User's Guide for LSFB Project

Documentation of the Francophone Belgian Sign Language Project

Loïc Franck KAMGAIN MEUPIAP

INFO B302 - Individual Project (2024 - 2025)
Faculty of Computer Science, University of Namur, Belgium

About this guide:

This **User's Guide** is intended to provide end users with all the necessary information to fully understand and utilize the **LSFB** (Langue des Signes Francophone de Belgique) project. This guide will accompany the user throughout their experience with the software, from installation to the advanced use of its features. It is designed to be both detailed and accessible so that any user, regardless of their technical skills, can interact effectively with the platform.

The guide covers the main functionalities of the software, installation steps, and instructions on how to navigate the user interface and make the most of the available data. Additionally, it provides solutions to common issues and includes a FAQ section to answer frequently asked questions by users.

Purpose of the guide:

The primary objective of this guide is to enable users to:

- **Install and configure** the software on their machine.
- **Search and view** signs in Langue des Signes Francophone de Belgique (LSFB).
- **Access videos and annotations** in 3D related to signs and associated gestures.
- **Analyze data** related to signs, their frequency, and their signers using the statistics generated by the software.

Target Audience:

This guide is aimed at anyone wishing to use the LSFB platform to explore, search, and analyze Langue des Signes Francophone de Belgique. It is particularly useful for:

- **Researchers and linguists** who want to study LSFB interactively and statistically.
- **Students** in linguistics or related fields.
- **Professionals in sign language**, including trainers and interpreters.

The guide is designed to be understood by users with basic computer knowledge, but no prior skills in programming or database management are required.

Contents

1 Preface

The LSFB (French Sign Language of Belgium) project is a digital platform dedicated to the visualization, exploration, and analysis of the French Sign Language of Belgium. This documentation aims to guide users in the installation and use of this platform, and to provide the necessary information for easy use. It is structured to meet the needs of both end users and developers wishing to contribute to or extend the project.

2 What is this software?

2.1 Its objective

The LSFB software aims to provide an interactive and dynamic platform for exploring videos and annotations of signs in the French Sign Language of Belgium (LSFB). Its goal is to make a corpus of videos and annotations accessible to a wide audience, enabling the linguistic and visual analysis of signs. The software is designed to:

- Assist in the study and analysis of signs in LSFB.
- Provide tools for research, 3D visualization, and statistical analysis of data.

2.2 Its main features

The software offers several features that allow for a complete exploration of the video and annotation database. The main features are as follows:

- **Sign Search**: Users can search for signs by glosses or phrases, with results detailing related videos and relevant segments.
- **2D Visualization**: The software allows you to visualize signs in 2D from videos and pose data (using MediaPipe).
- **Statistics**: The project generates detailed statistics on signs, their frequency, as well as on the signers.
- **Dynamic Display**: Users can explore videos and poses through an interactive interface, including controls to navigate through videos and segments.

2.3 What it is not

LSFB software is not designed to manage large-scale video content or to edit videos. It does not include features to:

- **Upload** or **edit** videos.
- **Manage annotations** outside the analysis context provided by the application.

It focuses solely on exploring, visualizing and analyzing videos and annotations.

2.4 Target audience

The target audience for this software includes

- **Linguistics researchers**, particularly those specializing in sign language and visual languages.
- **Students** in linguistics, computer science, or similar fields who wish to explore or analyze LSFB.
- **Sign language professionals**, including instructors and interpreters, who wish to study the characteristics of this visual language.

The software is designed to be accessible to all skill levels, although some familiarity with computers and databases is a plus.

2.5 Usage license

The LSFB software is distributed under the **MIT License**, which allows users to use, modify, and adapt it to their needs without restriction, including the right to copy, merge, publish, distribute, sublicense, and sell copies of the software. However, commercial use of derivative works of the software requires prior permission. The software is provided "as is," without warranty of any kind, either express or implied, including, but not limited to, the warranties of merchantability, fitness for a particular purpose, and non-infringement. In no event shall the authors or copyright holders be liable for any claims, damages, or other liabilities, whether in contract, tort, or otherwise, arising out of the use of the software or the performance of other operations in connection with the software.

2.6 Legal notices

- **Privacy**: This software respects privacy rules by not having access to users' personal information. No personal data is collected.
- **Disclaimer**: Users are responsible for their use of the videos and annotations. The project assumes no liability for misuse or damage related to the use of the software.

3 How to install it?

3.1 Where is the installation package?

The installation package is available as **source code** via the project's GitHub repository. The user can clone the repository or download a ZIP archive.

3.2 Prerequisites

- **Operating Systems**: The project is compatible with **Windows**, **macOS** and **Linux.**
- **Necessary libraries**:
  - Python 3.12+ installed on your machine.
  - FastApi for the backend
  - SQLAlchemy for database management.
  - PostgreSQL as a database management system.
  - TailwindCSS for frontend formatting.
  - JavaScript and associated libraries

requirements.txt file: This file contains all the dependencies needed to install the Python libraries.

3.3 Servers

- **Web Server**: To run the **FastAPI** application, you need to install **Uvicorn** or another **ASGI**-compatible server. You can install it with the following command: **pip install uvicorn**
- **Database Server: PostgreSQL** must be configured to store videos and annotations and any database items

3.4 Software components

- **Backend**: FastAPI and SQLAlchemy for query and database management.
- **Frontend**: HTML, CSS with TailwindCSS, and JavaScript for dynamic interface management.
- **Other tools**: MediaPipe for pose detection, Pyplot and Dash for 3D (actually 2D) display.

3.5 Accreditations

No special access is required to use the application, but a valid connection to a PostgreSQL database is essential. You must have access rights to a PostgreSQL server and configure the database correctly to work with the application. A user with the necessary rights must be created in PostgreSQL.

3.6 Disk space, infrastructure, hardware, and network

- **Hard disk space**: The project may require approximately **2TB** of space for large videos, annotations, pose files, and other files needed to complete the project.
- **Necessary infrastructure**: A working PostgreSQL database server and a web server to host the application.
- **Hardware**: No specific hardware required, but a machine with good processor for video processing is recommended.
- **Network**: A stable connection to the database and frontend is required.

3.7 Installation instructions

Steps for installation.

- Clone the repository: `git clone https://github.com/UNamurCSFaculty/2425_INFOB318_LSD_C`
- Install dependencies via `pip install -r requirements.txt`.
- Configure PostgreSQL with a suitable user and database. The first user is already configured and can be found in the `.env` file of the project.
- Connection to databases using information found in the environment variables:
  - **DB_HOST=localhost**
  - **DB_USER=your_username**
  - **DB_PASSWORD=your_password**
  - **DB_PORT=5432**
  - **DB_NAME=lsfb_database**
- Start the server with the following command: `python main.py`

3.8 Where to place the files?

Files must be placed in a local directory, for example C:/Project/LSFB or /home-/user/LSFB. Videos and annotations must be accessible via the path defined in the configuration file.

4 How to use it?

4.1 By features

Using the software by features.

- **Search by glosses or phrases**: In the interface, type a term into the search bar to find videos related to that symbol or phrase. These initial results can be refined using filters.
- **Viewing videos**: Select a video to view it with the 2D sketch representation of the sign (isolated dataset) or signs (continuous dataset) present in the video
- **Statistics**: View detailed statistics on signs, glosses, and signers, including their frequency, average duration, and number of appearances.

4.2 Start here (How to get started)

- Install the software as described in the previous section.
- Launch the server and access the interface through your browser at http://127.0.0.1:8000/.
- Start by searching for a sign or phrase to see results and explore videos.

4.3 By use case

Simple Scenario Example: Searching for a Specific Sign in Datasets with Filters

Here are the steps to follow to search for a sign in the LSFB software and display the associated results. This scenario covers the search for a **gloss** and a **sentence**, with a choice between the **continuous dataset** and the **isolated dataset**, while also allowing the refinement of search results with filters after the global search results are displayed.

4.3.1 Step 1: Access the Homepage

1. **Launch the application**: Open your browser and go to the address `http://127.0.0.1:8000/`

2. **Enter the search term**: On the homepage, you will see a **search bar** where you need to enter the term you want to search for. This term can be either a **gloss** (isolated word) or a **complete sentence** or **expression**.

- **Gloss**: A specific word (e.g., "house").
- **Sentence**: A full sequence of signs (e.g., "I am going to school").

4.3.2 Step 2: Select the Dataset

1. **Choose the dataset after entering the search term**: Once you have entered the search term (gloss or sentence), you will need to choose in which **dataset** you want to perform the search:
   - **Continuous Dataset**: Contains video segments with complete sentences. This dataset is used only for searching sentences.

- **Isolated Dataset**: Contains only isolated words (glosses). This dataset can be used for searching glosses, but not sentences.

2. **Click on "Search"**: Once you have selected the appropriate dataset (depending on the search type), click on **Search**.

4.3.3 Step 3: Viewing the Global Search Results

1. **Access the results**: After clicking on **Search**, the system will redirect you to the page showing results for the selected dataset (either the **continuous dataset** or the **isolated dataset**).

    - If you searched for a **gloss**, you can search it in **both datasets**. If you searched for a **sentence**, it can only be searched in the **continuous dataset**, as it contains videos with complete sentences.

2. **View the videos**: You will see a list of **videos** corresponding to your search, along with their associated annotations. You can **click on a video** to watch it. The videos come with **pose annotations**, which can be visualized in 2D.

4.3.4 Step 4: Refining the Results with Filters

Once the global search results are displayed, you can **refine** these results using the available filters.

1. **Available filters**:

- **Signer**: Filter the results to see only the videos signed by a specific signer. This filter is available in **both datasets**.
- **Sign Type**: Filter the results by **sign type** (normal, special) to restrict videos based on the type of gesture or sign performed.
- **Hand Type**: Filter by **hand type** (left, right, both) to see videos where the sign is performed with the left hand, the right hand, or both.
- **Duration**: You can also filter the videos by their **minimum or maximum duration**, allowing you to refine the results based on the length of the videos or segments.

2. **Apply the filters**: After selecting the desired filters, click on **Apply Filters** to see the refined results according to the criteria you defined.

4.3.5 Step 5: Changing the Dataset (if needed)

1. **Return to the homepage to change the dataset**:

- If you want to change the **dataset** (e.g., from the **isolated dataset** to the **continuous dataset**), you must return to the **homepage**.
- It is important to note that the dataset selection must be done **before** starting a new search. You will need to go back to the homepage, select the new dataset, and then start the search process again.

4.3.6 Step 6: Explore the Videos and View the Statistics

1. **Explore the videos**: On the results page, you can click on the videos to watch them. The videos come with annotations and 3D poses.
2. **View the statistics**: Depending on the search type (gloss or sentence), detailed **statistics** will be available, such as the frequency of the gloss or sentence, the average duration of the clips, and other relevant information about the signs.

4.3.7 Summary of the Simple Scenario

1. **Enter a search term** in the search bar (gloss or sentence).
2. **Select the dataset** (either continuous or isolated) after entering the term.
3. **Click on "Search"** to retrieve the results.
4. Explore the global results displayed for the search.
5. Refine the results with filters (signer, sign type, hand type, duration, etc.).
6. View the videos and **statistics**.
7. Return to the homepage to change the dataset if needed and start a new search.

This simple process allows you to easily navigate between the videos and better understand the structure of the Langue des Signes Francophone de Belgique, while offering maximum **flexibility** through filters to refine results and navigate through available videos.

4.4 What to do in case of errors?

When using the LSFB software, users may encounter technical errors related to installation, configuration, or usage of the application. This guide provides practical solutions for resolving the most common errors.

4.4.1 Errors related to the database

Database connectivity is crucial for the proper functioning of the application. The most frequent errors are usually related to incorrect configuration or access issues with the PostgreSQL database.

1.1. Error: "Failed to connect to the database"

- **Possible issue**: The connection to the database fails, usually due to an incorrect username, password, or server address in the configuration.
- **Solution**:

- **Check the `.env` file**: Ensure that the environment variables are correctly set in the `.env` file. This file should contain the following information (replace with your own values):

```
DB_HOST=localhost
DB_USER=your_db_user
DB_PASSWORD=your_db_password
DB_PORT=5432
DB_NAME=lsfb_database
```

- **Check database access rights**: Make sure the PostgreSQL user defined in the `.env` file has the necessary rights to connect to the database.
- **Test the connection manually**: If the database is local, you can test the connection via the terminal with the following command:

```
psql -h localhost -U your_db_user -d lsfb_database
```

  This will prompt you for the password. If the connection fails, there may be an issue with the password or connection settings in PostgreSQL.
- **Ensure PostgreSQL is running**: If the PostgreSQL server is not running, you can start it with the following command (for Linux):

```
sudo service postgresql start
```

  For Windows, open the service manager and ensure the PostgreSQL service is running.

1.2. Error: "Table or column does not exist in the database"

- **Possible issue**: This typically occurs if the database has not been properly initialized, or the tables were not created.
- **Solution**:
  - **Run the database initialization command**: If the tables were not created, run the initialization script using SQLAlchemy. You can do this by running the following command in the terminal:

```
python db_init.py
```

    This script will create all necessary tables in the PostgreSQL database.
  - **Check migrations**: If changes have been made to the database structure (adding new tables, columns, etc.), make sure to run the necessary migrations. Use **Alembic** or another migration tool to apply changes.

4.4.2 Errors related to videos

The LSFB project relies on video files stored in specific directories. Errors related to videos may occur if the files are not properly placed or the access paths are misconfigured.

2.1. Error: "Video file not found"

- **Possible issue**: The specified video file cannot be found at the expected location.
- **Solution**:
  - **Check the video access paths**: Ensure that the videos are present in the directories defined by the system and that the access paths are correctly recorded in the database. The videos should be located in directories specified in your configuration (e.g., in `./data/videos`).
  - **Test the path manually**: Go to the folder where the videos are supposed to be located and verify that the video file is present. For example, if a video is named `video_01.mp4`, you should ensure that the file is in `./data/videos/video_01.mp4`.

2.2. Error: "The video does not load correctly in the player"

- **Possible issue**: There may be an issue with the video format, or the access path may be misdefined in the frontend interface.
- **Solution**:
  - **Check the video format**: Ensure that the videos are in `MP4` format. This format is widely supported and compatible with HTML5 video players.
  - **Check the video URL**: If the video path is misconfigured, check the `main.py` file to ensure that the relative paths for the videos are correctly defined. The paths should be dynamically generated in the backend and then sent to the frontend.

5 Frequently Asked Questions (FAQ)

1. **What should I do if the software doesn't start after installation?**
   Answer: Ensure that all dependencies are installed by running the following command:

   ```
   pip install -r requirements.txt
   ```

   Additionally, verify that PostgreSQL is installed and the database is correctly configured. If the problem persists, check the error logs generated by FastAPI in the console to identify the source of the issue.

2. **How can I add new videos to the database?**
   Answer: To add new videos to the database, prepare the video files and annotations in the correct format. Use the video insertion script (`insert_db.py`) by specifying the directory of the videos and the corresponding annotation files. Ensure that the videos are properly named and the annotations are formatted as JSON.

3. **Why are the videos not loading in the interface?**
   Answer: If the videos are not loading, check the following:
   - Ensure that the video files exist at the specified location.
   - Verify that the video paths are correctly configured in the database.
   - Confirm that your web server (Uvicorn) is running correctly.
   - Check the file permissions to ensure the application has access to the video files.
4. **Can I modify the videos or annotations after they have been inserted into the database?**
   Answer: No, the software is designed to *visualize* and *analyze* the videos and annotations, but it does not allow direct modification of the videos or annotations once they have been inserted. If you need to modify the data, you must directly modify the source files (videos or annotations) and reinsert them into the database using the appropriate scripts.
5. **How does the search for glosses and phrases work?**
   Answer: The search for *glosses* allows you to find videos corresponding to a specific sign using a simple term (e.g., "house"). The search for *phrases* is more advanced and allows you to search for complete sequences of signs, providing more detailed results. If you are searching for a phrase, the system will search through the subtitle annotations associated with that phrase.

6 What artifact/support/format?

6.1 Nature of this documentation

The documentation is available as interactive web-based Latex code, downloadable as a PDF for offline viewing. It will be

6.2 Documentation versioning

The documentation was written all at once, after the project was completed. Unlike the project code, which was versioned over time, the documentation was not updated in parallel. It therefore represents the final and complete version of the project as it was completed, without any successive changes related to code versions.

6.3 Formats and accessibility

- **Formats**: Documentation in PDF.
- **Accessibility**: The documentation will be accessible online and downloaded as PDFs.

7 Glossary

The glossary contains definitions of key technical terms and concepts used in this guide, as well as their detailed explanations. This section helps clarify the notions that are specific to the field of sign language, multimodal data processing, and the technologies used in the LSFB project.

Glosses A **gloss** in the context of sign language refers to a word or sequence of signs that represents an idea or action in a sign language sentence. In computing, a gloss can also be considered as a textual annotation associated with a visual sign.

In the LSFB project, glosses are used to search and identify specific signs in videos. For example, searching for "house" in the search bar allows users to retrieve videos where this sign is used.

Pose A **pose** refers to the specific position of the body or hands of a signer when performing a sign. These poses are often represented as spatial coordinates in a 3D space, captured by motion tracking systems like **MediaPipe**.

In LSFB, poses are extracted from the videos using motion detection algorithms, and then visualized in 3D to allow users to study the gestures of the signers. Each pose is associated with a specific video segment and is used for sign analysis.

Signer A **signer** is a person who uses sign language to communicate. In this project, a signer is the individual performing the signs recorded in the videos.

Each video instance is associated with a unique signer, and this signer is recorded in the database with a specific identifier. The software allows analysis of the variability of signs performed by each signer.

Uvicorn **Uvicorn** is an ASGI (Asynchronous Server Gateway Interface) web server for Python, specifically designed to run asynchronous web applications like those built with FastAPI.

Uvicorn is used to run the web server for the LSFB application. When a user accesses the application via their browser, Uvicorn receives the requests and forwards them to FastAPI for processing.

3D Pose A **3D pose** is the three-dimensional representation of the position of the body or limbs of a person. In the context of sign language, the pose describes the movements and gestures made to represent a sign.

In the LSFB project, 3D poses are captured using MediaPipe, then visualized using Pyplot or Dash. They are associated with videos to allow detailed analysis of the signer's gestures.

Video Segment A **video segment** refers to a portion of a video where a particular sign or series of signs is performed. Each segment is defined by a start and end time interval.

In the LSFB project, video segments are used to precisely describe when and where a sign appears in the video. These segments are then displayed to the user, allowing them to jump directly to the part of the video that contains the searched sign.

8 Bibliography

Throughout the development of the LSFB project, various sources were consulted to ensure an efficient and high-quality implementation. These resources include official documentation, forums, online guides, and AI tools that facilitated research, coding, and solving technical issues. Below are the primary resources used:

8.1 FastAPI Official Documentation

Reference: `https://fastapi.tiangolo.com/`

Usefulness: The official FastAPI documentation was a key resource for understanding how to structure the backend API of the project. FastAPI, being a modern and fast framework for building APIs in Python, provides features like data validation, error handling, and automatic documentation generation. The fundamental concepts, as well as best practices for working with databases and SQLAlchemy models, were directly taken from this resource.

Importance: A deep understanding of how to integrate FastAPI with PostgreSQL was essential for creating the backend.

8.2 SQLAlchemy Official Documentation

Reference: `https://www.sqlalchemy.org/`

Usefulness: SQLAlchemy was used as the ORM (Object-Relational Mapping) to interact with the PostgreSQL database. The SQLAlchemy documentation was consulted to implement the necessary data models for managing videos, annotations, and poses. It also proved invaluable for understanding how to optimize queries and manage asynchronous sessions.

Importance: Proper management of relationships between different tables in the database, such as between videos and poses, was made possible with SQLAlchemy.

8.3 PostgreSQL Documentation

Reference: `https://www.postgresql.org/docs/`

Usefulness: PostgreSQL is the relational database management system used in this project. The PostgreSQL official documentation was referred to for best practices in database management, query optimization, and configuring the database server.

Importance: Proper configuration of the database and the use of the correct data types and indexes were made possible thanks to this documentation.

8.4 Python Official Documentation

Reference: `https://docs.python.org/3/`

Usefulness: The official Python documentation was used for all aspects of programming, particularly for file handling, modules, and data manipulation. It also allowed understanding how to use libraries for file processing and error handling.

Importance: Although the project primarily uses external frameworks like FastAPI, understanding Python fundamentals is essential for working with these tools.

8.5 Tailwind CSS Documentation

Reference: `https://tailwindcss.com/docs`

Usefulness: Tailwind CSS was used to style the user interface of the project. The Tailwind documentation helped in designing a clean, responsive interface by utilizing utility classes for layout, animations, and visual components.

Importance: Tailwind CSS played a crucial role in creating a modern and easy-to-navigate user interface for researchers and students, ensuring a smooth user experience.

8.6 ChatGPT

Reference: `https://openai.com/chatgpt`

Usefulness: ChatGPT was used to solve technical issues and to provide guidance on code structure and the implementation of new features. ChatGPT also helped in improving documentation and explaining complex concepts, such as integrating different libraries and tools.

Importance: As an AI tool, ChatGPT facilitated the development phase by providing quick answers and tailored suggestions to challenges faced in managing the backend, SQL queries, and 3D animation features.

8.7 Google Search

Reference: `https://www.google.com/`

Usefulness: Google was used to search for code examples, solutions to common errors, and information about specific libraries. Google also helped discover external tutorials and blog posts on topics such as video handling, database models, and 3D visualization.

Importance: Google helped in discovering external resources that were not directly covered by official documentation.

Importance: These resources enriched the understanding of tools used in the project and explored alternative approaches for solving certain technical problems.

This section provides an exhaustive list of all sources consulted during the development of the LSFB project. It includes official documentation, online guides, AI tools, and technical forums, each of which contributed to the success of the project by providing essential information, troubleshooting support, and guidance on best practices.

9 Index

Index of terms.