electronics

Article

Software Development for Processing and Analysis of Data Generated by Human Eye Movements

Radoslava Kraleva * and Velin Kralev

Department of Informatics, Faculty of Mathematics and Natural Sciences, South-West University, 2700 Blagoevgrad, Bulgaria
* Correspondence: rady_kraleva@swu.bg

Abstract: This research focuses on a software application providing opportunities for the processing and analysis of data generated by a saccade sensor with human eye movements. The main functional opportunities of the developed application are presented as well. According to the methodology of the experiments, three experiments were prepared. The first was related to visualization of the stimuli on a stimulation computer display that was integrated into the developed application as a separate module. The second experiment was related to an interactive visualization of the projection of the eye movement of the participants in the experiment onto the stimulation computer display. The third experiment was related to an analysis of aggregated data on the decision time and the number of correct responses given by the participants to visual tasks. The tests showed that the application can be used as a stimulation center to visualize the stimuli and to recreate the experimental sessions. The summary of the results led to the conclusion that the number of correct responses to the visual tasks depended both on the type of motion of the stimuli and on the size of displacement from the center of the aperture.

Keywords: data processing; data analysis; human eye movements; software development

check for updates

Citation: Kraleva, R.; Kralev, V. Software Development for Processing and Analysis of Data Generated by Human Eye Movements. *Electronics* **2023**, *12*, 485. https://doi.org/10.3390/electronics12030485

Academic Editors: Juan M. Corchado, Byung-Gyu Kim, Carlos A. Iglesias, In Lee, Fuji Ren and Rashid Mehmood

Received: 30 November 2022
Revised: 14 January 2023
Accepted: 16 January 2023
Published: 17 January 2023

1. Introduction

In modern scientific research, an essential aspect is the rapid increase in the size of the arrays storing the experimental data, which can be further processed and analyzed at a later stage [1,2]. Different approaches to collecting (summarizing) the experimental data also provide different advantages and disadvantages in the further processing and analysis of the data [3,4]. Furthermore, the different nature of these data, such as their source and their inhomogeneity, makes the task of modeling them even more difficult. Therefore, the use of experimental datasets often requires preprocessing of the raw data, with the aim of easier access to these data, easier sharing of these data between different researchers, as well as easier postprocessing. This preliminary preparation of the data includes the selection of an appropriate format for storing these data, as well as a method of accessing the data in order to more easily access, retrieve, and analyze the aggregated information gathered in one centralized place. After the data have been preprocessed they can be converted and inserted (or imported in bulk) into a purpose-built local or remote database. From this database, information can be aggregated at a later stage and parts of it can be extracted for further analysis. This process leads to a change in the original format of the data, which is necessary to achieve a higher degree of uniformity of the data, so that it is possible to process them in the same way at a later stage. When centralized storage of the experimental data is used and access to it is carried out through a local or global network (e.g., an Ethernet or the internet), specialized data storage and delivery systems are used. These systems are usually database management systems (DBMS) [5–7]. Different architectural models of multi-user systems based on dedicated database servers, which also provide additional possibilities for performing calculations and analysis of experimental data, have been presented in other scientific studies [8–10].

A specific area that is oriented towards the analysis of the human decision-making process is still actively researched [11,12]. Methods based on artificial intelligence are mainly used, which are also widely discussed in various scientific studies [13]. Through these methods, various processes are modeled [14,15], various experimental results are analyzed [16], and various essential characteristics of the participating subjects are investigated [17]. Many aspects of these methods, which are related to specific decision-making tasks based on visual stimuli, but which have only two possible responses, are also discussed in various scientific publications [18,19]. The goal is to study and analyze specific processes accompanying the decision-making process itself in humans [20,21]. Multiple approaches are also explored to determine the influence of decision time on decision accuracy in specifically used visual tasks (visual stimuli) [22,23]. Numerous approaches using neural networks have been developed for this purpose [24,25]. Through them, the generated experimental data are processed, after conducting the relevant sessions with the participants [26], after which the decision-making processes are analyzed [27,28]. All these studies discuss the importance of the data processing methods. Specific data models and technologies are used to access them, such as relational databases and web services, object-relational, and object-oriented [4]. In addition, various technologies and software methods have been used to access these data [29,30], as well as SQL servers for databases and web services [8].

Various aspects and results of a specific study related to decision-making when visualizing visual tasks (stimuli) are presented in detail in [9,13,31]. This research is oriented towards behavioral experiments and was conducted with real participants. The visual task stimuli are visualized on a computer display. The goal is that the participant should determine the direction of movement of the stimulus dots, left or right, respectively, by very quickly moving the gaze in the corresponding direction from the center of the computer display. This center is designated as the primary position in which the participant's gaze should be fixed at the beginning of the experimental session. The gaze shift should be to the left or to the right end zone of the computer display on which the corresponding stimulus is displayed, respectively. At this point, the participant can press a mouse button to instruct the visualization software to display the next stimulus in the series of generated visual stimuli. The purpose of this experiment is to summarize and analyze the data related to the human perception and the movement of the human eyes when visualizing, analyzing, and solving visual tasks such as visual stimuli.

Each experimental session represented a visualization of preset sequences of stimuli. Each stimulus contained 100 frames, and each frame contained 50 points. Depending on the defined coherence of the points, each point changed its coordinates in a different way (i.e., its location relative to the other points). If the stimulus coherence parameter had a value of 100 (in percentages), then all points changed their coordinates synchronously in a certain direction. If this parameter had a value of 0 (also in percentages), then the points changed their coordinates randomly, both on the screen and relative to each other. Another parameter that was adjusted was the displacement of the set of dots relative to the center of the computer display. The values of this parameter varied between 20 and 140 (in pixels) and were distributed exactly in 7 levels (across 20 pixels each). An experimental condition is called a variant and contains a specific set of parameters and values of those parameters. Each variant was repeated exactly 10 times during one experimental session. One experimental session consisted of sequential visualization of 140 stimuli, with a total duration not exceeding 10 min. Each stimulus was actually a recognition task, in the case of the direction of movement of a set of dots on a computer display [31,32].

According to the type of movement of the dots, 4 types of stimuli were used, respectively combined, flicker, motion, and static. The static stimuli contained only 1 frame with 50 points, and the coordinates of these points did not change. In the other types of stimuli, the coordinates of dots were changed according to a predetermined rule, for example, all dots changed their location at the same distance and in a certain direction, or all dots were rotated by a certain angle relative to the center of the screen. In the combined type of stimuli, the movement of the dots depended on the direction of their orientation. In the flicker type of stimuli, the motion of the dots depended on their current position [22]. The goal of the participant was to recognize the direction of motion of the dots, which was implemented in the corresponding visual task stimulus. The information for each stimulus was stored in a binary file with the extension "bin", which had a specific structure and naming convention: the type of movement of the points, type of orientation of the points, distance of displacement of the points from the center of aperture, and variant number. The parameters that were essential for each stimulus were: coherence, direction, and speed of movement of points, number of frames in which one dot was displayed, total number of frames for the corresponding stimulus, coordinates, and the colors of the points in each frame. Stimuli were generated in advance, but presented in random order [31].

The main part of the analyzed data was generated by a specific hardware device—a saccade sensor (Jazz novo eye tracking system, Ober Consulting Sp. Z o.o.). This device stores information about the eye movements of a particular subject at a sampling rate of 1000 Hz. The generated values are initially stored in a file with a specific file structure (and "jazz" extension) from which the data can be exported at a later stage to other files with flat text format [31]. These data may include: aperture center calibration information, eye position information (horizontal and vertical in degrees relative to the viewing angle), a screen sensor signal depending on whether there is currently on the computer display visual stimulus or not, audio signal from a microphone with a frequency of 8000 Hz, as well as additional metadata about the experimental environment [4,22]. Other files (with the extension "dat") store information about the order of presentation of the stimuli for each experimental session, as well as the main characteristics of the corresponding stimuli. These files also store the participants' responses to the respective tasks (stimuli). These files are further processed and the information from them is aggregated, stored, and can be retrieved in various ways presented in [4,8]. These data can be exported from the database to external files in various formats as per user/researcher requirements.

All information from the conducted experiments was stored in a centralized database named *mvsemdb* and contained 7 tables: "Participants", "Stimuli", "Frames", "Dots", "Sessions", "Movements", and "Experiments". More detailed information about the structure of the *mvsemdb* database is presented in [4]. Table 1 shows a brief description of the purpose of each of the tables included in the *mvsemdb* database.

In order to access the data in the *mvsemdb* database, it was necessary that the corresponding user had created a login with at least the privilege to read data (*dbreader*). Since this approach required the creation of multiple logins on the database server for more users, a specialized web service (*mvsemws*) was developed for easier sharing of these data, which is presented in detail in [8]. Web services provide opportunities through different technologies and approaches to reuse the same functionalities or data, which in turn can be sent to calling modules (applications and/or websites) and in different structural formats [33,34]. When using web services, this process is carried out both independently and very efficiently. Therefore, through this approach, functional and easily scalable software systems can be built by integrating multiple applications providing their functionality as web services in the form of web methods [35,36].

The *mvsemws* web service provides five web methods. Each of these methods retrieves (and provides to the caller) a subset of the data in the *mvsemdb* database. This web service is available on the internet at: http://194.141.86.222/mvsemws/MvsemWebService.asmx (accessed on 30 November 2022). The technology for creating this web service (and its web methods), as well as the ways to use it, are presented in detail in [8]. Table 2 shows a brief description of the purpose of each of the web methods provided by the *mvsemws* web service.

Table 1. Tables included in the MVSEMDB database.

Table	Purpose
Participants	This table contained data on the main characteristics of the participants as well as additional information that could be used by an expert analyzing the data.
Stimuli	This table contained data of the main characteristics of the stimuli, such as dot movement direction, dot presentation time, frame duration, coherence level, and others.
Frames	This table contained data for each individual frame of each stimulus. The belonging of each frame to the corresponding stimulus was determined by the relationship between the two tables "Stimuli" and "Frames", which was of type one-to-many.
Dots	This table contained data for all dots from all frames. The type of data stored was the coordinates of the dots. The belonging of a certain dot to the corresponding frame was determined by the specified relationship between the two tables "Frames" and "Dots", which was also of one-to-many type.
Sessions	This table contained data for each conducted session with a participant in the experiment. Associating a particular session with a particular participant was again implemented through a one-to-many relationship between the "Sessions" and "Participants" tables.
Movements	This table contained data about the eye movements of a given participant in the experiment and for a specific session. The relationship between the corresponding values from the "Movements" table and the entries in the "Sessions" table was again defined by a one-to-many relationship between these two tables.
Experiments	This table was an associative table and implemented the many-to-many relationship between the "Sessions" and "Stimuli" tables.

Table 2. Web methods provided by the MVSEM web service.

Web Method	Purpose
GetSubjects	This web method returned a result set of records (in the format specified by the passed parameter) from the "Subjects" table.
GetSessions	This web method returned a result set of records from the "Sessions" table.
GetMovementsBySessions	This web method returned a result set of records containing information from the "Movements" table. In addition to the parameter for the format of the returned data, it was also necessary to indicate which session the requested data was for.
GetMovementsByRange	This web method retrieved information from the "Movements" table by providing the ability to retrieve a specific subset of the stored records in the "Movement" table. The corresponding range of the result set could be determined from the values of the input parameters "Min" and "Max". These parameters were not optional, and therefore the web method required that they be passed before it executed.
GetMaxMovement	This web method returned the maximum record index from the "Movements" table.

More detailed information about the web methods, their input parameters, their speed of execution, as well as the main advantages and disadvantages of their use, is presented in [8].

The subject of the present study is the presentation of a software product developed by the authors (named "SimulationCenter"), which, first, allows processing the heterogeneous information that is used in conducting the experimental sessions, and second, allows recreating the experimental sessions themselves. All data used during the experimental sessions were available (in the original format) and could be retrieved either from the centralized database—*mvsemdb* or from the open access web service—*mvsemws*. Then, these data could be stored in files and processed according to the research objectives. The software that we have developed and present in this article can convert, process, and visualize all types of data used during experimental sessions, and through this functionality it can accurately reproduce each experimental session. The formats used by the software are mainly of three types—plain text, xml, and a specific binary format used by client datasets [9,31]. The main advantage of using the xml format is that it is platform-independent, meaning that it is easily portable while being self-descriptive [37]. Its main disadvantage is that the self-descriptive nature of the data multiplies the size of the xml files [38]. When software stores data in binary form (used by client datasets), the volume of data is significantly smaller compared to that of the xml format, while binary data are loaded (buffered) and processed much faster in memory than data formatted with xml tags [9].

The SimulationCenter application provides the possibility to process different types of "raw" data that are created beforehand (such as the stimulus files) or generated at a later stage during the experimental sessions (such as the files generated by the saccade sensor). Another important function of this application is the ability to convert all types of (heterogeneous) data from one format to another; thus, the application is also a powerful conversion module. This is possible because the software converts all types of data to a common binary format that it can process and convert back to the source format for each of the supported formats. After the application converts, i.e., reorganizes the data at a structural level and merges them, it can also export these data to other applications using the specific file formats that are supported. The most important function of the SimulationCenter application is the ability to recreate each experimental session by visualizing to the user both the stimuli themselves and the eye movements projected on the computer display. Various approaches described in detail in scientific publications were used for the development of the application [39,40]. In addition, various software development technologies and methods have been used, which are also widely discussed in the scientific literature [41,42]. From the point of view of software engineering, the creation of such software that provides functionalities for collecting, processing, summarizing, and analyzing experimental data, and such data collected from different and heterogeneous sources, is a serious problem [43,44]. The purpose of the present work was to test and analyze the main functions of the SimulationCenter application, as well as verify its effectiveness and functionality by conducting experimental tests, and then present the results of these tests.

2. Materials and Methods

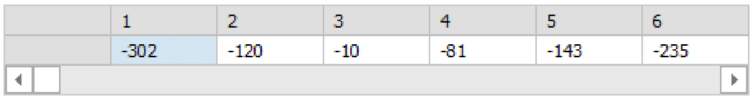
In order to recreate the experimental sessions, the SimulationCenter application first needs to process and convert 3 different types of files from their original "raw" form to a form ready to be used by the application itself. These are the stimulus files, the files storing the audio signal from the microphone, and the files storing the eye movement information. We will also note that the processed audio files are merged at a later stage with the eye movement files. This is necessary to synchronize the two files, since the sample rate of the eye movement file is 1000 Hz, and the audio signal that the microphone recorded is 8000 Hz. In order to synchronize the data in a common file, it is necessary to match each sample from the eye movement file with eight samples from the audio file that was recorded by the microphone. In this section, we will introduce the steps to convert all these files. We will introduce the conversion of the stimulus files first.

The coordinates of all 5000 points of all 100 frames in one stimulus are stored in a binary file. These files have a specific file structure. This requires additional processing to extract the necessary information from these files. This information contains the coordinates of certain points distributed in certain frames (for each stimulus file). Table 3 shows fragments of a stimulus file named: cl201.bin.

Table 3. Fragments of a stimulus file cl201.bin.

Byte	**0**	**2**	. . .	**9998**	**10,000**	**10,002**	. . .	**19,998**	**20,000**	. . .	**20,100**
Address (0×)	0000	0002	...	270E	2710	2712	...	4E1E	4E20	...	4E84
Hexadecimal	FED2	FF88	...	0012	00F7	00BE	...	01BB	00	...	00
16-bit Integer	−302	−120	...	18	247	190	...	443	0	...	0
Frame	F1	F1	...	F100	F1	F1	...	F100	F1	...	F100
Meaning	P1.X	P2.X	...	P50.X	P1.Y	P2.Y	...	P50.Y	Color	...	Color

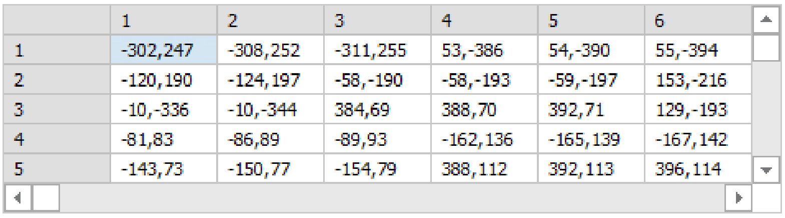
The following additional abbreviations are used in Table 3: F—frame, P—point, X—X coordinate, Y—Y coordinate. Each stimulus file is 20,100 bytes in size. The last 100 bytes contain color information, one value for each of the 100 frames. The values for the colors are respectively 0—white, 1—black, 2—combined. The remaining 20,000 bytes store information about the X and Y coordinates at 5000 points (distributed at 50 points in 100 frames). The X and Y coordinates at each point are written in two 16-bit integer values, which correspond exactly to 2 bytes. Since the points are 5000, and for each of them both the X coordinate and the Y coordinate are stored, the total number of values that need to be stored is 5000 × 2 = 10,000. Since each value is written in a 16-bit number (2 bytes), all 10000 values will require exactly 20,000 bytes. Table 1 also shows the hexadecimal equivalents of the coordinates of points 1 and 2 of frame 1, as well as point 50 of frame 100. One specific feature of the stimulus files is that in the first 10,000 bytes only the X coordinates of all 5000 points are stored, and in the next 10,000 bytes only the Y coordinates of all points are stored. In this way, the X coordinate of point 1 of frame 1 is "extracted" from bytes 0–1, respectively, and the Y coordinate from bytes 10,000–10,001. Similarly, the X coordinate at point 50 of frame 100 is "extracted" from bytes 9998–9999, respectively, and the Y coordinate from bytes 19,998 and 19,999, respectively. This way of storing the information about the coordinates of the points in the stimulus files is inconvenient to process (Figure 1).



	1	2	3	4	5	6
	-302	-120	-10	-81	-143	-235

Figure 1. One-dimensional array—points (for stimulus cl201.bin).

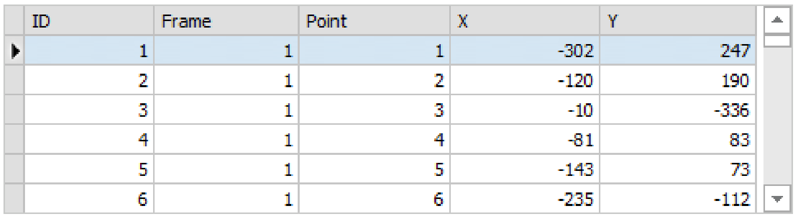
Therefore, in the presented ordinance of the coordinates of the points in a stimulus file, the software developed by us, first, converts into a two-dimensional matrix—PF (short for points by frames), which has a dimension of 50 rows per 100 columns (Figure 2). Each cell of this matrix contains one TPoint element, which is a structure (packed record) containing two integer values, one for the X coordinates and one for the Y coordinate of a given point. Each column of the matrix corresponds to exactly one frame of a given stimulus, and each row contains the X and Y coordinates of each of a total of 50 points for the specified frame.



	1	2	3	4	5	6
1	-302,247	-308,252	-311,255	53,-386	54,-390	55,-394
2	-120,190	-124,197	-58,-190	-58,-193	-59,-197	153,-216
3	-10,-336	-10,-344	384,69	388,70	392,71	129,-193
4	-81,83	-86,89	-89,93	-162,136	-165,139	-167,142
5	-143,73	-150,77	-154,79	388,112	392,113	396,114

Figure 2. Two-dimensional matrix—points by frames (for stimulus cl201.bin).

The PF matrix is transformative and is in fact an intermediate structure that stores the data on the coordinates of the points per frame (for a given stimulus) in a more convenient form for processing. However, for further processing of the information from the stimulus files, we converted a two-dimensional PF matrix into the "stimulus" relation, with the following analytical representation: stimulus = {ID, frame, point, Xcoord, Ycoord}, where the primary key ID contains unique values in the range from 1 to 5000 and corresponds to the index of a particular point of a frame in a particular stimulus. The values of the frame attribute change in the interval [1, 2, . . . , 100], as this is the total number of frames in each stimulus. The values of the point attribute change in the interval [1, 2, . . . , 50] and correspond to the point number in the current frame. The values X and Y, respectively, represent the coordinates of the current point of the current frame. Figure 3 shows the "stimulus" relation after transformation of the PF matrix (for stimulus cl201.bin).



ID	Frame	Point	X	Y
1	1	1	-302	247
2	1	2	-120	190
3	1	3	-10	-336
4	1	4	-81	83
5	1	5	-143	73
6	1	6	-235	-112

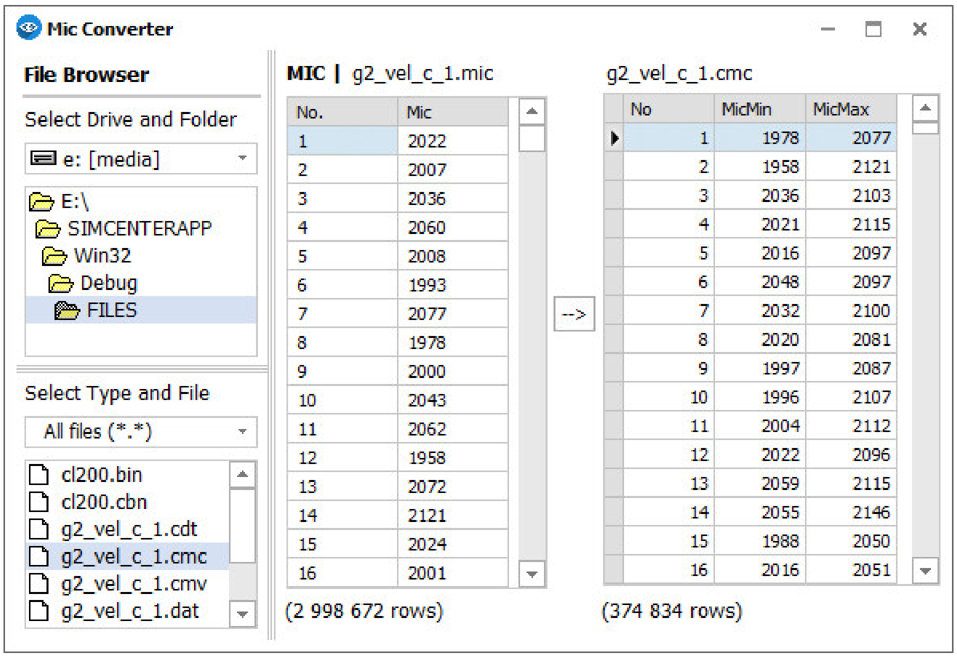
Figure 3. The "stimulus" relation after transformation of the PF matrix.

The ID attribute (which is the primary key) in the "stimulus" relation can be removed and the combination of the values of the frame and point attributes can be used instead. This combination of attribute values, as well as the ID attribute, fulfills the two conditions that must be met by a primary key in a relationship [6]: (1) uniqueness—to ensure the uniqueness of each value of a given attribute (or combination of attribute values) and (2) minimality—neither of the two frame nor point attributes can be removed without violating the uniqueness condition. Removing either of the two frame or point attributes will duplicate the values in the other attribute, because for each point in a frame, the frame number is repeated exactly 50 times. On the other hand, for each frame (a total of 100 for each stimulus) the numbers of points from 1 to 50 are repeated. However, the combination of frame number and point number (frame, point) is unique for each of the states of the stimulus relation. The ID attribute will be used to more quickly index the points and frames in the dataset that stores the information for a particular stimulus. The values of this attribute guarantee the uniqueness of each point of each frame in each stimulus. From each value of the ID attribute, the corresponding frame number and the corresponding point number can be calculated. Since each frame is known to contain exactly 50 points, then for each ID (in the range from 1 to 5000) the values of the variables PointIndex and FrameIndex can be calculated as follows:

Remainder = ID mod 50; if (Remainder = 0): PointIndex = 50; FrameIndex = ID div 50; if (Remainder > 0): PointIndex = Remainder; FrameIndex = (ID div 50) + 1;

The data organized in this way, containing the distribution of points by frames and the coordinates of these points, provide significant advantages in searching and filtering fragments of a stimulus or unidirectional extraction of coordinates of points distributed in frames when displaying a stimulus on the screen of the computer.

We will present the initial processing and conversion of the audio signal that is recorded by the microphone. All audio signal values that are stored by the microphone are saved in files with the extension "mic". Since these files have a specific structure, they need to be further processed. The microphone records the sound signal with a frequency of 8000 Hz, which means that for 1 millisecond the microphone stores 8 values. However, these 8 values should correspond to only 1 value, which is stored by the saccade sensor, which, however, works with a sampling rate of 1000 Hz. This requires additional processing of the audio signal and its synchronization with the signal stored by the saccade sensor, which corresponds to the eye movement. For this purpose, an additional module called MicConverter was developed and integrated into the SimulationCenter application, which allows processing, converting, and synchronizing the microphone signal with the signal from the saccade sensor. An example session of working with the MicConverter module is shown in Figure 4.


Mic Converter

File Browser

Select Drive and Folder

e: [media]

E:\
SIMCENTERAPP
Win32
Debug
FILES

Select Type and File

All files (*.*)

cl200.bin
cl200.cbn
g2_vel_c_1.cdt
g2_vel_c_1.cmc
g2_vel_c_1.cmv
g2_vel_c_1.dat

MIC | g2_vel_c_1.mic

No.	Mic
1	2022
2	2007
3	2036
4	2060
5	2008
6	1993
7	2077
8	1978
9	2000
10	2043
11	2062
12	1958
13	2072
14	2121
15	2024
16	2001

(2 998 672 rows)

-->

g2_vel_c_1.cmc

No	MicMin	MicMax
1	1978	2077
2	1958	2121
3	2036	2103
4	2021	2115
5	2016	2097
6	2048	2097
7	2032	2100
8	2020	2081
9	1997	2087
10	1996	2107
11	2004	2112
12	2022	2096
13	2059	2115
14	2055	2146
15	1988	2050
16	2016	2051

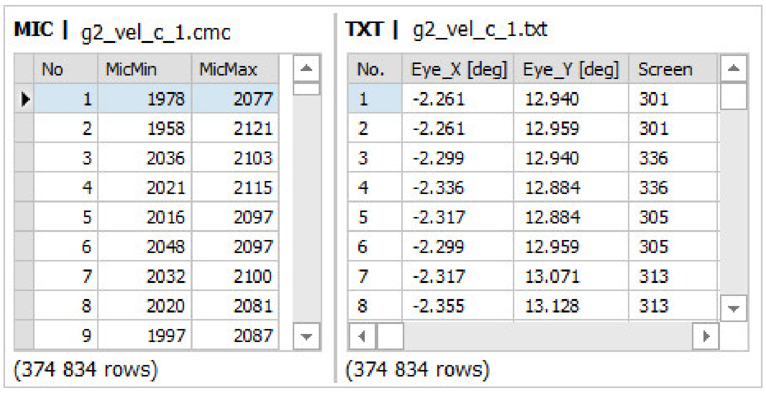
(374 834 rows)

Figure 4. A session of working with the MicConverter module.

The MicConveter module converts the original binary format of the audio signal that was recorded by the microphone, then converts it to a pair of integer values, respectively "sample number" and "audio value for this sample (in decibels)". In order to synchronize the two signals in duration, the first audio signal stored by the microphone and the second stored by the saccade sensor, it is necessary to match exactly 8 values of the audio signal for every value of the saccade sensor signal. To realize this, the MicConverter module divides the audio signal into segments, each segment containing exactly 8 values. These values (in decibels) overlap each other, thereby "merging" into a single discrete that has a corresponding minimum value (this is the lowest discrete value of the 8 considered) and maximum value (this is respectively the highest value of a discrete of the same 8 values in the corresponding segment). Thus, the processed microphone signal, containing an array of discrete signal value pairs, is actually an intermediate transform structure that the MicConverter module will use to generate the synchronized (reduced exactly 8 times) and converted audio signal.

For more convenient processing of the information from the audio signal (recorded by the microphone) and its synchronization with the signal recorded by the saccade sensor, the array containing the pairs of values "discrete", "signal" in the relation "MicSignal" was converted. This relation has the following analytical representation: MicSignal = {No, MicMin, MicMax}, where No is the primary key and contains unique values that are identical to the index values of the values generated by the saccade sensor. The MicMin and MicMax values respectively contain the minimum and maximum value of the audio signal (in decibels) recorded by the microphone for the corresponding segment. Figure 4 shows that the first 8 samples of the microphone signal form a single entry (with index 1) in the MicSignal relation. The corresponding minimum value in this 8-element segment is at a discrete with index 8, and the maximum value is at a discrete with index 7. From the next 8 values of the microphone signal, i.e., those contained in segment 2, a single entry (with index 2) is also formed in the MicSignal relation. In this segment, the corresponding minimum value is at a discrete with index 12, and the maximum value is at a discrete with index 14. It is observed that the original microphone signal contains exactly 2,998,672 values, and the converted signal contains 374,834 values, which is exactly 8 times less. This number of values will also uniquely and precisely match the number of values stored by the saccade sensor that is associated with eye movement information for the same experimental session. In this way, the data from the saccade sensor can be easily synchronized with the data from the audio signal that are recorded by the microphone. The audio signal is used for background noise analysis.

The process of merging the signal recorded by the saccade sensor (stores the eye movement information) and the signal recorded by the microphone (but already processed and converted) can be performed. For each session, the eye movement values generated by the saccade sensor are stored, as well as additional values generated by the application associated with the saccade sensor. The stored file contains for every 1 millisecond the values for the gaze deviation (i.e., their projection on the computer display), both the X coordinate and also the Y coordinate. In addition to these values, the index of the current discrete is also generated and stored, as well as an additional parameter indicating whether a stimulus is currently being visualized on the computer display or not. The eye movement values are calculated by the saccade sensor and represent the projection of the participant's gaze onto the computer display at a certain moment. For every 1 millisecond (of the corresponding experimental session), the saccade sensor generates one pair of such values, and the application associated with the saccade sensor adds the other two automatically. The data are then exported from this application to a flat text file. Since an experimental session can last 10 min or more, the amount of the data that will be generated can be very large. For example, with a session duration of 10 min (which is equal to 600 s or 600,000 milliseconds), the amount of data that must be stored after converting the text file to structures with certain data types will be 600,000 multiplied by the number of bytes to store each value from the 4 generated ones, and for each one discrete. We note that for each set of 4 values generated by the saccade sensor and its associated application (for each sample), it is necessary to add the values from the microphone signal for the same sample (MicMin and MicMax). Since these signals are already synchronized, they can easily be combined into one, which is shown in Figure 5.



MIC | g2_vel_c_1.cmc

No	MicMin	MicMax
1	1978	2077
2	1958	2121
3	2036	2103
4	2021	2115
5	2016	2097
6	2048	2097
7	2032	2100
8	2020	2081
9	1997	2087

(374 834 rows)

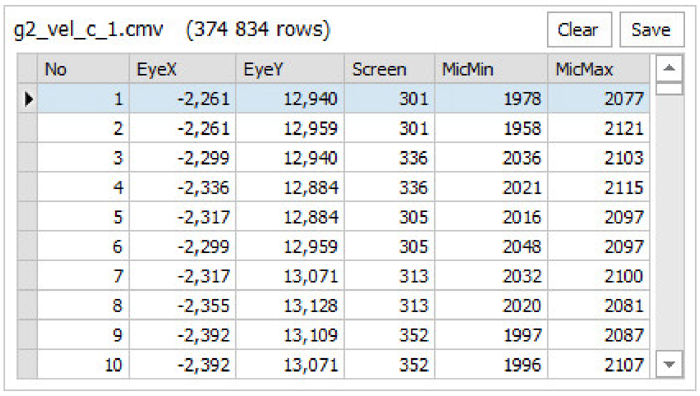
TXT | g2_vel_c_1.txt

No.	Eye_X [deg]	Eye_Y [deg]	Screen
1	-2.261	12.940	301
2	-2.261	12.959	301
3	-2.299	12.940	336
4	-2.336	12.884	336
5	-2.317	12.884	305
6	-2.299	12.959	305
7	-2.317	13.071	313
8	-2.355	13.128	313

(374 834 rows)

Figure 5. Combining the signals from the microphone and from the saccade sensor.

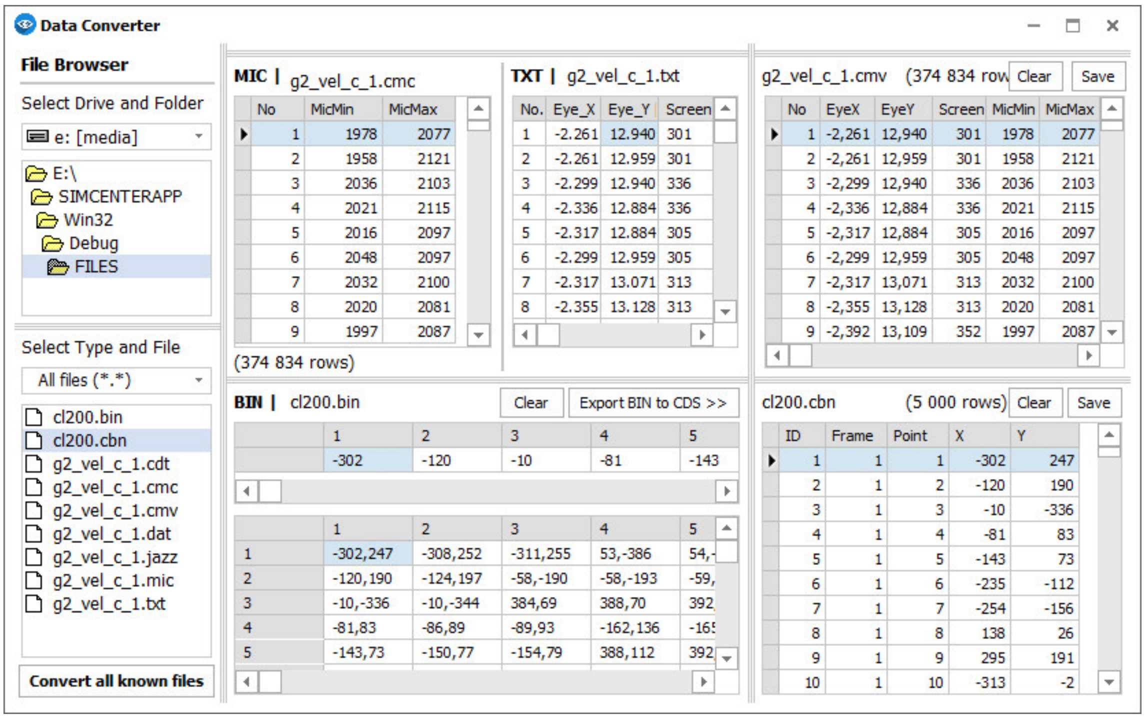
Handling the data from two structures that are different in data type, one of which contains string values, is inconvenient. It is necessary to convert the eye movement information data, which are currently of text type, to the corresponding base (structural) types. This conversion is done in parallel with the process of merging the eye movement data and the pairs of MicMin and MicMax values, which correspond to the minimum and maximum value of the audio signal for the corresponding segment, but extracted from the text file. The eye movement data are initially loaded and buffered into an array of strings whose elements are record type variables that hold 5 different values each. These values correspond to each field that is exported to the text file that stores the eye movement information. To join the two sets of records, the relation Movments = {No, EyeX, EyeY, Screen, MicMin, MicMax} was created. The first 4 fields correspond to the columns of the same name from a text file, and the last 2 fields correspond to the minimum and maximum value of the microphone signal for the current sample. In this relation, the "No" attribute actually represents a primary key and uniquely identifies each row of the formed record structure. The result of converting the text file containing the eye movement data and merging it with the microphone signal data is presented in Figure 6.


g2_vel_c_1.cmv (374 834 rows)

No	EyeX	EyeY	Screen	MicMin	MicMax
1	-2,261	12,940	301	1978	2077
2	-2,261	12,959	301	1958	2121
3	-2,299	12,940	336	2036	2103
4	-2,336	12,884	336	2021	2115
5	-2,317	12,884	305	2016	2097
6	-2,299	12,959	305	2048	2097
7	-2,317	13,071	313	2032	2100
8	-2,355	13,128	313	2020	2081
9	-2,392	13,109	352	1997	2087
10	-2,392	13,071	352	1996	2107

Figure 6. Converted and combined data generated by the microphone and saccade sensor.

Figure 7 shows a working session with the data converter unit of the SimulationCenter application.



MIC | g2_vel_c_1.cmc

No	MicMin	MicMax
1	1978	2077
2	1958	2121
3	2036	2103
4	2021	2115
5	2016	2097
6	2048	2097
7	2032	2100
8	2020	2081
9	1997	2087

(374 834 rows)

TXT | g2_vel_c_1.txt

No.	Eye_X	Eye_Y	Screen
1	-2.261	12.940	301
2	-2.261	12.959	301
3	-2.299	12.940	336
4	-2.336	12.884	336
5	-2.317	12.884	305
6	-2.299	12.959	305
7	-2.317	13.071	313
8	-2.355	13.128	313

g2_vel_c_1.cmv (374 834 row

No	EyeX	EyeY	Screen	MicMin	MicMax
1	-2,261	12,940	301	1978	2077
2	-2,261	12,959	301	1958	2121
3	-2,299	12,940	336	2036	2103
4	-2,336	12,884	336	2021	2115
5	-2,317	12,884	305	2016	2097
6	-2,299	12,959	305	2048	2097
7	-2,317	13,071	313	2032	2100
8	-2,355	13,128	313	2020	2081
9	-2,392	13,109	352	1997	2087

BIN | cl200.bin

	1	2	3	4	5
	-302	-120	-10	-81	-143

	1	2	3	4	5
1	-302,247	-308,252	-311,255	53,-386	54,-
2	-120,190	-124,197	-58,-190	-58,-193	-59,
3	-10,-336	-10,-344	384,69	388,70	392
4	-81,83	-86,89	-89,93	-162,136	-16
5	-143,73	-150,77	-154,79	388,112	392

cl200.cbn (5 000 rows)

ID	Frame	Point	X	Y
1	1	1	-302	247
2	1	2	-120	190
3	1	3	-10	-336
4	1	4	-81	83
5	1	5	-143	73
6	1	6	-235	-112
7	1	7	-254	-156
8	1	8	138	26
9	1	9	295	191
10	1	10	-313	-2

Figure 7. Working session with the data converter unit of the SimulationCenter application.

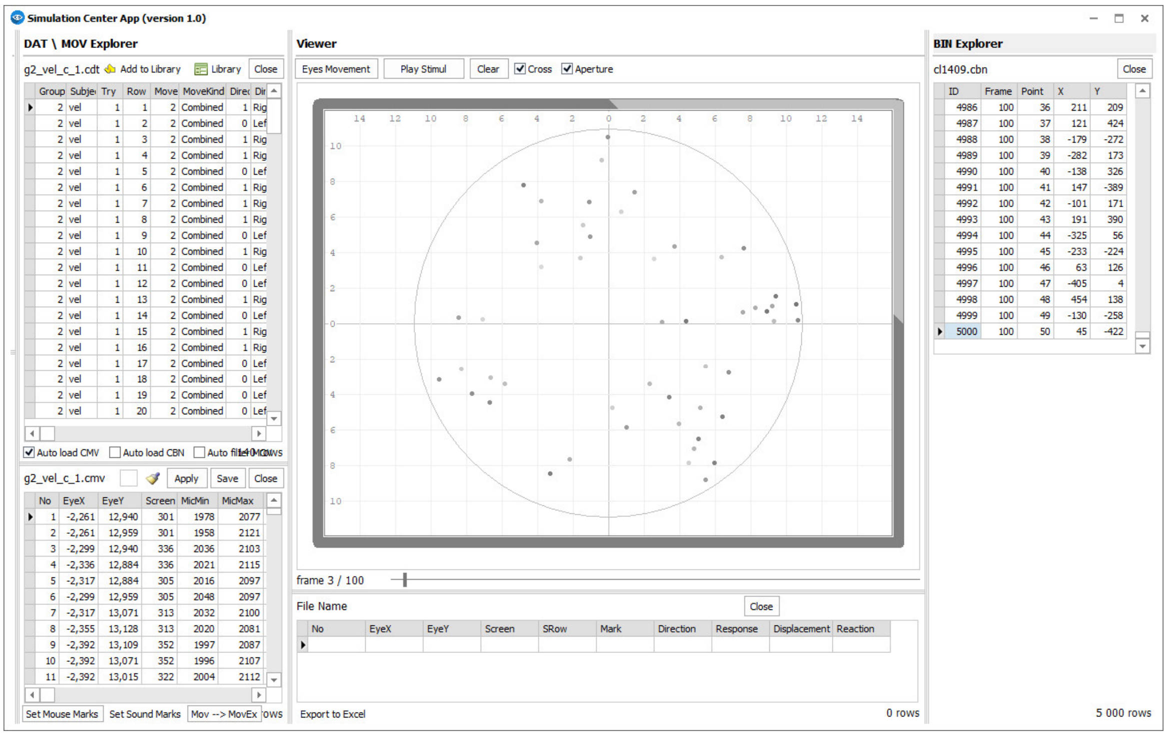
The data organized in this way already contain information about the index of the corresponding discrete, about the coordinates of the projection of the participant's averted gaze on the computer display, about the visualization or not of a stimulus at the current moment, as well as about the minimum and maximum values generated by the microphone for the corresponding segment. This data format already provides significant advantages when playing back and analyzing complete experimental sessions, or when working with partial fragments of them.

It is important to note that when the data are of a character type, mathematical calculations cannot be performed with these data. Apart from the stimulus binaries, all other files, i.e., those generated by the saccade sensor and the associated software, are of a character type. Therefore, data preprocessing and conversion is necessary, as this will enable the data to be converted to the appropriate base types (e.g., integer, word, double, etc.). After converting the data to the corresponding base types, they can be analyzed and visualized in a convenient way.

3. Results

The subject of this research is the SimulationCenter application and its functionalities. This application was run on a 32-bit Win 10 OS with the following hardware configuration: AMD (R) Ryzen (TM) 3 5300U (up to 3.85 GHz, 4 MB cache, 4 core) processor, 8 GB RAM, and AMD Radeon (TM) Desktop Graphics Card.

Figure 8 shows a session with the main unit of the SimulationCenter application. The main module of this application provides capabilities for visualization of various types of data in an interactive mode. After selecting a stimulus file, it can be visualized frame by frame on a simulation display. This display has the same aspect ratio as the original computer display on which the real stimuli are visualized. This module also provides options for selecting a specific experimental session that can be interactively played within the application.



Group	Subje	Try	Row	Move	MoveKind	Direc	Dir
2	vel	1	1	2	Combined	1	Rig
2	vel	1	2	2	Combined	0	Lef
2	vel	1	3	2	Combined	1	Rig
2	vel	1	4	2	Combined	1	Rig
2	vel	1	5	2	Combined	0	Lef
2	vel	1	6	2	Combined	1	Rig
2	vel	1	7	2	Combined	1	Rig
2	vel	1	8	2	Combined	1	Rig
2	vel	1	9	2	Combined	0	Lef
2	vel	1	10	2	Combined	1	Rig
2	vel	1	11	2	Combined	0	Lef
2	vel	1	12	2	Combined	0	Lef
2	vel	1	13	2	Combined	1	Rig
2	vel	1	14	2	Combined	0	Lef
2	vel	1	15	2	Combined	1	Rig
2	vel	1	16	2	Combined	1	Rig
2	vel	1	17	2	Combined	0	Lef
2	vel	1	18	2	Combined	0	Lef
2	vel	1	19	2	Combined	0	Lef
2	vel	1	20	2	Combined	0	Lef

No	EyeX	EyeY	Screen	MicMin	MicMax
1	-2,261	12,940	301	1978	2077
2	-2,261	12,959	301	1958	2121
3	-2,299	12,940	336	2036	2103
4	-2,336	12,884	336	2021	2115
5	-2,317	12,884	305	2016	2097
6	-2,299	12,959	305	2048	2097
7	-2,317	13,071	313	2032	2100
8	-2,355	13,128	313	2020	2081
9	-2,392	13,109	352	1997	2087
10	-2,392	13,071	352	1996	2107
11	-2,392	13,015	322	2004	2112

No	EyeX	EyeY	Screen	SRow	Mark	Direction	Response	Displacement	Reaction

ID	Frame	Point	X	Y
4986	100	36	211	209
4987	100	37	121	424
4988	100	38	-179	-272
4989	100	39	-282	173
4990	100	40	-138	326
4991	100	41	147	-389
4992	100	42	-101	171
4993	100	43	191	390
4994	100	44	-325	56
4995	100	45	-233	-224
4996	100	46	63	126
4997	100	47	-405	4
4998	100	48	454	138
4999	100	49	-130	-258
5000	100	50	45	-422

Figure 8. Working session with the main unit of the SimulationCenter application.

The first experiment with the SimulationCenter application that was done was to interactively visualize six stimuli using the built-in support for interactive visualization of stimuli. The application shows what the participant in the experiment saw during the experimental session itself. Interactive visualizations of these stimuli generated by the SimulationCenter application are shown in Figure 9a–f.

The difference is in the place where the stimulus will be visualized. In the real experiment, this is the computer display in front of the participant himself, but in the present test, it was the computer display on which the main window of the SimulationCenter application was displayed. Six files were selected, respectively divided into pairs for each of the combined, flicker, and motion types. For each of the stimulus dot movement types, one leftward and one rightward dot movement were selected, respectively. The stimuli were selected so that their dots had the largest displacement from the center of the aperture, i.e., 140 pixels. Variant 9 was selected for all stimuli.

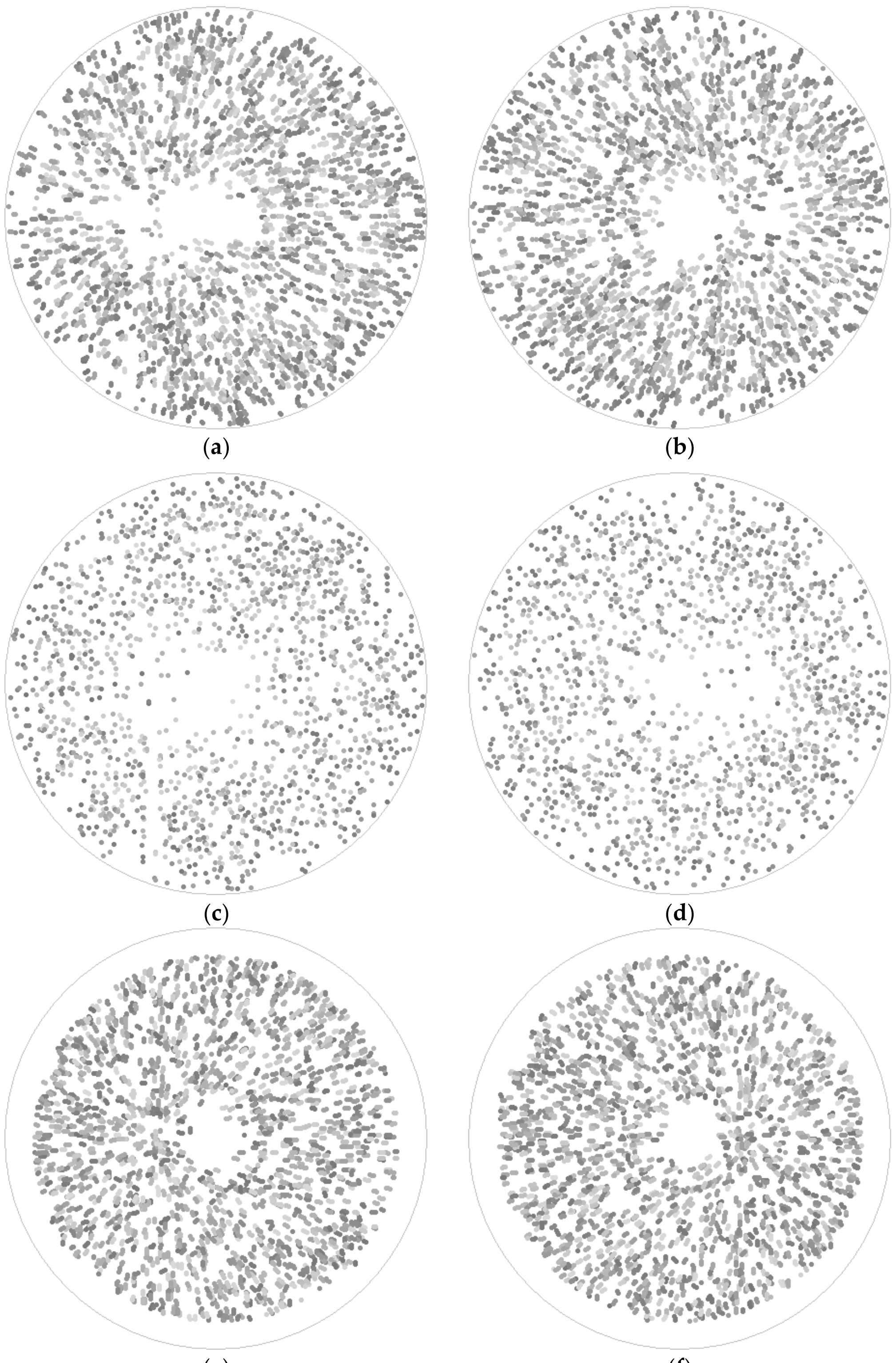


Figure 9. Stimuli (**a**) cl1409 (combined, left), (**b**) cr1409 (combined, right), (**c**) fl1409 (flicker, left), (**d**) fr1409 (flicker, right), (**e**) ml1409 (motion, left), (**f**) mr1409 (motion, right).

We note that after conducting this experiment, it was found that visualizing the stimuli frame by frame, although recreating what the participant in the experiment saw, could not create the sensation of movement of the set of dots. In fact, when previewing the stimuli, the application shows the frames one after the other but does not erase them. This way, the images shown in Figure 9a–f were generated. This way of visualizing the stimuli shows how the set of points forms the displacement from the center of the aperture to the left or right depending on the currently visualized stimulus and the type of motion that is selected, respectively combined, motion or flicker.

The second experiment that was done with the SimulationCenter application was to test its functionality in terms of the possibility to reproduce, interactively, the eye movement process on the computer display by a participant in the experiment during the visualization of the stimuli. This can be recreated and visualized to the user/analyst using the application. Figure 10 shows the eye movement visualization for an entire experimental session.

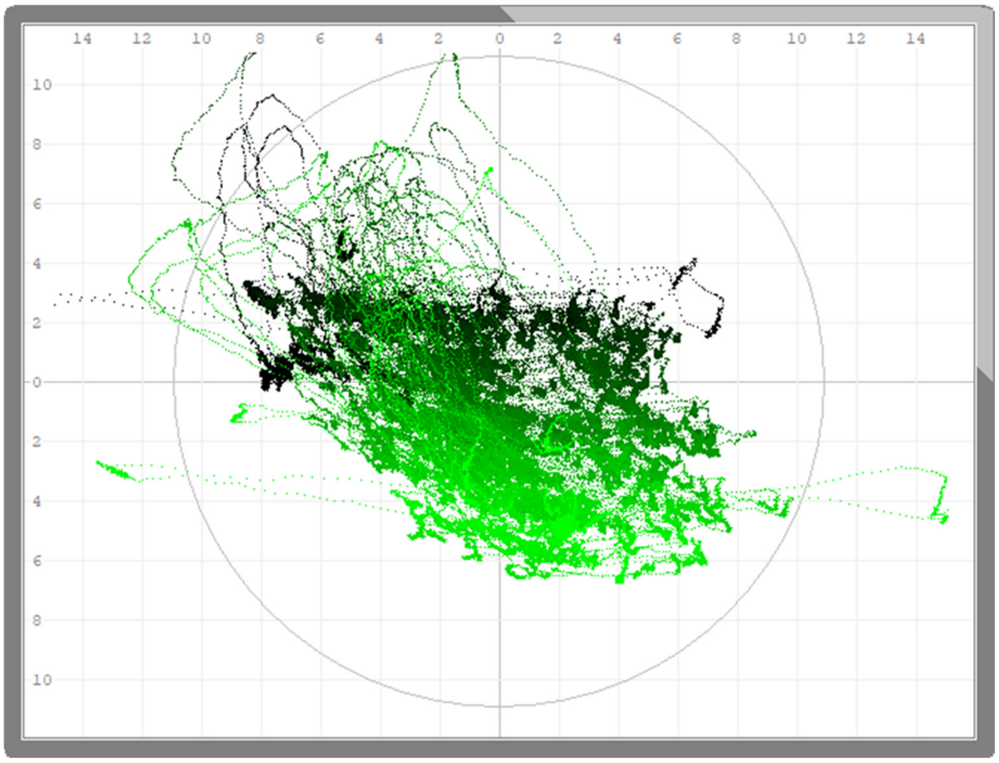


Figure 10. Working session with the main unit of the SimulationCenter application, showing the eye movements on the computer display of a participant during an entire experimental session.

In order that the user/analyst determines how the participant's eye movements were made over time, the application changes the color saturation of the graph by which it visualizes the gaze projection in the simulated computer display. Less saturated colors indicate projection coordinates in the earlier phases of the experimental session, and the darker tones indicate later moments of this process. We should also note that the user can simultaneously visualize both the eye movement and the corresponding stimulus.

The application provides functions of a group and summarizes data in an easy way. This provides an opportunity for these data to be more easily analyzed. In order to test this capability of the application, an analysis of the obtained results of the experiments was made from the point of view of the correct and incorrect answers given by the participants to the visual tasks. These visual tasks were based on stimuli for which two key parameters were defined, respectively, first according to the type of movement of the stimuli: combined, flicker, motion, and static, and second according to the size of displacement [4,8,9]. The summary data are presented in Tables 4 and 5.

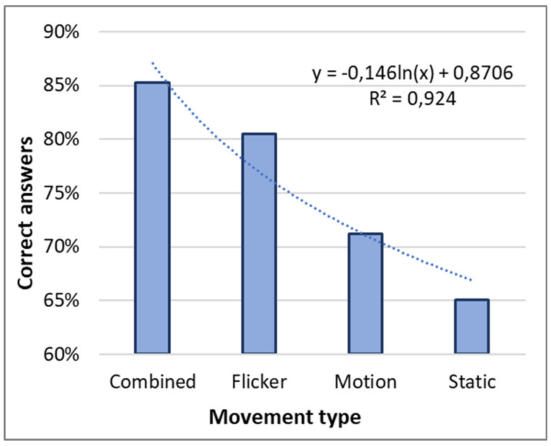
Table 4. Summarized data from the experimental results according to the type of movement.

Movement Type	Reaction Time		Stimulus	Correct Answers		Incorrect Answers	
	(ms)	(h, min)	(Count)	(Count)	(%)	(Count)	(%)
Combined	14,172,562	3 h 56 min	9768	8330	85.28%	1438	14.72%
Flicker	13,523,076	3 h 45 min	9076	7304	80.48%	1772	19.52%
Motion	16,479,249	4 h 34 min	9781	6963	71.19%	2818	28.81%
Static	13,669,203	3 h 47 min	9634	6266	65.04%	3368	34.96%
Summary	57,844,090	16 h 4 min	38,259	28,863	75.44%	9396	24.56%

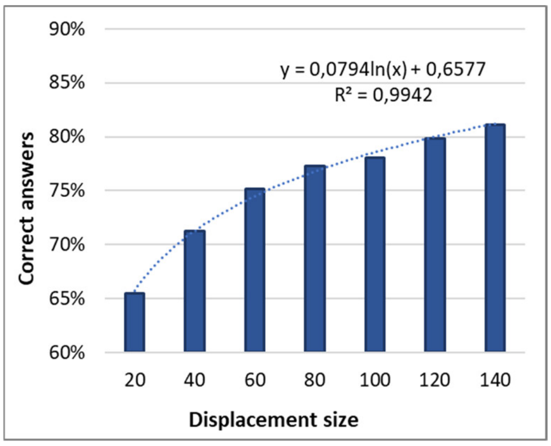
Table 5. Summarized data from the experimental results according to the size of displacement.

Displacement Size	Reaction Time		Stimulus	Correct Answers		Incorrect Answers	
	(ms)	(h, min)	(Count)	(Count)	(%)	(Count)	(%)
20	8,636,580	2 h 23 min	5472	3584	65.50%	1888	34.50%
40	8,568,584	2 h 22 min	5465	3892	71.22%	1573	28.78%
60	8,327,277	2 h 18 min	5470	4109	75.12%	1361	24.88%
80	8,226,506	2 h 17 min	5465	4223	77.27%	1242	22.73%
100	8,123,963	2 h 15 min	5462	4261	78.01%	1201	21.99%
120	8,039,586	2 h 13 min	5465	4365	79.87%	1100	20.13%
140	7,921,594	2 h 12 min	5460	4429	81.12%	1031	18.88%
Summary	57,844,090	16 h 4 min	38,259	28,863	75.44%	9396	24.56%

These data were analyzed with the goal of examining how the type of stimulus motion (i.e., combined, flicker, motion, and static) affects participants' responses and how the size of the displacement affects the responses of the visual tasks but is summarized to all participants in the experiment. The graphs of these results are shown in Figure 11a,b.



(**a**)



(**b**)

Figure 11. (**a**) Influence of the type of movement of the stimuli on the number (percentage) of correctly solved visual tasks by all participants in the experiment. (**b**) Influence of the size of displacement from the center of the set of dots on the number (percentage) of correctly solved visual tasks by all participants in the experiment.

Tables 4 and 5 and the diagrams in Figure 11a,b show that the type of movement of the stimuli has an impact on the number (percentage) of correctly solved visual tasks, with the best results obtained in the type of movement of the combined and flicker stimuli. In the motion type of the stimuli, given the complex dynamics of changing the locations and directions of the individual dots in the frames, it can also explain the difficulties that the participants had when making a decision in this type of motion. With the type of movement of the stimuli static, we note that the worst results were obtained, i.e., with this kind of stimulus movement, the information that the experimenters receive is as little as possible, since the dots are static and their positions do not change during visualization of the stimulus. On the other hand, the lack of dynamics of the points also leads to more difficult decision-making by the participants. Regarding the size of the displacement of the center of the set of dots from the center of the aperture, we can note that a tendency also stands out and it is the following: the number of correctly solved visual tasks depends on the size of the displacement. This is because the larger the displacement of the center of the set of dots from the center of the aperture, the more noticeable it is, and thus the participant makes a decision more easily. Because of this, the number of correctly recognized stimulus directions by participants (i.e., the number of correctly solved visual tasks) increased as the amount of displacement from the center of the set of dots relative to the center of the aperture increased. A detailed statistical analysis of the results is presented in [9,13,31].

We will now perform another analysis on the same data, examining how the type of stimulus motion (i.e., combined, flicker, motion, and static) affects the participants' decision times, and secondly how the displacement of the center of the cluster of points from the center of the aperture affected the decision time (summed) for all participants in the experiment. The graphs of these results are shown in Figure 12a,b.

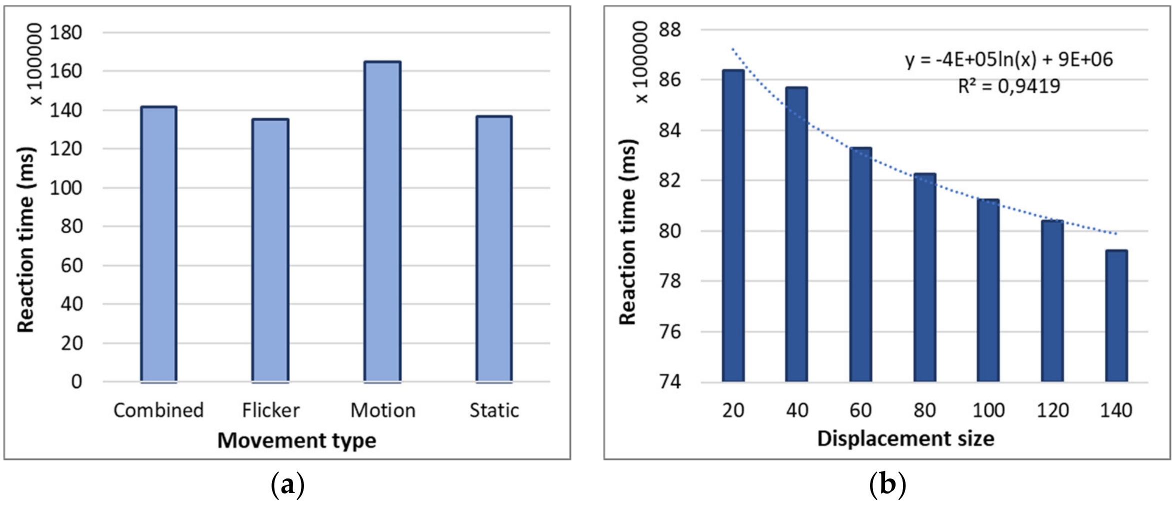


Figure 12. (**a**) Influence of the type of motion of the stimuli on the total decision time of the participants in the experiment. (**b**) Influence of the size of displacement from the center of the set of dots on the total decision time of the participants in the experiment.

Tables 4 and 5 and the diagrams in Figure 12a,b show that for the motion type of the stimuli, there was no clear trend in terms of the participants' decision times. We note that in the combined flicker and static movements, the total decision-making time was comparable for all three types, which varied in the range from 3 h 45 min to 3 h 56 min. In contrast to these movements, with the movement of the motion stimuli, the reaction time (decision time by the participants) was significantly longer: it was 4 h 34 min, which was generalized for all experimental sessions. We can summarize that with motion stimuli, in addition to the longest decision-making time, at the same time the number of correctly recognized stimuli was significantly lower. Regarding the size of the displacement of the set of dots from the center of the aperture and the time to make a decision, we note that there was a clearly marked trend and it was the following: the time to make the decisions depended inversely on the size of the displacement of the set of dots from the center of the aperture. This can be explained by the fact that the greater the displacement of the set of dots from the center of the aperture, the more noticeable this displacement was and the corresponding decision could be made by the participant faster. Furthermore, as noted in the analysis of Figure 11b, this displacement also resulted in a greater number of correctly solved visual tasks. We can summarize that the number (percentage) of correctly solved visual tasks was a function of the size of the displacement of the set of dots from the center of the aperture. Although the decision time decreased as the size of the displacement increased, the number of correct answers maintained an increasing trend.

4. Conclusions

This research focused on a software application providing opportunities for the processing and analysis of data generated by a saccade sensor with human eye movements. Various methods, approaches, and technologies for modeling, processing, summarizing, and analyzing the data generated during the experimental sessions have been presented. The main functional opportunities of the developed application have been described. They include processing, converting, and storing the different types of data that are generated during the experimental sessions. The application uses four types of information, corresponding to visual tasks: stimuli, an audio signal recorded by a microphone, the responses of the participants in the experiment, and the values generated by the saccade sensor. These values represent the coordinates of the projection of the respective participant's gaze onto the computer display on which the stimuli are visualized. According to the methodology of the experiments, three experiments were conducted. The first was related to visualization of the stimuli on a stimulation computer display that was integrated into the developed application as a separate module. The second experiment was related to an interactive visualization of the projection of the eye movement of the participants in the experiment onto the stimulation computer display. The third experiment was related to an analysis of aggregated data on the decision time and the number of correct responses given by the participants to visual tasks. The tests showed that the application can be used as a stimulation center to visualize the stimuli and to recreate the experimental sessions. The application also has a function to visualize the projection of the participants' eye movements on the computer display. The summary of the results led to the conclusion that the number of correct responses to the visual tasks depended both on the type of motion of the stimuli and on the size of displacement from the center of the aperture. The best results regarding the correct responses were obtained with the combined and flicker motion types, as well as displacement sizes above 100 pixels.

In conclusion, we can summarize that the SimulationCenter application has the necessary functionalities that have been experimentally tested and verified, and through which one can extract, process, convert, and visualize each of the types of data that are related to this experiment. The essential function of this application is the possibility, precisely by synchronizing the different signals (from the microphone and from the saccade sensor) and the stimulus files, to recreate each of all experimental sessions conducted with the participants of the experiment.

Author Contributions: Conceptualization, R.K. and V.K.; methodology, R.K.; software, R.K. and V.K.; formal analysis, R.K.; investigation, R.K.; data curation, V.K.; writing—original draft preparation, R.K. and V.K.; writing—review and editing, R.K.; visualization, V.K. All authors have read and agreed to the published version of the manuscript.

Funding: The APC was funded by MDPI under grant a5b667bc1618fac8 to R.K.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available via the web service at the internet address: http://194.141.86.222/mvsemws/MvsemWebService.asmx, as well as from a web page: http://194.141.86.222/work/mvsem/mvsemres.html and from an archive of summary data from: http://194.141.86.222/work/mvsem/mvsemrescsv.zip. The SimulationCenter application is still in the testing stage and unavailable for download.

Acknowledgments: Data collection for this project was supported by the Bulgarian Science Fund under grant No KP-06-N52/6 from 12 November 2021, project title "Modelling post-perceptual stages of cognitive processing and conscious representations of visual information".

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shu, Y. Experimental data analysis of college english teaching based on computer multimedia technology. *Comput. Aided Des. Appl.* **2020**, *17*, 46–56. [CrossRef]
2. Zong, C.; Zhang, D. Analysis of propagation characteristics along an array of silver nanorods using dielectric constants from experimental data and the drude-lorentz model. *Electronics* **2019**, *8*, 1280. [CrossRef]
3. Tosun, U. Distributed database design: A case study. *Procedia Comput. Sci.* **2014**, *37*, 447–450. [CrossRef]
4. Kraleva, R.; Kralev, V.; Sinyagina, N.; Koprinkova-Hristova, P.; Bocheva, N. Design and analysis of a relational database for behavioral experiments data processing. *Int. J. Online Eng.* **2018**, *14*, 117–132. [CrossRef]
5. Dimitrieski, V.; Čeliković, M.; Aleksić, S.; Ristić, S.; Alargt, A.; Luković, I. Concepts and evaluation of the extended entity-relationship approach to database design in a multi-paradigm information system modeling tool. *Comput. Lang. Syst. Struct.* **2015**, *44*, 299–318. [CrossRef]
6. Date, C.J. *Database Design and Relational Theory (Theory in Practice)*, 1st ed.; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2012.
7. Nicolaos, P.; Katerina, T. Simple-talking database development: Let the end-user design a relational schema by using simple words. *Comput. Hum. Behav.* **2015**, *48*, 273–289. [CrossRef]
8. Kralev, V.; Kraleva, R.; Sinyagina, N.; Koprinkova-Hristova, P.; Bocheva, N. An analysis of a web service based approach for experimental data sharing. *Int. J. Online Eng.* **2018**, *14*, 19–34. [CrossRef]
9. Kralev, V.; Kraleva, R.; Koprinkova-Hristova, P. Data modelling and data processing generated by human eye movements. *Int. J. Electr. Comput. Eng.* **2021**, *11*, 4345–4352. [CrossRef]
10. Bergamaschi, S.; Guerra, F.; Interlandi, M.; Trillo-Lado, R.; Velegrakis, Y. Combining user and database perspective for solving keyword queries over relational databases. *Inf. Syst.* **2016**, *55*, 1–19. [CrossRef]
11. Brown, S.D.; Heathcote, A. The simplest complete model of choice response time: Linear ballistic accumulation. *Cogn. Psychol.* **2008**, *57*, 153–178. [CrossRef]
12. Ratcliff, R.; Smith, P.L. Perceptual Discrimination in Static and Dynamic Noise: The Temporal Relation Between Perceptual Encoding and Decision Making. *J. Exp. Psychol. Gen.* **2010**, *139*, 70–94. [CrossRef] [PubMed]
13. Koprinkova-Hristova, P.; Stefanova, M.; Genova, B.; Bocheva, N.; Kraleva, R.; Kralev, V. Features extraction from human eye movements via echo state network. *Neural Comput. Appl.* **2020**, *32*, 4213–4226. [CrossRef]
14. Bayerl, P.; Neumann, H. Disambiguating visual motion through contextual feedback modulation. *Neural Comput.* **2004**, *16*, 2041–2066. [CrossRef] [PubMed]
15. Grossberg, S.; Mingolla, E.; Pack, C. A neural model of motion processing and visual navigation by cortical area MST. *Cereb. Cortex.* **1999**, *9*, 878–895. [CrossRef]
16. Bocheva, N.; Angelova, D.; Stefanova, M. Age-related changes in fine motion direction discriminations. *Exp. Brain Res.* **2013**, *228*, 257–278. [CrossRef]
17. Pratt, J.; Dodd, M.; Welsh, T. Growing older does not always mean moving slower: Examining aging and the saccadic motor system. *J. Mot. Behav.* **2006**, *38*, 373–382. [CrossRef]
18. Ratcliff, R.; McKoon, G. The diffusion decision model: Theory and data for two-choice decision tasks. *Neural Comput.* **2008**, *20*, 873–922. [CrossRef]
19. Zargari Marandi, R.; Madeleine, P.; Omland, Ø.; Vuillerme, N.; Samani, A. Eye movement characteristics reflected fatigue development in both young and elderly individuals. *Sci. Rep.* **2018**, *8*, 13148. [CrossRef]
20. Drugowitsch, J.; Deangelis, G.C.; Klier, E.M.; Angelaki, D.E.; Pouget, A. Optimal multisensory decision-making in a reaction-time task. *ELife.* **2014**, *2014*, e03005. [CrossRef]
21. Ratcliff, R.; Smith, P.L.; Brown, S.D.; McKoon, G. Diffusion Decision Model: Current Issues and History. *Trends Cogn. Sci.* **2016**, *20*, 260–281. [CrossRef]
22. Bocheva, N.B.; Genova, B.Z.; Stefanova, M.D. Drift diffusion modeling of response time in heading estimation based on motion and form cues. *Int. J. Biol. Biomed. Eng.* **2018**, *12*, 75–83.
23. Wiecki, T.V.; Sofer, I.; Frank, M.J. HDDM: Hierarchical bayesian estimation of the drift-diffusion model in Python. *Front. Neuroinform.* **2013**, *7*, 14. [CrossRef] [PubMed]
24. Prater, A. Spatiotemporal signal classification via principal components of reservoir states. *Neural Netw.* **2017**, *91*, 66–75. [CrossRef]
25. Koprinkova-Hristova, P.D.; Bocheva, N.; Nedelcheva, S.; Stefanova, M. Spike timing neural model of motion perception and decision making. *Front. Comput. Neurosci.* **2019**, *13*, 20. [CrossRef]
26. Layton, O.W.; Fajen, B.R. Possible role for recurrent interactions between expansion and contraction cells in MSTd during self-motion perception in dynamic environments. *J. Vis.* **2017**, *17*, 5. [CrossRef]
27. Pilz, K.S.; Kunchulia, M.; Parkosadze, K.; Herzog, M.H. Ageing and visual spatiotemporal processing. *Exp. Brain Res.* **2015**, *233*, 2441–2448. [CrossRef] [PubMed]

28. Dowiasch, S.; Marx, S.; Einhäuser, W.; Bremmer, F. Effects of aging on eye movements in the real world. *Front. Hum. Neurosci.* **2015**, *9*, 46. [CrossRef]
29. Manjunath, G.; Narasimha Murty, M.; Sitaram, D. Combining heterogeneous classifiers for relational databases. *Pattern Recognit.* **2013**, *46*, 317–324. [CrossRef]
30. Kulkarni, R.H.; Padmanabham, P.; Harshe, M.; Baseer, K.K.; Patil, P. Investigating agile adaptation for project development. *Int. J. Electr. Comput. Eng.* **2017**, *7*, 1278–1285. [CrossRef]
31. Kraleva, R.; Kralev, V.; Koprinkova-Hristova, P. Data analysis from Two-Choice decision tasks in visual information processing. *Int. J. Inform. Visualization* **2021**, *5*, 187–193. [CrossRef]
32. Bocheva, N.; Georgieva, O.; Stefanova, M. Data analysis of age-related changes in visual motion perception. In Proceedings of the 3rd International Conference on Agents and Artificial Intelligence, Rome, Italy, 28–30 January 2011; Volume 1, pp. 556–561.
33. Seghir, F.; Khababa, G. Fuzzy teaching learning based optimization approach for solving the QoS-aware web service selection problem in uncertain environments. *J. Ambient Intell. Humaniz. Comput.* **2021**, *12*, 10667–10697. [CrossRef]
34. Garba, S.; Mohamad, R.; Saadon, N.A. Web service discovery approaches for dynamic mobile environment: A systematic literature review. *Int. J. E-Serv. Mob. Appl.* **2021**, *13*, 16–38. [CrossRef]
35. Zhang, X.; Liu, J.; Cao, B.; Shi, M. Web service classification based on information gain theory and bidirectional long short-term memory with attention mechanism. *Concurr. Comput.* **2021**, *33*, e6202. [CrossRef]
36. Qiao, X.; Li, Z.; Zhang, F.; Ames, D.P.; Chen, M.; James Nelson, E.; Khattar, R. A container-based approach for sharing environmental models as web services. *Int. J. Digit. Earth.* **2021**, *14*, 1067–1086. [CrossRef]
37. Bikakis, N.; Tsinaraki, C.; Gioldasis, N.; Stavrakantonakis, I.; Christodoulakis, S. The XML and semantic web worlds: Technologies, interoperability and integration: A survey of the state of the art. *Stud. Comput. Intell.* **2013**, *418*, 319–360. [CrossRef]
38. Atilgan, D.; Altingovde, I.S.; Ulusoy, O. On the size of full element-indexes for XML keyword search. *Lect. Notes Comput. Sci.* **2012**, *7224*, 556–560. [CrossRef]
39. Gamido, H.V.; Gamido, M.V. Comparative review of the features of automated software testing tools. *Int. J. Electr. Comput. Eng.* **2019**, *9*, 4473–4478. [CrossRef]
40. Brata, A.H.; Liang, D.; Pramono, S.H. Software development of automatic data collector for bus route planning system. *Int. J. Electr. Comput. Eng.* **2015**, *5*, 150–157. [CrossRef]
41. Srinivas, M.; Ramakrishna, G.; Rajasekhara Rao, K.; Suresh Babu, E. Analysis of legacy system in software application development: A comparative survey. *Int. J. Electr. Comput. Eng.* **2016**, *6*, 292–297. [CrossRef]
42. Bhardwaj, M.; Rana, A. Key software metrics and its impact on each other for software development projects. *Int. J. Electr. Comput. Eng.* **2016**, *6*, 242–248. [CrossRef]
43. Jaspan, C.; Jorde, M.; Egelman, C.; Green, C.; Holtz, B.; Smith, E.; Hodges, M.; Knight, A.; Kammer, L.; Dicker, J.; et al. Enabling the Study of Software Development Behavior with Cross-Tool Logs. *IEEE Softw.* **2020**, *37*, 44–51. [CrossRef]
44. Khan, A.A.; Akbar, M.A. Systematic literature review and empirical investigation of motivators for requirements change management process in global software development. *J. Softw.: Evol. Process.* **2020**, *32*, e2242. [CrossRef]