mathematics

Article

The Serbian Sign Language Alphabet: A Unique Authentic Dataset of Letter Sign Gestures

Mladen Radaković [1,*], Marina Marjanović [2], Ivana Ristić [3], Valentin Kuleto [1], Milena P. Ilić [4] and Svetlana Dabić-Miletić [5]

[1] Information Technology School—ITS, 11000 Belgrade, Serbia; valentin.kuleto@its.edu.rs
[2] Faculty of Technical Sciences, Singidunum University, 11000 Belgrade, Serbia; mmarjanovic@singidunum.ac.rs
[3] Teacher Education Faculty, University of Priština in Kosovska Mitrovica, 38219 Kosovska Mitrovica, Serbia; ivana.ristic@pr.ac.rs
[4] Information Technology School ITS—Belgrade, LINK Group Belgrade, Faculty of Contemporary Arts Belgrade, University Business Academy in Novi Sad, 11000 Belgrade, Serbia; milena.ilic@its.edu.rs
[5] Faculty of Transport and Traffic Engineering, University of Belgrade, 11000 Belgrade, Serbia; cecad@sf.bg.ac.rs
* Correspondence: mladen.radakovic@its.edu.rs

Abstract: Language barriers and the communication difficulties of individuals with developmental disabilities are two major causes of communication problems that societies worldwide encounter. A particularly challenging group is hearing-impaired people who have difficulties with communication, reading, writing, learning, and social interactions, which have a substantial impact on their quality of life. This article focuses on detailing a Serbian Sign Language alphabet database and the method for creating it in order to provide a foundation for answering the various societal challenges of persons who use the Serbian language. In front of a computer camera, 41 people performed Serbian Sign Language sign movements that replicated the Serbian alphabet for this study's aims. Hand and body key points were identified using the recorded video clips, and the numerical values of the identified key points were then stored in a database for further processing. In total, 8.346 video clips of people making recognized hand gestures were gathered, processed, classed, and archived. This paper provides a thorough technique that may be applied to comparable tasks and details the process of constructing a dataset based on Serbian Sign Language alphabet signs. This dataset was created using custom-made Python 3.11 software. Data regarding dynamic video clips that capture entire subject movement were incorporated into this dataset to fill in the gaps in other similar efforts based on static photographs. Thus, the purpose of this investigation is to employ innovative technology to support the community of hearing-impaired people in areas such as general inclusion, education, communication, and empowerment.

Keywords: Serbian Sign Language; hearing-impaired persons; inclusion; machine learning; artificial intelligence; gestures

MSC: 68T45

Citation: Radaković, M.; Marjanović, M.; Ristić, I.; Kuleto, V.; Ilić, M.P.; Dabić-Miletić, S. The Serbian Sign Language Alphabet: A Unique Authentic Dataset of Letter Sign Gestures. *Mathematics* **2024**, *12*, 525. https://doi.org/10.3390/math12040525

Academic Editor: Jie Wen

Received: 12 January 2024
Revised: 2 February 2024
Accepted: 6 February 2024
Published: 8 February 2024

1. Introduction

The recognition of sign language motions presents a significant challenge in the discipline of computer vision in the rapidly evolving world of artificial intelligence (AI) and machine learning (ML) models [1]. Visual sign language recognition is a complex field for research in computer vision. Given that every sign language in the world has unique and distinct gestures, it is difficult to generalize potential problems for all of them. This study and dataset that was collected, assessed, processed, and presented form the basis for the development of fresh machine learning models and AI systems that will assist hearing-impaired people in Serbia to communicate with others. The provided algorithm is generalizable, and the technique can be used to create and gather similar datasets based on recording and identifying human motions. These types of data can be utilized as the foundation for the development of machine learning models that might be used in a variety of applications and projects. Machine learning models that have been taught to recognize generic or specialized hand gestures can be used to develop educational applications, games, or communication platforms for hearing-impaired persons. They can also be used in a variety of devices such as mobile phones, tablets, desktop computers, video conferencing devices, TVs, and others. Various sensor-based devices, such as Intel RealSenseTM, Microsoft Kinect, LeapMotion Controller, and others [2], have also been created for similar computer vision purposes. The assignment defined in this article is adaptable to numerous sign languages and other computer vision purposes.

Hearing-disabled people have numerous communication issues with the rest of the community. Research findings indicate that knowledge of sign language and the availability of sign language interpreters represent one of the indicators of the quality of life from the perception of deaf people [3,4]. Helping the hearing disabled learn the sign language alphabet using interactive computer software or to convert their gestures into computer-recognized alphabets is how SSL software can help with these challenges. The SSL dataset is designed to serve as a foundation for the creation of various machine learning-based software. Future machine learning models created and based on SSL can be used on different hardware platforms (mobile phones, tablets, PCs, etc.), offering various functionalities (recognizing presented SSL alphabets and making use of that information).

Along with the usual use of sign language in communication, dactylology and the finger alphabet are also used in case a sign for a certain term does not exist or the user does not know an adequate sign. Dactylology is an important part of sign languages because it allows people to spell out words or convey certain thoughts using manual signs that represent letters or words. In summary, dactylology and sign language are related because dactylology is a method of communication that uses physical signs and is an essential component of sign language. Dactylology is used for spelling people's first and last names, names of streets, squares, geographical terms, and terms from the field of politics and professional and non-academic terminology [5].

Creating tools and services that can help include the hearing impaired in work and everyday life can contribute to reducing the current barriers mostly present in communication with others. According to data from the World Health Organization in 2020, more than 450 million people were hearing impaired; therefore, sign language was their basic means of communication with others [6]. Due to difficult communication, hearing-impaired people face numerous obstacles when it comes to employment, socializing, and fitting into the wider social community, and, as a result, they are often ostracized from society. Therefore, as an imperative of social sustainability, the need to include these people in all segments of the social environment is emphasized. After the Serbian National Assembly passed the "Law on the Use of Sign Language" in 2015 [7], there have not been enough attempts to address issues concerning linguistic processes, such as the standardization of Serbian Sign Language, the creation of new lexical items, dictionaries, grammar reference books, etc. [8]. For this reason, numerous tools and applications are being developed to overcome existing barriers and create the basis for improving the position of hearing-impaired people in the wider social community. According to Marković, M. [9], 2% of the population in Serbia reported hearing problems of any level. Based on the latest available data for the census of the population from 2011, there were 144.648 persons with hearing disabilities. This corresponds to more than 25% of the population suffering from any disability. According to the previous findings, almost 30% of habitants in Serbia younger than 65 are in the hearing-impaired group, which is a serious social challenge [9].

Recognition of sign language motions presents a significant challenge in the discipline of computer vision in the rapidly evolving world of artificial intelligence (AI) and machine learning (ML) models. Given that every language in the world has unique and distinct gestures, it is difficult to generalize potential problems for all of them. This study was focused on collecting datasets that were assessed, processed, and presented to aid the creation of fresh ML models and AI systems that would assist hearing-impaired people in communicating with others. The provided algorithm is generalizable, and the technique can be used to create and gather similar datasets based on recording and identifying human motions. These types of data can be utilized as the foundation for the development of ML models that could be used in a variety of applications and projects. ML models that have been taught to recognize generic or specialized hand gestures can be used to develop educational applications, games, or communication platforms for hearing-impaired persons. They can also be used in a variety of devices such as mobile phones, tablets, desktop computers, video conferencing devices, TVs, and others. This type of work is easily adaptable to numerous sign languages and other computer vision purposes.

Bearing in mind the numerous challenges of hearing-disabled people in social environments, this paper provides practical guidelines for building a strong communication bridge between people with hearing and speech impairments and the rest of society. One of the most important obstacles to creating more universal sign language datasets is the regionalization of these languages. Regarding significant differences in dialects within one region, country, and city—even among different languages—the importance of creating a large number of region-specific sign language datasets for sign language applications is of high importance. This paper explains the collection procedure, creation, preparation, processing, and analysis of a unique dataset for the SSL alphabet to improve the social status of hearing-impaired persons. The obtained dataset is applicable for a community that expands to neighboring countries because of their numerous regional-wise similarities. With slight corrections to the dataset, it can become useable in the countries bordering Serbia.

2. Materials and Methods' Selection Process

This paper explains the way the configuration of a unique dataset that contains Serbian Sign Language (SSL) alphabets in a numerical form can be used in ML model training. The dataset was composed of a series of numbers that represented hand and body key points during movements and were identified using the MediaPipe [10] ML architecture. MediaPipe, which is based on a Python 3.11 software library for machine learning and the artificial intelligence TensorFlow [11], can produce very accurate 3D (x, y, z) numerical values that reflect meaningful spatial point locations on provided pictures or video clips.

In this study, the process of creating and preparing datasets was an essential component of creating machine learning models, as shown in Figure 1. Following the collection of video samples, important body and hand points were identified, converted to numerical values, and recorded in the database. Following that, data were assessed, and records made in error and extreme values were eliminated. The SSL dataset was built and based on short video clips, having 40 successive photographs in a 720p resolution. The resolution and number of frames required for machine learning model development could be decreased if required and sized to a performance-appropriate subset of data. Data were finally split into two sub-sets, for training and testing, before they were used for machine learning model training. If the dataset provided to the ML model was relevant, descriptive, and contained a large number of unique records, the ML model would be able to perform with more accuracy and reliability.

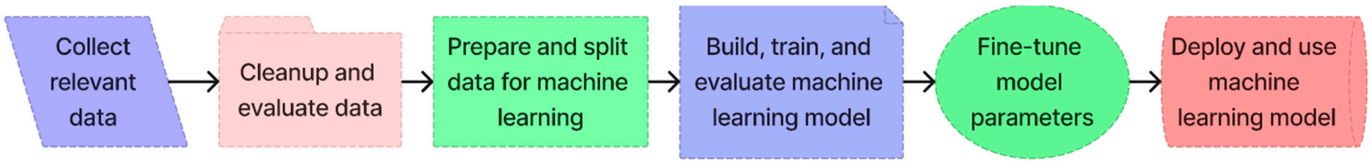

Collect relevant data
Cleanup and evaluate data
Prepare and split data for machine learning
Build, train, and evaluate machine learning model
Fine-tune model parameters
Deploy and use machine learning model

Figure 1. Generalized machine learning problem-solving algorithm steps.

Using Python (software version 3.11), a program was developed to collect recordings and build the SSL dataset. It was used on a PC running in a Windows operating system.

This type of scene setup could be readily rebuilt for data capture and could run the ML model using a standard laptop configuration with a common 720p webcam while running Python source code. Figure 2 depicts the conceptual process behind the SSL dataset creation process.

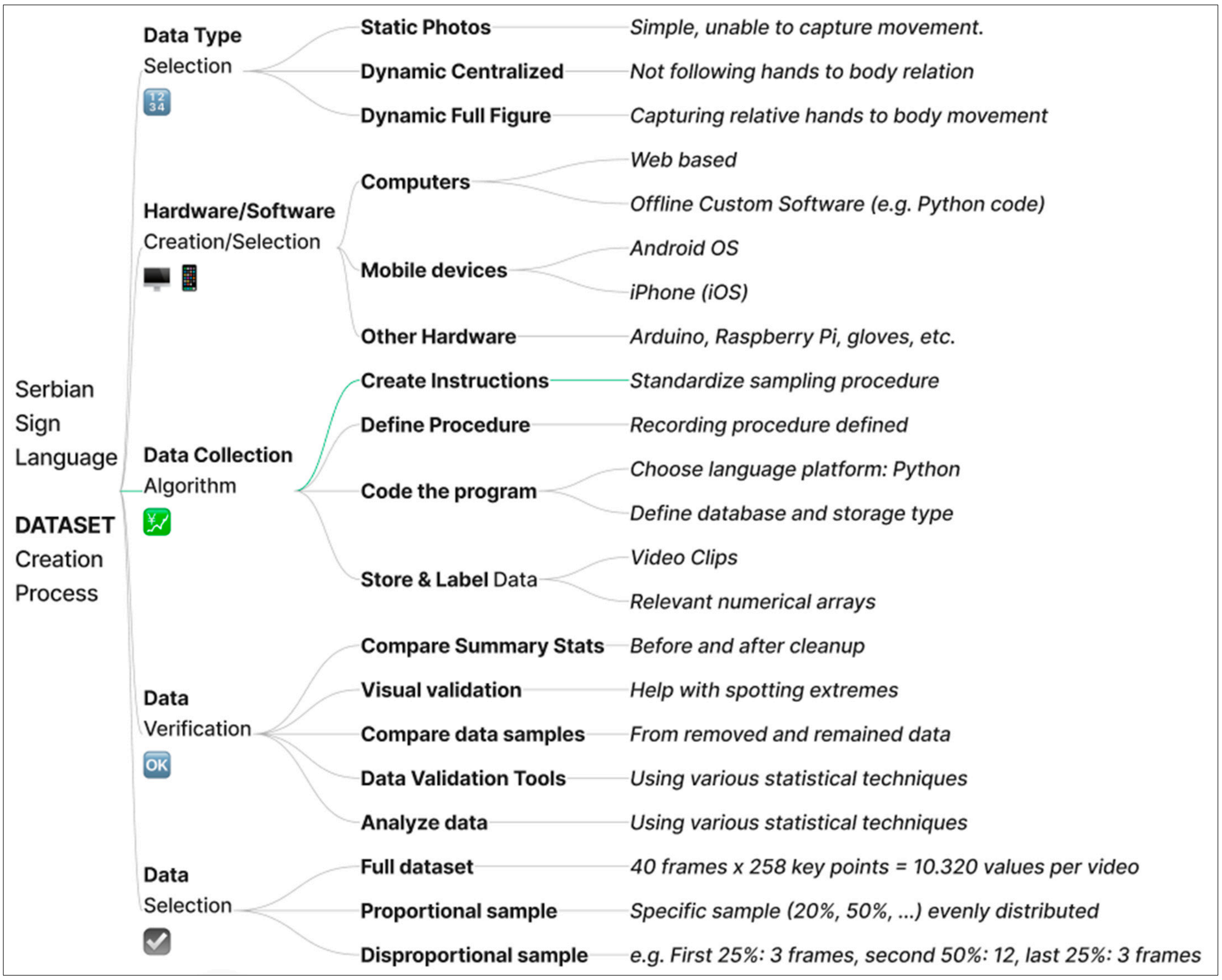
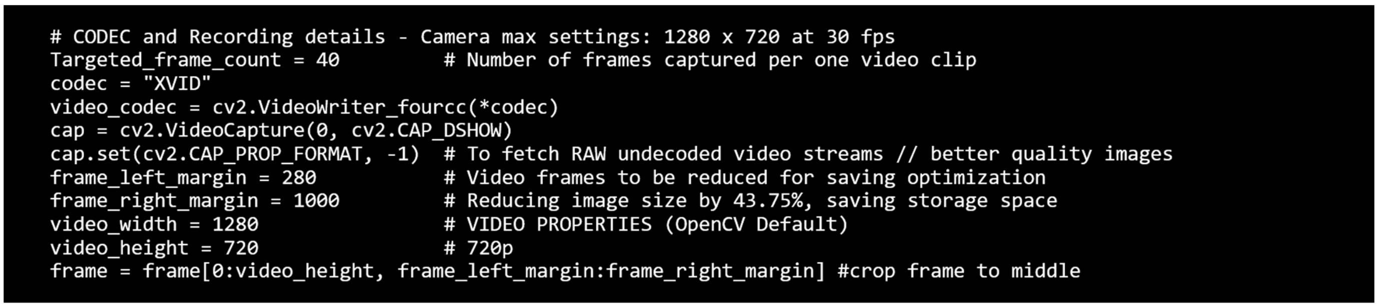

Serbian Sign Language DATASET Creation Process
Data Type Selection
Static Photos
Simple, unable to capture movement.
Dynamic Centralized
Not following hands to body relation
Dynamic Full Figure
Capturing relative hands to body movement
Hardware/Software Creation/Selection
Computers
Web based
Offline Custom Software (e.g. Python code)
Mobile devices
Android OS
iPhone (iOS)
Other Hardware
Arduino, Raspberry Pi, gloves, etc.
Data Collection Algorithm
Create Instructions
Standardize sampling procedure
Define Procedure
Recording procedure defined
Code the program
Choose language platform: Python
Define database and storage type
Store & Label Data
Video Clips
Relevant numerical arrays
Data Verification
Compare Summary Stats
Before and after cleanup
Visual validation
Help with spotting extremes
Compare data samples
From removed and remained data
Data Validation Tools
Using various statistical techniques
Analyze data
Using various statistical techniques
Data Selection
Full dataset
40 frames x 258 key points = 10.320 values per video
Proportional sample
Specific sample (20%, 50%, ...) evenly distributed
Disproportional sample
e.g. First 25%: 3 frames, second 50%: 12, last 25%: 3 frames

Figure 2. Creation procedure of the Serbian Sign Language dataset.

2.1. Data Type Selection

The following 5 sign language alphabets in SSL were not static but represented with a movement: J (Ј), R (Р), DZ (Џ), C (Ц), and CC (Ћ). To capture motion, dynamic images (videos) were chosen as the dataset's base for all gestures. In contrast to photographs, the machine learning system based on this dataset with dynamic values would be able to differentiate between very similarly displayed letters using video clips.

All video clips were stored and compressed with the XVID video codec inside a universal AVI container. The Python source code containing file recording details is presented in Figure 3. The full-sized camera recording frame of 1280 × 720 px was reduced to 720 × 720 px, lowering the image size by 43.75%. Separate images during video capture were forwarded to a MediaPipe framework for the detection of hand and body key points and were automatically saved in a numerical NumPy file. Video recordings were saved alongside numerical datasets for future data validation, error removal, classification, and potential changes. Video files were saved in *.avi format, and NumPy numerical values were saved in *.npy files with the same names. File names were generated automatically and were based on a computer clock.



```
# CODEC and Recording details - Camera max settings: 1280 x 720 at 30 fps
Targeted_frame_count = 40         # Number of frames captured per one video clip
codec = "XVID"
video_codec = cv2.VideoWriter_fourcc(*codec)
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
cap.set(cv2.CAP_PROP_FORMAT, -1)  # To fetch RAW undecoded video streams // better quality images
frame_left_margin = 280           # Video frames to be reduced for saving optimization
frame_right_margin = 1000         # Reducing image size by 43.75%, saving storage space
video_width = 1280                # VIDEO PROPERTIES (OpenCV Default)
video_height = 720                # 720p
frame = frame[0:video_height, frame_left_margin:frame_right_margin] #crop frame to middle
```

Figure 3. Video codec parameter settings and a recording frame size.

2.2. Hardware and Software Selection Process

When deciding on the hardware platform to be used, personal computers, mobile devices (phones and tablets), and other hardware (Arduino, Raspberry PI, gloves, wearable sensors [12], and so on) were all considered. To evaluate this type of solution on mobile devices, an Android application was developed. Several recordings were collected, but the quality was inadequate. The recordings were made on a mobile phone and then uploaded to the cloud. Some of the main reasons for abandoning this way of development were inconsistent records, cloud limitations, storage costs, and the slowness of the process.

Since webcam specifications have remained relatively constant over the years (720p), and personal computers are now widely accessible, PC programming has emerged as the preferred method for developing SSL software. As a result, the chosen platform simplified the process of sharing dataset collection software between contributors. Python, an object-oriented, high-level programming language, was selected for its universality, adaptability, resources, and wide programming community support. As a cross-platform language, it is compatible with and can run on a variety of operating systems, including Windows, Linux, and macOS.

This project also examined the usage of certain hardware devices such as Arduino or Raspberry PI. These devices have some hardware and, more significantly, software constraints. It is possible to create this kind of sign language recognition system on them, but it is a much more complex process. This was an important factor for why these devices were not the preferred choice for the SSL software development process, and we decided to use a standard Windows-based PC.

3. The Dataset Collection Process

The dataset collection process is essential to an ML model development procedure. It is of great importance that the objects in the dataset be of appropriate and applicable quality and quantity. If the dataset is too small or does not accurately reflect the problem data, the resulting machine learning model may be ineffective and imprecise, often producing wrong results. When the dataset is of high quality, containing relevant data, modifying and adapting machine learning parameters, such as the number, structure, and type of layers, is possible. For satisfactory and effective outcomes, creating ML models on large and accurate datasets is required.

3.1. Participant Structure and Ethical Standard Compliance

In this exercise, 41 participants recorded all 30 gestures that represent letters in Serbian Sign Language. Students, faculty professors, and special education teachers aged 19 to 50 took part. They were all informed about the study and signed consent forms before the recording began. This study was conducted in accordance with the Declaration of Helsinki, and the protocol was approved by the Ethics Committee of the Faculty of Contemporary Arts in Belgrade (Project code: SSL23). Also, all participants were trained in the video recording process by a short video that explained the process and proper recording procedure. Participants were instructed to use standard SSL gestures when recording them so that the dataset would not contain many deviations.

Artificial intelligence raises questions about privacy and ethics every time it is used. Questions concerning data privacy and user consent are valid and should be raised whenever activities involve recording and analyzing sensitive personal information that is presented through gestures.

The only output from the generated and released SSL dataset were numerical key points, with no way to identify individuals or return to an original image. Both the SSL dataset and the process of creating it followed and satisfied ethical and data security standards. Real video recordings will not be disseminated; instead, they were only utilized to generate numerical values that correlated to hand and body movements. These numerical values served as the foundation of a dataset and could not be associated with video recordings of the participants. They are the real value of the published dataset, as they were the basis for ML model creation. Video recordings were temporarily stored locally and could potentially be re-used to change the resolution or frame rate of a future machine learning model. Furthermore, the videos were used for visual control of the recorded gestures in the process of data cleanup and validation.

3.2. Setting up the Environment and the Recording Scene

Recordings needed to include subjects of various sizes and proportions to produce a usable and meaningful dataset. The camera needed to be positioned close to or far away from the participants (as indicated in Figure 4). In this study, the subjects were asked to display sign language letters in a variety of speeds and styles, positioning their hands in different positions. A larger number of variables in a dataset should be used to generate higher-quality datasets for training more effective machine learning models.

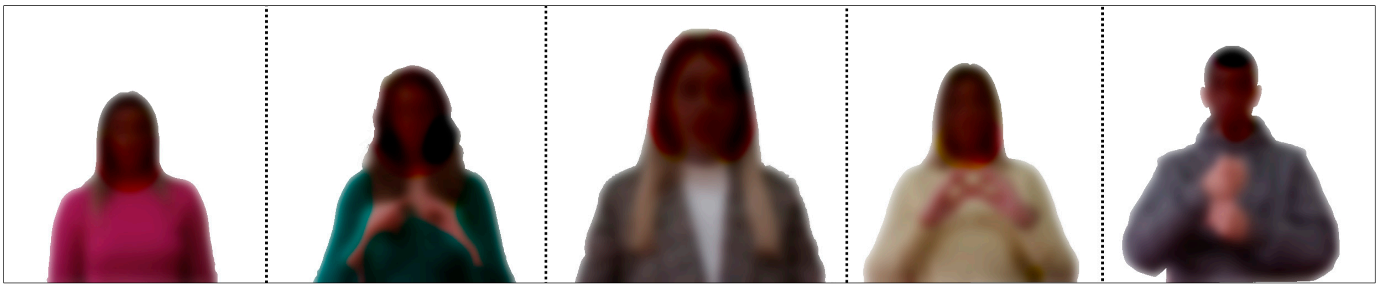


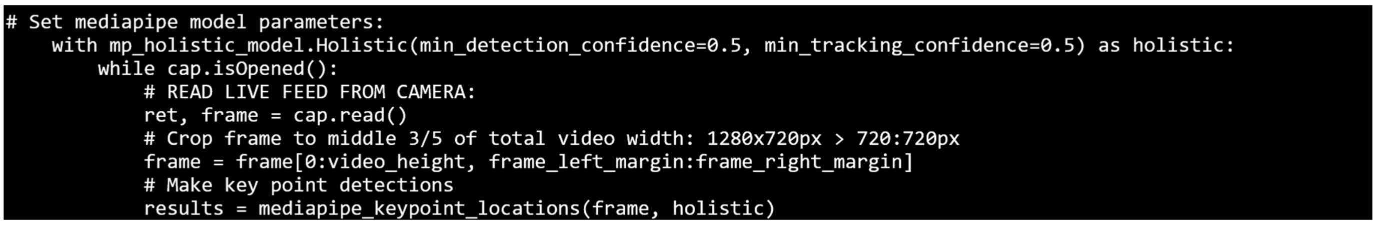
Figure 4. Size, proportion, and location of subjects in the video recording scenarios.

Color and patterns in the background, and on the participants' clothing, can have an impact on the hand and body recognition framework, either positively or negatively. Highly colorful backgrounds and clothing on the recorded subjects were found to harm the MediaPipe recognition framework. If the background and/or clothing were very colorful, the model provided less accurate results. By contrast, if the background and/or clothing were only one color, the MediaPipe framework detected key points much more accurately.

3.3. Recording Procedure

Before initiating recording, the participant needed to be positioned in an optimal camera area, without displaying their hands. When their hands entered the camera recording scene, they were recognized, and recording began immediately after. The program then began to save 40 subsequent frames while displaying a red line timer to the subject to ensure that the entire movement had been recorded as expected. The full movement included hands entering the scene, exhibiting a gesture sign, and leaving the recorded scene. If the hands remained in the scene for longer than 40 frames, the application would not begin recording the next movie clip until the hands left the scene. The application limited the recording to exactly 40 frames. As there was a chance that some key points on the hands would not be identified in some of the frames during the recording of a video, a fail-safe had been programmed to recognize the hands outside of the recording window for 5 consecutive frames in order for them to be considered out. The next recording sequence could only start after both hands left the scene.

Figure 5 depicts a section of source code illustrating certain MediaPipe options. The parameter "holistic" activates, a part of the framework that was based on the provided photo, recognizes individual key points for the whole-body posture, face, and both hands.



```
# Set mediapipe model parameters:
    with mp_holistic_model.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
        while cap.isOpened():
            # READ LIVE FEED FROM CAMERA:
            ret, frame = cap.read()
            # Crop frame to middle 3/5 of total video width: 1280x720px > 720:720px
            frame = frame[0:video_height, frame_left_margin:frame_right_margin]
            # Make key point detections
            results = mediapipe_keypoint_locations(frame, holistic)
```

Figure 5. Size, proportion, and location of subjects in the video recording scenarios.

As a result of the recording process, the application saved two files: a brief video clip (about 4 s long) and a corresponding NumPy array with numerical values of the detected hands and body key points (10.320 values per file). The NumPy array files were to be used as components of the dataset in the development of machine learning models, and the stored video clips would be used to visually examine and validate the recordings. As recordings can be generated by mistake or because they might belong to a different class, files containing visual information are important for further data processing. Figures 6 and 7 represent two files that were saved.

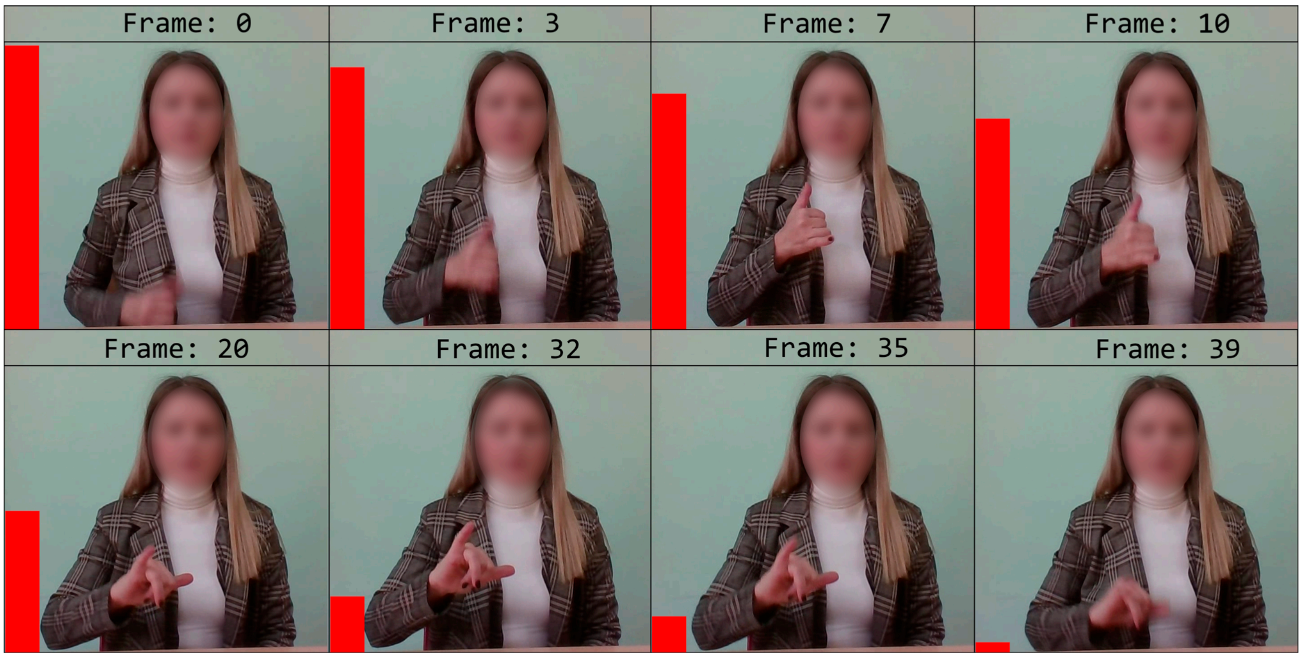


Figure 6. Video showing a full gesture movement of the letter "J".

Figure 6 shows a selection of static frames (out of 40) from a recorded video clip of the letter "J". With this example, it is demonstrated that a single static photo would not be sufficient to detect the meaning of the gesture. The gesture shown could not be distinguished using only one photo. The use of full video clips for data processing will allow for the detection of more complex movements and gestures.

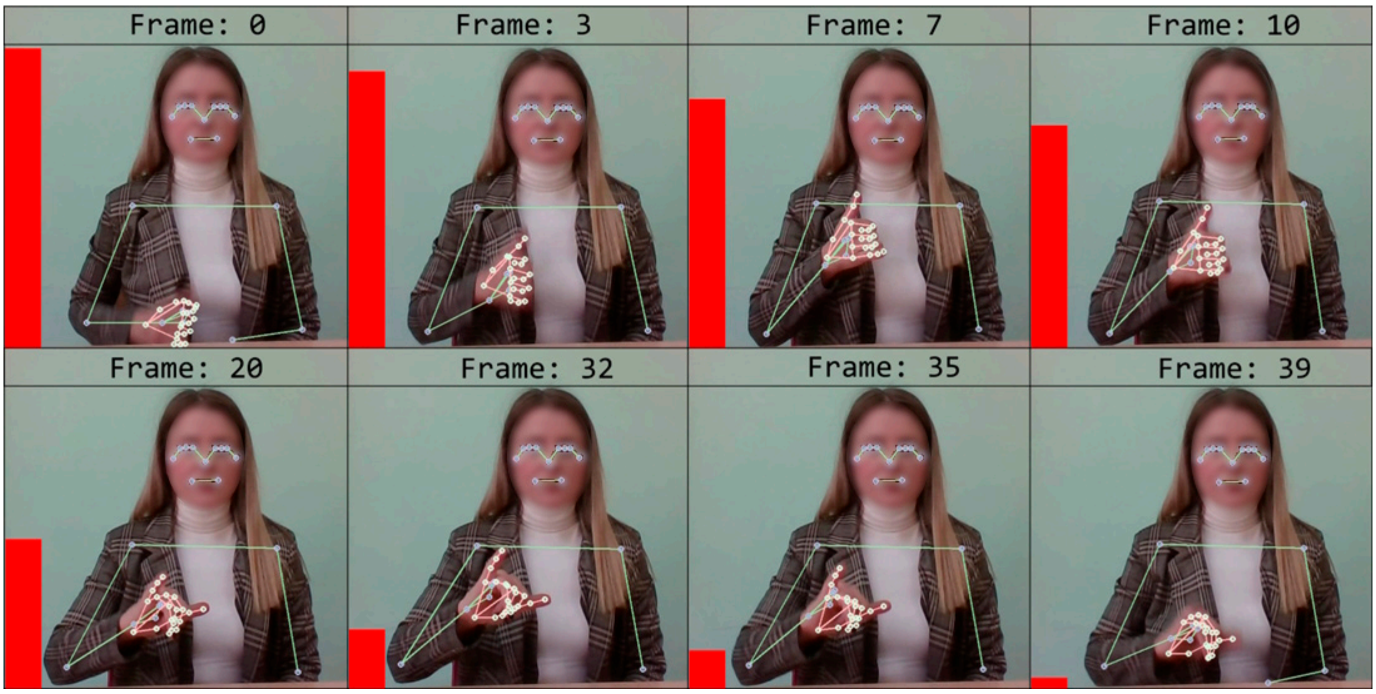


Figure 7. Recognized hand movements and body posture of the letter "J", with marked key points.

Figure 7 depicts the corresponding frames to Figure 6 of the MediaPipe body and hand key point identification procedure. Key point numerical values were recognized and saved as NumPy files before being included in the final dataset.

3.4. Recording Goal: 30 Sign Language Alphabets

The purpose of creating a dataset is to have a sufficient number of distinct recordings of each of the 30 Serbian alphabet signs. The dataset collection and Python source code have been made public, allowing for existing dataset upgrades, the development of new applications, and the implementation of further theoretical and practical research.

The entire dataset creation process is depicted in Figure 8. Before the dataset collection process started, the recording scene needed to be set up according to the instructions. The subject needed to sit across from the computer, keeping his torso in the targeted frame, with his hands away from it. The subjects were instructed on the recording procedure and goals to create better-quality video clips with fewer errors. Following data collection, the data evaluation and cleanup phase employed both visual data verification and numerical validation. Dataset values could be processed further by reducing their size to improve processing performance. Then, data were separated into subsets for testing and training. Finally, the created dataset was sent for further AI/ML use.

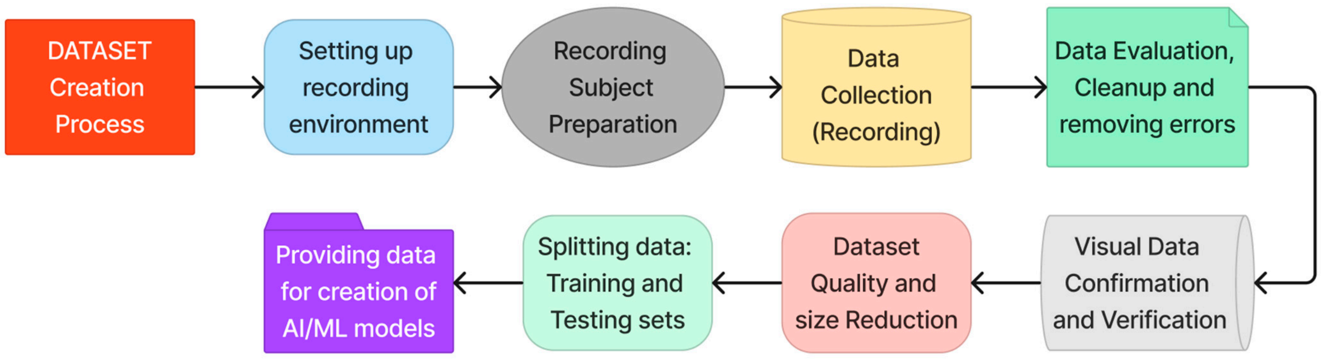

DATASET Creation Process
Setting up recording environment
Recording Subject Preparation
Data Collection (Recording)
Data Evaluation, Cleanup and removing errors
Visual Data Confirmation and Verification
Dataset Quality and size Reduction
Splitting data: Training and Testing sets
Providing data for creation of AI/ML models

Figure 8. Dataset creation procedure.

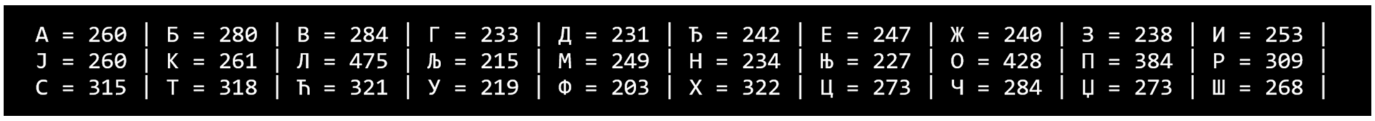
Table 1 includes the 30 alphabets for which we have video recordings. The original SSL alphabet can be found in the bottom row. The middle row of the table depicts a readable and computer-friendly alternative form of the original Serbian Cyrillic letters. The middle row characters were used in the SSL Python program as well in order to organize the dataset file and folder structure.

Table 1. Recording goal: annotation of all 30 gestures depicting the Serbian Sign Language alphabet.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
A	B	V	G	D	DJ	E	ZH	Z	I	J	K	L	LJ	M	M	NJ	O	P	R	S	T	CC	U	F	H	C	CH	DZ	SH
А	Б	В	Г	Д	Ђ	Е	Ж	З	И	Ј	К	Л	Љ	М	Н	Њ	О	П	Р	С	Т	Ћ	У	Ф	Х	Ц	Ч	Џ	Ш

4. Results

The final SSL dataset includes 8.346 video recording clips and accompanying files containing numerical values of recognized body and hand key points. As seen in Figure 9, each letter was recorded an average of 278 times. The database developed is approximately 4.15 GB in size and contains a total of 16.740 files. With an average duration of 4 s for each video recording, the total net time spent creating the dataset without preparations or pauses is approximately 33.384 s or 9.3 h.



```
A = 260 | Б = 280 | В = 284 | Г = 233 | Д = 231 | Ђ = 242 | Е = 247 | Ж = 240 | З = 238 | И = 253 |
Ј = 260 | К = 261 | Л = 475 | Љ = 215 | М = 249 | Н = 234 | Њ = 227 | О = 428 | П = 384 | Р = 309 |
С = 315 | Т = 318 | Ћ = 321 | У = 219 | Ф = 203 | Х = 322 | Ц = 273 | Ч = 284 | Џ = 273 | Ш = 268 |
```

Figure 9. Initial number of recorded SSL video clip gestures.

Figure 10 presents the folder structure for all 30 alphabets for accepted and deleted records. All not-validated recordings were moved to corresponding deleted files folders for later evaluation and statistical processing. For further upgrades and expansions of the database, all dataset files and folders, as well as Python source code that can be used to create new recordings, have been made available online.

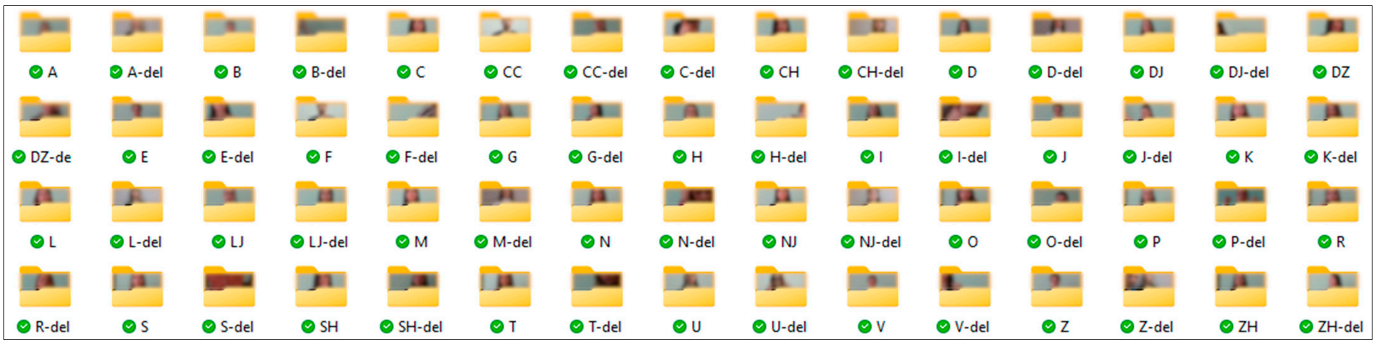
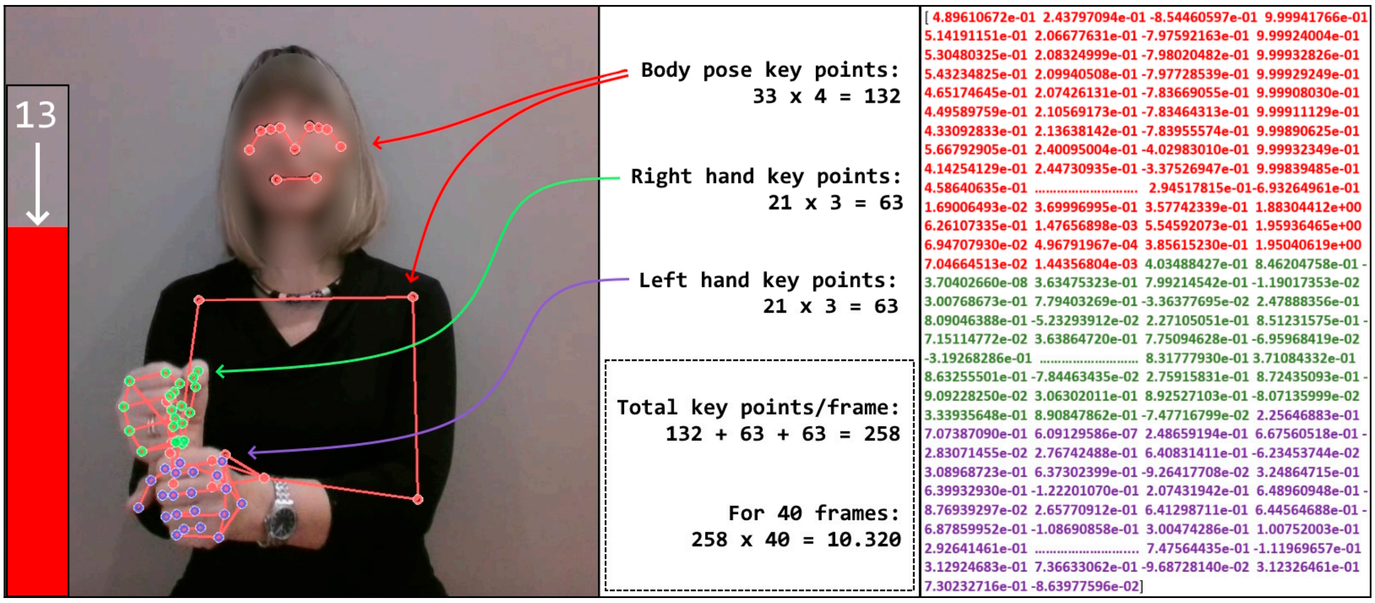


Figure 10. Dataset folder structure.

4.1. *Recorded Video Clips*

As indicated in Figure 11, the red line on the left side of the video functioned as a timer. When the recording began, the red timer was expanded completely, indicating that 40 frames remained to be recorded. The red line reduced as the recording progressed, and, in the example shown, it showed the status of the timer on the 13th frame.



```
[ 4.89610672e-01 2.43797094e-01 -8.54460597e-01 9.99941766e-01
5.14191151e-01 2.06677631e-01 -7.97592163e-01 9.99924004e-01
5.30480325e-01 2.08324999e-01 -7.98020482e-01 9.99932826e-01
5.43234825e-01 2.09940508e-01 -7.97728539e-01 9.99929249e-01
4.65174645e-01 2.07426131e-01 -7.83669055e-01 9.99908030e-01
4.49589759e-01 2.10569173e-01 -7.83464313e-01 9.99911129e-01
4.33092833e-01 2.13638142e-01 -7.83955574e-01 9.99890625e-01
5.66792905e-01 2.40095004e-01 -4.02983010e-01 9.99932349e-01
4.14254129e-01 2.44730935e-01 -3.37526947e-01 9.99839485e-01
4.58640635e-01 .......................... 2.94517815e-01-6.93264961e-01
1.69006493e-02 3.69996995e-01 3.57742339e-01 1.88304412e+00
6.26107335e-01 1.47656898e-03 5.54592073e-01 1.95936465e+00
6.94707930e-02 4.96791967e-04 3.85615230e-01 1.95040619e+00
7.04664513e-02 1.44356804e-03 4.03488427e-01 8.46204758e-01 -
3.70402660e-08 3.63475323e-01 7.99214542e-01 -1.19017353e-02
3.00768673e-01 7.79403269e-01 -3.36377695e-02 2.47888356e-01
8.09046388e-01 -5.23293912e-02 2.27105051e-01 8.51231575e-01 -
7.15114772e-02 3.63864720e-01 7.75094628e-01 -6.95968419e-02
-3.19268286e-01 .......................... 8.31777930e-01 3.71084332e-01
8.63255501e-01 -7.84463435e-02 2.75915831e-01 8.72435093e-01 -
9.09228250e-02 3.06302011e-01 8.92527103e-01 -8.07135999e-02
3.33935648e-01 8.90847862e-01 -7.47716799e-02 2.25646883e-01
7.07387090e-01 6.09129586e-07 2.48659194e-01 6.67560518e-01 -
2.83071455e-02 2.76742488e-01 6.40831411e-01 -6.23453744e-02
3.08968723e-01 6.37302399e-01 -9.26417708e-02 3.24864715e-01
6.39932930e-01 -1.22201070e-01 2.07431942e-01 6.48960948e-01 -
8.76939297e-02 2.65770912e-01 6.41298711e-01 6.44564688e-01 -
6.87859952e-01 -1.08690858e-01 3.00474286e-01 1.00752003e-01
2.92641461e-01 .......................... 7.47564435e-01 -1.11969657e-01
3.12924683e-01 7.36633062e-01 -9.68728140e-02 3.12326461e-01
7.30232716e-01 -8.63977596e-02]
```

Figure 11. One frame example (13 out of 40) of a letter "Д (D)" gesture with key point values.

On the right side of the image, some of the real data values that form the foundation of the dataset are displayed. As shown in Figure 11, there are 258 numerical values recorded for one recorded frame. Arrows from the center of the image point to the body and hand key points where these numerical values are created. As a result of the key point recognition process, the software generates matrices that are converted to a one-dimensional numerical array of 10.320 items for each video clip.

If a MediaPipe framework does not recognize a body or a key point, a 0 value is written into the resulting array. In the case of a video clip from Figure 11, which only shows frame number 13, there are some zero values in other frames. They are mostly present in the first and last frames, when hands enter and exit the recording scene.

The number of frames with zero key point values is shown in Figure 12. From the start of the recording, the subject had one hand in the scene (63 key points signify key points for one hand), and the other hand entered the scene in the third frame. Beginning with frame 3, both the left and right hands, as well as all body posture key points, were being detected, recognized, and recorded (zero records). All the key points were recognized and documented from frame 3 to frame 33. One hand exited the recording scene in frame 33, whereas the other exited in frame 37, as confirmed by the number of zero values.

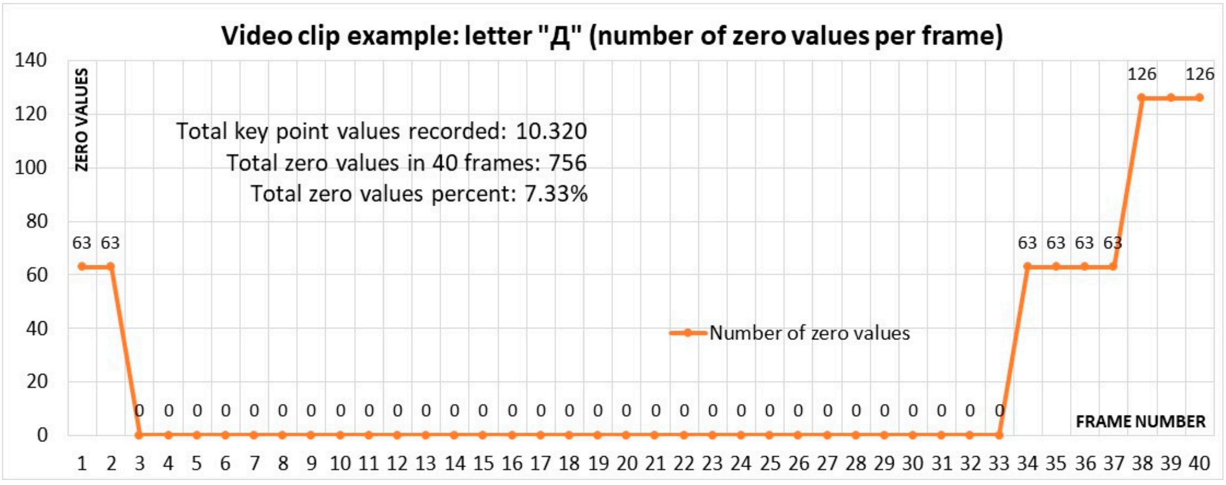


Figure 12. Number of zero key point values in one letter gesture video clip.

4.2. Numpy Dataset

The number of recognized key points for left and right hands is 21 each. In addition, there are also 33 body pose key points that can be detected on the scene. Depicted in Equation (7) are the total number of collected numerical values. The returning values for detected key points on the right and left hands are represented in an array of x, y, and z coordinates [Hand landmarks det...]. Matrix "Handedness" returns the detected hand's value—left or right (Equation (1)). "WorldLandmarks" (Equation (2)) contains absolute recognized values in meters with the origin at the hand's geometric center, and the "Landmarks" matrix contains normalized relative numerical values between −1 and 1 (Equation (3)). To create a dataset, it is required to save "Landmarks" values.

$$\text{HandLandmarkerResult} = [[Handedness] \quad [Landmarks] \quad [WorldLandmarks]] \tag{1}$$

$$Handedness = \begin{bmatrix} index \\ score \\ hand \end{bmatrix} WorldLandmarks = \begin{bmatrix} ax1 & ax2 & ax3 & \dots & ax21 \\ ay1 & ay2 & ay3 & \dots & ay21 \\ az1 & az2 & az3 & \dots & az21 \end{bmatrix} \tag{2}$$

$$Landmarks = \begin{bmatrix} x1 & x2 & x3 & \dots & x21 \\ y1 & y2 & y3 & \dots & y21 \\ z1 & z2 & z3 & \dots & z21 \end{bmatrix}, \text{ where one key point is : } HandKeyPoint1 = \begin{bmatrix} x1 \\ y1 \\ z1 \end{bmatrix} \tag{3}$$

Right-hand (*rh*) and left-hand (*lh*) key points can be expressed in a matrix, as shown in Equation (4):

$$rh = \begin{bmatrix} rhx1 & rhx2 & rhx3 & \dots & rhx21 \\ rhy1 & rhy2 & rhy3 & \dots & rhy21 \\ rhz1 & rhz2 & rhz3 & \dots & rhz21 \end{bmatrix} lh = \begin{bmatrix} lhx1 & lhx2 & lhx3 & \dots & lhx21 \\ lhy1 & lhy2 & lhy3 & \dots & lhy21 \\ lhz1 & lhz2 & lhz3 & \dots & lhz21 \end{bmatrix} \tag{4}$$

Three-dimensional values with x, y, and z coordinates are "flattened" via conversion into a numerical one-dimensional array. Body posture key points are reduced from four dimensions to one. One additional matrix dimension for body posture represents the visibility coefficient array. The body pose detection function, similar to the hand detection function, produces the matrix shown in Equation (5).

$$\text{PoseLandmarkerResult} = [[Landmarks] \quad [WorldLandmarks]] \tag{5}$$

$$Landmarks = \begin{bmatrix} px1 & px2 & px3 & \dots & px33 \\ py1 & py2 & py3 & \dots & py33 \\ pz1 & pz2 & pz3 & \dots & pz33 \\ pv1 & pv2 & pv3 & \dots & pv33 \end{bmatrix} \tag{6}$$

Presented in Figure 13 is a visual transformation procedure for creating a resulting one-dimensional array with values that represent all detected key points in one frame. Three matrices represent coordinates of recognized 33 body and general face landmarks (poseLandmarks), 21 left-hand key points (*lh*), and 21 right-hand (*rh*) key points.

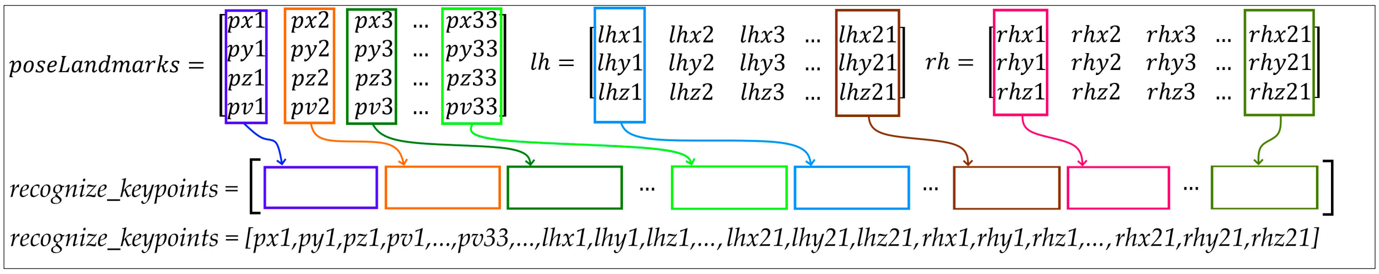
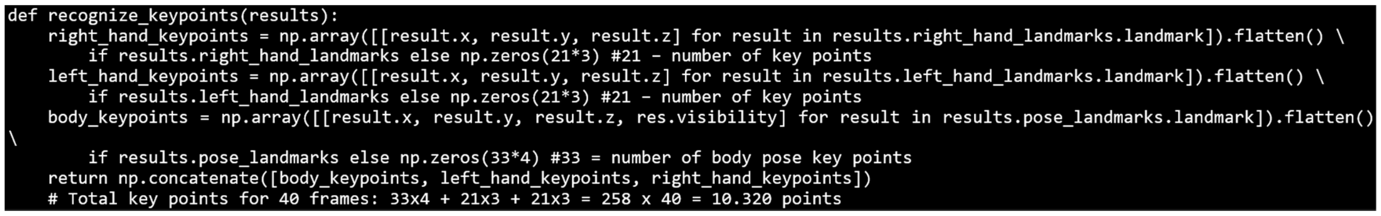


Figure 13. Transforming three matrices into a one-dimensional numerical array.

All the numerical values representing key point coordinates are grouped in one frame of detected right and left hands and general body posture. More precisely, the right hand has 21 points defined by three number values, the left hand has the same, and the body posture has 33 points defined by four numerical values. When matrices are converted to a one-dimensional vector, the resultant array representing one video frame has 258 key points, as shown in Equation (7).

$$\text{number\_of\_keypoints\_per\_frame} = \text{pose} + \text{lh} + \text{rh} = 33 \times 4 + 21 \times 3 + 21 \times 3 = 132 + 63 + 63 = 258 \text{ key points} \quad (7)$$

The Python source code that transforms matrices into a one-dimensional array is presented in Figure 14. The Python method "flatten()" returns a copy of the two-dimensional array collapsed into one dimension.



```
def recognize_keypoints(results):
    right_hand_keypoints = np.array([[result.x, result.y, result.z] for result in results.right_hand_landmarks.landmark]).flatten() \
        if results.right_hand_landmarks else np.zeros(21*3) #21 - number of key points
    left_hand_keypoints = np.array([[result.x, result.y, result.z] for result in results.left_hand_landmarks.landmark]).flatten() \
        if results.left_hand_landmarks else np.zeros(21*3) #21 - number of key points
    body_keypoints = np.array([[result.x, result.y, result.z, res.visibility] for result in results.pose_landmarks.landmark]).flatten()
\
        if results.pose_landmarks else np.zeros(33*4) #33 = number of body pose key points
    return np.concatenate([body_keypoints, left_hand_keypoints, right_hand_keypoints])
    # Total key points for 40 frames: 33x4 + 21x3 + 21x3 = 258 x 40 = 10.320 points
```

Figure 14. Procedure for converting key point matrices into a one-dimensional array.

Each sign language alphabet is recorded in a single file that contains data for the 40 frames. Each file is saved in NumPy format and contains exactly 258 key points multiplied by 40 frames for a total of 10.320 numerical values. This presents a good base that can optionally be reduced and used in an appropriate quantity for further processing.

4.3. Dataset Validation and Cleanup

Cleaning and validating a dataset are critical steps in the data preparation process for any machine learning project. This ensures the accuracy, reliability, and suitability of the data used to train and test models for an intended task. The SSL database generation technique did not allow for the fabrication of duplicates because it recorded unique video snippets. In the produced dataset, there were also no empty data or information that needed to be additionally filled in.

Reasons for dismissing and removing video recordings from the final dataset comprised the following:

- Records made accidentally by presenting hands to the camera;
- Footage recorded while a person was settling themselves in front of the camera;
- Records classified as a wrong letter (wrong letter recording);
- The presented sign gesture not conforming to the SSL standard;
- The speed of the gesture movement being inadequate for a 40-frame-per-video clip.

The dataset validation, cleanup, and classification methods used in the SSL database creation process were as follows:

- Visual validation and classification of data based on video clips;
- Numerical validation and classification based on Python source code-generated data.

Visual validation of video recordings for a dataset is the first step in the data cleanup process. This involves inspecting the films to make sure they conform to the desired standards and criteria. All inadequately recorded video clips were removed and placed in separate folders for deleted letters. Figure 15 illustrates a collection of wrongfully recorded video clips.

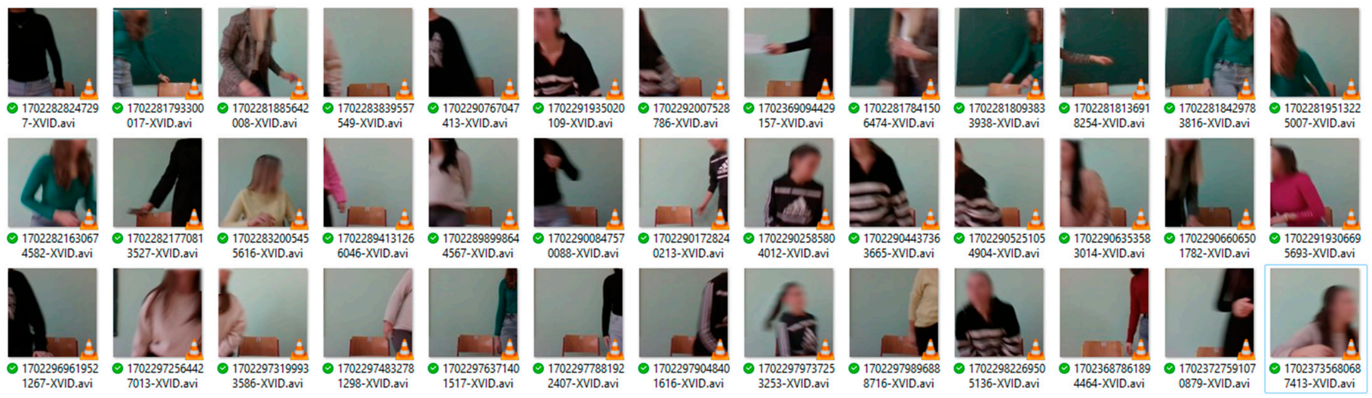
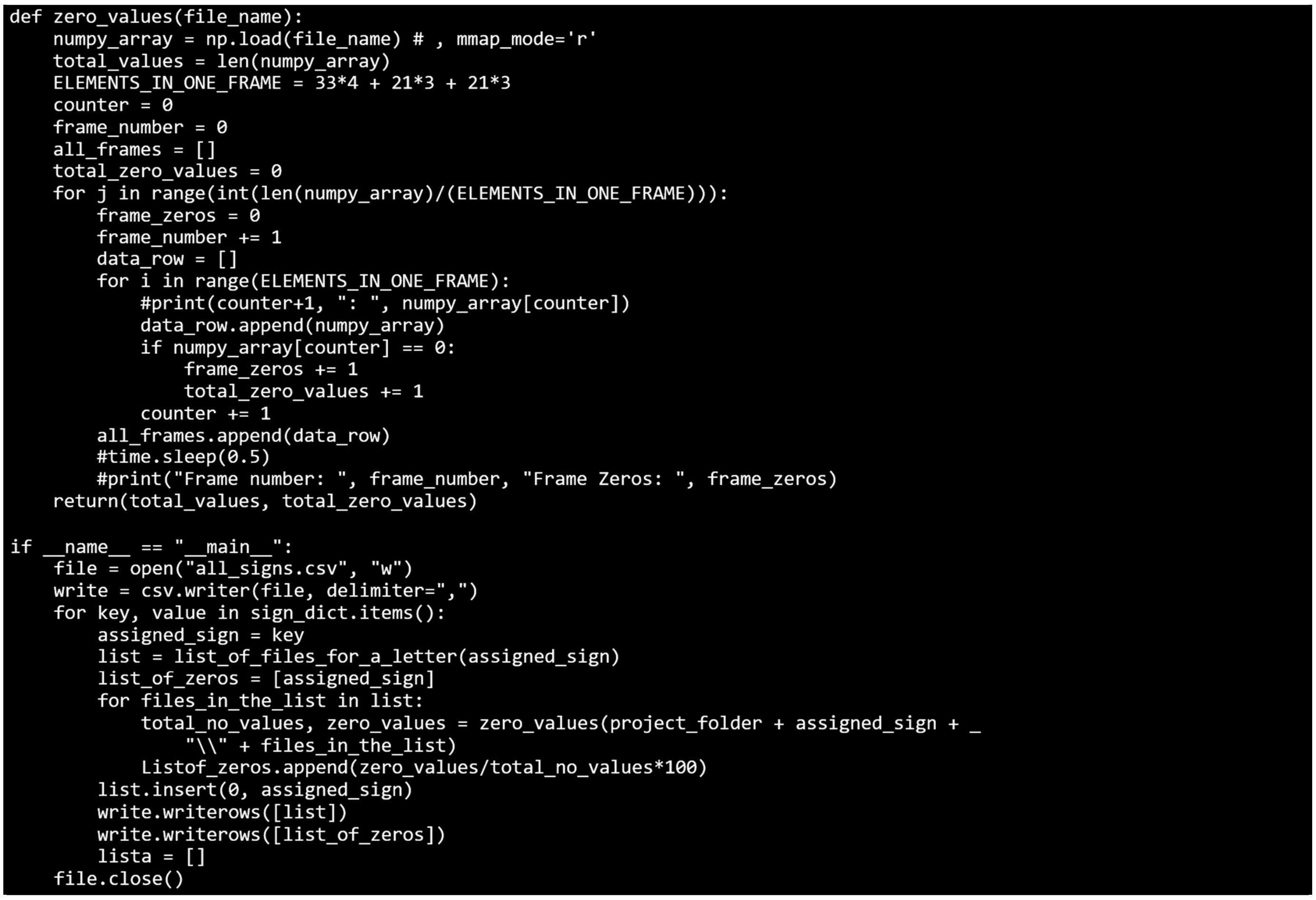

1702282824729 7-XVID.avi
1702281793300 017-XVID.avi
1702281885642 008-XVID.avi
1702283839557 549-XVID.avi
1702290767047 413-XVID.avi
1702291935020 109-XVID.avi
1702292007528 786-XVID.avi
1702369094429 157-XVID.avi
1702281784150 6474-XVID.avi
1702281809383 3938-XVID.avi
1702281813691 8254-XVID.avi
1702281842978 3816-XVID.avi
1702281951322 5007-XVID.avi
1702282163067 4582-XVID.avi
1702282177081 3527-XVID.avi
1702283200545 5616-XVID.avi
1702289413126 6046-XVID.avi
1702289899864 4567-XVID.avi
1702290084757 0088-XVID.avi
1702290172824 0213-XVID.avi
1702290258580 4012-XVID.avi
1702290443736 3665-XVID.avi
1702290525105 4904-XVID.avi
1702290635358 3014-XVID.avi
1702290660650 1782-XVID.avi
1702291930669 5693-XVID.avi
1702296961952 1267-XVID.avi
1702297256442 7013-XVID.avi
1702297319993 3586-XVID.avi
1702297483278 1298-XVID.avi
1702297637140 1517-XVID.avi
1702297788192 2407-XVID.avi
1702297904840 1616-XVID.avi
1702297973725 3253-XVID.avi
1702297989688 8716-XVID.avi
1702298226950 5136-XVID.avi
1702368786189 4464-XVID.avi
1702372759107 0879-XVID.avi
1702373568068 7413-XVID.avi

Figure 15. Incorrectly recorded video clips detected visually.

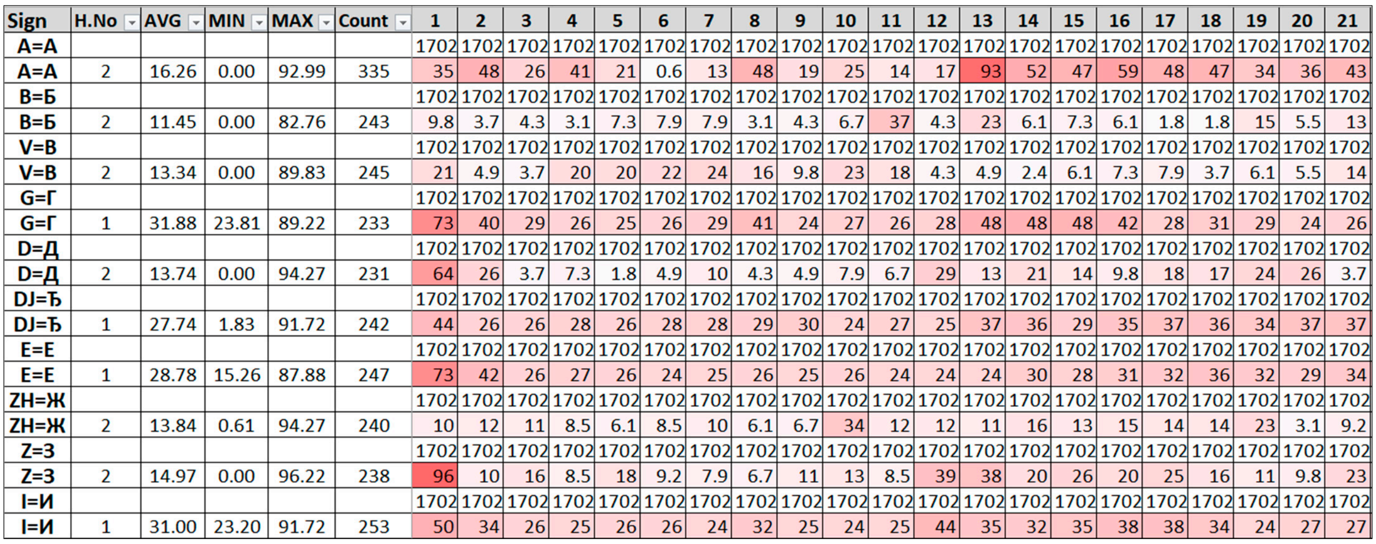
Various programming languages and libraries can also be used to automate dataset cleanup. The tools used are determined by the nature of the dataset and the specific cleanup tasks required. Python was used for this task because it was chosen as the primary programming platform for the project, and Figure 16 presents a procedure for this task.


```
def zero_values(file_name):
    numpy_array = np.load(file_name) # , mmap_mode='r'
    total_values = len(numpy_array)
    ELEMENTS_IN_ONE_FRAME = 33*4 + 21*3 + 21*3
    counter = 0
    frame_number = 0
    all_frames = []
    total_zero_values = 0
    for j in range(int(len(numpy_array)/(ELEMENTS_IN_ONE_FRAME))):
        frame_zeros = 0
        frame_number += 1
        data_row = []
        for i in range(ELEMENTS_IN_ONE_FRAME):
            #print(counter+1, ": ", numpy_array[counter])
            data_row.append(numpy_array)
            if numpy_array[counter] == 0:
                frame_zeros += 1
                total_zero_values += 1
            counter += 1
        all_frames.append(data_row)
        #time.sleep(0.5)
        #print("Frame number: ", frame_number, "Frame Zeros: ", frame_zeros)
    return(total_values, total_zero_values)

if __name__ == "__main__":
    file = open("all_signs.csv", "w")
    write = csv.writer(file, delimiter=",")
    for key, value in sign_dict.items():
        assigned_sign = key
        list = list_of_files_for_a_letter(assigned_sign)
        list_of_zeros = [assigned_sign]
        for files_in_the_list in list:
            total_no_values, zero_values = zero_values(project_folder + assigned_sign + _
                "\\" + files_in_the_list)
            Listof_zeros.append(zero_values/total_no_values*100)
        list.insert(0, assigned_sign)
        write.writerows([list])
        write.writerows([list_of_zeros])
        lista = []
    file.close()
```

Figure 16. Procedure for detecting incorrectly recorded video clips.

The file "all_signs.csv" produced by the source code in Figure 16 shows the total percentage of zeros present within a video clip. Zero values represent "unrecognized" body poses and left- and right-hand key points. The "AVG" column displays the average percent of zero values for a specific alphabet, whereas the "MIN" and "MAX" columns display extreme values, and “H.No” denotes the number of hands used to present a particular sign/letter. Figure 17 displays darker values that require additional visual inspection or removal from the dataset due to a higher percentage of zero values. By comparing these percentages, the process of removing recordings with extreme values can be automated.



Sign	H.No	AVG	MIN	MAX	Count	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
A=A						1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702
A=A	2	16.26	0.00	92.99	335	35	48	26	41	21	0.6	13	48	19	25	14	17	93	52	47	59	48	47	34	36	43
B=Б						1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702
B=Б	2	11.45	0.00	82.76	243	9.8	3.7	4.3	3.1	7.3	7.9	7.9	3.1	4.3	6.7	37	4.3	23	6.1	7.3	6.1	1.8	1.8	15	5.5	13
V=B						1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702
V=B	2	13.34	0.00	89.83	245	21	4.9	3.7	20	20	22	24	16	9.8	23	18	4.3	4.9	2.4	6.1	7.3	7.9	3.7	6.1	5.5	14
G=Г						1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702
G=Г	1	31.88	23.81	89.22	233	73	40	29	26	25	26	29	41	24	27	26	28	48	48	48	42	28	31	29	24	26
D=Д						1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702
D=Д	2	13.74	0.00	94.27	231	64	26	3.7	7.3	1.8	4.9	10	4.3	4.9	7.9	6.7	29	13	21	14	9.8	18	17	24	26	3.7
DJ=Ђ						1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702
DJ=Ђ	1	27.74	1.83	91.72	242	44	26	26	28	26	28	28	29	30	24	27	25	37	36	29	35	37	36	34	37	37
E=E						1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702
E=E	1	28.78	15.26	87.88	247	73	42	26	27	26	24	25	26	25	26	24	24	24	30	28	31	32	36	32	29	34
ZH=Ж						1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702
ZH=Ж	2	13.84	0.61	94.27	240	10	12	11	8.5	6.1	8.5	10	6.1	6.7	34	12	12	11	16	13	15	14	14	23	3.1	9.2
Z=З						1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702
Z=З	2	14.97	0.00	96.22	238	96	10	16	8.5	18	9.2	7.9	6.7	11	13	8.5	39	38	20	26	20	25	16	11	9.8	23
I=И						1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702	1702
I=И	1	31.00	23.20	91.72	253	50	34	26	25	26	26	24	32	25	24	25	44	35	32	35	38	38	34	24	27	27

Figure 17. Number of total video clips before and after data validation.

When one hand is missing from the scene, the program generates zero values for it because the missing hand key points are not recognized, resulting in zeros. Consequently, gestures used to present signs with only one hand have a higher number of zero-value percentages. Two-handed gestures have an average of 9.6% of zero values per video clip, whereas one-handed gestures have an average of 27.8% zero values per video clip.

Figure 18 illustrates the percentage of zero values before and after data cleanup. The red line shows the extreme, maximum zero value percentages of video clips that were mistakenly recorded. The blue line represents the new maximum values after the extremes have been removed. The “AVG Before” and “AVG After” lines represent the difference after data cleanup.

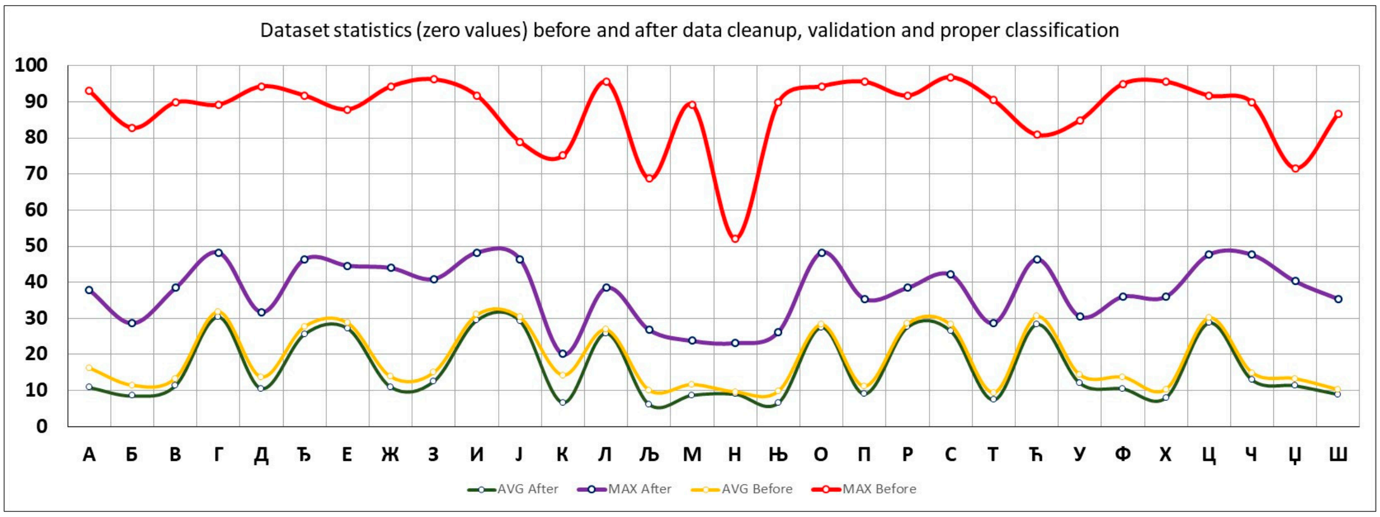


Figure 18. Video clip statistics before and after data validation—percent of zero values.

The difference between two-handed (on the left) and one-handed (on the right) sign gestures is shown in Figure 19. It is noticeable that two-handed gestures have fewer zero-values in their files.

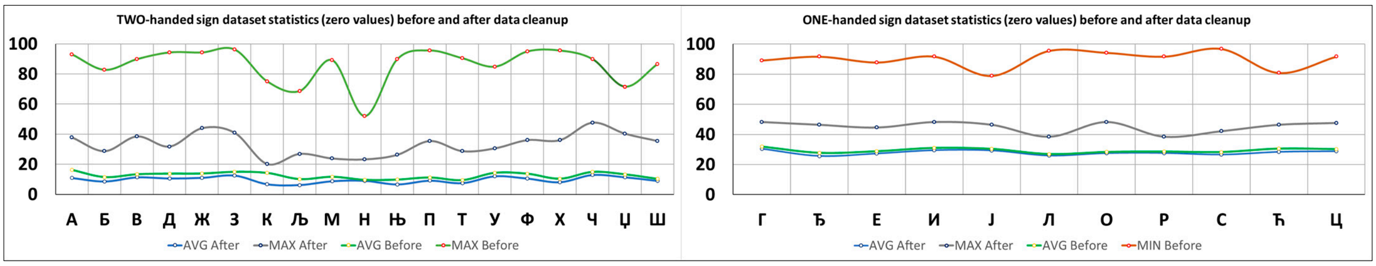


Figure 19. Video clip statistics before and after data cleanup.

4.4. Final Dataset Shape and Size

After recording and data cleanup, the dataset reached its final shape and size. The proportion of all recorded video clips and the ones removed after data validation are presented visually in Figure 20.

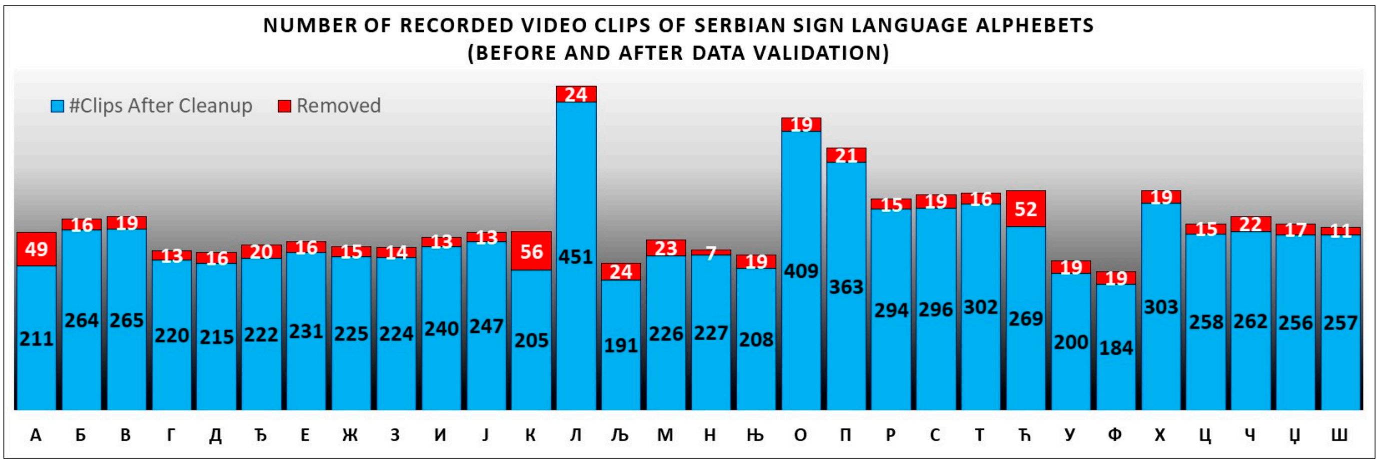


Figure 20. Number of recorded validated and removed video clips in the dataset.

The total number of recorded video clips is 8.346, out of which 621 have been removed, leaving 7.725 in the final dataset. With an average of 257.5 records per alphabet, all of them are well described and supported by validated data.

5. Discussion of Results

The dataset developed and presented in this paper is based on SSL gestures and can be used to build ML models. Key points extracted from video clips can be used as a foundation for future data processing using computer vision techniques and the MediaPipe machine learning framework.

The recordings were collected from 41 individuals while they were in classrooms, offices, or at home using an ordinary computer. This method of gathering data is not impacted by external conditions and may be replicated in various environments. The subject was sitting in a natural position, while the computer was on the desk. With this configuration, it is possible to expand the dataset further by replicating similar scenarios. Furthermore, when using the ML model based on this dataset for recognizing sign language gestures, users will be in a similar position as during the data collection process. Therefore, the model will be more reliable and will produce more accurate results.

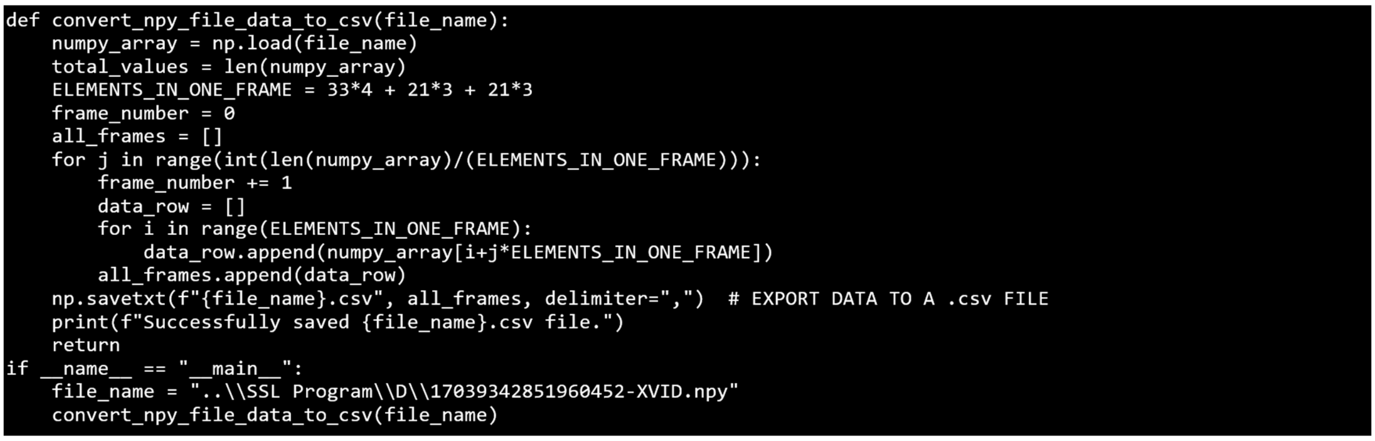
To achieve higher quality outcomes and less useless data, an instructional video was prepared and presented to all participants. The standardization of gestures, movement dynamics, and general recording procedures resulted in the generation of a more effective SSL dataset.

The average recording duration is composed mostly of 20% of hands entering the scene, 60% of hands showing sign language movement, and 20% of hands leaving the recording scene. Having 40 frames in these recordings allows researchers to modify the number of frames utilized in the building of the ML model. Using computer vision techniques, the moment that hands enter the recording scene is recognized, and the recording of 40 frames begins automatically, making the process faster and more efficient.

As various recording participants entered and exited the scene, a couple of recording errors occurred. A final number of 621 video clips were removed out of a total of 8.436. Together with recordings that were not compliant with SSL criteria (wrong hand, improper movement, etc.), only 7.44% of records were made in error, which is a very good result.

A reliable measure of the importance of the data is the number of zero values in the identified frames of the hands and body key points. One useful way to identify recordings that are not compliant is to process all of the resultant data files and identify those that have a higher percentage of zero values or that deviate from the average. Visual control of video files was utilized to ensure the removal of the records from the final dataset, following numerical evaluation and labeling of the files with extreme values.

The resulting dataset of NumPy arrays can be loaded and manipulated using the source code shown in Figure 21. This loads data from one "npy" file into a Python list before saving it to a "csv" human-readable file type. Data, like in this source code, can be used for a variety of similar purposes.



```
def convert_npy_file_data_to_csv(file_name):
    numpy_array = np.load(file_name)
    total_values = len(numpy_array)
    ELEMENTS_IN_ONE_FRAME = 33*4 + 21*3 + 21*3
    frame_number = 0
    all_frames = []
    for j in range(int(len(numpy_array)/(ELEMENTS_IN_ONE_FRAME))):
        frame_number += 1
        data_row = []
        for i in range(ELEMENTS_IN_ONE_FRAME):
            data_row.append(numpy_array[i+j*ELEMENTS_IN_ONE_FRAME])
        all_frames.append(data_row)
    np.savetxt(f"{file_name}.csv", all_frames, delimiter=",")  # EXPORT DATA TO A .csv FILE
    print(f"Successfully saved {file_name}.csv file.")
    return
if __name__ == "__main__":
    file_name = "..\\SSL Program\\D\\17039342851960452-XVID.npy"
    convert_npy_file_data_to_csv(file_name)
```

Figure 21. Python source code for loading and using recorded NumPy dataset files.

Depending on its intended use, the dataset can be reduced to a more manageable size. Downsizing from 40 frames per second and 720 × 720-pixel videos can lead to faster data manipulation and less machine learning processing time. Each frame's data can be reduced from 40 to a smaller subset proportionally, arbitrarily, or by using custom data distribution.

After the creation of a validated dataset with a large number of files with data, splitting a dataset into subsets is an important step in the process of training and testing machine learning models. Subsets are created for training, validation, and testing. The data splitting strategy to be used is determined by the features of the dataset, the problem that is being solved, and the aims of the planned machine learning project. It is critical to carefully select a split strategy to ensure the model's generalization and performance on unknown input.

Some documented attempts have been implemented globally to establish databases for sign language research purposes. Each has unique characteristics and varies in terms of technology. Some of the existing datasets that can be found worldwide are shown in Table 2. SSL describes 30 alphabets in comparison to other datasets that describe 104 to 9083 signs.

Table 2. Comparing SSL with other data sets, authors' calculation based on [13].

Dataset	Language	No of Signs	Co-Articulated	Avg Videos Per Sign	No of Participants	Source	No of Hours
ASLLVD [14]	ASL (American)	2742	-	3	6	lab	4
MSASL [15]	ASL (American)	1000	-	25	222	lexicons, web	25
WLASL [16]	ASL (American)	2000	-	11	119	lexicons, web	14
BOSTON104 [17]	ASL (American)	104	+	10	3	lab	1
BSLDict [18]	BSL (British)	9283	-	1	148	lexicons	9
BSL-1K [19]	BSL (British)	1064	+	257	40	TV	1060
BOBSL [13]	BSL (British)	2281	+	198	39	TV	1467
Devisign [20]	CSL (Chinese)	2000	-	12	8	lab	13–33
CSL500 [21]	CSL (Chinese)	500	-	250	50	lab	69–139
CSL-Daily [22]	CSL (Chinese)	2000	+	76	10	lab	23
SIGNUM [23]	DGS (German)	450	+	304	25	lab	55
Phoenix14T [24]	DGS (German)	1066	+	71	9	TV	11
S-pot [25]	FinSL (Finish)	1211	+	5	5	lab	9
GSL [26]	GSL (Greek)	310	+	132	7	lab	10
INCLUDE [27]	ISL (Indian)	263	-	16	7	lab	3
KETI [28]	KSL (Korean)	524	+	28	14	lab	28
SERBIAN SL	SSL (Serbian)	30	-	280	41	lab	10
BosphorusSign22k [29]	TSL (Turkish)	744	-	30	6	lab	19
AUTSL [30]	TSL (Turkish)	226	-	170	43	lab	21

As seen in Figure 22, comparable datasets have between 3 and 304 entries per sign, whereas SSL has an average of 280 records per sign. All 30 alphabets in the SSL dataset are well represented and recorded, and there are enough records to describe sign gestures for all of them in detail and in a variety of ways. A greater quantity of records per alphabet ensures more accurate training of future machine learning models. As the source code for collecting SSL records and the dataset are available online, the quantity of records is projected to grow over time.

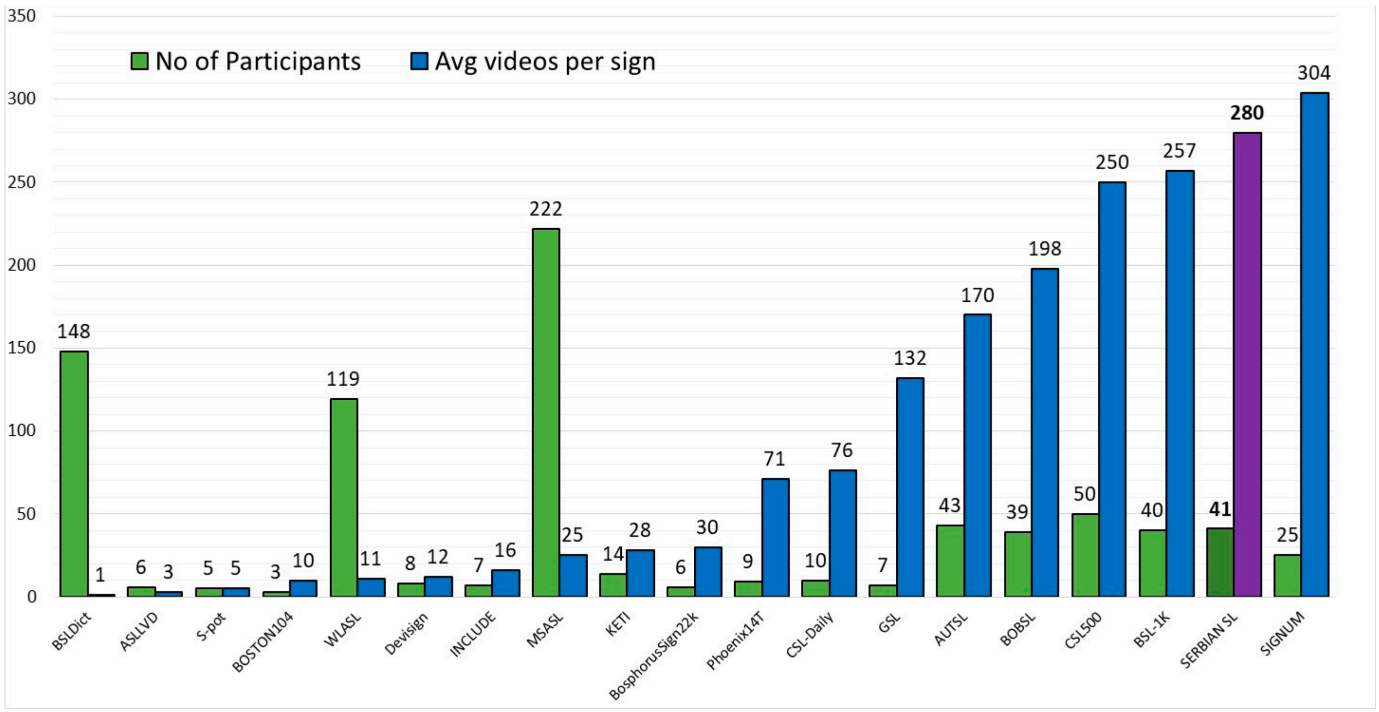


Figure 22. Dataset parameters comparison.

The number of existing datasets presented in Table 2, which were previously used for sign language recognition in research papers, focuses predominantly on continuous sign language recognition and sign language production. They can be separated into a group of datasets that is based on recognizing isolated signs (one sign recognized at a time) and a group that is based on continuous sign recognition (co-articulated) [13]. The shortfalls in several of the following datasets are in their limited numbers of participants demonstrating gestures: ASLLVD (3), BSLDict (1), Devisign (12), WLASL (11), BOSTON104 (10), S-pot (5), INCLUDES (16), etc. The SSL dataset, with 41 presenters for 30 signs, has a very good number of unique demonstrators. Many datasets have many signs but only a few records per gesture. Datasets with a very low average of videos per sign include BSLDict (1), ASLLVD (3), S-pot (5), BOSTON104 (10), WLASL (11), Devisign (12), INCLUDE (16), etc. The SSL dataset, with 280 records per sign, is well supported and describes all 30 possible outcomes.

Intelligent machine learning classifiers commonly used for sign language identification include Deep Learning [31], K-nearest neighbor (KNN), artificial neural network (ANN), support vector machine (SVM), hidden Markov Model (HMM), Convolutional Neural Network (CNN), fuzzy logic, and ensemble learning [32]. With the dataset generated, all of these predictors can also be used for further SSL software development.

6. Conclusions

The World Federation of the Deaf (WFD) is a global organization led by the deaf, and their vision is to promote the human right to sign language for all deaf people, i.e., to ensure equal rights for over 70 million deaf people worldwide [33,34]. In Serbia alone, around 150,000 people with hearing impairment are registered [9]. Faced with inequality every day, they struggle to improve communication opportunities and the quality of their lives. Research findings show that there is insufficient knowledge and use of sign language and typing not only by professionals who participate in the process of education and upbringing of the hearing impaired but also by parents and the wider social community [35]. Nevertheless, the notable significance of the application of sign language and typing can be found in social situations, education, the workplace and areas of professional development, access to legal and administrative services, as well as health and social care services [36]. In addition, an open approach to communication significantly contributes to the improvement of social interactions in the everyday environment and the achievement of the successful social inclusion of hearing-impaired people, which is the ultimate goal of all-inclusive practices in the world [1].

Intending to advance in the direction of improving the communication possibilities of deaf people and increasing the availability of various services, this paper presents a unique dataset of values describing 30 SSL alphabets. It shows 8.346 distinct dynamic video records and corresponding files containing numerical values with recognized hand and body key points. The dataset created is intended to be used in the development of machine learning models capable of recognizing SSL alphabets. The setup and positioning of participants recording gestures during dataset creation were typical. Therefore, the final machine learning model can be employed in everyday situations and environments.

Other datasets may consist of a spoken language source (interpreted into signed language, for example, with a picture-in-picture translator) or a sign language source (interpreted into spoken language via voice-over). The interpretation of broadcasts, especially television weather broadcasts, has been the most commonly used source of data for sign machine learning datasets [35]. These databases are derived from a variety of sources, including continuous natural studio-recorded datasets initially meant for linguistic use [37,38], project-specific isolated studio-recorded datasets [15,16], and sign-interpreted broadcast footage [39,40]. Comparatively, SSL records were collected in a regular, everyday classroom environment.

The records collected are related to the Serbian alphabet, which comprises 30 letters. Future research could focus on increasing the number of samples for the same dataset to improve precision, reduce errors, and increase reliability. Data cleaning and validation processes can be additionally improved and automated. Considering that the dataset described in this paper was produced for 30 alphabets, future research can be expanded to include more gestures that represent full words. The SSL dataset is limited to the Serbian alphabet, which is insufficient for developing advanced language recognition and translation systems. It allows developers to create machine learning models that will only recognize 30 predetermined alphabets. Extending the dataset to include a broader variety of identifiable gestures will require significantly more resources, participants, time, and effort, but it might cover a larger section of the already-used spoken sign language.

This article provides a software solution that can also be used for increasing the number of records in the existing dataset. The same method can be applied to other sign languages or similar computer vision applications. The proposed methodology for dataset collecting is flexible, widely applicable, and can be easily extended to include an unlimited number of sign languages. Different sign languages, body gestures, and movements can be analyzed and recognized similarly. The SSL dataset has the potential to be a valuable resource for the further growth of machine-learning models that can make use of it. This dataset can be used as the foundation for future machine learning models that can be used to construct educational applications (with interactive content) and computer games (that children can use to master sign language gestures), translate gestures into computer commands (to recognize alphabets and use them in other programs), or be incorporated into communicational programs (add-ons for alphabet recognition in MS Teams, Viber, WhatsApp, Webex, Zoom, etc.); alternatively, the SSL recognition model can be integrated into the software of existing video conferencing hardware devices (Polycom, Cisco, Avaya, Logitech, etc.). Interacting with computers using simple hand gestures can provide users with a natural and intuitive interface while also assisting hearing-impaired individuals with communication issues [41]. The quality of life, level of workplace participation, status in society, and general everyday communication abilities of people with impaired hearing are all intended to be improved by the applications proposed.

Author Contributions: Conceptualization, M.R.; methodology, M.R.; software, M.R.; validation, M.R., M.M., I.R., M.P.I., V.K. and S.D.-M.; formal analysis, M.R., M.M., V.K., I.R., S.D.-M. and M.P.I.; resources, M.R., M.M., V.K., M.P.I., S.D.-M. and I.R.; data curation, M.R., M.M., I.R., M.M., M.P.I., and S.D.-M.; writing—original draft preparation, M.R.; writing—review and editing, M.R., M.P.I., S.D.-M., V.K., I.R. and M.M.; visualization, M.R.; supervision, M.M., V.K. and S.D.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Science Fund of the Republic of Serbia, Grant No. 7502, Intelligent Multi-Agent Control and Optimization applied to Green Buildings and Environmental Monitoring Drone Swarms—ECOSwarm.

Data Availability Statement: The publicly available SSL dataset presented in this article can be found on GitHub: https://github.com/mlradak/serbian-sign-language (accessed on 5 February 2024).

Acknowledgments: Katarina Radakovic, a graduate special education teacher, has been a great inspiration and idea generator. She was able to recognize the genuine obstacles that hearing-impaired children face when working with them in elementary school. The initial concept for this study was to assist hearing-impaired children in acquiring the fundamentals of sign language. She served as an inspiration, a research advisor, and a participant in the dataset creation process. Her extensive school experience with hearing-impaired children and her understanding of Serbian Sign Language gestures was invaluable to this research.

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

References

1. Fox, N.F.; Woll, B.W.; Cormier, K.C. Best practices for sign language technology research. *Univers. Access Inf. Soc.* **2023**. [CrossRef]
2. Joksimoski, B.; Zdravevski, E.; Lameski, P.; Pires, I.M.; Melero, F.J.; Martinez, T.P.; Garcia, N.M.; Mihajlov, M.; Chorbev, I.; Trajkovik, V. Technological Solutions for Sign Language Recognition: A Scoping Review of Research Trends, Challenges, and Opportunities. *IEEE Access* **2022**, *10*, 40979–40998. [CrossRef]
3. La Grutta, S.; Piombo, M.A.; Spicuzza, V.; Riolo, M.; Fanara, I.; Trombini, E.; Andrei, F.; Epifanio, M.S. The Relationship between Knowing Sign Language and Quality of Life among Italian People Who Are Deaf: A Cross-Sectional Study. *Healthcare* **2023**, *11*, 1021. [CrossRef] [PubMed]
4. Mijatović, S.; Ristić, I. Znakovni jezik—Razvoj i karakteristike. In Proceedings of the Međunarodne Konferencije Aktuelnosti u Logopediji, Okupacionoj Terapiji i Socijalnom Radu: Izmedu Tradicije i Tranzicije, Belgrade, Serbia, 28–29 November 2019; pp. 25–39.

5. Dimić, N.; Isaković, L. *O Znakovnom Jeziku*; Univerzitet u Beogradu—Fakultet za Specijalnu Edukaciju i Rehabilitaciju, Izdavački Centar (IFC): Belgrade, Serbia, 2018; ISBN 978-86-6203-110-5.
6. Tateno, S.; Liu, H.; Ou, J.O.J. Development of sign language motion recognition system for hearing-impaired people using electromyography signal. *Sensors* **2020**, *20*, 5807. [CrossRef]
7. Law on the Use of Sign Language. Official Gazette of the Republic of Serbia, Number 38. 2015. Available online: http://demo.paragraf.rs/WebParagrafDemo/?did=272323 (accessed on 2 January 2024).
8. Bajić, D.R.; Nikolić, G.; Gordić, M.; Mouvet, K.; Herreweghe, M.V. Language attitudes towards Serbian Sign Language and experiences with deaf education in Serbia. *DiGeSt J. Divers. Gend. Stud.* **2021**, *8*, 1.
9. Marković, M.M. *Osobe sa Invaliditetom u Srbiji*; Statistical office of the Republic of Serbia: Belgrade, Serbia, 2014; p. 35.
10. Hand Landmarks Detection Guide for Python: MediaPipe—Google for Developers. Available online: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker/python (accessed on 5 January 2024).
11. TensorFlow Machine Learning Platform. Available online: https://www.tensorflow.org/ (accessed on 12 December 2023).
12. Jeon, H.; Choi, H.; Noh, D.; Kim, T.; Lee, D. Wearable Inertial Sensor-Based Hand-Guiding Gestures Recognition Method Robust to Significant Changes in the Body-Alignment of Subject. *Mathematics* **2022**, *10*, 4753. [CrossRef]
13. Albanie, S.; Varol, G.; Momeni, L.; Bull, H.; Afouras, T.; Chowdhury, H.; Fox, N.; Woll, B.; Cooper, R.; McParland, A.; et al. BBC-Oxford British Sign Language Dataset. *arXiv* **2021**, arXiv:2111.03635. [CrossRef]
14. Athitsos, V.; Neidle, C.; Sclaroff, S.; Nash, J.; Stefan, A.; Yuan, Q.; Thangali, A. The American Sign Language lexicon video dataset. In Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Anchorage, AK, USA, 23–28 June 2008.
15. Joze, H.R.V.; Koller, O. MS-ASL: A large-scale data set and benchmark for understanding American Sign Language. In Proceedings of the British Machine Vision Conference (BMVC), Cardiff, UK, 9–12 September 2019.
16. Li, D.; Opazo, C.R.; Yu, X.; Li, H. Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), Aspen, CO, USA, 2–5 March 2020.
17. Dreuw, P.; Forster, J.; Deselaers, T.; Ney, H. Efficient approximations to model-based joint tracking and recognition of continuous sign language. In Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition, Amsterdam, The Netherlands, 17–19 September 2008.
18. Momeni, L.; Varol, G.; Albanie, S.; Afouras, T.; Zisserman, A. Watch, read and lookup: Learning to spot signs from multiple supervisors. In Proceedings of the Asian Conference on Computer Vision (ACCV), Kyoto, Japan, 30 November–4 December 2020.
19. Albanie, S.; Varol, G.; Momeni, L.; Afouras, T.; Chung, J.S.; Fox, N.; Zisserman, A. BSL-1K: Scaling up co-articulated sign language recognition using mouthing cues. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020.
20. Chai, X.; Wang, H.; Chen, X. *The Devisign Large Vocabulary of Chinese Sign Language Database and Baseline Evaluations*; Technical Report VIPL-TR-14-SLR-001; Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS: Beijing, China, 2014.
21. Huang, J.; Zhou, W.; Li, H.; Li, W. Attention-based 3D-CNNs for large-vocabulary sign language recognition. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *29*, 2822–2832. [CrossRef]
22. Zhou, H.; Zhou, W.G.; Qi, W.; Pu, J.; Li, H. Improving sign language translation with monolingual data by sign back-translation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021.
23. von Agris, U.; Knorr, M.; Kraiss, K. The significance of facial features for automatic sign language recognition. In Proceedings of the 8th IEEE International Conference on Automatic Face Gesture Recognition, Amsterdam, The Netherlands, 17–19 September 2008.
24. Camgoz, N.C.; Hadfield, S.; Koller, O.; Ney, H.; Bowden, R. Neural sign language translation. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
25. Viitaniemi, V.; Jantunen, T.; Savolainen, L.; Karppa, M.; Laaksonen, J. S-pot—A benchmark in spotting signs within continuous signing. In Proceedings of the 9th international conference on Language Resources and Evaluation (LREC 2014), Reykjavik, Iceland, 26–31 May 2014.
26. Adaloglou, N.; Chatzis, T.; Papastratis, I.; Stergioulas, A.; Papadopoulos, G.T.; Zacharopoulou, V.; Xydopoulos, G.J.; Atzakas, K.; Papazachariou, D.; Daras, P. A comprehensive study on sign language recognition methods. *arXiv* **2020**, arXiv:2007.12530.
27. Sridhar, A.; Ganesan, R.G.; Kumar, P.; Khapra, M.M. Include: A large scale dataset for indian sign language recognition. In Proceedings of the 28th ACM International Conference on Multimedia, Seattle, WA, USA, 12–16 October 2020.
28. Ko, S.-K.; Kim, C.J.; Jung, H.; Cho, C. Neural sign language translation based on human keypoint estimation. *Appl. Sci.* **2019**, *9*, 2683. [CrossRef]
29. Özdemir, O.; Kındıroğlu, A.A.; Camgoz, N.C.; Akarun, L. BosphorusSign22k Sign Language Recognition Dataset. In Proceedings of the LREC2020 9th Workshop on the Representation and Processing of Sign Languages: Sign Language Resources in the Service of the Language Community, Technological Challenges and Application Perspectives, Marseille, France, 11–16 May 2020.
30. Sincan, O.M.; Keles, H. AUTSL: A large scale multi-modal Turkish Sign Language dataset and baseline methods. *IEEE Access* **2020**, *8*, 181340–181355. [CrossRef]
31. Buttar, A.M.; Ahmad, U.; Gumaei, A.H.; Assiri, A.; Akbar, M.A.; Alkhamees, B.F. Deep Learning in Sign Language Recognition: A Hybrid Approach for the Recognition of Static and Dynamic Signs. *Mathematics* **2023**, *11*, 3729. [CrossRef]

32. Adeyanju, I.A.; Bello, O.O.; Adegboye, M.A. Machine learning methods for sign language recognition: A critical review and analysis. *Intell. Syst. Appl.* **2021**, *12*, 200056. [CrossRef]
33. World Federation of the Deaf. Available online: http://wfdeaf.org/our-work/ (accessed on 7 January 2024).
34. Jones, G.A.; Ni, D.; Wang, W. Nothing about us without us: Deaf education and sign language access in China. *Deaf. Educ. Int.* **2021**, *23*, 179–200. [CrossRef]
35. Xiao, X.; Chen, X.; Palmer, J. Chinese Deaf viewers' comprehension of sign language interpreting on television: An experimental study. *Interpreting* **2015**, *17*, 91–117. [CrossRef]
36. Ristić, I. *Partnerska Uloga Porodice u Inkluzivnom Obrazovanju*; Univerzitet u Prištini, Kosovska Mitrovica, Učiteljski fakultet Prizren-Leposavić: Mitrovica, Kosovo, 2023; ISBN 978-86-84143-63-3.
37. Schembri, A.; Fenlon, J.; Rentelis, R.; Reynolds, S.; Cormier, K. Building the British Sign Language Corpus. *Lang. Doc. Conserv.* **2013**, *7*, 136–154.
38. Neidle, C.; Thangali, A.; Sclarof, S. Challenges in the development of the American Sign Language Lexicon Video Dataset (ASLLVD) Corpus. In Proceedings of the 5th Workshop on the Representation and Processing of Sign Languages: Interactions between Corpus and Lexicon, Language Resources and Evaluation Conference (LREC), Istanbul, Turkey, 21–27 May 2012; pp. 143–150. Available online: https://hdl.handle.net/2144/31899 (accessed on 10 January 2024).
39. Buehler, P.; Zisserman, A.; Everingham, M. Learning sign language by watching TV (using weakly aligned subtitles). In Proceedings of the IEEE Conference Computer and Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 2961–2968. [CrossRef]
40. Camgöz, N.C.; Saunders, B.; Rochette, G.; Giovanelli, M.; Inches, G.; Nachtrab-Ribback, R.; Bowden, R. Content4All Open Research Sign Language Translation Datasets. In Proceedings of the 2021 16th IEEE International Conference on Automatic Face and Gesture Recognition, Jodhpur, India, 15–18 December 2021; pp. 1–5. [CrossRef]
41. Awaluddin, B.-A.; Chao, C.-T.; Chiou, J.-S. Investigating Effective Geometric Transformation for Image Augmentation to Improve Static Hand Gestures with a Pre-Trained Convolutional Neural Network. *Mathematics* **2023**, *11*, 4783. [CrossRef]