

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Ilja Jakobovitš 205894IAIB

**APPLICATION FOR THE ANALYSIS OF DRAWING AND
WRITING TESTS**

Bachelor's Thesis

Supervisor: Prof. Sven Nõmm
PhD

Tallinn 2024

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ilja Jakobovitš 205894IAIB

**RAKENDUS JOONISTUS- JA KIRJUTAMISTESTIDE
ANALÜÜSIMISEKS**

Bakalaureusetöö

Juhendaja: Prof. Sven Nõmm
PhD

Tallinn 2024

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Ilja Jakoboviš

27.05.2024

Abstract

This thesis aims to develop a desktop application to examine and analyze drawing and writing tests employed in assessing various neurological conditions. Initially, those tests were conducted using pen and paper, but with the recent development of tablet computers, it is possible to carry out the testing digitally. Digital tests have many advantages over paper analogs, allowing one to gather more information like pen velocity, pressure, and angle. Although applications exist to perform drawing and writing on a tablet computer, there is a lack of tools for further analysis of the test samples.

The analyzing process consists of visualization of test samples, calculating various features, and employing machine learning models. The application is designed to import data in the same format as stored in tablet computers. It features a graphical user interface that allows medical professionals to navigate through test samples, play or rewind the drawing process, segment drawings into different parts, view calculated features with the output of machine learning models, and generate reports. The machine learning models are not hard-coded and are open to extension and modification by the user. Alongside the graphical interface, a command line tool allows retraining and verifying a model on gathered training samples, if needed.

As a result, most of the planned functionality was implemented. The application works as intended. Due to the scarcity of available data, the machine learning models must be further improved, fine-tuned, and verified before being used in a real production setting.

The thesis is written in English and is 27 pages long, including 5 chapters and 11 figures.

Annotatsioon

Rakendus joonistus- ja kirjutamistestide analüüsimiseks

Selle lõputöö eesmärk on töötada välja töölauarakendus neuroloogiliste joonistus- ja kirjutamistestide vaatamiseks ja analüüsimiseks. Algselt viidi need testid läbi pliitsi ja paberiga, kuid tahvelarvutite hiljutise arenguga on võimalik testida ka digitaalselt. Digitaaltestidel on paberanalooide ees palju eeliseid, mis võimaldavad meil koguda rohkem teavet, nagu pliitsi kiirus, rõhk ja nurk. Kuigi tahvelarvutis joonistamiseks ja kirjutamiseks on olemas rakendused, joonistuste analüüsimine tööriist puudub.

Analüüsi protsess koosneb joonistuste visualiseerimisest, erinevate parameetrite arvutamisest ja masinõppemudelite kasutamisest. Rakendus on loodud andmete importimiseks samas vormingus, mis on salvestatud tahvelarvutitesse. Sellel on graafiline kasutajaliides, mis võimaldab meditsiinitöötajatel navigeerida katsenäidistes, mängida või tagasi kerida joonistusprotsessi, segmentida jooniseid erinevateks osadeks, vaadata arvutatud parameetrit masinõppemudelite väljundiga ja koostada aruandeid. Masinõppemudelid ei ole kõvakodeeritud ning kasutaja saab neid laiendada ja muuta. Lisaks graafilisele liidesele võimaldab käsurea tööriist vajaduse korral koondatud andmete põhjal mudelit ümber õpetada ja kontrollida.

Selle töö tulemusena rakendati suurem osa kavandatud funktsioonidest. Rakendus töötab ettenähtud viisil, kuid sellel on mitme kriitilised vead ja puudused, mida saaks edasise arendusega parandada. Saadaolevate andmete nappuse tõttu tuleb masinõppemudeleid enne reaalses tootmiskeskkonnas kasutamist veelgi täiustada, peenhäälestada ja kontrollida.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 27 leheküljel, 5 peatükki, 11 joonist.

List of Abbreviations and Terms

AI	Artificial Intelligence
API	Application Programming Interface
CLI	Command Line Interface
GUI	Graphical User Interface
JSON	JavaScript Object Notation
ML	Machine Learning
MVP	Minimum Viable Product
PD	Parkinson's Disease
PDF	Portable Document Format

Table of Contents

1	Introduction	8
1.1	Problem statement	8
1.2	Existing solutions	10
2	Project design	11
2.1	Target platform	11
2.1.1	Comparison of web and desktop application	11
2.1.2	Operating system	12
2.2	Technology overview	12
2.2.1	Programming language	12
2.2.2	User interface library	12
2.2.3	Machine learning library	13
2.2.4	Version control system	13
2.2.5	Dependency management and software build tools	14
2.3	Data format	14
3	Implementation	16
3.1	Graphical interface overview	16
3.2	Visualization	17
3.2.1	Animations	17
3.2.2	Customization	18
3.2.3	3D graph	18
3.3	Segmentation	19
3.3.1	Training data extraction	21
3.3.2	Machine learning model selection	22
3.4	Generating reports	22
3.5	Saving and loading	22
3.6	Machine learning interface overview	23
3.6.1	Features	23
3.6.2	Command line tool	24
4	Evaluation	25
4.1	Result	25
4.2	Considerations for future development	25
5	Summary	27

Acknowledgments	28
References	29
Appendix 1 – Non-Exclusive Licence for Reproduction and Publication of a Graduation Thesis	30
Appendix 2 – Formulas for the computation of features	31
Appendix 3 – A screenshot of calculated features in the application	32
Appendix 4 – An example from a generated PDF report	33
Appendix 5 – A link to GitHub repository	34

List of Figures

1	<i>The main window of the application.</i>	16
2	<i>The 2D plot customization window.</i>	18
3	<i>The 3D plot window.</i>	19
4	<i>The analysis window.</i>	20
5	<i>The process of selecting a segment with a mouse. It is done by left-clicking and dragging the red box around the desired area.</i>	21
6	<i>Saving a sample for the future machine learning model training.</i>	22
7	<i>Selecting a machine learning model to evaluate the drawing.</i>	22
8	<i>Selecting segments to include in a report.</i>	23
9	<i>The CLI tool help command output.</i>	24
10	<i>A screenshot of calculated features in the application.</i>	32
11	<i>An example of a PDF report generated from the application.</i>	33

1. Introduction

Drawing tests have been used in psychiatry and neurology for more than a century. Initially designed to support the diagnosis of neurodegenerative disorders and brain injuries, the tests were adapted to be used in other areas, such as the assessment of cognitive functions and the detection of fatigue. The beginning of the twenty-first century witnessed rapid development of tablet PCs, which were noticed as a viable alternative to the paper and pen setting used to conduct drawing tests before. Compared to the paper-and-pen setting, the tablet PC offers the possibility of capturing many parameters of drawing kinematics and pressure, opening the door to a more objective approach in the assessment of testing results. Together with statistical machine learning methods [1] and more recently deep learning techniques [2] tablet PC-based testing leads to highly accurate results in supporting diagnostics of neurodegenerative diseases [3]. Despite this, the full potential of tablet PC-based testing has yet to be used. One area requiring more attention is the visual analysis and comparison of the tests. An algorithm based on artificial intelligence (AI) may provide a prediction of the test result, but it does not allow the practitioner to explore and compare the tests of the same patient. The present work aims to bridge this gap by providing the tool to examine the test of a patient alongside with the output of assisting AI models. The application would allow users to visually evaluate the drawing tests, compare them across different time points for the same patient, and interpret the AI model's outputs in a meaningful context. By doing so, the practitioners could get an insight into the patient's condition and progression. One key aspect of the system is the incorporation of animated visualizations.

1.1 Problem statement

The problem statement of this thesis is to develop an application designed for viewing and analyzing drawing tests.

The desktop application resulting from this thesis would be a tool designed for healthcare professionals to load, visualize, analyze, and apply machine learning to assess drawings made by patients with neurological diseases. The application would have an intuitive interface. The main dashboard would allow easy navigation between different functionalities such as loading data, visualization, analysis, and reports. Users can upload drawings from digital tablets via a USB port and then load them from the application. Once the data is loaded, the application provides visualization tools. These tools would enable users to view

the drawings in both 2D or 3D mode, zoom in and out, and play and rewind the drawing process. As a part of visualization, the user can fully customize graphs where drawings are rendered.

In the analysis window, the user can segment the drawing into simpler shapes, choose the appropriate machine learning model, and view its output together with calculated kinematic, geometric, and other features.

As an end result, the following requirements should be completed:

- The application must be able to read the drawing extracted from the tablet.
- The application must provide customizable 2D graphs and 3D graphs of drawings with a time component.
- The application must provide play, pause, rewind, and forward skip functions to observe the drawing process.
- The application must be able to segment drawings into different parts both automatically or manually
- For each segment, the application must calculate different parameters like pen velocity, acceleration, and higher-order derivatives.
- The application must be able to extract and save training data samples for machine learning models.
- The application must provide an interface to train, modify, and verify new machine learning models.
- The application must be able to generate PDF reports containing the drawings and associated information.
- The application must be able to save the current progress.
- The application must work on the Windows operating system.

The development should be conducted in an iterative manner, enabling incremental enhancements and the integration of feedback at various stages. By starting with a core set of functionalities, the focus is set on establishing a solid foundation, which can then be expanded and refined based on user needs and emerging technologies. The goals outlined serve as a set of requirements for a minimal viable product which already could be used but upon completion and serve as a foundation for a project that can evolve into bigger, more comprehensive and professional tool.

1.2 Existing solutions

Previously, E.Tamm and A.Pärnoja have proposed interfaces to facilitate the processing of digitized drawing tests. The thesis of Tamm was devoted to the analysis of the entire battery of tests, and Pärnoja concentrated his attention on allowing the analysis of one test as a whole [4, 5]. This thesis has two major novel components. From the point of view of analysis, it will provide the possibility of using machine learning techniques to find and classify the components of the drawings. Unlike previous work, this thesis targets more complex tests such as Peuplereuter's and other tests consisting of multi-element drawings. From the software architecture viewpoint, the software will be built to allow the integration of different components to be used to analyze drawing and writing tests.

The drawing process itself happens on a tablet computer using the application developed by Sergei Zarembo and called Draw Test [6].

2. Project design

Before development began, a planning phase was undertaken, which consisted of reviewing and selecting the right technologies to complete the goals. To help select the appropriate technologies, a list of essential questions regarding project design was composed.

- Should it be the desktop application or the web application?
- Which programming language to use?
- Which frameworks or libraries to use to develop a user interface?
- Which machine learning frameworks or libraries to use?
- How will the application be built and distributed?

The following section provides an overview of how those questions were answered and, subsequently, how the technology stack was decided.

2.1 Target platform

2.1.1 Comparison of web and desktop application

Web applications are accessible from any device with an internet connection, making them convenient for users who need to access the application from different locations or devices. They are easier to maintain since updates and bug fixes are implemented on the server, and users can always access the latest version without needing to update their software. However, web applications typically require a stable internet connection and can be slower than desktop applications due to their reliance on web server performance and internet speed. They also have limited access to a user's local system resources.

Desktop applications, on the other hand, are known for their performance and efficiency. They run directly on the user's computer and can take full advantage of the computer's processing power and memory. This makes them more suitable for applications that require intensive data processing or graphic capabilities. Desktop applications also work offline, providing uninterrupted access regardless of internet connectivity. However, they are platform-dependent and often need to be individually installed and updated on each user's computer. Additionally, if they need to be compatible with multiple operating systems, desktop applications could require significantly more time to develop [7].

For the task of analyzing drawings, a desktop application appears to be more suitable. This suitability stems from the desktop application's superior performance capabilities, which are essential for handling the potentially large and complex data sets involved in drawing analysis. The application's ability to function offline ensures reliability and continuous accessibility. Furthermore, the enhanced security offered by desktop applications is vital for protecting sensitive data often associated with such analyses.

2.1.2 Operating system

Windows operating system was selected as the main target platform. Windows is one of the most prevalent operating systems used on personal computers worldwide. In the context of the medical sector, Windows holds a particularly significant position. Most medical workers, including doctors, nurses, and administrative staff, rely on Windows-based computers for their daily operations [8].

Although the main target is Windows, since the application is built upon Python, it makes it easier to make the application cross-platform in the future.

2.2 Technology overview

2.2.1 Programming language

Python is a suitable programming language for developing a desktop application that utilizes machine learning to analyze drawings. Python has many machine learning libraries, such as TensorFlow, PyTorch, scikit-learn, and Keras. These libraries simplify the process of implementing complex algorithms, offering pre-built functions and models. Python has many tools for data manipulation and processing, which is vital for handling drawing inputs. Libraries like NumPy and Pandas provide tools for data analysis and transformation, making it easier to preprocess images for machine learning models. Python is inherently cross-platform, meaning a Python-based desktop application can run on various operating systems such as Windows, macOS, and Linux without significant modifications. Python supports multiple frameworks for developing graphical user interfaces, such as Tkinter, PyQt, and Kivy.

2.2.2 User interface library

Python has a variety of GUI libraries, each with its unique features and capabilities. Among these, PyQt and Tkinter are particularly notable, along with others like Kivy, wxPython,

and PyGTK. Among these options, PyQt was chosen for the development. PyQt is a set of Python bindings for the Qt application framework. It offers comprehensive tools and widgets to create desktop applications. It allows the integration of matplotlib which enables applications to have graphical plotting and data visualization. This integration enables the embedding of graphs and charts directly into the GUI [9].

For the rendering of the drawings matplotlib was chosen. Matplotlib is a plotting library in Python that offers capabilities for creating both static and animated 2D graphs. Its animation functionality can be used for visualizing drawings

2.2.3 Machine learning library

Keras is a high-level neural networks API, written in Python, acting as an interface for the TensorFlow library. Keras is generally easier for beginners and for applications that require fast development. TensorFlow is more low-level compared to Keras; it offers finer control over model architecture and training processes. Keras runs on top of TensorFlow, simplifying its complexity with a more intuitive interface. PyTorch, like TensorFlow, is more flexible but also more complex than Keras. Scikit-learn is primarily focused on traditional machine learning algorithms. For the current task, Keras is chosen due to its ease of use, and the ability to quickly prototype, save, and load neural networks. Its high-level functionality abstracts away many of the complex aspects of building, training, and persisting neural network models, making it an ideal choice for tasks that require efficient development without the need for in-depth manipulation of lower-level API functionalities [10].

In Keras, models are made either by Functional API or Sequential API. The sequential model in Keras is used to create models layer-by-layer in a step-by-step fashion, the functional API allows you to create models that have a lot more flexibility. You can connect any layer to any other layer, rather than just the next one in a sequence.

Keras provides a straightforward way to save and load models, which enables the preservation of trained models and deployment of them for future use. This functionality in Keras covers saving the entire model, just the model's architecture, or only the weights.

2.2.4 Version control system

For this project, GitHub was chosen as a version control tool. GitHub is an online platform for software development. It's built around Git, a version control system that enables

developers to track changes and collaborate on coding projects.

Throughout the development, the project was private, with only one developer working on it. However, after the release, the code is planned to be opened to the public allowing other collaborators to contribute to the project.

2.2.5 Dependency management and software build tools

Dependency management is achieved through the use of virtual environments and the package manager "pip". A virtual environment is an isolated Python environment where dependencies for a specific project are installed. This isolation prevents conflicts between project dependencies and ensures that each project has access to only the packages it needs [11].

The application itself is distributed as a stand-alone executable which is packed together with Python environment and necessary dependencies. For that PyInstaller is used. PyInstaller is a tool used for converting Python applications into stand-alone executables, under Windows, Linux, and MacOS. Its primary purpose is to bundle a Python application and all its dependencies into a single package. This package can then be distributed and run on systems without requiring a separate Python installation. PyInstaller is a tool that simplifies the distribution process and ensures that Python applications can be run on various platforms without additional dependencies [12].

After the building process, the resulting release is uploaded to GitHub from where users can download it freely.

2.3 Data format

The application accepts drawings in exactly the same format as the drawing application for tablet computers made by Sergei Zarembo. Each drawing is saved as a JSON file. A drawing is represented by a list of points spaced out in time sampled with high frequency. Each point, alongside its timestamp, x, and y coordinate, also contains pen pressure, pen altitude, and pen longitude.

For the handling and manipulation of data Pandas and Numpy libraries are used. The core of Pandas is its DataFrame - a two-dimensional, size-mutable, tabular data structure with labeled (or indexed) axes (rows and columns). It is versatile for data manipulation tasks like data cleaning, transformation, analysis, and visualization. Pandas provide a set

of functions to easily handle missing data, merge, reshape, and slice datasets. NumPy provides an array-object or ndarray, which is a N-dimensional array that allows for the efficient manipulation of large datasets. NumPy arrays are faster and more compact than Python lists. The library also offers a comprehensive set of mathematical functions to perform operations on these arrays [13, 14].

3. Implementation

This chapter provides a complete overview of the functionality of the application. The application can be divided into two parts. The first part is a graphical interface that physicians can use directly to view and analyze the tests performed. The second part is the command interface for working with machine learning models. While the GUI does not require knowledge of technical details, the command interface requires the user to have experience in programming and machine learning, and is intended primarily for machine learning specialists.

3.1 Graphical interface overview

The interface consists of a main window and additional windows that are opened by clicking the corresponding buttons.

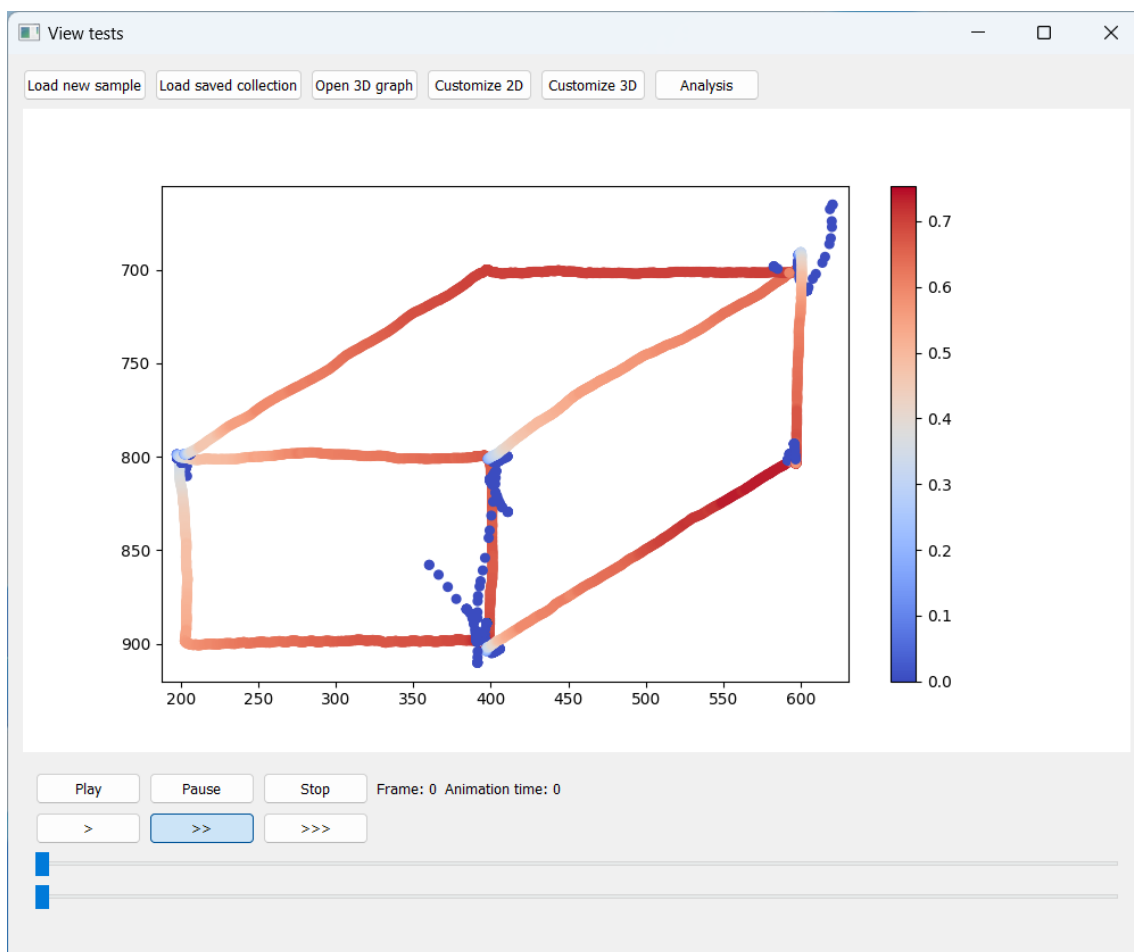


Figure 1. *The main window of the application.*

The main buttons are located in the upper part of the main window. Reading them from left to right gives the following.

- "Load new test" button opens a window where the user can choose and load a new test without associated information
- "Load saved test" button opens a window where the user can choose and load a new test with associated information (segments, features, notes)
- "Open 3D graph" button opens a window with the 3D graph of a drawing, the third coordinate being time
- "Customize 2D" button opens a window where the user can change the appearance of the 2D graph of the drawing
- "Customize 3D" button opens a window where the user can change the appearance of 3D graph of the drawing
- "Analysis" button opens a window where the user can select segments and machine learning models, view segment's information, attach notes to the segments, and save training samples

The following sections give an overview of each application component in more detail.

3.2 Visualization

A drawing is rendered at the main window in the form of a 2D scatter plot. The scatter plot consists of a series of tightly packed points which together form a picture of the drawing.

In order to see the drawing, a user must first load it using either of two loading options. After a test sample is loaded, the user can see a 2D graph of a drawing in the main window. By default, a pressure color bar is activated, coloring the drawing with red or blue color hues according to pressure at certain areas. Color gradation allows for the visual identification of regions with different pressure intensities. Red signifies areas under high pressure, while blue indicates lower-pressure regions. The size of the drawing is not fixed and depends on the whole application window size. The user can interact with the drawing, for example zooming in or out, using a toolbox at the upper part of the canvas.

3.2.1 Animations

The animation control panel is located below the picture. This panel contains buttons that allow the user to start the animation. Animation shows how the drawing was carried out from start to finish. The user can also pause or stop the animation altogether, which resets

it to the original state. Below the buttons, there are two sliders that allow the user to select a specific part of the drawing by specifying the start by the bottom slider and the end by the upper slider. These sliders also play a part in the segmentation, as the selected segment could be saved by pressing the corresponding button in the analysis window.

By default, the animation speed matches the actual drawing speed. To speed up or slow down the animation, the user can click on the corresponding buttons that control the speed of the animation. There are three speed options in total. The first option is twice slowed down, the second one is normal, the third one is twice sped up.

3.2.2 Customization

The graph properties and appearance could be customized in the customization window, which is opened by the corresponding button at the top panel. Modifiable options include the thickness of points, the color of the drawing, enabling or disabling the pressure bar and pressure coloring, enabling or disabling the axis of the graph, and filtering points with pressure below some threshold.

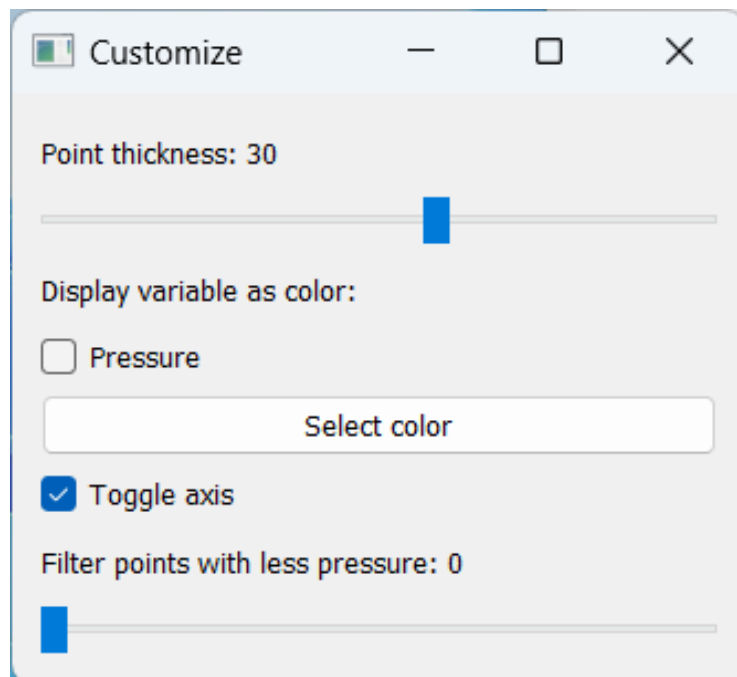


Figure 2. *The 2D plot customization window.*

3.2.3 3D graph

To open the 3D graph, the user can click on the corresponding button on the top panel. The third coordinate in the 3D graph is time. The graph itself is also a scatter plot. The 3D graph has the same properties as the 2D counterpart and can be customized in the 3D

graph customization window. All settings possible for customization for the 3D graph are the same. The user can also enable animation for the 3D graph using a button on the animation control panel, but this may lead to excessive use of computer resources and, as a result, slowed animation.

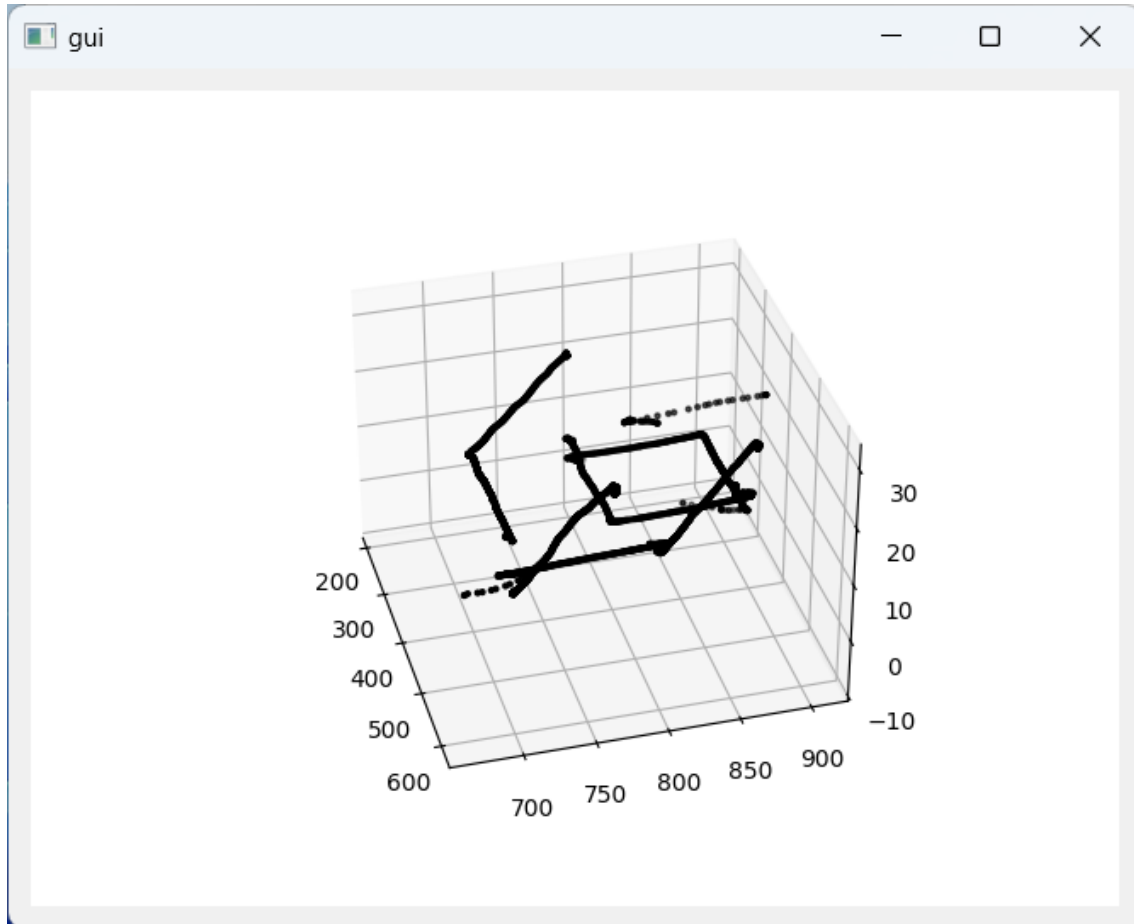


Figure 3. *The 3D plot window.*

3.3 Segmentation

Interaction with segments occurs in the “Analysis” window. This window can be divided into three parts. On the top left side, there are various buttons that enable the user to interact with the underlying data. The first three buttons are responsible for three different segmentation methods available to the user. The fourth button opens a window to save a training sample. The fifth button opens a window to select the current machine learning model.

In total, there are three segmentation methods in the application:

1. Automatic segmentation by strokes.

A drawing essentially consists of strokes. One stroke is a continuous line that starts

when the drawer's pen touches the tablet screen and ends with the lifting of the pen from the tablet screen. By pressing the corresponding button, the application will automatically divide the drawing into individual strokes and save them as segments.

2. Manual segmentation by sliders.

Using the sliders at the animation control panel, the user can select a part of the drawing within some time span. Then clicking the corresponding button saves the selected segment.

3. Manual segmentation by mouse.

Clicking the corresponding button will activate segmentation by mouse mode. When this mode is active, the user can left-click on the drawing by mouse and drag the cursor across the drawing. This will produce a selection box that contains some parts of the drawing. Releasing the mouse button will save the selected area as a new segment. In case there are multiple strokes in the selected area, they are automatically saved as separate segments.

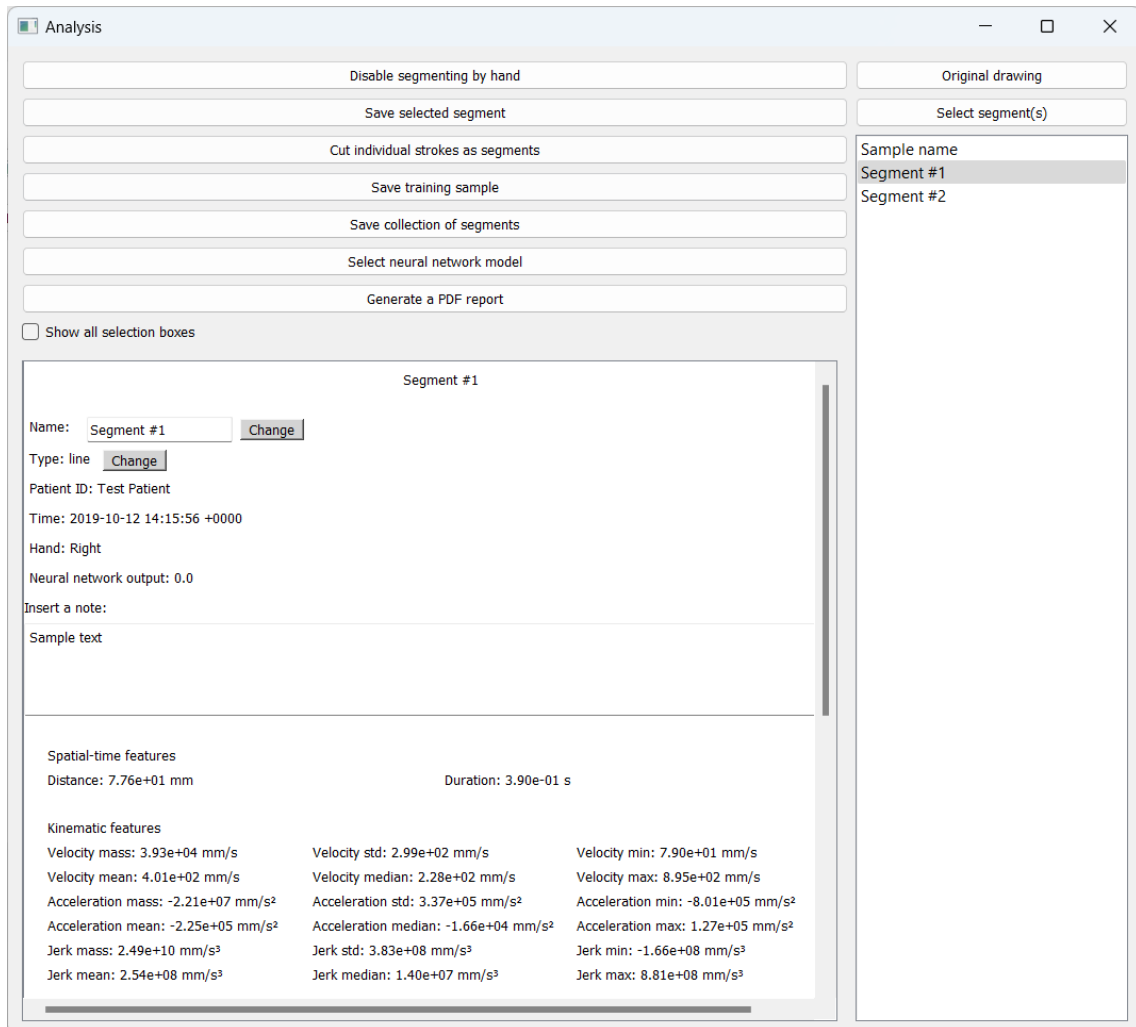


Figure 4. The analysis window.

All saved segments of a given drawing are displayed in a list on the right side of the screen.

By default, each new segment is given an auto-incrementing identifier as its name, but the user can change the segment name. However, each segment name must be unique.

In the bottom left part of the window, the user can see the segment's name, type, machine learning model output, and all calculated features from the drawing data. In addition to that, the user can change the segment name or type and attach a text note using the corresponding text field.

In total, there are 62 features. A more detailed breakdown of features is presented in [].

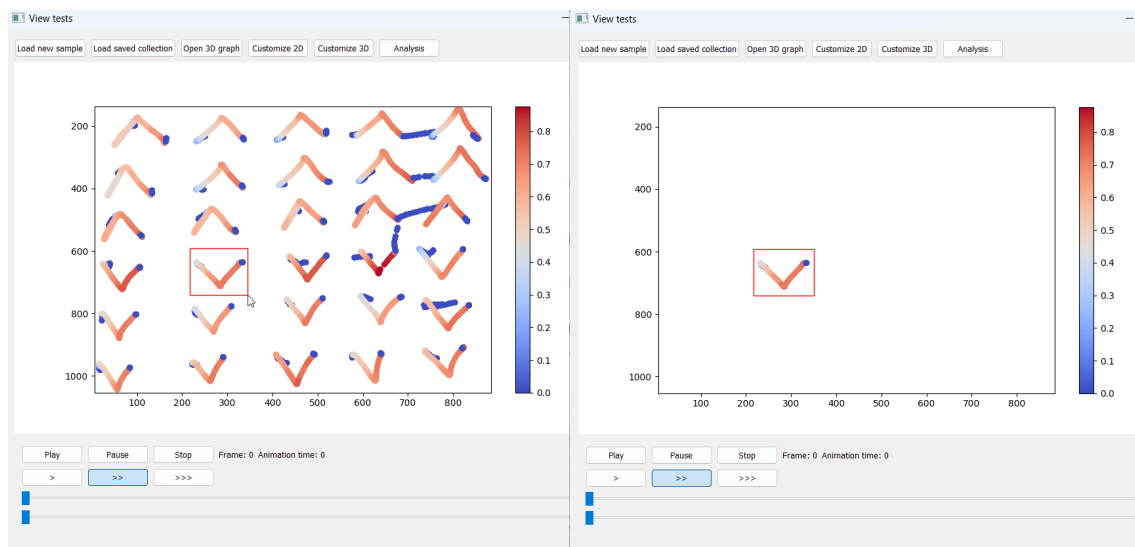


Figure 5. *The process of selecting a segment with a mouse. It is done by left-clicking and dragging the red box around the desired area.*

3.3.1 Training data extraction

One of the benefits of segmentation is the extraction of training samples. The amount of training data for the purpose of drawing analysis could be scarce, with each drawing providing exactly one training sample. However, the drawing can be a complex shape consisting of many simpler shapes. Extracting these shapes provides more training data and, as a consequence, enables more versatile and precise analysis.

In order to save a new sample, it should be first labeled. For that, the user can press the button in the upper left to open a new window. There, the user can select appropriate labels like the presence of PD or segment type and then save it. Using the extracted training samples, the user can later retrain the existing model or train a new one.

3.3.2 Machine learning model selection

It is also possible to select the currently used machine learning model using the last button in the upper left control panel. It opens a window with a list of all available models. After the selection of the model, the application will take all the calculated features, run them through the model and output the result in the same spot where segment info is located.

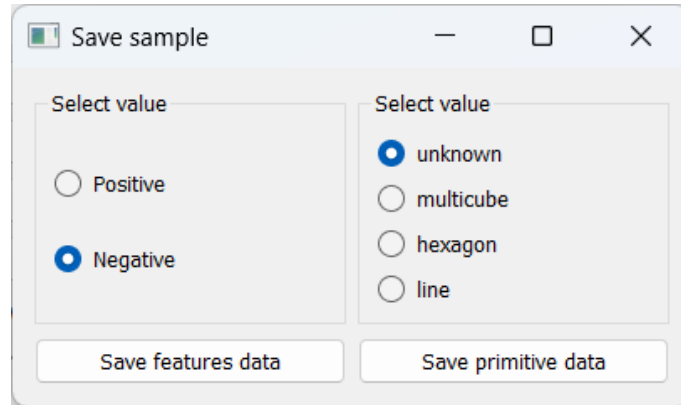


Figure 6. Saving a sample for the future machine learning model training.

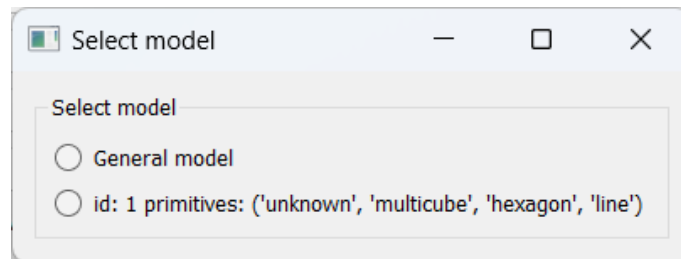


Figure 7. Selecting a machine learning model to evaluate the drawing.

3.4 Generating reports

The user can generate a PDF report. To do this, the user can click the corresponding button in the top panel and then select which segments to include in the report together with the location where the report would be saved. The generated report will contain a drawing with the current customization, calculated parameters, notes, and the output of the machine learning model.

3.5 Saving and loading

The user can save achieved progress by pressing the corresponding button in the uppermost panel. The application will save all segments and related information in a separate folder. Subsequently, the user can load the saved project.

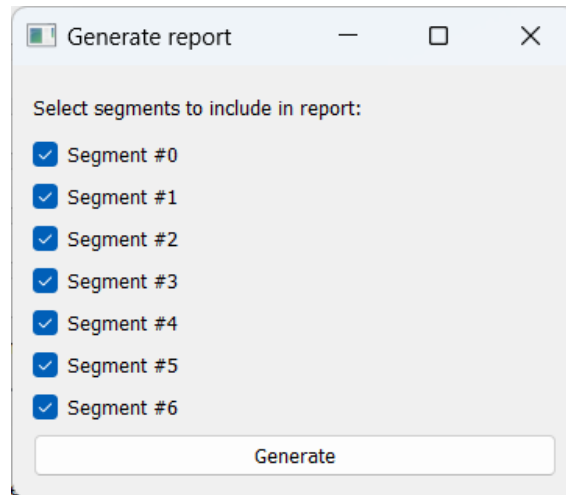


Figure 8. *Selecting segments to include in a report.*

3.6 Machine learning interface overview

For the purpose of analyzing the data of a patient’s drawing, many machine learning methods could be used. They include Logistic Regression, AdaBoost, Naive Bayes, Support Vector Machine, K-nearest neighbors algorithm, Random Forests, Decision Trees, Linear Discriminant Analysis [1, 2]. Another prominent methods are neural networks and convolutional neural networks [3]. The Keras library used within the application is mainly suited for deep learning neural networks. Any model defined in Keras API and adhering to input and output format could be used by the application. The next sections describe which features are available to the user and how a new model could be defined and trained.

The training data format is different from the format the drawings themselves are presented. For the purpose of training, a feature set and the corresponding labels are represented by a NumPy array.

3.6.1 Features

The original time-series data, including the pen’s position in terms of x and y coordinates, the time at which the data is recorded, the pressure applied by the pen, and the pen’s altitude and azimuth, can be used to calculate various features in a vector form. Then, single-valued statistical parameters are calculated on the vector features.

The velocity is computed as the distance between two neighbor points divided by the time difference between the same points. The pressure, azimuth, and latitude difference are computed as the difference of the value between two neighbor points divided by the time difference between the same points.

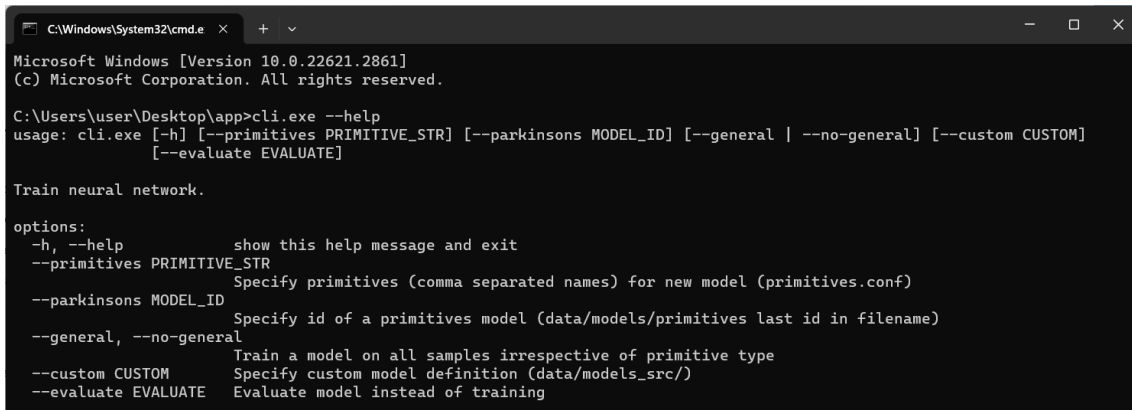
Alpha angle, rotational angle, and yaw angle calculations are provided in Appendix 2. The derivatives are approximated by dividing the forward difference (Appendix 2.5) by the time difference between two points.

3.6.2 Command line tool

Alongside the primary executable, which opens the graphical user interface, there is also an executable providing a command line interface that enables the user to interact with machine learning models. This tool allows the user to train a new model using custom definitions and training data saved in a specific folder within the application folder structure.

The new model definitions could be added in the folder "models_src". They are essentially Python files consisting of a single function that takes in all specified training data, which consists of features described above and a corresponding label, and outputs a trained model by Keras. The model is automatically saved with the model name being equal to the name of the Python file. These models could then be used in the "Analysis" window of the application.

To obtain specific commands acceptable by the command tool the "--help" argument could be used.



```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user\Desktop\app>cli.exe --help
usage: cli.exe [-h] [--primitives PRIMITIVE_STR] [--parkinsons MODEL_ID] [--general | --no-general] [--custom CUSTOM]
              [--evaluate EVALUATE]

Train neural network.

options:
  -h, --help            show this help message and exit
  --primitives PRIMITIVE_STR
                        Specify primitives (comma separated names) for new model (primitives.conf)
  --parkinsons MODEL_ID
                        Specify id of a primitives model (data/models/primitives last id in filename)
  --general, --no-general
                        Train a model on all samples irrespective of primitive type
  --custom CUSTOM      Specify custom model definition (data/models_src/)
  --evaluate EVALUATE  Evaluate model instead of training
```

Figure 9. The CLI tool help command output.

4. Evaluation

4.1 Result

As a result of the work, a functioning application was produced that meets the stated requirements. All planned functions of the application were implemented. The resulting build of the application consists of two executable files, a runtime environment (Python interpreter and packages), and application internal files and folders (data samples, reports, saved projects, machine learning models). The build is published in a public GitHub repository from which it can be downloaded as a ZIP archive. Running the application does not require installing any additional dependencies. Some exemplary drawing tests are provided from the get-go. To interact with the actual drawing tests, they first should be transferred from the tablet computer to the computer where the application runs.

One of the goals of this work was to create an interface for interacting with machine learning models working with drawing data. To create a sufficiently accurate model, it is necessary to collect a sufficient number of samples for training. Different neural network designs can be defined and trained directly by the user. This allows for a more flexible approach to machine learning, as users can experiment with different neural network configurations and different types of tests.

During the development, some critical bugs were encountered and subsequently fixed. All the primary functionalities of the application are expected to work correctly. However, some minor bugs and inconveniences may appear. The following section will explore potential enhancements and functionalities that could be added to the application, although these suggestions extend beyond the current project's scope.

4.2 Considerations for future development

Currently, the user needs to move files with the drawings from the tablet manually. A possible big improvement is the seamless connection between the app and the data on the tablet. This means the test samples become available in the application right after the drawing is finished. It would demand some kind of network connection between the host where the application is running and the tablet computer. Various methods could be employed to accomplish this, and it's important that any chosen method considers and addresses data privacy issues.

In the current version of the application only neural networks defined using Keras API are supported. The more general interface that supports any machine learning backend and model is desirable.

At present, each test exists separately from each other. A useful addition would be to create a summary that includes and provides statistics on a set of separate tests. For example, make it possible to create a portfolio of all tests of a certain patient over a certain period of time and present in the application various parameters and graphs based on a collection of tests. This could help to analyze changes in drawing characteristics over time to assess possible disease progression.

5. Summary

The goal of this work is to create an application for viewing and analyzing drawing tests for the assessment of neurological disorders. By applying visualization, segmentation and machine learning techniques the medical practitioner could get an additional insight into the condition of the patient. Initially, the project began by clearly listing and identifying specific goals to ensure that the development was aligned with the needs of medical professionals. An analysis of potential technologies was then conducted, leading to the selection of the most appropriate tools and frameworks that would support data processing and interface designs. During the development of the project, an iterative development methodology was employed, incorporating feedback from the users who evaluated preliminary versions of the application.

The first version of the application included the ability to load, customize, and view animated plots of drawings. In a subsequent version the calculation of various features of the drawing process was added, and an interface with which you can modify and train machine learning models has been developed. Due to the fact that machine learning models are not hard-coded into the program, the developed interface allows specialists in the field of machine learning to define and apply their own models. At the moment, the program supports the creation of neural networks using the Keras library, but in the future it is possible to generalize the interface and support any backends. The latest version was updated with the ability to create PDF reports, attach notes to drawings, save and load progress. Reports contain a drawing, or its segments, notes and meta information associated with the drawing, as well as calculated features.

Overall, the result of this work is an MVP that satisfies the initial requirements set at the project's start, but also holds great potential for further development. Enhancements such as improving the user interface and the categorization of drawing tests will enhance usability and functionality. Additionally, refining existing machine learning models and integrating new ones will increase the application's analytical precision. Integrating the application directly with drawing software would make an easier workflow for doctors, making it an even more valuable tool in their diagnostic and monitoring toolkit. These improvements would ensure that the application not only meets current professional standards but also adapts to future advancements in neurological assessment.

Acknowledgments

This thesis is partially supported by the Estonian Research Council ETAG grant PRG 2100.

References

- [1] P. Drotar et al. "Evaluation of handwriting kinematics and pressure for differential diagnosis of parkinson's disease". In: *Artificial Intelligence in Medicine* (2016).
- [2] Sergei Zarembo et al. "CNN based analysis of the Luria's alternating series test for Parkinson's disease diagnostics." In: 2021.
- [3] C. Kotsavasiloglou et al. "Machine learning-based classification of simple drawing movements in Parkinson's disease". In: *Biomedical Signal Processing and Control* (2017).
- [4] Elisa Tamm. *Associations Between Fine Motor Tests among Parkinson's Disease Patients and Control Groups*. 2020.
- [5] Artur-Aleksander Pärnoja. *Application for the analysis of drawing and writing tests*. 2020.
- [6] Sergei Zarembo. *Adaptation Affect on Fine Motor Motions*. 2019.
- [7] Jemin Desai. *Web Application Vs Desktop Application: Pros and Cons*. [Accessed: 01-12-2023]. 2023. URL: <https://positiwise.com/blog/web-application-vs-desktop-application-pros-and-cons>.
- [8] GlobalStats. [Accessed: 01-12-2023]. 2023. URL: <https://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-202208-202306>.
- [9] [Accessed: 01-12-2023]. 2023. URL: <https://wiki.python.org/moin/PyQt>.
- [10] [Accessed: 01-12-2023]. 2023. URL: <https://keras.io/about/>.
- [11] [Accessed: 01-12-2023]. 2023. URL: <https://docs.python.org/3/tutorial/venv.html>.
- [12] [Accessed: 01-12-2023]. 2023. URL: <https://pyinstaller.org/en/stable/>.
- [13] [Accessed: 01-12-2023]. 2023. URL: <https://pandas.pydata.org/about/index.html>.
- [14] [Accessed: 01-12-2023]. 2023. URL: https://numpy.org/doc/stable/user/absolute_beginners.html.

Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis¹

I Ilja Jakuboviš

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Application for the Analysis of Drawing and Writing Tests”, supervised by Prof. Sven Nõmm
 - 1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
 - 1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons’ intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

27.05.2024

¹The non-exclusive licence is not valid during the validity of access restriction indicated in the student’s application for restriction on access to the graduation thesis that has been signed by the school’s dean, except in case of the university’s right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 – Formulas for the computation of features

Slope

$$k = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \quad (1)$$

Alpha angle

$$\alpha = \arctan(k) \quad (2)$$

Rotational angle

$$\phi_i = \pi + \alpha_{i-1} - \alpha_i \quad (3)$$

Yaw angle

$$\psi_i = \alpha_i - \alpha_{i-1} \quad (4)$$

Forward difference

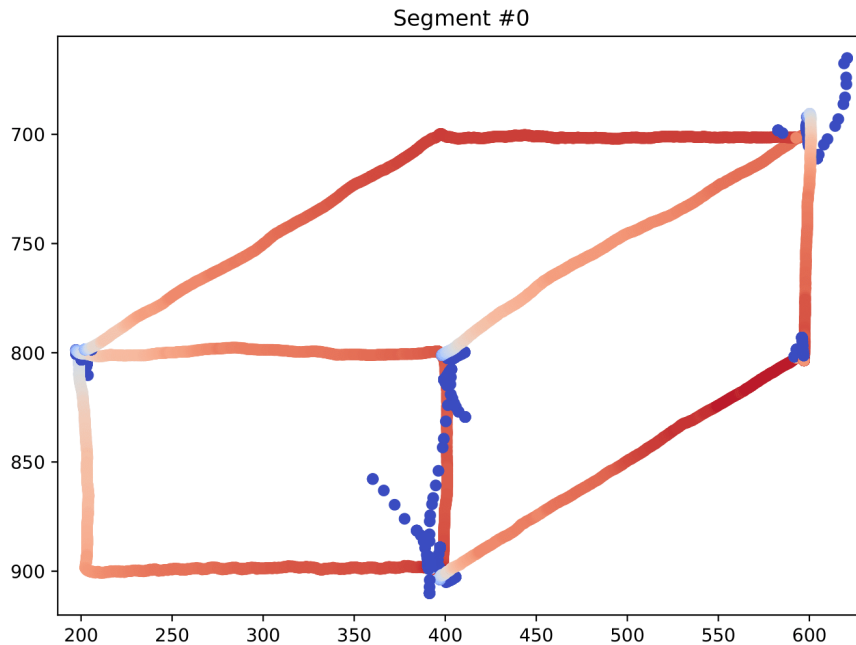
$$\Delta_h[f](x) = f(x + h) - f(x) \quad (5)$$

Appendix 3 – A screenshot of calculated features in the application

Spatial-time features		
Distance: 3.24e+03 mm	Duration: 2.34e+01 s	
Kinematic features		
Velocity mass: 1.05e+06 mm/s	Velocity std: 5.36e+02 mm/s	Velocity min: 0.00e+00 mm/s
Velocity mean: 2.63e+02 mm/s	Velocity median: 9.65e+01 mm/s	Velocity max: 1.09e+04 mm/s
Acceleration mass: -6.45e+08 mm/s ²	Acceleration std: 5.26e+05 mm/s ²	Acceleration min: -1.05e+07 mm/s ²
Acceleration mean: -1.61e+05 mm/s ²	Acceleration median: 0.00e+00 mm/s ²	Acceleration max: 1.23e+06 mm/s ²
Jerk mass: 7.15e+11 mm/s ³	Jerk std: 5.61e+08 mm/s ³	Jerk min: -8.66e+08 mm/s ³
Jerk mean: 1.79e+08 mm/s ³	Jerk median: 0.00e+00 mm/s ³	Jerk max: 1.08e+10 mm/s ³
Pressure features		
Pressure difference mass: 0.00e+00 units	Pressure difference std: 2.00e-02 units	Pressure difference min: -6.00e-01 units
Pressure difference mean: 0.00e+00 units	Pressure difference median: 0.00e+00 units	Pressure difference max: 1.60e-01 units
Yank mass: -1.09e+03 units/s	Yank std: 1.13e+01 units/s	Yank min: -4.77e+02 units/s
Yank mean: -2.70e-01 units/s	Yank median: 0.00e+00 units/s	Yank max: 6.67e+01 units/s
Tug mass: 6.56e+05 units/s ²	Tug std: 1.17e+04 units/s ²	Tug min: -2.39e+05 units/s ²
Tug mean: 1.64e+02 units/s ²	Tug median: 0.00e+00 units/s ²	Tug max: 5.13e+05 units/s ²
Geometric features		
Azimuth difference mass: 0.00e+00 rad	Azimuth difference std: 1.58e+00 rad	Azimuth difference min: -3.83e+01 rad
Azimuth difference mean: 0.00e+00 rad	Azimuth difference median: 0.00e+00 rad	Azimuth difference max: 3.80e+01 rad
Latitude difference mass: 0.00e+00 rad	Latitude difference std: 2.27e+00 rad	Latitude difference min: -4.73e+01 rad
Latitude difference mean: 0.00e+00 rad	Latitude difference median: 0.00e+00 rad	Latitude difference max: 4.84e+01 rad
Alpha angle mass: 1.80e+02 rad	Alpha angle std: 8.00e-01 rad	Alpha angle min: -1.57e+00 rad
Alpha angle mean: 6.00e-02 rad	Alpha angle median: 0.00e+00 rad	Alpha angle max: 1.57e+00 rad
Rotational angle mass: 8.98e+03 rad	Rotational angle std: 6.90e-01 rad	Rotational angle min: 2.00e-02 rad
Rotational angle mean: 3.14e+00 rad	Rotational angle median: 3.14e+00 rad	Rotational angle max: 6.25e+00 rad
Yaw angle mass: 5.68e+00 rad	Yaw angle std: 6.90e-01 rad	Yaw angle min: -3.11e+00 rad
Yaw angle mean: 0.00e+00 rad	Yaw angle median: 0.00e+00 rad	Yaw angle max: 3.12e+00 rad

Figure 10. A screenshot of calculated features in the application.

Appendix 4 – An example of a PDF report generated from the application



Patient ID: Test Patient
 Time: 2019-10-12 14:15:56 +0000
 Hand: Right
 Neural network output: 1.0
 Note: Sample text

Features

Distance: 3.24e+03 mm	Duration: 2.34e+01 s	
Velocity mass: 1.05e+06 mm/s	Velocity std: 5.36e+02 mm/s	Velocity min: 0.00e+00 mm/s
Velocity mean: 2.63e+02 mm/s	Velocity median: 9.65e+01 mm/s	Velocity max: 1.09e+04 mm/s
Acceleration mass: -6.45e+08 mm/s ²	Acceleration std: 5.26e+05 mm/s ²	Acceleration min: -1.05e+07 mm/s ²
Acceleration mean: -1.61e+05 mm/s ²	Acceleration median: 0.00e+00 mm/s ²	Acceleration max: 1.23e+06 mm/s ²
Jerk mass: 7.15e+11 mm/s ³	Jerk std: 5.61e+08 mm/s ³	Jerk min: -8.66e+08 mm/s ³
Jerk mean: 1.79e+08 mm/s ³	Jerk median: 0.00e+00 mm/s ³	Jerk max: 1.08e+10 mm/s ³
Pressure difference mass: 0.00e+00 units	Pressure difference std: 2.00e-02 units	Pressure difference min: -6.00e-01 units
Pressure difference mean: 0.00e+00 units	Pressure difference median: 0.00e+00 units	Pressure difference max: 1.60e-01 units
Yank mass: -1.09e+03 units/s	Yank std: 1.13e+01 units/s	Yank min: -4.77e+02 units/s
Yank mean: -2.70e-01 units/s	Yank median: 0.00e+00 units/s	Yank max: 6.67e+01 units/s
Tug mass: 6.56e+05 units/s ²	Tug std: 1.17e+04 units/s ²	Tug min: -2.39e+05 units/s ²
Tug mean: 1.64e+02 units/s ²	Tug median: 0.00e+00 units/s ²	Tug max: 5.13e+05 units/s ²
Azimuth difference mass: 0.00e+00 rad	Azimuth difference std: 1.58e+00 rad	Azimuth difference min: -3.83e+01 rad
Azimuth difference mean: 0.00e+00 rad	Azimuth difference median: 0.00e+00 rad	Azimuth difference max: 3.80e+01 rad
Latitude difference mass: 0.00e+00 rad	Latitude difference std: 2.27e+00 rad	Latitude difference min: -4.73e+01 rad
Latitude difference mean: 0.00e+00 rad	Latitude difference median: 0.00e+00 rad	Latitude difference max: 4.84e+01 rad
Alpha angle mass: 1.80e+02 rad	Alpha angle std: 8.00e-01 rad	Alpha angle min: -1.57e+00 rad
Alpha angle mean: 6.00e-02 rad	Alpha angle median: 0.00e+00 rad	Alpha angle max: 1.57e+00 rad
Rotational angle mass: 8.98e+03 rad	Rotational angle std: 6.90e-01 rad	Rotational angle min: 2.00e-02 rad
Rotational angle mean: 3.14e+00 rad	Rotational angle median: 3.14e+00 rad	Rotational angle max: 6.25e+00 rad
Yaw angle mass: 5.68e+00 rad	Yaw angle std: 6.90e-01 rad	Yaw angle min: -3.11e+00 rad
Yaw angle mean: 0.00e+00 rad	Yaw angle median: 0.00e+00 rad	Yaw angle max: 3.12e+00 rad

Figure 11. An example of a PDF report generated from the application.

Appendix 5 – A link to GitHub repository

<https://github.com/illyria1/drawing-tests>