channel 0,1 period, 16bit */
#define     PWMPER23_16BIT      PORTIO_16(IO_BASE + 0xb6)     /* pwm channel 2,3 period, 16bit */
#define     PWMPER45_16BIT      PORTIO_16(IO_BASE + 0xb8)     /* pwm channel 4,5 period, 16bit */
#define     PWMPER67_16BIT      PORTIO_16(IO_BASE + 0xba)     /* pwm channel 6,7 period, 16bit */
#define     PWMDTY01_16BIT      PORTIO_16(IO_BASE + 0xbc)     /* pwm channel 0,1 duty cycle, 16bit
*/
```

```
#define     PWMDTY23_16BIT      PORTIO_16(IO_BASE + 0xbe)     /* pwm channel 2,3 duty cycle, 16bit
*/
#define     PWMDTY45_16BIT      PORTIO_16(IO_BASE + 0xc0)     /* pwm channel 4,5 duty cycle, 16bit
*/
#define     PWMDTY67_16BIT      PORTIO_16(IO_BASE + 0xc2)     /* pwm channel 6,7 duty cycle, 16bit
*/


/*
 * below are some temp. vars. to save the values we get from PC
 */
char pwm_mode, pwm_channel_number, pwm_polarity, pwm_alignment, pwm_is_using_scaled_clock,
pwm_main_clock_divider;
unsigned int pwm_scale_clock_divider, pwm_period_reg_value, pwm_duty_reg_value;

/*
 * bit value of enable and disable flags in PWME
 */
#define ENABLE 1
#define DISABLE 0

/*
 * bit value of polarity
 * polarity can be high at start of the pulse or can be low
 */
#define PWM_POLARITY_IS_HIGH_AT_BEGINNING_OF_THE_PULSE 1
#define PWM_POLARITY_IS_LOW_AT_BEGINNING_OF_THE_PULSE 0

/*
 * alignment: pulse can be left or center aligned
 */
#define PWM_ALIGNMENT_CENTER_ALIGN 1
#define PWM_ALIGNMENT_LEFT_ALIGN 0


/*
 * 16bit or 8bit of opration
 */
#define PWM_8BIT_OPERATION_MODE 8
#define PWM_16BIT_OPERATION_MODE 16

/*
 * channels
 */
#define PWM_8BIT_CHANNEL_0 0
#define PWM_8BIT_CHANNEL_1 1
#define PWM_8BIT_CHANNEL_2 2
#define PWM_8BIT_CHANNEL_3 3
#define PWM_8BIT_CHANNEL_4 4
#define PWM_8BIT_CHANNEL_5 5
#define PWM_8BIT_CHANNEL_6 6
#define PWM_8BIT_CHANNEL_7 7
#define PWM_16BIT_CHANNEL_0 0 //combined 8bit channel1 and 0, CH1 is used for output pin
#define PWM_16BIT_CHANNEL_1 1 //combined 8bit channel3 and 2, CH3 is used for output pin
#define PWM_16BIT_CHANNEL_2 2 //combined 8bit channel5 and 5, CH5 is used for output pin
#define PWM_16BIT_CHANNEL_3 3 //combined 8bit channel7 and 6, CH7 is used for output pin


/*
 * we have two clock source ... A and B
 * some channels use A and some use B
 */
#define PWM_IS_USING_CLOCK_A 0
#define PWM_IS_USING_CLOCK_B 1

/*
 * are we going to use scale our clock source further or not
 */
#define PWM_IS_USING_SCALED_CLOCK 1
```

```
#define PWM_IS_NOT_USING_SCALED_CLOCK 0



/**
 * cannel: 8 channel in 8bit or 4 channel in 16 bit
 * status: enable, disable ...
 * mode: 8bit mode or 16bit
 * polarity: hight at start or low at start
 * alignment: left or center aligned
 * is_using_scaled_clock: SA or SB ...
 */
typedef struct pwm_struct
  {
    char channel;
    char status;
    char mode;
    char polarity;
    char alignment;
    char is_using_scaled_clock;
    char main_clock_divider_reg_value;
    int scaled_clock_divider_reg_value;
    unsigned int duty_reg_value;
    unsigned int period_reg_value;
  }
PWM;

/*
 * We can precalculate and preset the setting for different Freq.
 * and whenever we want just call the related index in batch to simply run that
 * settings ...
 *
 * here we can preload 50 different settings for channels.
 */
PWM pwm_batch[50];


/**
 * helps us to see which of clock A or B this givven channle is using ...
 * channel: 0-7 for 8bit and 0-3 for 16bit modes
 * mode: is 16bit or 8bit
 */
char pwm_is_using_which_clock(char channel, char mode)
{
  if (mode == PWM_8BIT_OPERATION_MODE)
    {
      if (channel == PWM_8BIT_CHANNEL_0 ||
          channel == PWM_8BIT_CHANNEL_1 ||
          channel == PWM_8BIT_CHANNEL_4 ||
          channel == PWM_8BIT_CHANNEL_5
         )
        return PWM_IS_USING_CLOCK_A;
      else
        return PWM_IS_USING_CLOCK_B;
    }
  else // mode == PWM_16BIT_OPERATION_MODE
    {
      if (channel == PWM_8BIT_CHANNEL_0 ||
          channel == PWM_8BIT_CHANNEL_2)
        return PWM_IS_USING_CLOCK_A;
      else
        return PWM_IS_USING_CLOCK_B;
    }
}



/*
 * get a port and set the coresponding port number to 1 or 0
 */
void helper__bit_setter(volatile unsigned char * PORT, char bit_number, char bit_value)
{
  if (bit_value == 1)
    {
      *PORT |= 1 << bit_number;
    }
  else if (bit_value == 0)
    {
      *PORT &= ~(1 << bit_number);
    }
}


/**
 * set the priod and duty cycel registers for a channel
 */
void pwm_set_period_and_duty_registers(char channel, char mode, unsigned int period_value,
unsigned int duty_value )
{
  if (mode == PWM_8BIT_OPERATION_MODE)
    {
      char new_period_value = (char) period_value;
      char new_duty_value = (char) duty_value;

      switch (channel)
        {
        case 0:
          {
            PWMPER0 = new_period_value;
            PWMDTY0 = new_duty_value;
          }
        case 1:
          {
            PWMPER1 = new_period_value;
            PWMDTY1 = new_duty_value;
          }
        case 2:
          {
            PWMPER2 = new_period_value;
            PWMDTY2 = new_duty_value;
          }
        case 3:
          {
            PWMPER3 = new_period_value;
            PWMDTY3 = new_duty_value;
          }
        case 4:
          {
            PWMPER4 = new_period_value;
            PWMDTY4 = new_duty_value;
          }
        case 5:
          {
            PWMPER5 = new_period_value;
            PWMDTY5 = new_duty_value;
          }
        case 6:
          {
            PWMPER6 = new_period_value;
            PWMDTY6 = new_duty_value;
          }
        case 7:
          {
            PWMPER7 = new_period_value;
            PWMDTY7 = new_duty_value;
          }
        }
    }
  else
    {
      switch (channel)
        {
        case 0:
          {
            PWMPER01_16BIT = period_value;
            PWMDTY01_16BIT = duty_value;
          }
        case 1:
          {

            PWMPER23_16BIT = period_value;
            PWMDTY23_16BIT = duty_value;
          }
        case 2:
          {
            PWMPER45_16BIT = period_value;
            PWMDTY45_16BIT = duty_value;
          }
        case 3:
          {

            PWMPER67_16BIT = period_value;
            PWMDTY67_16BIT = duty_value;
          }
        }
    }

}




/**
 * PWME - PWM Enable Register
 * To enable PWM on channel0 and 1 you should write PWME = 0x03 ...
 * Once concatenated mode is enabled (CONxx bits set in PWMCTL register) then
 * enabling/disabling the corresponding 16-bit PWM channel is controlled by
 * the low order PWMEx bit.
 *
 * status is enabled or disabled ...
 * mode is 16bit or 8bit
 */
void pwm_set_status(char pwm_channel_number, char mode, char status)
{

  if (mode == PWM_8BIT_OPERATION_MODE)
    {
      helper__bit_setter(&PWME, pwm_channel_number, status);
    }
  else
    {
      helper__bit_setter(&PWME, pwm_channel_number+pwm_channel_number+1, status);
    }
}



/*
 * if we like to change the settings for a PWM we have to reset the
 * configurations first by writing somthing to PWM counter to reset
 * the counter ...
 */
void pwm_reset_settings(char channel, char mode)
{
  if (mode == PWM_8BIT_OPERATION_MODE)
    {
      switch (channel)
```

```
        {
        case 0:
          {
            PWMCNT0 = 1;
            pwm_set_status(channel,mode, DISABLE);
            PWMCNT0 = 1;
          }
        case 1:
          {
            PWMCNT1 = 1;
            pwm_set_status(channel,mode, DISABLE);
            PWMCNT1 = 1;
          }
        case 2:
          {
            PWMCNT2 = 1;
            pwm_set_status(channel,mode, DISABLE);
            PWMCNT2 = 1;
          }
        case 3:
          {
            PWMCNT3 = 1;
            pwm_set_status(channel,mode, DISABLE);
            PWMCNT3 = 1;
          }
        case 4:
          {
            PWMCNT4 = 1;
            pwm_set_status(channel,mode, DISABLE);
            PWMCNT4 = 1;
          }
        case 5:
          {
            PWMCNT5 = 1;
            pwm_set_status(channel,mode, DISABLE);
            PWMCNT5 = 1;
          }
        case 6:
          {
            PWMCNT6 = 1;
            pwm_set_status(channel,mode, DISABLE);
            PWMCNT6 = 1;
          }
        case 7:
          {
            PWMCNT7 = 1;
            pwm_set_status(channel,mode, DISABLE);
            PWMCNT7 = 1;
          }
        }
    }
  else
    {
      switch (channel)
        {
        case 0:
          {
            PWMCNT01_16BIT = 1;
            pwm_set_status(channel,mode, DISABLE);
            PWMCNT01_16BIT = 1;
          }
        case 1:
          {
            PWMCNT23_16BIT = 1;
            pwm_set_status(channel,mode, DISABLE);
            PWMCNT23_16BIT = 1;
          }
        case 2:
          {
            PWMCNT45_16BIT = 1;
            pwm_set_status(channel,mode, DISABLE);
            PWMCNT45_16BIT = 1;
          }
        case 3:
          {
            PWMCNT67_16BIT = 1;
            pwm_set_status(channel,mode, DISABLE);
            PWMCNT67_16BIT = 1;
          }
        }
    }

}






/**
 * PWMPOL - PWM Polarity Register
 * If the polarity bit is one, the PWM channel output is high at the beginning
 * of the cycle and then goes low when the duty count is reached. Conversely,
 * if the polarity bit is zero, the output starts low and then goes high when
 * the duty count is reached.
 */
void pwm_set_polarity(char pwm_channel_number, char mode,char polarity)
{
  if (mode == PWM_8BIT_OPERATION_MODE)
    {
      helper__bit_setter(&PWMPOL, pwm_channel_number, polarity);
    }
  else
    {
      helper__bit_setter(&PWMPOL, pwm_channel_number+pwm_channel_number+1, polarity);
    }
}




/**
 * PWMCAE  - PWM Center Align Enable Register
 * If the CAEx bit is set to a one, the corresponding PWM output will be center
 * aligned. If the CAEx bit is cleared, the corresponding PWM output will be left
 * aligned
 */
void pwm_set_alignment(char pwm_channel_number, char mode, char alignment)
{
  if (mode == PWM_8BIT_OPERATION_MODE)
    {
      helper__bit_setter(&PWMCAE, pwm_channel_number, alignment);
    }
  else
    {
      helper__bit_setter(&PWMCAE, pwm_channel_number+pwm_channel_number+1, alignment);
    }
}

/*
 * helps us to set the 16bit or 8bit mode of opration for PWM channel.
 */
void pwm_set_operation_mode_flags(char pwm_channel_number, char mode)
{
  if (mode == PWM_16BIT_OPERATION_MODE)
    {
      // why 4??  just an ofset to reach CONxx bits
      helper__bit_setter(&PWMCTL, (pwm_channel_number+4), ENABLE);
    }
  else if (mode == PWM_8BIT_OPERATION_MODE)
    {
      helper__bit_setter(&PWMCTL, ((pwm_channel_number/2) +4), DISABLE);
    }
}



/*
 * set the main clock settings for a channel
 */
void pwm_set_main_clock(char channel, char mode, char is_using_scaled)
{
  if (mode == PWM_8BIT_OPERATION_MODE)
    {
      if (is_using_scaled)
        {
          helper__bit_setter(&PWMCLK,channel, PWM_IS_USING_SCALED_CLOCK);
        }
      else
        {
          helper__bit_setter(&PWMCLK,channel, PWM_IS_NOT_USING_SCALED_CLOCK);
        }
    }
  else // mode == PWM_16BIT_OPERATION_MODE
    {
      char bit = channel+channel+1; //to reach proper bit

      if (is_using_scaled)
        {
          helper__bit_setter(&PWMCLK,bit, PWM_IS_USING_SCALED_CLOCK);
        }
      else
        {
          helper__bit_setter(&PWMCLK,bit, PWM_IS_NOT_USING_SCALED_CLOCK);
        }
    }
}

/*
 * set the main clock divider which can be 0-7
 *
 * 2^0 or 2^1 ... 2^7
 */
void pwm_set_main_clock_divider(char channel, char mode, char divider)
{
  //Selects prescale clock source for clocks A and B independently: XXX  PCKB2 PCKB1 PCKB0 XXX
PCKA2 PCKA1 PCKA0
  if (pwm_is_using_which_clock(channel, mode) == PWM_IS_USING_CLOCK_A)
    {
      PWMPRCLK &= 0xf0; //reset the pins for A
      PWMPRCLK |=  divider;
    }
  else
    {
      PWMPRCLK &= 0x0f; //reset the pins for B
      PWMPRCLK |=  divider << 4;
    }
}



/*
 * set the scaled clock divider register, IF we are using scaled mode
 */
void pwm_set_scale_clock_divider(char channel, char mode, char is_using_scaled, char divider )
{
  if (is_using_scaled)
```

```
    {
      if (pwm_is_using_which_clock(channel, mode) == PWM_IS_USING_CLOCK_A)
        {
          PWMSCLA = 0;
          PWMSCLA |= divider;
        }
      else
        {
          PWMSCLB = 0;
          PWMSCLB |= divider;
        }
    }
}



/**
 * pc can send commands to micro controller. Format will be
 * pc:command_code&arg1&arg2&
 * comand code can be from 1 to (2^16  -1)
 * args can be from 1 to (2^16  -1)
 * sample:
 * "pc:125&0&10"
 */
void process_the_retrieved_command_from_pc(int command,  unsigned int arg1, unsigned int arg2)
{
  // these series of commands will control the PWM section
  if (command > 100 && command < 200)
    {
      /*
       * command for pwm channel///////////////////////////////////////////////
       *
       * examples:
       * pc:111&0&1& means use PWM 8bit on channel 1
       * pc:111&1&0& means use PWM 16bit on channel 0
       */
      if (command == 111)
        {
          //16bit or 8bit
          if (arg1 == 0)
            pwm_mode = PWM_8BIT_OPERATION_MODE;
          else if (arg1 == 1)
            pwm_mode = PWM_16BIT_OPERATION_MODE;

          pwm_channel_number = (char) arg2;
        }


      /*
       * set the boolean flags ////////////////////////
       *
       * exampels:
       * pc:122&0&1& means use polarity high at start
       * pc:122&1&0& means use left aligned for the pulse
       */
      else if (command == 122)
        {
          if (arg1 == 0) //polarity
            {
              if (arg2 == 0)
                pwm_polarity = PWM_POLARITY_IS_LOW_AT_BEGINNING_OF_THE_PULSE;
              else if (arg2 == 1)
                pwm_polarity = PWM_POLARITY_IS_HIGH_AT_BEGINNING_OF_THE_PULSE;
            }
          else if (arg1 == 1) //align
            {
              if (arg2 == 0)
                pwm_alignment = PWM_ALIGNMENT_LEFT_ALIGN;
              else if (arg2 == 1)
```

```
                pwm_alignment = PWM_ALIGNMENT_CENTER_ALIGN;
              }
            else if (arg1 == 2) //is using SA clock
              {
                if (arg2 == 0)
                  pwm_is_using_scaled_clock = PWM_IS_NOT_USING_SCALED_CLOCK;
                else if (arg2 == 1)
                  pwm_is_using_scaled_clock = PWM_IS_USING_SCALED_CLOCK;
              }
          }

        else if (command == 133) ///values //////////////////////////////////
          {
            if (arg1 == 0) //main divider value
              pwm_main_clock_divider = arg2;
            else if (arg1 == 1) //scaled divider value
              pwm_scale_clock_divider = arg2;
            else if (arg1 == 2) //period value
              pwm_period_reg_value = arg2;
            else if (arg1 == 3) //duty value
              pwm_duty_reg_value = arg2;
          }
        else if (command == 144) ///UPDATE SETTINGS//////////////////////////////
          {
            if (arg1 == 1 && arg2 == 1)
              {
                pwm_reset_settings(pwm_channel_number, pwm_mode);

                pwm_set_polarity(pwm_channel_number, pwm_mode, pwm_polarity);
                pwm_set_alignment(pwm_channel_number, pwm_mode, pwm_alignment);
                pwm_set_operation_mode_flags(pwm_channel_number, pwm_mode);
                pwm_set_main_clock(pwm_channel_number, pwm_mode, pwm_is_using_scaled_clock);
                pwm_set_main_clock_divider(pwm_channel_number, pwm_mode, pwm_main_clock_divider);
                pwm_set_scale_clock_divider(pwm_channel_number, pwm_mode, pwm_is_using_scaled_clock,
pwm_scale_clock_divider);
                pwm_set_period_and_duty_registers(pwm_channel_number, pwm_mode,
pwm_period_reg_value, pwm_duty_reg_value);

                pwm_set_status(pwm_channel_number, pwm_mode, ENABLE);


              }
          }
        else if (command == 155) ///UPDATE the BATCH PWM//////////////////////////////////
          {
            if (arg1 == 1)//update batch array
              {
                pwm_batch[arg2].alignment = pwm_alignment;
                pwm_batch[arg2].channel = pwm_channel_number;
                pwm_batch[arg2].duty_reg_value = pwm_duty_reg_value;
                pwm_batch[arg2].is_using_scaled_clock = pwm_is_using_scaled_clock;
                pwm_batch[arg2].main_clock_divider_reg_value = pwm_main_clock_divider;
                pwm_batch[arg2].mode = pwm_mode;
                pwm_batch[arg2].period_reg_value = pwm_period_reg_value;
                pwm_batch[arg2].polarity = pwm_polarity;
                pwm_batch[arg2].scaled_clock_divider_reg_value = pwm_scale_clock_divider;


              }
            else if (arg1 == 2) //load a pwm settings from batch array
              {

                pwm_alignment =  pwm_batch[arg2].alignment ;
                pwm_channel_number =  pwm_batch[arg2].channel;
                pwm_duty_reg_value = pwm_batch[arg2].duty_reg_value ;
                pwm_is_using_scaled_clock =  pwm_batch[arg2].is_using_scaled_clock ;
                pwm_main_clock_divider =  pwm_batch[arg2].main_clock_divider_reg_value ;
                pwm_mode =  pwm_batch[arg2].mode ;
                pwm_period_reg_value = pwm_batch[arg2].period_reg_value ;
                pwm_polarity = pwm_batch[arg2].polarity  ;
                pwm_scale_clock_divider =  pwm_batch[arg2].scaled_clock_divider_reg_value ;


                pwm_reset_settings(pwm_channel_number, pwm_mode);
                pwm_set_polarity(pwm_channel_number, pwm_mode, pwm_polarity);
                pwm_set_alignment(pwm_channel_number, pwm_mode, pwm_alignment);
                pwm_set_operation_mode_flags(pwm_channel_number, pwm_mode);
                pwm_set_main_clock(pwm_channel_number, pwm_mode, pwm_is_using_scaled_clock);
                pwm_set_main_clock_divider(pwm_channel_number, pwm_mode, pwm_main_clock_divider);
                pwm_set_scale_clock_divider(pwm_channel_number, pwm_mode, pwm_is_using_scaled_clock, 
  pwm_scale_clock_divider);
                pwm_set_period_and_duty_registers(pwm_channel_number, pwm_mode, 
  pwm_period_reg_value, pwm_duty_reg_value);
                pwm_set_status(pwm_channel_number, pwm_mode, ENABLE);
               }

          }
      }

  }
```

COLLECTION AND VALIDATION OF TEST ANSWERS

As briefly mentioned in the sensory system, NGRID desktop application provides a secure WebSocket infrastructure to interconnect all subsystems and allow two-way communication as well as aggregation of test answers and events. Below sample code shown how messages are collected, validated against a set of predefined commands and stored in logs:

```
namespace NGRIDDesktopApp
{
    public partial class NGRIDDesktopAppMainForm : Form
    {
        private const int WM_SYSCOMMAND = 0x0112;
        private const int SC_MINIMIZE = 0xf020;
        private const int SC_MOVE = 0xF010;

        //...
        public Process httpd_process;
        public NGRIDSpeechToTextEngine NgridSpeechToTextEngine;
        public WebSocketServer websocket_app_server = new WebSocketServer();

        public NGRIDDesktopAppMainForm()
        {
            InitializeComponent();
            NavidInit();
        }


        [DllImport("winmm.dll", EntryPoint = "mciSendStringA", ExactSpelling = true, CharSet = 
CharSet.Ansi,
            SetLastError = true)]
        private static extern int wave_record(string lpstrCommand, string lpstrReturnString, int
uReturnLength,
            int hwndCallback);


        [DllImport("user32.dll")]
        private static extern void ClipCursor(ref Rectangle rect);


        public void NavidInit()
        {
            keyboardHook.KeyPressed += keypPressed_CtrlAlrtShiftF11;
            keyboardHook.RegisterHotKey(
                NGRIDDesktopApp.ModifierKeys.Control | NGRIDDesktopApp.ModifierKeys.Alt |
                NGRIDDesktopApp.ModifierKeys.Shift, Keys.F11);

            //Start inetrnall HTTPD
            httpd_process = new Process();
            httpd_process.StartInfo.FileName = "NGRID" +
                "HttpdConsole.exe";
            httpd_process.StartInfo.Arguments = "-n";
            httpd_process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
            httpd_process.Start();


            //Setup the websocket app server
            if (!websocket_app_server.Setup(2012)) //Setup with listening port
            {
                return -1;
            }

            websocket_app_server.NewMessageReceived += Websocket_new_message_received;

            //Try to start the appServer
            if (!websocket_app_server.Start())
            {
                return -2;
            }
        }


        //got a new message on websocket
        public void Websocket_new_message_received(WebSocketSession session, string message)
        {
            if (
                message.ToLower().Contains("globalsetting") && message.ToLower().Contains("style")
||
                message.ToLower().Contains("objs_obj_stg3")
               )
            {
                websocket_message_handle(session, message);
                return;
            }
        }

        //how to handle a new message:
        public void websocket_message_handle(WebSocketSession session, string message)
        {
            var command_stream = new MemoryStream(Encoding.UTF8.GetBytes(message));
            var command_js = new DataContractJsonSerializer(typeof(CommandObject));
            var objective = (CommandObject) command_js.ReadObject(command_stream);

            if (objective.command.Contains("store"))
            {
                if (objective.related_class.Contains("log"))
                {
                    try
                    {
                        var log_stream = new
MemoryStream(Encoding.UTF8.GetBytes(objective.payload));
                        var log_js = new DataContractJsonSerializer(typeof(LogsObject));
                        var log_object = (LogsObject) log_js.ReadObject(log_stream); //object from
JSON
                        //do DAO
                        db_run_statment(log_object.sqlite_create_insert_statment()); //store to db
                    }
                    catch (Exception e)
                    {
                        Console.WriteLine(exception_string + e.Message);
                    }
                }
                //else if ... many more cases ... we didn'y show them here as there many ... over
300 lines of code
            }
        //else if ... many more cases ... we didn'y show them here as there many ... over 300
lines of code

        }

        //send/broadcast a message down the webscoket
        public void websocketSend(string message)
        {
            foreach (var session in websocket_app_server.GetAllSessions()) session.Send(message);
        }


        protected override void WndProc(ref Message m)
        {
            if (m.Msg == WM_SYSCOMMAND)
            {
                var command = m.WParam.ToInt32() & 0xfff0;

                if (m.WParam.ToInt32() == SC_MINIMIZE)
                {
                    m.Result = IntPtr.Zero;
                    Size = new Size(400, 400);
                    return;
                }

                if (!allow_window_move && command == SC_MOVE)
                {
                    return;
                }
            }

            base.WndProc(ref m);
        }

        //on closing shutdown everything
        private void onFormClosing(object sender, FormClosingEventArgs e)
        {
            try
            {
                httpd_process.Kill();
                Process.Start("taskkill", "/f /im HttpdConsole.exe");

                websocket_app_server.Stop();
                websocket_app_server.Dispose();

                stop_wave_recording = true;
                thread_for_wave_saving.Shutdown();
            }
            catch (Exception e)
            {
                Console.WriteLine(exception_string + e.Message);
            }
        }
        //restart internal HTTPD
        private void onButtonRestartHTTPD_click(object sender, EventArgs e)
        {
            httpd_process.Kill();
            Process.Start("taskkill", "/f /im HttpdConsole.exe");

            httpd_process = new Process();
            httpd_process.StartInfo.FileName = "HttpdConsole.exe";
            httpd_process.StartInfo.Arguments = "-n";
            httpd_process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
            httpd_process.Start();
        }

    }
}
```

ARCHIVAL OF THE TEST ANSWERS

Test data, events and test answers need to be structurally and securely saved and archived for post-processing in NGRID datacenter. We use SQLite for this purpose. A password protected local SQLite file is used to temporarily store the test data and aggregated events as well as test answers. Once internet connection is established (i.e. device is online), the file will be uploaded securely via HTTPS to NGRID datacenter.

Storing the logs in SQLite allows for easy access and query of the test answers and events as well as native compression of the logs. In below sample code, we demonstrate how this is done:

```
namespace NGRIDDesktopApp
{
    public partial class NGRIDDesktopAppMainForm : Form
    {

        private void createNewDatabase()
        {
            if (File.Exists(filename_db))
            {
                Console.WriteLine("** DB exist: " + filename_db);
                return;
            }
            SQLiteConnection.CreateFileWithPassword(filename_db, this.fetched_temp_password);
        }

        // Creates a connection with our database file.
        private void connectToDatabase()
        {
            db_connection = new SQLiteConnection("Data Source=" + filename_db +
";Version=3;Password=" + this.fetched_temp_password);
            db_connection.Open();
        }

        //e.g. to insert, update or drop etc
        private void db_run_statment(string sql)
        {
            var command = new SQLiteCommand(sql, db_connection);
            command.ExecuteNonQuery();
        }


        //how to handle a new message:
        public void websocket_message_handle(WebSocketSession session, string message)
        {
            var command_stream = new MemoryStream(Encoding.UTF8.GetBytes(message));
            var command_js = new DataContractJsonSerializer(typeof(CommandObject));
            var objective = (CommandObject) command_js.ReadObject(command_stream);

            if (objective.command.Contains("store"))
            {
                if (objective.related_class.Contains("log"))
                {
                    try
                    {
                        var log_stream = new
MemoryStream(Encoding.UTF8.GetBytes(objective.payload));
                        var log_js = new DataContractJsonSerializer(typeof(LogsObject));
                        var log_object = (LogsObject) log_js.ReadObject(log_stream); //object from
JSON
                        //do DAO
                        db_run_statment(log_object.sqlite_create_insert_statment()); //store to db
                    }
                    catch (Exception e)
                    {
                        Console.WriteLine(exception_string + e.Message);
                    }
                }
                //else if ... many more cases ... we didn'y show them here as there many ... over
300 lines of code
            }
        //else if ... many more cases ... we didn'y show them here as there many ... over 300
lines of code
        }

    }


    //Log objects to persist to sqlite
    [DataContract]
    internal class LogsObject
    {
        [DataMember] public string action;
        [DataMember] public int id;
        [DataMember] public string message;
        [DataMember] public string participant_id;
        [DataMember] public string test_type;
        [DataMember] public string timestamp;
        [DataMember] public string trial_id;

        public static string GetMemberName<T, TValue>(Expression<Func<T, TValue>> memberAccess)
        {
            return ((MemberExpression) memberAccess.Body).Member.Name;
        }

        public static string db_header_maker(string header, bool start = false)
        {
            if (!start) return ", [" + header + "]";
            return " [" + header + "]";
        }
        public static string db_value_maker(string val, bool start = false)
        {
            if (!start) return ", '" + val + "'";
            return " '" + val + "'";
        }


        public string sqlite_create_insert_statment()
        {
            var result = "";

            //GetMemberName((LogsObject c) => c.id);
            result = "INSERT INTO [Logs] ("
                     + db_header_maker(GetMemberName((LogsObject c) => c.id), true)
                     + db_header_maker(GetMemberName((LogsObject c) => c.timestamp))
                     + db_header_maker(GetMemberName((LogsObject c) => c.trial_id))
                     + db_header_maker(GetMemberName((LogsObject c) => c.participant_id))
                     + db_header_maker(GetMemberName((LogsObject c) => c.test_type))
                     + db_header_maker(GetMemberName((LogsObject c) => c.action))
                     + db_header_maker(GetMemberName((LogsObject c) => c.message))
                     + " ) VALUES ("
                     + " NULL"
                     + db_value_maker(timestamp)
                     + db_value_maker(trial_id)
                     + db_value_maker(participant_id)
                     + db_value_maker(test_type)
                     + db_value_maker(action)
                     + db_value_maker(message)
                     + ");";


            return result;
        }

            //DROP TABLE [Logs];
                //
                //CREATE TABLE [Logs] ( [id] INTEGER PRIMARY KEY
                //                        , [timestamp] datetime NOT NULL
                //                        , [trial_id] nvarchar(15) NOT NULL
                //                        , [participant_id] nvarchar(15) NOT NULL
                //                        , [test_type] nvarchar(100) NOT NULL
                //                        , [action] nvarchar(100) NOT NULL
                //                        , [message] ntext NOT NULL
                //                        , CONSTRAINT [PK_Logs] );
                //
                //sqlite_create_insert_statment();

        public string sqlite_create_update_statment()
        {
            var result = "";

            //GetMemberName((LogsObject c) => c.id);
            result = "UPDATE [Logs] "
                     + "SET [timestamp] = '" + GetMemberName((LogsObject c) => c.timestamp) + "'"
                     + "    ,[trial_id] = '" + GetMemberName((LogsObject c) => c.trial_id) + "'"
                     + "     ,[participant_id] = '" + GetMemberName((LogsObject c) =>
c.participant_id) + "'"
                     + "     ,[test_type] = '" + GetMemberName((LogsObject c) => c.test_type) + "'"
                     + "     ,[action] = '" + GetMemberName((LogsObject c) => c.action) + "'"
                     + "     ,[message] = '" + GetMemberName((LogsObject c) => c.message) + "'"
                     + " WHERE id = " + GetMemberName((LogsObject c) => c.id) + ";";

            return result;
        }
    }
}
```

APPENDIX B – NGRID DATABASE

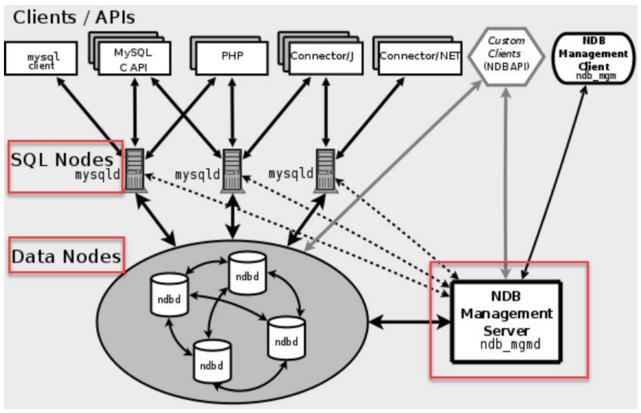
NGRID platform provides a highly available relational database within NGRID datacenter that keeps all the test SVG data and credentials. MySQL NDB Cluster is a high-availability and high-redundancy version of MySQL adapted for the distributed computing environment. We use NDB storage engine and ndb_mgm command line interface (CLI) to manage the nodes ndbd, ndb_mgmd and mysqld that are explained below.

MySQL NDB Cluster is a technology that enables clustering of in-memory databases in a shared-nothing fashion. The shared-nothing architecture enables the systems to work with very inexpensive hardware (Wikipedia is using MySQL Cluster to serve massive amount of data in a distributed fashion). The Cluster does not have any single point of failure. Each node has its own independent CPU, memory and disk.

NDB Cluster integrates the standard MySQL server with an in-memory clustered storage engine called Network DataBase (NDB). MySQL NDB Cluster refers to the cluster of MySQL servers which use the NDB storage engine. The cluster has a series of computer nodes which can be:

- One or more SQL API Server nodes (mysqld) to access the NDB data nodes.
- One or more NDB Data nodes (ndbd) that store the database data with high-availability and high-redundancy.
- One or more management server nodes (ndb_mgmd) to manage the overall cluster.

The relationship of these components in an NDB Cluster is shown here:



We also developed various SQL stored procedures to embed frequently used database queries. Below is sample stored procedure which copy SVGs of one NGRID test to another.

```
BEGIN
        DECLARE bDone INT DEFAULT 0;
        DECLARE XMLCONTENT text;
        DECLARE SEQ INT(11);
        DECLARE SVGID INT(11);
        DECLARE TESTID INT(11);
        DECLARE curs CURSOR FOR SELECT * FROM SVG
            WHERE SVG.TESTID = _OLD_TEST_ID ORDER BY SEQ;
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET bDone = 1;
    DROP TEMPORARY TABLE IF EXISTS `SVG_TEMP`;
    CREATE  TEMPORARY  TABLE  IF  NOT  EXISTS  `SVG_TEMP`  (`SVGID`  int(11)  NOT  NULL
    AUTO_INCREMENT, `XMLCONTENT` text COLLATE utf8_unicode_ci NOT NULL, `TESTID` int(11)
    NOT NULL, `SEQ` int(11) NOT NULL, PRIMARY KEY (`SVGID`), KEY `TESTID` (`TESTID`)  )
    AUTO_INCREMENT = 0 DEFAULT CHARSET = utf8 COLLATE = utf8_unicode_ci;
    OPEN curs;
        REPEAT
                FETCH curs INTO SVGID, XMLCONTENT, TESTID, SEQ;
                INSERT INTO `SVG` VALUES ( NULL, XMLCONTENT, _NEW_TEST_ID, SEQ ) ;
        UNTIL bDone
        END REPEAT;
    CLOSE curs;
    SELECT * FROM `svg_temp`;
END
```

We also use a cluster of Apache Cassandra NoSQL paired with a cluster of Kafka to perform stream processing on test answers. The goal here is to start analyzing and processing the test answers as they arrive to NGRID datacenter to generate heatmap. The Apache Cassandra is a NoSQL database. It does not have the overhead of the relational databases. It is designed to be linearly scalable, highly available and performant NoSQL database. We use Cassandra to make sure test results can be accessed and stored rapidly.

Apache Kafka is a streaming platform that can Publish and Subscribe to a streams of records, similar to a message queues or enterprise transactional messaging systems. It can store streams of records in high available, fault-tolerant and durable cluster. It also provides APIs to help process the streams of records as they occur. Kafka allows for building real-time streaming data pipelines that reliably get the data between systems and applications. Moreover it appropriately transforms and react to the streams of data. It is noteworthy to mention that the Kafka cluster stores streams of records in categories called Topics. Each record consists of a key, a value, and a timestamp. Below is a sample code that shows how we receive and send batch work messages to and from our Kafka cluster.

```java
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.TimeUnit;

import org.apache.kafka.clients.consumer.ConsumerRecord;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.kafka.core.KafkaTemplate;

    //...
    @Autowired
    private KafkaTemplate<String, String> ngridKafkaTemplate;
    private final CountDownLatch ngridKafkaLatch = new CountDownLatch(1);
    private static final int NGRID_KAFKA_TIMEOUT = 60;
    private static final String NGRID_KAFKA_TOPIC = "ngrid_batch_topic";

    public void NgridKafkaSend(String msg) {
        this.ngridKafkaTemplate.send(NGRID_KAFKA_TOPIC, msg);
        try
        {
            ngridKafkaLatch.await(NGRID_KAFKA_TIMEOUT, TimeUnit.SECONDS);
            logger.info("Sent message " + msg + " to Kafka topic " + NGRID_KAFKA_TOPIC );
        }
        catch (InterruptedException e)
        {
            e.printStackTrace();
        }
    }

    @KafkaListener(topics = NGRID_KAFKA_TOPIC)
    public void NgridKafkaListener(ConsumerRecord<?, ?> r) throws Exception {
        logger.info("Got message " + r.toString() + " from Kafka topic " + NGRID_KAFKA_TOPIC );
        ngridKafkaLatch.countDown();

        //... process the message  and do the additional work
    }
```

APPENDIX C – NGRID SVGEDITOR

NGRID platform aims to provide an easy way to create and modify NGRID VD Tests. The tests use Standard Vector Graphics (SVG). Hence, NGRID provides an easy to use SVGEditor that can create and modify test graphics. MIT licensed online open source SVGEditor was heavily modified to add functionalities to allow multiple SVGs to be stored centrally at NGRID datacenter. NGRID SVGEditor allows loading and modifying NGRID VD Tests through an online interface at http://ngrid.website. Below we demonstrate a sample code that is responsible for loading and uploading edited SVG content to NGRID Server at NGRID Datacenter.

```
'use strict';

/**
 * Global is svgeditor
 *
 * @ngdoc function
 * @name NGRID Server - amdappApp.controller - SVGEditor Controller of the NGRID
 *       Server - amdappApp.controller - SVGEditor
 */
angular.module('amdappApp')
  .controller('svgEditorController', function ($http, $log, $state, $stateParams, AlertService,
$sanitize, $sce, svgs) {
    var svgEditorCtrl = this;

    /**
         * Controller-specific objects attached to svgEditorCtrl:
         *
         * svgs: SVGS array deleteHistory: Stringified version of object for history
         * undoing toDelete: svgEditor objects to be deleted when resequenced.
         * tempIndexes: Indexes of temporary elements. Can be removed without
         * querying the database. remove function checks these indexes before
         * pushing an item into the toDelete stack testName: name corresponding to
         * the test where the SVGs are being added
         */
    svgEditorCtrl.svgs = svgs;
    svgEditorCtrl.deleteHistory = [];
    svgEditorCtrl.toDelete = [];
    svgEditorCtrl.tempIndexes = [];
    svgEditorCtrl.testName = $stateParams.testName;
    svgEditorCtrl.testId = parseInt($sanitize($stateParams.testId));

    var deletedTemps = [];

    if (!$stateParams.testId || !$stateParams.testName){
      $state.go('adminLoggedIn.main');
      return;
    }

    function sortBySeq(svg){
      return svg.seq;
    }

    function defaultSVG(){
      return {svgId: 0, xmlContent : svgEditor.getContent(), testId: svgEditorCtrl.testId, seq:
1};
    }

    function fixIndexes(){
      for(var i = 0; i < svgEditorCtrl.svgs.length; i++){
        svgEditorCtrl.svgs[i].seq = i+1;
      }
    }


    /**
         * Initialize function, will only run on document ready Loads the first SVG
         * defined by the seq variable.
         */
    svgEditorCtrl.init = function()
    {
      svgEditorCtrl.makeAlert = new makeAlert();
      if (svgs.length > 0)
      {
        svgEditorCtrl.svgs = _.sortBy(svgEditorCtrl.svgs, sortBySeq);
        svgEditorCtrl.currSVG = svgs[0];
        svgEditorCtrl.currIndex = 0;
        fixIndexes();
        loadCurrent();
        clearHist();
        return;
      };
      // on empty:
      svgEditorCtrl.svgs[0] = defaultSVG();
      svgEditorCtrl.currSVG = svgEditorCtrl.svgs[0];
      svgEditorCtrl.currIndex = 0;
      loadCurrent();
      clearHist();
    };

    // listen for page load
    // this ensures no svgEditor global functions will be instantiated before
    // in case the controller loads first.
    // function needs a timeout delay to clear the history properly.
    angular.element(document).ready(setTimeout(svgEditorCtrl.init, 1000));

    /**
         * Function Loads the current SVG in the frame.
         */
    function loadCurrent(){
      svgEditor.loadFromString(svgEditorCtrl.currSVG.xmlContent);
      $('#tool_navid_frame_number').val(svgEditorCtrl.currIndex);
    }

    function clearHist(){
      svgEditor.clearHistory();
    }


    /**
         * Function scrolls to next frame.
         */
    svgEditorCtrl.nextFrame = function(){
      if (svgEditorCtrl.currIndex >= (svgEditorCtrl.svgs.length-1)){
        return;
      }
      svgEditorCtrl.saveCurrent();
      svgEditorCtrl.currIndex++;
      svgEditorCtrl.currSVG = svgEditorCtrl.svgs[svgEditorCtrl.currIndex];
      loadCurrent();
      clearHist();
    };


    /**
         * Function to scroll to previous frame
         */
    svgEditorCtrl.prevFrame = function(){
      if (svgEditorCtrl.currIndex <= 0){
        return;
      }
      svgEditorCtrl.saveCurrent();
      svgEditorCtrl.currIndex--;
      svgEditorCtrl.currSVG = svgEditorCtrl.svgs[svgEditorCtrl.currIndex];
      loadCurrent();
      clearHist();
    };

    svgEditorCtrl.saveCurrent = function(){
      svgEditorCtrl.currSVG.xmlContent = svgEditor.getContent();
      svgEditorCtrl.svgs[svgEditorCtrl.currIndex] = svgEditorCtrl.currSVG;
    };


    /**
     * Get SVG IDs for a given test
     */
    function getSVGIds() {
      return $http.get('/app/getSVGIds/' + svgEditorCtrl.testId)
    };

    /*
     * Add/save multiple edited SVGs back to server
     */
    function postMultiple(svgs){

      var req =
      {
        method: 'POST',
        url: '/app/addMultiSVGs',
        headers:
        {
          'Content-Type': 'application/json',
          'Content-Encoding': 'gzip'
        },
        data: pako.gzip(JSON.stringify(svgs)),
        transformRequest: []
      };
      return $http(req);
    }
    // Clear all history.
    svgEditorCtrl.saveAll = function(){
      svgEditorCtrl.deleted = 0;
      svgEditorCtrl.added = 0;
      svgEditorCtrl.makeAlert.confirm("Are you sure you want to save? All changes are final!",
function(ok){
        if(ok){
          svgEditorCtrl.saveCurrent();
          svgEditorCtrl.tempIndexes = [];
          deletedTemps = [];
          // If there is deletions to be made, delete the current Frames
          if(svgEditorCtrl.toDelete.length > 0){
            $http.post('/app/deleteMultiSVGs', svgEditorCtrl.toDelete).then(function(response){
              svgEditorCtrl.deleted = response.data.deleted;
              svgEditorCtrl.toDelete = [];
              return postMultiple(svgEditorCtrl.svgs)
            }).then(function(response){
              svgEditorCtrl.added = response.data.added;
              return getSVGIds()
            }).then(function(response){
              for(var i =0; i< svgEditorCtrl.svgs.length; i++){
                svgEditorCtrl.svgs[i].svgId = response.data.ids[i]
              }
              loadCurrent();
              svgEditorCtrl.makeAlert.alert("Success!\n" + svgEditorCtrl.added + " Added/Updated
and " + svgEditorCtrl.deleted +" deleted!")
            })
              .catch(function(err){
                svgEditorCtrl.makeAlert.alert("Error: "+ err.data.message)
              })
          }
          // If there are no Deletions
          else{
            postMultiple(svgEditorCtrl.svgs).then(function(response){
              svgEditorCtrl.added = response.data.added;

              return getSVGIds()
            }).then(function(response){
              for(var i =0; i< svgEditorCtrl.svgs.length; i++){
                svgEditorCtrl.svgs[i].svgId = response.data.ids[i]
              }
              svgEditorCtrl.makeAlert.alert("Success!\n" + response.data.ids.length + "
Added/Updated and " + svgEditorCtrl.deleted +" deleted!")
            }).catch(function(err){
              svgEditorCtrl.makeAlert.alert("Error: "+ err.data.message)
            })
          }
        }
      })
    };


    /**
         * Function copies the frame at the index, and inserts it at the current
         * sequence.
         *
         * Inserts number of elements == iterations of svg[frameIndex]. Essentially
         * clones the current element JSON.parse and Stringify ensure no element is
         * being copied by reference. Insertion running in linear time
         */
    function insertFrames(frameIndex, iterations){
      var copyElem;

      for (var i = 1; i<=iterations; i++){
        copyElem = JSON.parse(JSON.stringify(svgEditorCtrl.svgs[frameIndex]));
        copyElem.seq += i;
        // Set Copy element ID to 0, to not conflict with the current REST calls
        copyElem.svgId = 0;
        svgEditorCtrl.svgs.splice(frameIndex+i, 0, copyElem);
        svgEditorCtrl.tempIndexes.push(frameIndex+i);
      }

      for(var j=frameIndex+iterations+1; j < svgEditorCtrl.svgs.length; j++){
        svgEditorCtrl.svgs[j].seq++;
      }

      // Points current element to last
      svgEditorCtrl.currIndex = frameIndex+iterations;
      svgEditorCtrl.currSVG = svgEditorCtrl.svgs[frameIndex+iterations];
      loadCurrent();
      clearHist();
    }

    svgEditorCtrl.insertSingle = function(){
      svgEditorCtrl.saveCurrent();
      insertFrames(svgEditorCtrl.currIndex, 1);
    };
```

```
/**
     * JQuery event listener for button change.
     *
     * Filters any and all symbols outside of 0 to 9
     */
$(function(){
  $('#tool_navid_frame_number').keypress(function(e){
    // has enter been pressed?
    var keycode = e.keyCode ? e.keyCode : e.which;

    if (keycode === 13 || keycode === 10){
      var reg = /[^0-9]+/;
      if(svgEditorCtrl.jumpToFrame.toString().search(reg) >= 0){
        $('#tool_navid_frame_number').val(svgEditorCtrl.currIndex);
        return;
      }


      if(svgEditorCtrl.jumpToFrame > (svgEditorCtrl.svgs.length-1))
      {
        svgEditorCtrl.jumpToFrame = svgEditorCtrl.svgs.length -1;
        svgEditorCtrl.currIndex = parseInt(svgEditorCtrl.jumpToFrame);
        svgEditorCtrl.currSVG = svgEditorCtrl.svgs[svgEditorCtrl.jumpToFrame];
        $('#tool_navid_frame_number').val(svgEditorCtrl.jumpToFrame);
        loadCurrent()
      }
      else if(svgEditorCtrl.jumpToFrame < 0){
        svgEditorCtrl.jumpToFrame = 0;
        svgEditorCtrl.currIndex = parseInt(svgEditorCtrl.jumpToFrame);
        svgEditorCtrl.currSVG = svgEditorCtrl.svgs[svgEditorCtrl.jumpToFrame];
        $('#tool_navid_frame_number').val(svgEditorCtrl.jumpToFrame);
        loadCurrent()
      }
      else {
        svgEditorCtrl.currIndex = parseInt(svgEditorCtrl.jumpToFrame);
        svgEditorCtrl.currSVG = svgEditorCtrl.svgs[svgEditorCtrl.jumpToFrame];
        loadCurrent()
      }
      svgEditorCtrl.saveCurrent();
      clearHist();
    }
  })
});


/**
     * Function on exit
     */
svgEditorCtrl.exit = function()
{
  $("link").each(function(){
    $(this).prop('disabled', false);
  });

  $state.go('adminLoggedIn.viewTests')
};

/**
     * Function to delete a specific frame at index frameIndex
     *
     * @param frameIndex
     */
function deleteFrame(frameIndex){
  svgEditorCtrl.makeAlert.confirm("Are you sure you want to delete?", function(confirmation){
    if(confirmation) {
      if (frameIndex < 0 || frameIndex >= svgEditorCtrl.svgs.length) {
        return;
      }
```

```
          var deletedElem = JSON.parse(JSON.stringify(svgEditorCtrl.svgs[frameIndex]));

          svgEditorCtrl.deleteHistory.push(deletedElem);

          // If the item is a temporary item, delete it, push it into temp
          // register deleted
          if (svgEditorCtrl.tempIndexes.indexOf(frameIndex) < 0) {
            svgEditorCtrl.toDelete.push(deletedElem);
          } else {
            deletedTemps.push(svgEditorCtrl.tempIndexes.pop());
          }

          svgEditorCtrl.svgs.splice(frameIndex, 1)

          $log.debug(svgEditorCtrl.svgs)

          if (svgEditorCtrl.svgs.length > 0) {
            for (var i = frameIndex; i < svgEditorCtrl.svgs.length; i++) {
              svgEditorCtrl.svgs[i].seq--;
            }
          }

          // Currentindex
          if(svgEditorCtrl.svgs.length === 0){
            svgEditor.clearDocument();
            svgEditorCtrl.currSVG = defaultSVG();
            svgEditorCtrl.currIndex = 0;
            svgEditorCtrl.saveCurrent();
            $log.debug(svgEditorCtrl.currSVG)
            $log.debug(frameIndex)
            loadCurrent();
            clearHist();
            return;
          }
          else if(svgEditorCtrl.svgs.length === frameIndex){
            frameIndex--;
          }
          svgEditorCtrl.currSVG = svgEditorCtrl.svgs[frameIndex];
          svgEditorCtrl.currIndex = frameIndex;
          $log.debug(svgEditorCtrl.currSVG)
          $log.debug(frameIndex)
          loadCurrent();
          clearHist();
        }

      });
    }

    /**
         * Delete the current frame
         */
    svgEditorCtrl.deleteCurrent = function(){
      deleteFrame(svgEditorCtrl.currIndex);
    };

    /**
         * Alert Dialog Taken from svg-editor.js into an object
         */
    function makeAlert() {
      $('#dialog_container').draggable({cancel: '#dialog_content, #dialog_buttons *', containment: 
'window'});
      var box = $('#dialog_box'),
        btn_holder = $('#dialog_buttons'),
        dialog_content = $('#dialog_content'),
        dbox = function(type, msg, callback, defaultVal, opts, changeCb, checkbox) {
          var ok, ctrl, chkbx;
          dialog_content.html('<p>'+msg.replace(/\n/g, '</p><p>')+'</p>')
            .toggleClass('prompt', (type == 'prompt'));
          btn_holder.empty();
          ok = $('<input type="button" value="' + svgEditor.uiStrings.common.ok +
'">').appendTo(btn_holder);

          if (type !== 'alert') {
            $('<input type="button" value="' + svgEditor.uiStrings.common.cancel + '">')
              .appendTo(btn_holder)
              .click(function() { box.hide(); if (callback) {callback(false);}});
          }

          if (type === 'prompt') {
            ctrl = $('<input type="text">').prependTo(btn_holder);
            ctrl.val(defaultVal || '');
            ctrl.bind('keydown', 'return', function() {ok.click();});
          }
          else if (type === 'select') {
            var div = $('<div style="text-align:center;">');
            ctrl = $('<select>').appendTo(div);
            if (checkbox) {
              var label = $('<label>').text(checkbox.label);
              chkbx = $('<input type="checkbox">').appendTo(label);
              chkbx.val(checkbox.value);
              if (checkbox.tooltip) {
                label.attr('title', checkbox.tooltip);
              }
              chkbx.prop('checked', !!checkbox.checked);
              div.append($('<div>').append(label));
            }
            $.each(opts || [], function (opt, val) {
              if (typeof val === 'object') {
                ctrl.append($('<option>').val(val.value).html(val.text));
              }
              else {
                ctrl.append($('<option>').html(val));
              }
            });
            dialog_content.append(div);
            if (defaultVal) {
              ctrl.val(defaultVal);
            }
            if (changeCb) {
              ctrl.bind('change', 'return', changeCb);
            }
            ctrl.bind('keydown', 'return', function() {ok.click();});
          }
          else if (type === 'process') {
            ok.hide();
          }

          box.show();

          ok.click(function() {
            box.hide();
            var resp = (type === 'prompt' || type === 'select') ? ctrl.val() : true;
            if (callback) {
              if (chkbx) {
                callback(resp, chkbx.prop('checked'));
              }
              else {
                callback(resp);
              }
            }
          }).focus();

          if (type === 'prompt' || type === 'select' ) {
            ctrl.focus();
          }
        };
```

```
      this.alert = function(msg, cb) { dbox('alert', msg, cb);};
      this.confirm = function(msg, cb) { dbox('confirm', msg, cb);};
      this.process_cancel = function(msg, cb) { dbox('process', msg, cb);};
      this.prompt = function(msg, txt, cb) { dbox('prompt', msg, cb, txt);};
      this.select = function(msg, opts, cb, changeCb, txt, checkbox) { dbox('select', msg, cb,
txt, opts, changeCb, checkbox);};
    };

    svgEditorCtrl.showToDelete = function(){$log.debug(svgEditorCtrl.currSVG);}

      });
```

APPENDIX D – NGRID APPLICATION SERVER

NGRID platform provides NGRID Application Server which is a server-side online portal for NGRID admin staff and medical practitioners to login and create trials, add patients, as well as add or modify NGRID VD tests. The server-side codes are written as a microservice architecture that allows linear scalability under load.

As mentioned in NGRID SVGEditor, all changes to VD tests need to be persisted to NGRID application server. Below code is the server side codes that SVGEditor will call to fetch or modify NGRID VD Tests:

```
package controllers

import java.io.{ByteArrayInputStream, ByteArrayOutputStream, InputStreamReader}
import java.sql.SQLTimeoutException
import java.util.Calendar
import java.util.zip.GZIPInputStream
import javax.inject.{Inject, Singleton}

import scala.util.Try
import scala.concurrent.Future
import scala.concurrent.ExecutionContext.Implicits.global

import models._
import services._

import play.api.mvc._
import play.api.libs.json._
import play.api.cache.CacheApi
import play.api.libs.functional.syntax._

import com.google.common.io.CharStreams
import com.mohiva.play.silhouette.api.Silhouette

@Singleton
class SVGController @Inject()(svgService: SVGs, silhouette: Silhouette[DefaultEnv], cacheApi:
CacheApi) extends Controller {

  val cacheKeyWord = "SVGCache"

  implicit val svgRead: Reads[SVG] = (
    (JsPath \ "svgId").read[Int] and
      (JsPath \ "xmlContent").read[String] and
      (JsPath \ "testId").read[Int] and
      (JsPath \ "seq").read[Int]
    )(SVG.apply _)

  implicit val svgWrites: Writes[SVG] = new Writes[SVG] {
    def writes(s: SVG): JsValue = Json.obj(
      "svgId" -> s.svgId,
      "xmlContent" -> s.xmlContent,
      "testId" -> s.testId,
      "seq" -> s.seq
    )
  }

  def addSVG() = silhouette.SecuredAction.async { implicit request =>
    SVGForm.form.bindFromRequest.fold(
      errorForm => Future.successful(Ok(JsObject(Seq(
        "success" -> JsBoolean(false),
        "message" ->
          JsString("SVG form has errors"))))),
      data => {
        val svg = SVG(0,
          data.xmlContent,
          data.testId,
          data.seq)
        svgService.add(svg).map {
          case 0 => Ok(JsonServerMessage(false, "SVG already exists"))
          case 1 => Ok(JsonServerMessage(true, "SVG add successful"))
          case _ => InternalServerError(JsonServerMessage(false,
            """Critical SQL error.
              |Multiple SVGs with the same credentials.
              |Look into database at once.
            """.stripMargin))
        }.recover {
          case sq: SQLTimeoutException => InternalServerError(JsonServerMessage(false, "timeout"))
          case ex: Exception => InternalServerError(JsonServerMessage(false, "SQL ERROR: " +
ex.getMessage()))
        }
      }
    )
  }

  def deleteSVG() = silhouette.SecuredAction.async(parse.json) {
    implicit request =>
      val id = (request.body \ "svgId").asOpt[Int]
      id match {
        case Some(x: Int) =>
          svgService.delete(x).map(res => Ok(JsObject(Seq(
            "success" -> JsBoolean(true),
            "message" ->
              JsString("SVG Delete Successful"))))).recover {
            case ex: Exception => InternalServerError(JsObject(Seq(
              "success" -> JsBoolean(false),
              "message" ->
                JsString("SVG Delete Unsuccessful"))))
          }
        case None => Future(Ok(JsObject(Seq(
          "success" -> JsBoolean(false),
          "message" ->
            JsString("request did not contain proper format")))))
      }
  }

  def getSVGIds(testId: Int) = silhouette.SecuredAction.async {
    implicit request =>
      svgService.getAllIds(testId).map(res => Ok(JsObject(Seq("ids" ->
JsArray(res.map(JsNumber(_))))))).recover {
        case ex: Exception => InternalServerError(JsObject(Seq(
          "success" -> JsBoolean(false),
          "message" ->
            JsString("Could not Retrieve SVGs. SQL: " + ex.getMessage()))))
      }
  }

  def getSVGs() = silhouette.SecuredAction.async(parse.json) {
    implicit request =>
      val exId = (request.body \ "testId").asOpt[Int]
      exId match {
        case Some(id: Int) =>
          svgService.listAll(id).map(res => {
            Ok(Json.toJson(res.sortBy(_.seq)))
          }).recover {
            case ex: Exception => InternalServerError(JsObject(Seq(
              "success" -> JsBoolean(false),
              "message" ->
                JsString("Could not Retrieve SVGs. Please Try again later"))))
          }
        case None => Future(Unauthorized(JsObject(Seq(
          "success" -> JsBoolean(false),
          "message" ->
            JsString("Error In format")))))
      }
  }

  def deleteMultiSVGs() = silhouette.SecuredAction.async(parse.json){
    implicit request =>
      val svgSeq = request.body.asOpt[List[SVG]]
      svgSeq match{
        case Some(x: List[SVG]) => svgService.deleteMultiple(x.toSeq).map(res =>
Ok(JsObject(Seq("success" ->
          JsBoolean(true), "deleted" -> JsNumber(res.sum)))))
        case None => Future(Ok(JsObject(Seq(
          "success" -> JsBoolean(true),
          "deleted" ->
            JsNumber(0)))))
      }
  }

  def addMultiSVGs() = silhouette.SecuredAction.async(parse.json(maxLength = 1024*10000)){
    implicit request =>
      val svgSeq = request.body.asOpt[List[SVG]]
      svgSeq match{
        case Some(x: List[SVG]) => svgService.addMultiple(x.toSeq).map(res => {
          Ok(JsObject(Seq("success" ->
          JsBoolean(true), "added" -> JsNumber(res.sum)
          )))}).recover {
          case sq: SQLTimeoutException => InternalServerError(JsonServerMessage(false, "timeout"))
          case ex: Exception => InternalServerError(JsonServerMessage(false, "SQL ERROR: " +
ex.getMessage()))
        }
        case None => {
          Future(Ok(JsObject(Seq(
            "success" -> JsBoolean(true),
            "added" ->
              JsNumber(0)))))
        }
      }

  }
```

APPENDIX E – NGRID HEATMAP GENERATOR

```
from builtins import print
import numpy as np
import scipy as scipy
from scipy import stats
import matplotlib.pyplot as plt
from matplotlib import colors
import matplotlib.lines as mlines
from PIL import Image


# Helper function: compare against a given value, if smaller return zero.
def compare_with_value(x,value):
    if x < value:
        return 0
    else:
        return x

plot_test_frames = True

# define a set of frames to be shown to patient
f1  = np.matrix( '1 1 1 1 1 ; '
                 '0 0 0 0 0 ; '
                 '0 0 0 0 0 ; '
                 '0 0 0 0 0 ; '
                 '0 0 0 0 0'
                 )
f2  = np.matrix( '0 0 0 0 1 ; '
                 '0 0 0 0 1 ; '
                 '0 0 0 0 1 ; '
                 '0 0 0 0 1 ; '
                 '0 0 0 0 1'
                 )

f3  = np.matrix( '0 0 0 0 0 ; '
                 '0 0 0 0 0 ; '
                 '0 0 0 0 0 ; '
                 '0 0 0 0 0 ; '
                 '1 1 1 1 1'
                 )

f4  = np.matrix( '1 0 0 0 0 ; '
                 '1 0 0 0 0 ; '
                 '1 0 0 0 0 ; '
                 '1 0 0 0 0 ; '
                 '1 0 0 0 0'
                 )

f5  = np.matrix( '0 0 0 0 0 ; '
                 '0 1 1 1 0 ; '
                 '0 1 1 1 0 ; '
                 '0 1 1 1 0 ; '
                 '0 0 0 0 0'
                 )

f6  = np.matrix( '0 0 0 0 0 ; '
                 '1 1 1 1 1 ; '
                 '0 0 0 0 0 ; '
                 '0 0 0 0 0 ; '
                 '0 0 0 0 0'
```

```
                  )

f7  = np.matrix( '0 0 0 0 0 ; '
                 '0 0 0 0 0 ; '
                 '1 1 1 1 1 ; '
                 '0 0 0 0 0 ; '
                 '0 0 0 0 0'
                 )

f8  = np.matrix( '0 0 0 0 0 ; '
                 '0 0 0 0 0 ; '
                 '0 0 0 0 0 ; '
                 '1 1 1 1 1 ; '
                 '0 0 0 0 0'
                 )

f9  = np.matrix( '0 1 0 0 0 ; '
                 '0 1 0 0 0 ; '
                 '0 1 0 0 0 ; '
                 '0 1 0 0 0 ; '
                 '0 1 0 0 0'
                 )

f10 = np.matrix( '0 0 1 0 0 ; '
                 '0 0 1 0 0 ; '
                 '0 0 1 0 0 ; '
                 '0 0 1 0 0 ; '
                 '0 0 1 0 0'
                 )

f11 = np.matrix( '0 0 0 1 0 ; '
                 '0 0 0 1 0 ; '
                 '0 0 0 1 0 ; '
                 '0 0 0 1 0 ; '
                 '0 0 0 1 0'
                 )

f12 = np.matrix( '0 0 0 0 1 ; '
                 '0 0 0 1 0 ; '
                 '0 0 1 0 0 ; '
                 '0 1 0 0 0 ; '
                 '1 0 0 0 0'
                 )

f13 = np.matrix( '1 0 0 0 0 ; '
                 '0 1 0 0 0 ; '
                 '0 0 1 0 0 ; '
                 '0 0 0 1 0 ; '
                 '0 0 0 0 1'
                 )

f14 = np.matrix( '1 1 1 1 1 ; '
                 '1 1 1 1 1 ; '
                 '0 0 0 0 0 ; '
                 '0 0 0 0 0 ; '
                 '0 0 0 0 0'
                 )

f15 = np.matrix( '0 0 0 0 0 ; '
                 '1 1 0 1 1 ; '
                 '1 1 0 1 1 ; '
                 '1 1 0 1 1 ; '
                 '1 1 0 1 1'
                 )

f16 = np.matrix( '1 1 0 1 1 ; '
                 '1 1 0 1 1 ; '
                 '1 1 0 1 1 ; '
                 '1 1 0 1 1 ; '
                 '1 1 0 1 1'
                 )


# simulate patient answers to the defined frames above while assuming patient
is seeing bad/wavy
# in pixels that fall under cells located at column 3 and row 3,4,5
an = {
1:  "good",
2:  "good",
3:  "bad",
4:  "good",
5:  "bad",
6:  "good",
7:  "bad",
8:  "bad",
9:  "good",
10: "bad",
11: "good",
12: "bad",
13: "bad",
14: "good",
15: "good",
16: "good"
}



# Show 64 frames to patient and collect the answers
frames_seen_by_patient = [
   f1 ,f2 ,f3 ,f4 ,f5 ,f6 ,f7 ,f8 ,f9 ,f10 ,f11 ,f12 ,f13, f14, f15, f16
]
patient_answers = [
   an[1], an[2], an[3], an[4], an[5], an[6], an[7], an[8], an[9], an[10],
an[11], an[12], an[13], an[14], an[15], an[16]
]


#init PM, HM and HMM as a sparse matrix of 5x5
perfect_matrix = np.zeros((5, 5))
heat_matrix    = np.zeros((5, 5))
heatmap_matrix = np.zeros((5, 5))


if plot_test_frames == True:
    Nr = 4
    Nc = 4
    cmap = "cool"

    fig, axs = plt.subplots(Nr, Nc, figsize=(20, 16), dpi=100, facecolor='w',
edgecolor='k')
    fig.canvas.set_window_title('Designed Frames')
    fig.subplots_adjust(hspace=0.25, wspace=0.25 )
    plt.rcParams.update({'font.size': 22})
    #fig.suptitle('Designed Frames F1 to F16')
    k = 0
    images = []
    for i in range(Nr):
        for j in range(Nc):
            data = frames_seen_by_patient[k]
            images.append(axs[i, j].imshow(data, cmap=cmap))
            #axs[i, j].set_axis_off()
            axs[i, j].set_facecolor('#eafff5')
            axs[i, j].get_xaxis().set_ticks([0,1,2,3,4])
            axs[i, j].get_yaxis().set_ticks([0,1,2,3,4])
            frame_name = "F(" + str(k+1) + ")"
            #axs[i, j].text(0.01, 0.05, frame_name , fontsize=18)
            k = k + 1

    #plt.show()
    plt.savefig('frames.png', bbox_inches='tight')


k = 0
print("=================")
for frame, answer in zip(frames_seen_by_patient, patient_answers):
    k = k + 1
    print("Frame" + str(k) + ":")
    print (frame)
    print("Patient marked this frame as seen " + answer)
print("=================")




#Go through the frames and answers to calculate PM, HM
print("Calculating Perfect Matrix (PM):")
k = 0
for frame, answer in zip(frames_seen_by_patient, patient_answers):
    k = k + 1
    print("PM value after considering frame" + str(k) + ":")
    perfect_matrix = perfect_matrix + frame
    heat_matrix = heat_matrix + frame
    print (perfect_matrix)
    print("------")
print("=================")


print("Heat Matrix (HM) is initiated as the same as Perfect Matrix (PM):")
print (heat_matrix)
print("------")
print("Now going through the frames and corresponding patients answer. Heat
Matrix (HM) value is printed at each state:")
k = 0
for frame, answer in zip(frames_seen_by_patient, patient_answers):
    k = k + 1
    print("HM value after considering frame" + str(k) + " (patients answer was:
" + answer + "):")
    if answer == 'good':
        heat_matrix = heat_matrix + frame
    elif answer == 'bad':
        heat_matrix = heat_matrix - 2 * frame
    print (heat_matrix)
    print("------")
print("=================")

print("HeatMap Matrix (HMM) is: PM - HM")
print("Please note in HMM, the higher the value of a cell, the more heat it has
(e.g. higher chance of VD at that location)")
print("PM was:")
print(perfect_matrix)
print("HM was:")
print(heat_matrix)
print("Thus HMM is:")
#Calculate HMM, the higher the value of a cell, the more heat it has (e.g.
higher chance of VD at that location)
heatmap_matrix = perfect_matrix - heat_matrix
print(heatmap_matrix)
print("=================")


print ("If our SVG test has good area coverage (e.g. test more than 90% of the
visual field), we can eliminate the lower than mean heat values. This will help
to produce cleaner heatmap.")
# If our SVG test has good area coverage (e.g. test more than 90% of the visual
field), we can eliminate the lower than mean heat values
# This will help to produce cleaner heatmap
vectorized__compare_with_value = np.vectorize(compare_with_value)
mean = np.floor(np.mean(heatmap_matrix)).astype(int)

print("Clamping HMM based on mean value of HMM which is " + str(mean) + ":")
heatmap_matrix = vectorized__compare_with_value(heatmap_matrix, mean)
print(heatmap_matrix)
print("=================")


print ("Now we find the z-score of the heatmap values and clamp on heatmap z-
score values that are higher than 75% SD heat")
# find the z-score of the heat values and only look for heat values that are
higher than 75% SD
heatmap_zscore_matrix = np.array(stats.zscore(heatmap_matrix.A1)).reshape((5,
5))
print ("HMM z-scores will be:")
print(heatmap_zscore_matrix)
vectorized__compare_with_value = np.vectorize(compare_with_value,
otypes=[np.float])
heatmap_zscore_matrix = vectorized__compare_with_value(heatmap_zscore_matrix,
0.75)
print ("After clamping on values less than 75% SD heat, we will get:")
print(heatmap_zscore_matrix)
print("=================")



print("opacity_matrix derived from normalized_heatmap_zscore_matrix: ")
opacity_matrix = heatmap_zscore_matrix / np.amax(heatmap_zscore_matrix)
opacity_matrix = opacity_matrix * 100
#mask_zeros = opacity_matrix == 0
#mask_nonzeros = opacity_matrix != 0
#opacity_matrix[mask_nonzeros] = abs (opacity_matrix[mask_nonzeros] - 100.0)
#opacity_matrix[mask_zeros] = opacity_matrix[mask_zeros] + 100.0
print(opacity_matrix)
print("=================")


oneDimention_heatmap_zscore_matrix = heatmap_zscore_matrix.flatten()
radious_matrix = np.array([float(i) / max(oneDimention_heatmap_zscore_matrix)
for i in oneDimention_heatmap_zscore_matrix]).reshape((5, 5))
print("radious_matrix derived from normalized_heatmap_zscore_matrix: ")
print(radious_matrix)
print("=================")
```

```
fig, ax = plt.subplots( figsize=(20, 20), dpi=100 )
plt.axis('equal', bbox_inches=0)
ax.set_xlim((-1, 5))
ax.set_ylim((5, -1))
#ax.set_facecolor('xkcd:salmon')
ax.set_facecolor((0, 1, 0))
#plt.gca().invert_yaxis()

plt.savefig('stage0.png', bbox_inches='tight')

print("And the location of heat cell indices are:")
reshaped__heatmap_matrix = heatmap_matrix.A1.reshape((5, 5))
k = 0
avg__heat_i = 0
avg__heat_j = 0
heat_cell_locations = np.nonzero(heatmap_zscore_matrix > 0)
#heat_cell_locations is a two-dimensional array. index 0 is array of i'th
indices and 1 is array of corresponding js.
for i in heat_cell_locations[0]:
    j = heat_cell_locations[1][k]
    k = k + 1
    print("\tAt Zij location (i=" + str(j) + ",j=" + str(i) + ") found heat-z-
score of " +
          '{0:.2f}'.format(heatmap_zscore_matrix[i][j]) +
          " Opacity is " + '{0:.2f}'.format(opacity_matrix[i][j]) +
          " Radius is " + '{0:.2f}'.format(radious_matrix[i][j]) +
          " and heat-value of " + str(reshaped__heatmap_matrix[i][j])
         )

    avg__heat_i = avg__heat_i + i * opacity_matrix[i][j] / 100
    avg__heat_j = avg__heat_j + j * opacity_matrix[i][j] / 100

    ax.add_artist(plt.Circle((j,i), radious_matrix[i][j] , color='r',
alpha=opacity_matrix[i][j] / 100))

avg__heat_i = avg__heat_i / len(heat_cell_locations[0])
avg__heat_j = avg__heat_j / len(heat_cell_locations[0])

avg__heat_i = (avg__heat_i + np.average(heat_cell_locations[0]))/2
avg__heat_j = (avg__heat_j + np.average(heat_cell_locations[1]))/2
print("Center of the heat is " +  str((avg__heat_j, avg__heat_i)))

print("================")
#plt.axis('off') #don't use as it will remove the bgcolor for axises
plt.savefig('stage1.png', bbox_inches='tight')


im = Image.open('stage0.png')
greens_in_im0 = 0
for pixel in im.getdata():
    if pixel == (0, 255, 0, 255): # this is for RGBA, for RGB remove the last
255
        greens_in_im0 += 1

im = Image.open('stage1.png')
greens_in_im1 = 0
for pixel in im.getdata():
    if pixel == (0, 255, 0, 255):
        greens_in_im1 += 1


heat_coverage = 100 - 100 * (greens_in_im1/greens_in_im0)
```

```
print('Heat coverage is {0:.2f}%'.format(heat_coverage))


ax.set_facecolor((1, 1, 1))
ax.add_artist(plt.Circle( (avg__heat_j, avg__heat_i), 0.05 , color='b',
alpha=0.5))
#ax.text(avg__heat_j + 0.05, avg__heat_i + 0.05, 'χ =
({0:.2f},'.format(avg__heat_j) + '{0:.2f})'.format(avg__heat_i) , fontsize=20)
#ax.text(avg__heat_j + 0.05, avg__heat_i + 0.20, 'η =
{0:.1f}%'.format(heat_coverage) , fontsize=20)

i = 0
j = 0
i_max = 6
i_min = -2
j_max = 6
j_min = -2
for i in np.arange(i_min, i_max, 0.1):
        ax.add_line(mlines.Line2D([i, i], [j_min, j_max]))

for j in np.arange(j_min, j_max, 0.1):
        ax.add_line(mlines.Line2D([i_min, i_max], [j, j] ))


plt.savefig('final.png', bbox_inches='tight')

#colorImage = Image.open("results.png")
#rotated = colorImage.rotate(-90)
#rotated.show()
#plt.show()
```

APPENDIX F – NGRID VD SIMULATOR

Here the details for the program we wrote to create the VD simulations:

```
namespace VDSimul
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button_Click_create_vd(object sender, EventArgs e)
        {
            this.textBox_temp_code.Text = this.simpleEditor1.Text;

            openFileDialog1.Title = "Browse for an Input Image File";
            openFileDialog1.CheckFileExists = true;
            openFileDialog1.CheckPathExists = true;
            openFileDialog1.DefaultExt = "sqlite";
            openFileDialog1.Filter = "PNG Files (*.png)|*.png|JPG Files (*.jpg)|*.jpg|BMP Files
(*.bmp)|*.bmp|All files (*.*)|*.*";
            openFileDialog1.FilterIndex = 1;
            openFileDialog1.RestoreDirectory = true;
            openFileDialog1.ReadOnlyChecked = true;
            openFileDialog1.ShowReadOnly = true;
            if (openFileDialog1.ShowDialog() != DialogResult.Cancel)
            {
                selected_file = openFileDialog1.FileName;
                var task = Task.Run(() => backgroundWorker_create_the_vd_image(selected_file));
            }
        }


        [DllImport("user32.dll")]
        public static extern int MessageBox2(int hWnd, String text, String caption, uint type);



        public bool isPointInTheCircle(double cx, double cy, double cr, double x, double y)
        {
            double d = Math.Pow(cx - x, 2) + Math.Pow(cy - y, 2);
            if (d > Math.Pow(cr, 2))
            {
                return false;
            }
            else
            {
                return true;
            }
        }


        public async void backgroundWorker_save_an_image(string filepath, string append_ext,
Bitmap image, System.Drawing.Imaging.ImageFormat format)
        {
            image.Save(filepath + append_ext, format);
        }

        public async void backgroundWorker_create_the_vd_image(string filepath)
        {
            this.CompileAndRun(this.textBox_temp_code.Text);
            if (VDmethod == null)
            {
```

```
        MessageBox.Show("VD function compilation failed! Please correct the VD Code.");
        return;
    }

    VDs_mins = new Dictionary<int, int>();
    VDs_maxs = new Dictionary<int, int>();

    Bitmap b1 = new Bitmap(filepath);

    int height = b1.Height;
    int width = b1.Width;

    Invoke((MethodInvoker)delegate ()
    {
        this.textBox_Image_Height.Text = height + "";
        this.textBox_Image_Width.Text = width + "";
    });

    int cylinder_radius = int.Parse(this.textBox_VD_Sphere_Radius.Text);
    int origin_x_axis = int.Parse(this.textBox_VD_center_x.Text);
    int origin_y_axis = int.Parse(this.textBox_VD_center_y.Text);

    int cylinder_start_x = origin_x_axis -cylinder_radius;
    int cylinder_finish_x = origin_x_axis + cylinder_radius;

    double[][] org = new double[width][];
    Color[][] org_color = new Color[width][];
    Color[][] vd_color = new Color[width][];
    double[][] vd = new double[width][];
    for (int i = 0; i < width; i++)
    {
        org[i] = new double[height];
        vd[i] = new double[height];
        org_color[i] = new Color[height];
        vd_color[i] = new Color[height];
    }

    var img = new Bitmap(width, height);
    var vd_img = new Bitmap(width, height);
    for (int i = 0; i < width; i++)
    {
        for (int j = 0; j < height; j++)
        {
            img.SetPixel(i, j, Color.White);
            vd_img.SetPixel(i, j, Color.White);
            org[i][j] = 0;
            vd[i][j] = 0;

            vd_color[i][j] = Color.Transparent;
            org_color[i][j] = Color.Transparent;
        }
    }

    for (int i = 0; i < width; i++)
    {
        for (int j = 0; j < height; j++)
        {
            if (b1.GetPixel(i, j).ToArgb() != Color.White.ToArgb())
            {
                org[i][j] = 1;
                org_color[i][j] = b1.GetPixel(i, j);
                img.SetPixel(i, j, b1.GetPixel(i, j));
            }
        }
    }

    if (checkBox_add_org_to_vd.Checked)
    {
        for (int i = 0; i < width; i++)
```

```
                {
                    for (int j = 0; j < height; j++)
                    {
                        vd[i][j] = org[i][j];
                    }
                }
            }

            for (int i = 0; i < width; i++)
            {
                for (int j = 0; j < height; j++)
                {
                    if (i > cylinder_start_x && i < cylinder_finish_x)
                    {
                        if (this.checkBox1.Checked)
                        {
                            vd_img.SetPixel(i, j, Color.White);
                        }
                        else
                        {
                            if (checkBox_add_org_to_vd.Checked)
                            {
                                vd_img.SetPixel(i, j, org_color[i][j]);
                            }
                        }

                        vd[i][j] = 0;
                    }
                    else if (org[i][j] == 1)
                    {
                        if (checkBox_add_org_to_vd.Checked)
                        {
                            vd_img.SetPixel(i, j, org_color[i][j]); //changed from BLACK
                        }
                    }
                }
            }

            double R1 = double.Parse(textBox_VD_Sphere_R1.Text);
            for (int i = cylinder_start_x; i < cylinder_finish_x; i++)
            {
                for (int j = 0; j < height; j++)
                {
                    if (org[i][j] == 0) continue;

                    Color orginal_pixel = org_color[i][j];
                    if (!this.isPointInTheCircle(origin_x_axis, origin_y_axis, cylinder_radius, i,
j))
                    {
                        vd[i][j] = 1;
                        if (checkBox_add_org_to_vd.Checked)
                            vd_img.SetPixel(i, j, orginal_pixel);
                        continue;
                    }

                    VDPoint.Point p = (VDPoint.Point) VDmethod.Invoke(null, new object[] { i, j,
cylinder_radius, R1, origin_x_axis, origin_y_axis });

                    if (!VDs_maxs.ContainsKey(p.j))
                        VDs_maxs.Add(p.j, p.i);
                    if (!VDs_mins.ContainsKey(p.j))
                        VDs_mins.Add(p.j, p.i);

                    if (VDs_maxs[p.j] < p.i) VDs_maxs[p.j] = p.i;
                    if (VDs_mins[p.j] > p.i) VDs_mins[p.j] = p.i;

                    vd[p.i][p.j] = 1;

                    Color c = orginal_pixel;
                    if (checkBox_transparency.Checked)
                    {
                        c = Darken(orginal_pixel,  p.h);
                    }

                    vd_img.SetPixel(p.i, p.j, c);

                    if (checkBox_pxel_pack.Checked)
                    {
                        vd[p.i - 1][p.j - 1] = 1;
                        vd_img.SetPixel(p.i - 1, p.j - 1, c);
                        vd[p.i - 1][p.j + 1] = 1;
                        vd_img.SetPixel(p.i - 1, p.j + 1, c);
                        vd[p.i + 1][p.j - 1] = 1;
                        vd_img.SetPixel(p.i + 1, p.j - 1, c);
                        vd[p.i + 1][p.j + 1] = 1;
                        vd_img.SetPixel(p.i + 1, p.j + 1, c);
                    }
                }
            }


            if (checkBox_Draw_TS_Circles.Checked)
            {
                for (int i = cylinder_start_x; i < cylinder_finish_x; i++)
                {
                    for (int j = 0; j < height; j++)
                    {
                        Color orginal_pixel = vd_img.GetPixel(i, j);
                        Color c = orginal_pixel;

                        if (isPointInTheCircle(origin_x_axis, origin_y_axis, cylinder_radius, i,
j))

                        {

                            int xr = i - origin_x_axis;
                            int yr = j - origin_y_axis;
                            int rr = xr * xr + yr * yr;
                            int sphere_rr = cylinder_radius * cylinder_radius;
                            double r_val = 1;
                            r_val = 1 - (sphere_rr - rr) / (double) (sphere_rr) +
double.Parse(textBox_ts_circle_offset.Text);
                            c = Darken(orginal_pixel, r_val);
                            vd_img.SetPixel(i, j, c);
                            continue;
                        }
                    }
                }
            }


            if (checkBox_new_method_to_overlay_vd_on_org.Checked)
            {
                for (int i = 0; i < width; i++)
                {
                    for (int j = 0; j < height; j++)
                    {
                        Color orginal_pixel = img.GetPixel(i, j);

                        if (!VDs_mins.ContainsKey(j) && !VDs_maxs.ContainsKey(j))
                        {
                            vd_img.SetPixel(i, j, orginal_pixel);
                            continue;
                        }

                        if (i <= VDs_mins[j] || i >= VDs_maxs[j])
                        {
                            vd_img.SetPixel(i, j, orginal_pixel);
                        }
                    }
                }
            }


            Invoke((MethodInvoker)delegate ()
            {
                this.kpImageViewer2.Image = img;
                this.kpImageViewer2.ShowPreview = false;
                this.kpImageViewer2.save_format = ImageFormat.Bmp;
                this.kpImageViewer2.file_path = filepath + ".org.bmp";
                this.kpImageViewer2.save_status_label = this.statusStrip_src_image_label1;
            });

            Invoke((MethodInvoker) delegate()
            {
                this.kpImageViewer1.Image = vd_img;
                this.kpImageViewer1.ShowPreview = false;
                this.tabPage5.Select();
                this.tabControl1.SelectedTab = this.tabPage5;

                this.kpImageViewer1.save_format = ImageFormat.Bmp;
                this.kpImageViewer1.file_path = filepath + ".VDed.bmp";
                this.kpImageViewer1.save_status_label = this.statusStrip_vd_img_label1;
            });

        }

        protected override bool ProcessCmdKey(ref Message msg, Keys keyData)
        {
            if (keyData == (Keys.Alt | Keys.O))
            {
                button_Click_create_vd(null, null);
                return true;
            }

            if (keyData == (Keys.Alt | Keys.T))
            {
                textBox_VD_center_x.Text = kpImageViewer1.target_mode_cursor_loc_x + "";
                textBox_VD_center_y.Text = kpImageViewer1.target_mode_cursor_loc_y + "";

                textBox_VD_Sphere_Radius.Text = kpImageViewer1.textBox_R.Text;
                textBox_VD_Sphere_R1.Text = kpImageViewer1.textBox_R1.Text;

                this.textBox_temp_code.Text = this.simpleEditor1.Text;
                var task = Task.Run(() => backgroundWorker_create_the_vd_image(selected_file));

                return true;
            }


            if (keyData == (Keys.Alt | Keys.R))
            {
                this.textBox_temp_code.Text = this.simpleEditor1.Text;
                var task = Task.Run(() => backgroundWorker_create_the_vd_image(selected_file));

                return true;
            }

            if (keyData == (Keys.Alt | Keys.D4))
            {
                this.tabPage5.Select();
                this.tabControl1.SelectedTab = this.tabPage5;

                return true;
            }

            if (keyData == (Keys.Alt | Keys.D3))
            {
```

```
            this.tabPage2.Select();
            this.tabControl1.SelectedTab = this.tabPage2;

            return true;
        }

        if (keyData == (Keys.Alt | Keys.D1))
        {
            this.tabPage3.Select();
            this.tabControl1.SelectedTab = this.tabPage3;

            return true;
        }

        if (keyData == (Keys.Alt | Keys.D2))
        {
            this.tabPage4.Select();
            this.tabControl1.SelectedTab = this.tabPage4;

            return true;
        }



        return base.ProcessCmdKey(ref msg, keyData);
    }

    public Color Darken(Color color, double darkenAmount)
    {
        HSLColor hslColor = new HSLColor(color);
        hslColor.Luminosity *= darkenAmount;
        return hslColor;
    }

    Dictionary<int, int> VDs_mins = new Dictionary<int, int>();
    Dictionary<int, int> VDs_maxs = new Dictionary<int, int>();

    private void tabControl1_DrawItem(object sender, DrawItemEventArgs e)
    {
        TabPage page = tabControl1.TabPages[e.Index];
        e.Graphics.FillRectangle(new SolidBrush(page.BackColor), e.Bounds);

        Rectangle paddedBounds = e.Bounds;
        int yOffset = (e.State == DrawItemState.Selected) ? -3 : 1;
        paddedBounds.Offset(1, yOffset);

        if (e.Index == 0)
        {
            e.Graphics.FillRectangle(new SolidBrush(Color.LightPink), e.Bounds);
            TextRenderer.DrawText(e.Graphics, page.Text, Font, paddedBounds, Color.Black);
        }
        else if (e.Index == 1)
        {
            e.Graphics.FillRectangle(new SolidBrush(Color.LightGreen), e.Bounds);
            TextRenderer.DrawText(e.Graphics, page.Text, Font, paddedBounds, Color.Black);
        }
        else if (e.Index == 2)
        {
            e.Graphics.FillRectangle(new SolidBrush(Color.LightBlue), e.Bounds);
            TextRenderer.DrawText(e.Graphics, page.Text, Font, paddedBounds, Color.Black);
        }
        else if (e.Index == 3)
        {
            e.Graphics.FillRectangle(new SolidBrush(Color.LightYellow), e.Bounds);
            TextRenderer.DrawText(e.Graphics, page.Text, Font, paddedBounds, Color.Black);
        }

        else
        {
                TextRenderer.DrawText(e.Graphics, page.Text, Font, paddedBounds, page.ForeColor);
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            WindowState = FormWindowState.Maximized;
            MinimumSize = this.Size;
            MaximumSize = this.Size;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            toolStrip_main_StatusLabel1.Text = "";
            this.textBox_temp_code.Text = this.simpleEditor1.Text;

            this.CompileAndRun(this.textBox_temp_code.Text);
            if (VDmethod == null)
            {
                MessageBox.Show("VD function compilation failed! Please correct the VD Code.");
                return;
            }

            this.textBox1_testPoint.Text = this.textBox1_testPoint.Text.Replace(" ", ",");
            this.textBox1_testPoint.Text = this.textBox1_testPoint.Text.Replace(",,", ",");
            this.textBox1_testPoint.Text = this.textBox1_testPoint.Text.Replace(",,", ",");
            this.textBox1_testPoint.Text = this.textBox1_testPoint.Text.Replace(",,", ",");

            string[] toks = this.textBox1_testPoint.Text.Split(',');
            if (toks.Length < 2) return;
            int x = int.Parse(toks[0]);
            int y = int.Parse(toks[1]);

            int cylinder_radius = int.Parse(this.textBox_VD_Sphere_Radius.Text);
            int origin_x_axis = int.Parse(this.textBox_VD_center_x.Text);
            int origin_y_axis = int.Parse(this.textBox_VD_center_y.Text);
            double R1 = double.Parse(textBox_VD_Sphere_R1.Text);


            VDPoint.Point p = (VDPoint.Point)VDmethod.Invoke(null, new object[] { x, y,
cylinder_radius, R1, origin_x_axis, origin_y_axis });

            toolStrip_main_StatusLabel1.Text = "" + p.i + "," + p.j + "     " + p.h;
        }

        private void textBox1_testPoint_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == (Keys.Enter))
            {
                button2_Click(null, null);
                return;
            }
        }
    }
}
```

REFERENCES

[1] The Eye Diseases Prevalence Research Group, "Prevalence of age-related macular degeneration in the united states," *Archives of Ophthalmology*, vol. 122, no. 4, pp. 564–572, 2004.

[2] M. Wang, I. C. Munch, P. W. Hasler, C. Prünte, and M. Larsen, "Central serous chorioretinopathy," *Acta Ophthalmologica*, vol. 86, no. 2, pp. 126–145, Mar. 2008.

[3] W. L. Wong *et al.*, "Global prevalence of age-related macular degeneration and disease burden projection for 2020 and 2040: a systematic review and meta-analysis," *The Lancet Global Health*, vol. 2, no. 2, pp. e106–e116, Feb. 2014.

[4] D. J. Taylor, A. E. Hobby, A. M. Binns, and D. P. Crabb, "How does age-related macular degeneration affect real-world visual ability and quality of life? A systematic review," *BMJ Open*, vol. 6, no. 12, p. e011504, Dec. 2016.

[5] N. M. Bressler, S. B. Bressler, and S. L. Fine, "Age-related macular degeneration," *Survey of Ophthalmology*, vol. 32, no. 6, pp. 375–413, 1988.

[6] L. The, "Age-related macular degeneration: treatment at what cost?," *Lancet (London, England)*, vol. 392, no. 10153, p. 1090, 2018.

[7] C. Busch *et al.*, "Retinal Microvasculature and Visual Acuity after Intravitreal Aflibercept in Diabetic Macular Edema: An Optical Coherence Tomography Angiography Study," *Scientific reports*, vol. 9, no. 1, p. 1561, 2019.

[8] R. F. Spaide, J. G. Fujimoto, N. K. Waheed, S. R. Sadda, and G. Staurenghi, "Optical coherence tomography angiography," *Progress in Retinal and Eye Research*, vol. 64, pp. 1–55, May 2018.

[9] A. H. Rogers and J. S. Duker, *Retina*. Philadelphia: Mosby Elsevier, 2008.

[10] F. G. Holz and R. F. Spaide, *Medical Retina: Focus on Retinal Imaging*. Springer, 2010.

[11] E. Midena and E. Pilotto, "Microperimetry in age: related macular degeneration," *Eye*, vol. 31, no. 7, p. 985, 2017.

[12] E. Midena and S. Vujosevic, "Metamorphopsia: An Overlooked Visual Symptom," *Ophthalmic Research*, vol. 55, no. 1, pp. 26–36, Nov. 2015.

[13] J. C. Besharse and D. Bok, *The Retina and Its Disorders*. Academic Press, 2011.

[14] E. R. F. Collaboration and others, "Diabetes mellitus, fasting blood glucose concentration, and risk of vascular disease: a collaborative meta-analysis of 102 prospective studies," *The Lancet*, vol. 375, no. 9733, pp. 2215–2222, 2010.

[15] R. R. A. Bourne *et al.*, "Causes of vision loss worldwide, 1990-2010: a systematic analysis," *Lancet Glob Health*, vol. 1, no. 6, pp. e339-349, Dec. 2013.

[16] R. F. Spaide *et al.*, "Central serous chorioretinopathy in younger and older adults," *Ophthalmology*, vol. 103, no. 12, pp. 2070–2079; discussion 2079-2080, Dec. 1996.

[17] T. J. Peiris, H. E. El Rami, and J. K. Sun, "CENTRAL SEROUS CHORIORETINOPATHY ASSOCIATED WITH STEROID ENEMA," *Retin Cases Brief Rep*, Jul. 2018.

[18] D. S. Friedman *et al.*, "Prevalence of age-related macular degeneration in the united states," *Archives of Ophthalmology*, vol. 122, no. 4, pp. 564–572, 2004.

[19] T. Schlote, M. Grueb, J. Mielke, and J. M. Rohrbach, *Pocket Atlas of Ophthalmology*. 2006.

[20] R. E. Hogg and U. Chakravarthy, “Visual function and dysfunction in early and late age-related maculopathy,” *Progress in Retinal and Eye Research*, vol. 25, no. 3, pp. 249–276, May 2006.
[21] S. Parmet, C. Lynm, and R. M. Glass, “Age-related macular degeneration,” *JAMA*, vol. 295, no. 20, pp. 2438–2438, 2006.
[22] S. Plainis, P. Tzatzala, Y. Orphanos, and M. K. TSILIMBARIS, “A modified ETDRS visual acuity chart for European-wide use,” *Optometry & Vision Science*, vol. 84, no. 7, pp. 647–653, 2007.
[23] C. Springer, S. Bültmann, H. E. Völcker, and K. Rohrschneider, “Fundus Perimetry with the Micro Perimeter 1 in Normal Individuals: Comparison with Conventional Threshold Perimetry,” *Ophthalmology*, vol. 112, no. 5, pp. 848–854, May 2005.
[24] F. L. Ferris III, A. Kassoff, G. H. Bresnick, and I. Bailey, “New visual acuity charts for clinical research,” *American journal of ophthalmology*, vol. 94, no. 1, pp. 91–96, 1982.
[25] I. A. Falkenstein *et al.*, “Comparison of Visual Acuity in Macular Degeneration Patients Measured with Snellen and Early Treatment Diabetic Retinopathy Study Charts,” *Ophthalmology*, vol. 115, no. 2, pp. 319–323, Feb. 2008.
[26] M. Amsler, “Earliest symptoms of diseases of the macula,” *The British journal of ophthalmology*, vol. 37, no. 9, p. 521, 1953.
[27] M. Amsler, “Earliest symptoms of diseases of the macula,” *The British journal of ophthalmology*, vol. 37, no. 9, p. 521, 1953.
[28] J. E. Dowling, *The Retina: An Approachable Part of the Brain*. Belknap Press of Harvard University Press, 1987.
[29] D. Thomas and G. Duguid, “Optical coherence tomography--a review of the principles and contemporary uses in retinal investigation,” *Eye (Lond)*, vol. 18, no. 6, pp. 561–570, Jun. 2004.
[30] “Macular Degeneration | Lake Travis Eye & Laser Center.” [Online]. Available: https://laketraviseyecenter.com/macular-degeneration/. [Accessed: 20-Feb-2019].
[31] N. Yoshimura, *Oct-atlas*, 1st edition. New York: Springer, 2013.
[32] P. J. Patel *et al.*, “Repeatability of Stratus Optical Coherence Tomography Measures in Neovascular Age-Related Macular Degeneration,” *Investigative Opthalmology & Visual Science*, vol. 49, no. 3, p. 1084, Mar. 2008.
[33] R. Trevino, “Recent progress in macular function self-assessment,” *Ophthalmic and Physiological Optics*, vol. 28, no. 3, pp. 183–192, May 2008.
[34] E. Collazo, *Portable electronic amsler test*. US8047652, 2011.
[35] R. Hirji, *Near eye opthalmic device*. US20080309879, 2008.
[36] M. C. Roser, *Visual and memory stimulating retina self-monitoring system*. US7798645, 2010.
[37] A. A. Sadun and M. Wall, *System and method of detecting visual field defects*. US4818091, 1989.
[38] W. Fink and A. A. Sadun, “Three-dimensional computer-automated threshold Amsler grid test,” *Journal of biomedical optics*, vol. 9, no. 1, pp. 149–153, 2004.
[39] A. Loewenstein *et al.*, “Replacing the Amsler grid,” *Ophthalmology*, vol. 110, no. 5, pp. 966–970, May 2003.
[40] D. Palanker, *Metamorphopsia testing and related methods*. WO2014022850A1, 2014.

[41] W. Kohn and J. A. Klingshirn, *Characterization and correction of macular distortion*. US8708495, 2014.
[42] V. Lakshminarayanan and J. M. Enoch, “Vernier acuity and aging,” *International Ophthalmology*, vol. 19, no. 2, pp. 109–115, 1995.
[43] J. H. Kaas, L. A. Krubitzer, Y. M. Chino, A. L. Langston, E. H. Polley, and N. Blair, “Reorganization of retinotopic cortical maps in adult mammals after lesions of the retina,” *Science*, vol. 248, no. 4952, pp. 229–231, 1990.
[44] Notal Vision Inc., “ForeseeHome,” 31-Oct-2015. [Online]. Available: http://www.foreseehome.com. [Accessed: 31-Oct-2015].
[45] C. D. Robison, R. V. Jivrajka, S. R. Bababeygy, W. Fink, A. A. Sadun, and J. Sebag, “Distinguishing wet from dry age-related macular degeneration using three-dimensional computer-automated threshold Amsler grid testing,” *British Journal of Ophthalmology*, vol. 95, no. 10, pp. 1419–1423, Oct. 2011.
[46] A. Loewenstein, A. Pollack, and A. Schachat, “Results of a Multicentered, Masked Clinical Trial to Evaluate the Macular Computerized Psychophysical Test (MCPT) for Detection of Age-related Macular Degeneration (AMD),” *Investigative Ophtalmology and Visual Science*, vol. 43, no. 12, p. 1213, 2002.
[47] K. Nowomiejska *et al.*, “M-charts as a tool for quantifying metamorphopsia in age-related macular degeneration treated with the bevacizumab injections,” *BMC ophthalmology*, vol. 13, no. 1, p. 13, 2013.
[48] Inami Ltd., “Quantitatable Metamorphopsia Chart,” 31-Oct-2015. [Online]. Available: http://www.inami.co.jp/english/surgical_instruments/innovations/kdm1. [Accessed: 31-Oct-2015].
[49] I. Wada *et al.*, “Quantifying metamorphopsia with M-CHARTS in patients with idiopathic macular hole,” *Clinical Ophthalmology*, 20-Sep-2017. [Online]. Available: https://www.dovepress.com/quantifying-metamorphopsia-with-m-charts-in-patients-with-idiopathic-m-peer-reviewed-fulltext-article-OPTH. [Accessed: 16-Apr-2019].
[50] E. Arimura *et al.*, “Correlations between M-CHARTS and PHP findings and subjective perception of metamorphopsia in patients with macular diseases,” *Investigative ophthalmology & visual science*, vol. 52, no. 1, pp. 128–135, 2011.
[51] Y.-Z. Wang, E. Wilson, K. G. Locke, and A. O. Edwards, “Shape discrimination in age-related macular degeneration.,” *Investigative ophthalmology & visual science*, vol. 43, no. 6, pp. 2055–2062, 2002.
[52] J. M. Enoch and R. A. Knowles, *Method for evaluating metamorphopsia*. US4798456, 1989.
[53] E. Wiecek, K. Lashkari, S. Dakin, and P. J. Bex, “Novel Quantitative Assessment of Metamorphopsia in Maculopathy,” *Investigative ophthalmology & visual science*, p. IOVS–14, 2014.
[54] J. L. Stewart, *System and method for full field oscillating stimulus perimeter*. US6742894, 2004.
[55] “Compass - Fundus Automated Perimetry,” *CenterVue*. .
[56] P. P. Nazemi, W. Fink, A. A. Sadun, B. Francis, and D. Minckler, “Early detection of glaucoma by means of a novel 3D computer-automated visual field test,” *British Journal of Ophthalmology*, vol. 91, no. 10, pp. 1331–1336, 2007.
[57] N. Mohaghegh, E. Ghafar-Zadeh, S. Munidasa, and S. Magierowski, “Toward Age-related Macular Degeneration (AMD) Big Data: Hardware and software design and

implementation," in *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2017, pp. 1–4.

[58] K. J. Cocce *et al.*, "Visual Function Metrics in Early and Intermediate Dry Age-related Macular Degeneration for Use as Clinical Trial Endpoints," *American Journal of Ophthalmology*, vol. 189, pp. 127–138, May 2018.

[59] A. G. Bennett and R. B. Rabbetts, "Proposals for new reduced and schematic eyes," *Ophthalmic Physiol Opt*, vol. 9, no. 2, pp. 228–230, Apr. 1989.

[60] A. S. Kitzmann, J. S. Pulido, N. N. Diehl, D. O. Hodge, and J. P. Burke, "The incidence of central serous chorioretinopathy in Olmsted County, Minnesota, 1980-2002," *Ophthalmology*, vol. 115, no. 1, pp. 169–173, Jan. 2008.

[61] M. Crossland and G. Rubin, "The Amsler chart: absence of evidence is not evidence of absence," *British Journal of Ophthalmology*, vol. 91, no. 3, pp. 391–393, Mar. 2007.

[62] E. Y. Chew *et al.*, "Randomized trial of the ForeseeHome monitoring device for early detection of neovascular age-related macular degeneration. The HOme Monitoring of the Eye (HOME) study design — HOME Study report number 1," *Contemporary Clinical Trials*, vol. 37, no. 2, pp. 294–300, 2014.

[63] H. DP, "The foreseehome device and the home study: A milestone in the self-detection of neovascular age-related macular degeneration," *JAMA Ophthalmology*, vol. 132, no. 10, pp. 1167–1168, 2014.

[64] "How the ForeseeHome AMD Monitoring Program Works," *ForeseeHome*. .

[65] M. K. Schmid, L. Faes, L. M. Bachmann, and M. A. Thiel, "Accuracy of a Self-monitoring Test for Identification and Monitoring of Age-related Macular Degeneration: A Diagnostic Case-control Study," *The Open Ophthalmology Journal*, vol. 12, no. 1, pp. 19–28, Mar. 2018.

[66] D. A. Atchison and L. N. Thibos, "Optical models of the human eye," *Clinical and Experimental Optometry*, vol. 99, no. 2, pp. 99–106, Mar. 2016.

[67] L. Thibos, A. Bradley, D. Still, X. Zhang, and P. Howarth, "Theory and measurement of ocular chromatic aberration," *Vision research*, vol. 30, no. 1, pp. 33–49, 1990.

[68] A. N. S. Institute, *American National Standard for Ophthalmics: Methods for Reporting Optical Aberrations of Eyes*. Optical Laboratories Association, 2004.

[69] C. Matsumoto, "Quantification of Metamorphopsia in Patients with Epiretinal Membranes," *Investigative Ophthalmology & Visual Science*, vol. 44, no. 9, pp. 4012–4016, Sep. 2003.

[70] J. D. Foley, *Computer Graphics: Principles and Practice*. Addison-Wesley, 1996.

[71] Adobe Systems Inc., "Adobe Photoshop," 31-Oct-2015. [Online]. Available: http://www.adobe.com/products/photoshopfamily.html. [Accessed: 31-Oct-2015].

[72] Gimp Org., "Gimp," 31-Oct-2015. [Online]. Available: http://www.gimp.org/. [Accessed: 31-Oct-2015].

[73] Adobe Systems Inc., "Adobe Illustrator," 31-Oct-2015. [Online]. Available: http://www.adobe.com/products/illustrator.html. [Accessed: 31-Oct-2015].

[74] Inkscape Org., "Inkscape," 31-Oct-2015. [Online]. Available: http://www.inkscape.org. [Accessed: 31-Oct-2015].

[75] R. R. Everett, C. A. Zraket, and H. D. Benington, "SAGE: A Data-processing System for Air Defense," in *Papers and Discussions Presented at the December 9-13, 1957,*

Eastern Joint Computer Conference: Computers with Deadlines to Meet, New York, NY, USA, 1958, pp. 148–155.
[76] W3 Org., "W3C Standards," 31-Oct-2015. [Online]. Available: http://www.w3.org/standards. [Accessed: 31-Oct-2015].
[77] W3 Org., "W3C SVG," 31-Oct-2015. [Online]. Available: http://www.w3.org/Graphics/SVG. [Accessed: 31-Oct-2015].
[78] Microsoft Corporation, "Microsoft Introduction to SVG," 31-Oct-2015. [Online]. Available: https://msdn.microsoft.com/en-us/library/gg589525%28v=vs.85%29.aspx. [Accessed: 31-Oct-2015].
[79] A. Kaufman, *Rendering, Visualization and Rasterization Hardware*. Springer Science & Business Media, 1993.
[80] "DOM Standard." [Online]. Available: https://dom.spec.whatwg.org/. [Accessed: 12-Sep-2018].
[81] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement.," *Journal of experimental psychology*, vol. 47, no. 6, p. 381, 1954.
[82] A. G. Hauptmann and A. I. Rudnicky, "A comparison of speech and typed input," in *Proceedings of the workshop on Speech and Natural Language - HLT '90*, Hidden Valley, Pennsylvania, 1990, pp. 219–224.
[83] Y. Ji, M. Li, X. Zhang, Y. Peng, and F. Wen, "Poor Sleep Quality Is the Risk Factor for Central Serous Chorioretinopathy," *J Ophthalmol*, vol. 2018, p. 9450297, 2018.
[84] J. C. Besharse and D. Bok, *The Retina and Its Disorders*. Academic Press, 2011.
[85] C. Arndt, O. Rebollo, S. Séguinet, P. Debruyne, and G. Caputo, "Quantification of metamorphopsia in patients with epiretinal membranes before and after surgery," *Graefe's Archive for Clinical and Experimental Ophthalmology*, vol. 245, no. 8, pp. 1123–1129, Jun. 2007.
[86] Q. Yao, Y. Tian, P.-F. Li, L.-L. Tian, Y.-M. Qian, and J.-S. Li, "Design and Development of a Medical Big Data Processing System Based on Hadoop," *J Med Syst*, vol. 39, no. 3, p. 23, Feb. 2015.
[87] J. Kim and W. Lee, "Stochastic Decision Making for Adaptive Crowdsourcing in Medical Big-Data Platforms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 11, pp. 1471–1476, Nov. 2015.
[88] S. Siuly and Y. Zhang, "Medical Big Data: Neurological Diseases Diagnosis Through Medical Data Analysis," *Data Sci. Eng.*, vol. 1, no. 2, pp. 54–64, Jun. 2016.