University of Economics and Human Sciences
in Warsaw
Computer Engineering

Ömer Zırh
37344
Full-Time studies

PJM: Polish Sign Language Recognition using Leap Motion Controller

Bachelor's thesis written under the supervision of Ruslana Prus
PhD
Warsaw,2021

Table of contents

Table of figures

Introduction

Until today, intelligent interfaces have been designed to increase human and computer interaction, to facilitate our daily work, in short, to simplify life. Studies on human computer interaction are generally aimed at the remote control of the newly developed technological products. In order to command these systems, sound and image signals are used and preferred depending on the need.

Sign language is an important tool that enables communication between hearing impaired individuals and the rest of society. With the rapid development of technology, the communication gap between the society and hearing impaired individuals was increasing. As a solution to this, studies were carried out to recognize the sign language on the computer. Computer translation systems and development of remote control systems will create innovations and facilities in many technological and social areas with the successful recognition of sign languages.

There have been different approaches to recognizing sign languages by computer. These are basically image and glove based systems. Electronically supported gloves, cameras, Kinect and Leap motion devices are preferred in those systems. Although these two approaches -image and glove-based systems- have high accuracy success, their reliability is low in natural environments due to their disadvantages. The most important disadvantage is that a slight difference in hand signs changes the meaning.

This project aimed to develop a functional system for recognizing fixed hand gestures using the Leap Motion Controller. Based on previous studies and relatively simple approaches to recognizing fixed hand gestures, a machine learning approach was considered.

In this study, the recognition of Polish Sign Language (PJM) letters by using a digital sensor based Leap Motion Controller was studied. Leap Motion Controller uses a pair of cameras and infrared LEDs to obtain a hand image with depth information. With the help of motion sensors and cameras, Leap Motion can detect and record hand and finger movements in three-dimensional space with a precision of up to one percent of a millimeter with zero delay. Leap Motion Controller can monitor movements of up to 200 frames per second. This device scans hand and finger movements in the field of view, providing a continuous data series and image frames. Each image frame contains basic scanning information of hand and finger movements that are tracked and within the field of view, movement and status information.

Hand and finger movements belonging to sign languages consist of fixed and moving gestures. A system has been designed to recognize fixed gestures of PSLI. First of all, the data types provided by Leap Motion Controller were examined. Important and distinctive features have been identified for the sign language. The data was created by taking different points of the hand. In the interface part, the picture of the hand sign that the user should show is displayed. If the user demonstrates the action correctly, the next sign is passed.

'Polish Sign Language Recognition Using Leap Motion Controller' capable of recognizing fixed signs. However, this study will not be sufficient when it is desired to recognize signs that are shown with two hands or moving. For the moment, it can be used in studies about the recognition of sign language or in sign language education at a basic level.

Chapter 1

Analysis of sign language recognition systems using Leap Motion

In this chapter, Previous studies about Recognition of Sign Languages is presented. Previous studies have been presented because comparing this study will provide further understanding. Most of the studies have been done in languages other than Polish. Examining the previous studies has provided us with some ideas about the study we have done and has shown that our study is genuine.

Mariusz Oszust and Marian Wysocki discuss recognition of Polish sign language words with Kinect in their articles. The classifier used two sets of features, the first to define the hands produced by Kinect and the second to identify the hands that are traced as flesh-colored regions in images acquired by Kinect. The resulting feature vectors representing the PSL words are aggregated to explore the natural data segments between them. This step was intended to help uncover potential problems that might arise during recognition and provided additional information about the data being processed. Results of tenfold cross validation tests for the classifier with two feature sets are given. The classifier with data provided by Kinect gave promising recognition accuracy (89.33%), but it was noted to consider adding new features responsible for hand shape description to achieve better results (98.33%). (Oszust & Wysocki, 2013)

The project of Catarina Araujo and Sofia Santos had developed an application which called Leap Reader. It has an interesting design, is a wearable product that decodes sign language into text and displays it on the attached screen worn around the neck. In this way, it will be easier to communicate with deaf people. The translator is to work in real time, and the pickup will use the Leap Motion module located on the back of the housing. (Sitnik, 2014)

Chuan developed a motion recognition system of American Sign Language (ASL) using Leap Motion. A recognition system was targeted using the properties of the data obtained from the sensors and the K-Nearest Neighbor algorithm (KNN) and the Support Vector Machine (SVM) classifier method of 26 English letters in American Sign Language. For each finger, features such as direction, length, joint connection positions, palm orientation vector positions, open-closed state of the fingers and squeezing power have been extracted. These hand and finger features are used for the recognition of sign language letters. The test results were obtained with the highest classification rate of

72.78% with the K-Nearest Neighbor algorithm and with the Support Vector Machine with a rate of 79.83% (Chuan, 2014)

Mohandes aimed a new approach on the Arabic Sign Language (ArSLR) recognition system using Leap Motion. Leap Motion tracks and detects the position and movement information of hand and fingers and sends this information to the instant computer. A fixed motion recognition system has been developed using all the Arabic alphabet signs with a single hand. For the training and testing data set, a total of 2,800 frames of information in total, 100 for each letter, were collected. MATLAB was used in the analysis and operation stages. In this study, the most suitable 12 properties were selected. In addition to data collection, the system consists of three stages: pre-processing stage, feature extraction and classification stage. In the study performed on 28 signs belonging to Arabic Sign Language (ArSL), 98% accuracy rate was achieved with Naive Bayes classifier and 99% accuracy rate with Multi-Layer Sensor (MLP) classifier. (Mohandes, 2014)

Again, in a similar approach, Elons et al. Has developed a recognition system for Arabic Sign Language (ArSL) gestures using the Leap Motion device. Leap Motion device detects and captures hand and finger movements in 3D digital format. Leap Motion sends frame information of each motion acquired. These obtained spatio-temporal properties are interpreted with the Multilayer Perceptron (MLP). It has been tested on 50 different dynamic signs belonging to Arabic Sign Language (ArSL). The recognition accuracy rate was achieved at 88%. (Samir, 2014)

In this paper Naglot and Kulkarni display the importance of American Sign Language and proposed technique for classification and their efficient results. American Sign Language uses only one hand to demonstrate gestures, so interpreting and understanding become easier. Gestures are captured using the “Leap Motion Controller” digital sensor. The proposed system has created a classification model that takes the feature set as input by using a Multilayer Perceptron (MLP) neural network with the Back Propagation (BP) algorithm. They studied 26 different letters of American Sign Language. The Multilayer Perceptron (MLP) is run on a dataset of 520 samples (20 samples from each letter). Recognition rate of proposed system is 96.15%. (Kulkarni, 2016)

After researching various works that had previously been done, it was realized that there were very few researches on the Polish Sign language with the Leap Motion Controller with this approach. The most important feature that distinguishes this study from other studies is the use of leap motion controller device. Leap Motion Controller

was chosen among various devices. The greatest motivation for choosing Leap Motion Controller was its easy availability, cost. portability and its magical software. Leap Motion, which has developed a lot in recognition of hand gestures with its constantly developing software, was seen as the most reasonable device for recognizing the sign language.

We can enable machines to learn sign language and classify sign language images quickly and effectively with the help of machine learning. Here, the variation of gestures in sign language from person to person, from display to display makes it more difficult to define. It may take a long time to code all of this diversity, but thanks to machine learning, we have established a more stable and open model by transferring various signals to the machine.

Chapter 2

Description of device used to recognition

2.1 General description of Leap Motion Controller and its software structure

Leap Motion is a peripheral input device that allows the user to interact with the software using hand gestures. Leap Motion can detect hand and finger movements in three-dimensional space with an accuracy of up to one percent of a millimeter. This device, which can scan 10 fingers with a high speed of 290 fps, offers a very high sensitivity rate. This 8 cm long device that can be easily operated on the computer by simply moving the fingers of the user without touching the computer and anything around it by connecting to the computer via USB channel. Schematic and physical view of leap motion showed on Figure 1 and Figure 2.

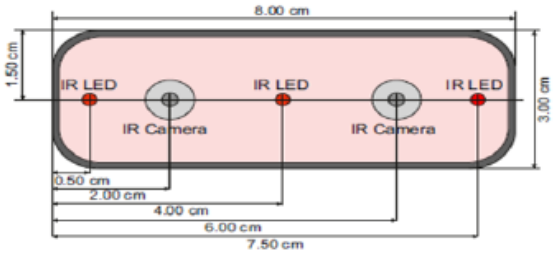

Figure 1 Schematic display of Leap Motion Controller. (Telli, 2020)

Figure 2 Physical display of Leap Motion Controller. (Ultraleap, 2020)

This device can detect and track objects such as hands, fingers and arms. The use of the device by detecting hand movements is shown in Figure 3.

Figure 3 Hand detection example. (roboticsbd)

Leap Motion detects the movements of hands and 10 fingers with its highly sensitive sensors and transfers them to the computer and allows the computer to operate according to these data. When connected to the computer, this small device illuminates the hand with three LED lights, while motion sensors and cameras can detect and record every movement of 10 fingers up to a hundredth of a millimeter with zero delay.

Leap Motion Controller is very sensitive device and successful in detecting and monitoring hand and finger movements in 3 dimensions. In this way, it is easily separated from other systems and devices. Although Leap Motion Controller is a new technology, it is in a very attractive position compared to other systems in terms of product cost. Sample movements detected by the device are shown below. Illustrating of a hand that tracing circle has shown on Figure 4. Illustrating of swiping has shown on Figure 5. Illustrating of tapping a button with finger has shown on Figure 6. Illustrating of tapping a vertical screen has shown on Figure 7.

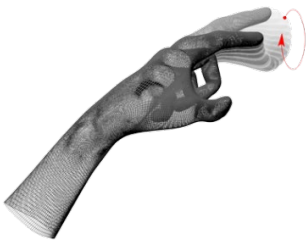

Figure 4 Circle – A finger tracing circle. (Motion, Gestures)

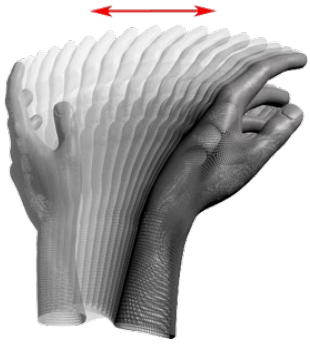

Figure 5 Swipe — A long, linear movement of a hand and its fingers. (Motion, Gestures)

Figure 6 Key Tap — A tapping movement by a finger as if tapping a keyboard key (Motion, Gestures)

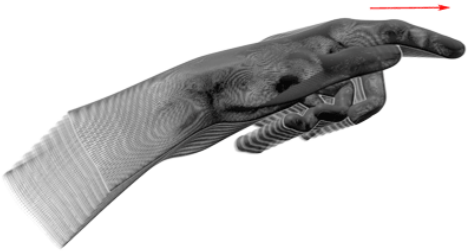

Figure 7 Screen Tap — A tapping movement by the finger as if tapping a vertical computer screen. (Motion, Gestures)

2.2 Hardware structure and field of vision of Leap Motion Controller

Leap motion controller is not as complex as it seems. What makes it complex is the software. Device's main parts are two cameras and three infrared LEDs. These parts track infrared light with a 850 nanometers wavelength, which is outside the visible light spectrum. The hardware structure has shown in Figure 8.

Figure 8 The hardware structure of Leap Motion Controller. (Introducing LeapUVC: A New API for Education, Robotics and More, 2018)

Thanks to its wide-angle lenses, the device has a large eight-cubic foot area of interaction that takes the form of an inverted pyramid, which is the intersection point of the field of vision of binocular cameras. It has a limited field of view, 60 cm above the device. Field of vision has shown in Figure 9. Since the device displays the hand in 3d in space, it is difficult to show at a distance beyond this range. The LED light intensity is ultimately limited by the maximum current that can be transferred through the USB connection.

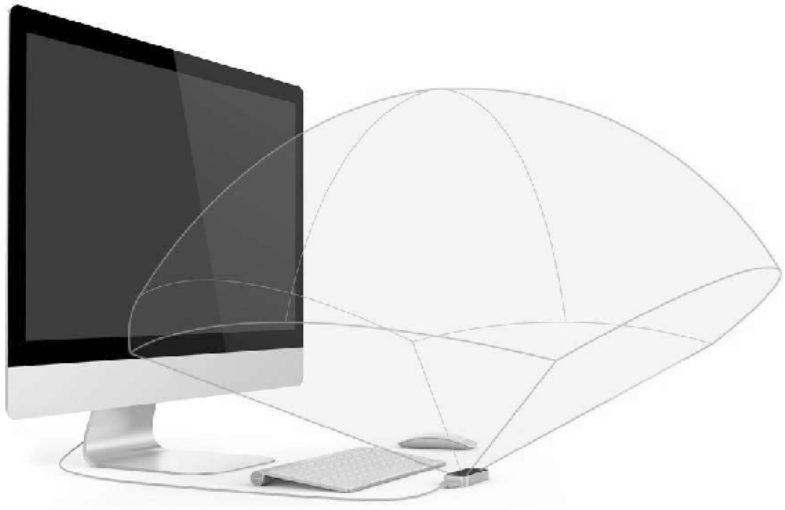

Figure 9 Field of view of the Leap Motion device. (Dobosz, 2018)

The data detected by the sensor is transferred to the local memory of the USB controller and the necessary resolution adjustments are made. At last, the final data are transferred to the Leap Motion tracking software over USB.

As the Leap Motion Controller tracks the near infrared, images appear grayscale. Intense sources or reflectors of infrared light can make it difficult to distinguish and track hands and fingers. The device is still under development and it is highly competitive in the market. (Motion, How Does the Leap Motion Controller Work?, 2015)

Chapter 3

Characteristics and recognition of sign languages

3.1. Classification of the sign language alphabet

Classification is to distribute data between various classes defined on a data set. The training set contains a data set created by the records from which the classification model will be obtained. Each record in the data set contains a set of attributes and one of these is the class attribute information. A model is created with objects known to which class (learning set - training set). The classification model is the function of the class attribute with the values of other attributes. The success of the created model is tested with objects (test set) that are not included in the learning set. The purpose of classification is to correctly assign new records to previously determined classes. Usually, the data set is divided into training and test sets, while the training set and the model are built, the test set is used for model validation.

In this chapter, Machine learning and the Random Forest Algorithm that will be used in the study will be explained.

Machine learning

Machine Learning is programmed to increase performance or reduce error based on a criterion using sample data or past experience. There are many approaches to machine learning and most of these approaches are capable of prediction, classification, reduction. We can use any of the machine learning methods for a sample data or a problem which has no algorithm. At the same time, machine learning is more useful if there is a few expert in a job or people cannot explain their expertise to the algorithm.

Used machine learning libraries and algorithm

There are a wide variety of machine learning libraries and algorithms written in the Python language. These libraries and algorithms can be used with different machine learning methods. In this section, only used ones in the project will be explained.

Libraries

Pandas

Pandas is a Python library that provides fast, flexible and impressive data structures designed to make, working with "relational" and "tagged" data, easy and intuitive. It aims to be the basic building block for practical, real-world data analysis in Python. It also has the broader purpose of being the most powerful and flexible open source data analysis /

manipulation tool available in any language. In this project, it is used to read data from the csv file for training.

NumPy

NumPy (Numeric Python) is a math library that enables rapid scientific calculations. Numpy arrays are more useful than python lists in terms of speed and functionality. NumPy is critical in this project; linear algebra methods were used to calculate the angle of each bone data. It was also used to append the last calculated data to the arrays.

Scikit Learn

Scikit-learn (Sklearn) is the most transformative and robust library for machine learning in Python. It offers a range of efficient tools for machine learning and virtual modeling, including classification, regression, clustering, and dimensionality reduction in a planning commitment in Python. This library, written largely in Python, is built on NumPy, SciPy and Matplotlib. (tutorialspoint, n.d.)

In this project Random Forest Classifier library imported from Scikit Learn library.

Random Forest Classifier

One tree structure is created in the decision tree algorithm. Then, a root for the tree structure and branches are created according to the values of the variables and the result is reached. If n of these trees are formed by using the values in the obtained data set randomly, it is called a random forest algorithm. In other words, it is the forest which is the whole tree with branches formed according to random values. (Smolyakov, 2017) Figure 10 can be examined to make it more understandable.

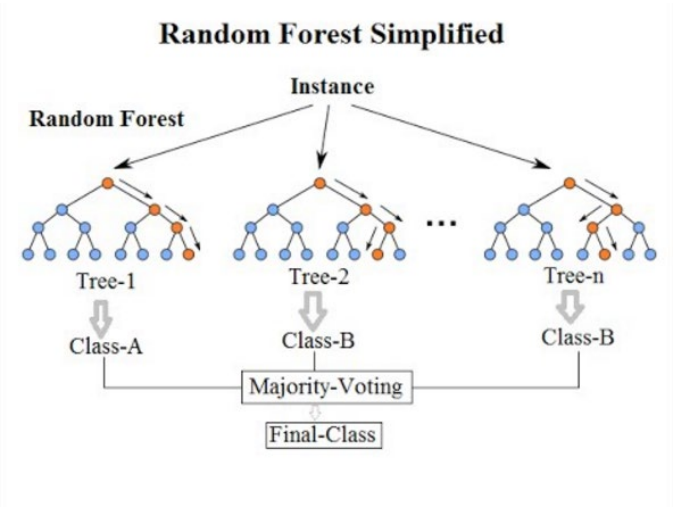

Figure 10 Random Forest simplified chart. (Koehrsen, 2017)

The algorithm creates too many tree structures. The best results are chosen from each tree. Correct branches are formed by making a comparison between the obtained results. A summary can be made by saying that Random Forest is an algorithm that produces the most compatible result by processing the results of different algorithms.

3.2 Characteristics of sign languages

Sign language is a language consisting of visual elements that enable, hearing and speech impaired people to communicate with each other, using hand gestures and facial expressions. The use of these visual elements; The signs may differ according to their usage in the meaning and sentence. There are two types of approaches in sign language. In the first approximation type, users make a mark for each word or phrase, while in the second approximation type, they make a mark for each letter. The second type of approach is often used to express words that do not have word marks or cannot be understood. Sign languages have their own formal structure like spoken languages. There are over 40 sign languages in the world in which various finger alphabets are used. Sign languages of countries are generally not alike. Examples of these are ASL for American Sign Language, BSL for British Sign Language, DGS for German Sign Language, and TID for Turkish Sign Language. In the Polish language, there are signs for the finger of 36 letters, 4 of which are compound letters. However, there are finger signs of a limited number of letters in phonetic alphabets such as Chinese or Japanese.

3.3 Sign language recognition systems

Until today, many systems have been designed using machine learning methods in sign language recognition. The systems used in these studies are divided into two, as electronic glove and image based systems according to their data types. The first studies on sign languages were electronic glove based systems. These systems use information about the hand and fingers, such as various positions and angles. Since the person performing the sign must wear gloves, it has features that are not easy to use and restrict the person's movements. In image-based systems, it is used to recognize signs and to extract the attributes of images taken by one or more cameras. A lot of studies and research have been done on these systems in recent years, especially due to ease of use, low cost and similar important factors.

Sign languages consist of fixed and moving gestures. Fixed gestures are represented by the stance of the hand at any given moment or by a single image. On the other hand, moving signs are represented by the changes in the position and shape of the hand during the representation.

There were advantages and disadvantages in all systems, but the shape-based system is the most suitable for this project in terms of both cost, accessibility and convenience. For this reason, Leap Motion Controller device has been purchased to realize the recognition of sign language idea.

3.3.1 Electronic glove based systems

The first studies on the recognition of sign languages were electronic glove-based systems. Data devices are sensor and electronic gloves. The pointer performs the gestures while wearing electronic gloves. The rotation, speed, angle, position and similar electrical signals received from the gloves during the demonstrating of the signs are used for the recognition process. In these systems, the reliability of the data is high and the data purity, i.e., the rate of exposure to noise, is low. However, there are also difficulties brought by gloves. Markers performing the marks must wear gloves and be familiar with its use. In addition, these systems are not very useful due to the high cost of gloves, restricting movements and being unavailable in all environments.

3.3.2 Image-based systems

These systems consist of still and sequential images. Although image-based systems are advantageous in terms of ease of use, cost and system simplicity. These systems can cause problems at data acquisition and processing stages. In addition to the lighting conditions of the environment, image quality and similar problems, it can also occur during segmentation and modeling of the elements that make up the movement in the images. However, the high usability and applicability of image-based systems by pre-processing such as object segmentation, noise filtering and dimension reduction is a reason for preference. Image-based systems can be divided into two categories as motion-based and shape-based systems.

3.3.3 Motion based systems

Sign languages use fixed gestures to express a word or sentence. Gestures consist of changes in the position and shape of the movement when demonstrating the hand sign. One or two hands may be used when showing Gesture.

Chapter 4

Data collection and application

Gesture movements performed with one hand were used in this study. The data received from the Leap Motion controller is transferred to the computer by USB. These transferred data are used to create data and to recognize hand gestures. Leap Motion Controller sensors present hand and finger movements as a data string. In this chapter all steps while building a machine learning model will be explained as shown on Figure 11.

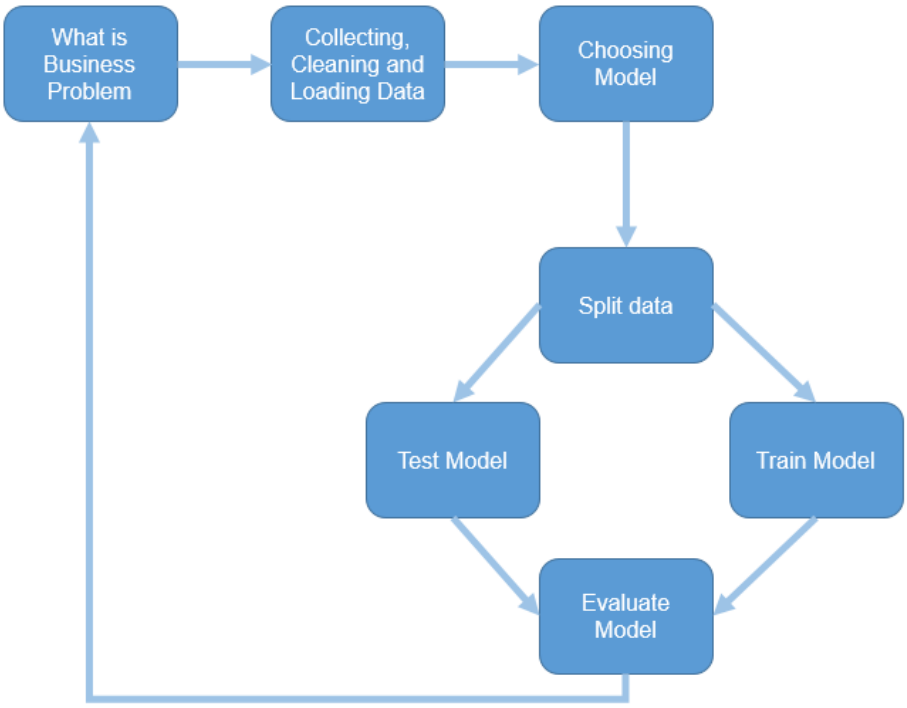

Figure 11 Implementing machine learning steps chart.

Firstly, the problem was Polish Sign Language shall be recognized by the computer. This request is made after passing the thought steps specified in the introduction. To summarize briefly, this idea was developed due to the limited research on the recognition of the Polish Sign Language and the need of recognition systems of PSL in this field.

There were various devices and methods specified in the chapter "Characteristics and Recognition of Sign Languages" for the recognition, but the most reasonable of them was Leap motion which was chosen as the most suitable device for this project. Leap motion Controller detects hand and arm movements, determines their positions in space and presents them to the user. How this data will be processed is up to the developer's choice and method. Accordingly, work started to create the data set.

First of all, an algorithm was established to process the data coming from Leap Motion Controller and increase its accuracy. According to this algorithm, the data received by the

leap motion sensor was processed. The positions of the points shown in the Figure 12 were obtained using the Leap Motion API.

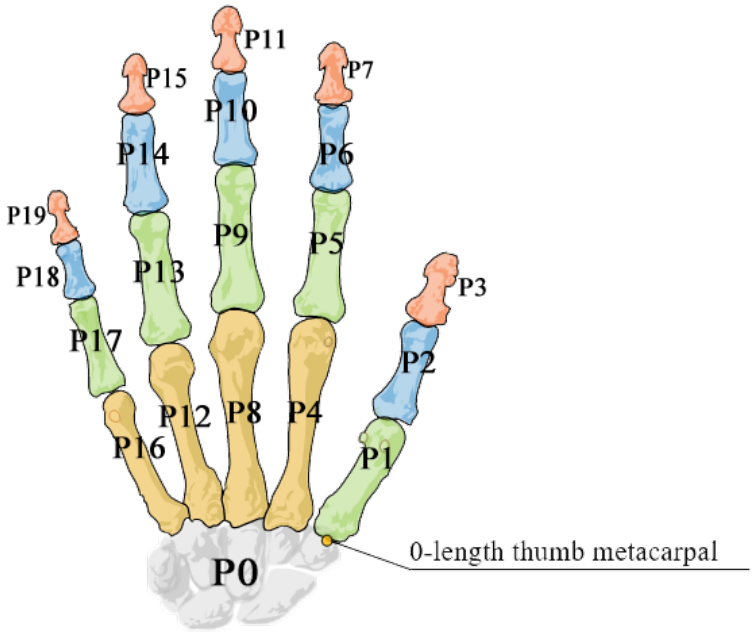

Figure 12 Points in hand obtained from Leap Motion Controller. (Villarreal, 2007)

The distances of these positions to the one behind them and their angles to each other at these distances were calculated. In other words, instead of taking the positions, it was aimed to increase the accuracy by taking the angle of the locations to each other. If we give an example from the code, p0 was subtracted from p1, p0 was subtracted from p4, and the angle of these results was calculated, so it was aimed to create a dataset by calculating the angles of each knuckle of the fingers separately. The code block that performs these operations is shown in Figure 13 and Figure 14.

```
# palm
P0 = np.array(frame.hands[0].palm_position.to_float_array())
# thumb
P1 = np.array(frame.hands[0].fingers[0].bone(1).next_joint.to_float_array())
P2 = np.array(frame.hands[0].fingers[0].bone(2).next_joint.to_float_array())
P3 = np.array(frame.hands[0].fingers[0].bone(3).next_joint.to_float_array())
# index
P4 = np.array(frame.hands[0].fingers[1].bone(0).next_joint.to_float_array())
P5 = np.array(frame.hands[0].fingers[1].bone(1).next_joint.to_float_array())
P6 = np.array(frame.hands[0].fingers[1].bone(2).next_joint.to_float_array())
P7 = np.array(frame.hands[0].fingers[1].bone(3).next_joint.to_float_array())
# middle
P8  = np.array(frame.hands[0].fingers[2].bone(0).next_joint.to_float_array())
P9  = np.array(frame.hands[0].fingers[2].bone(1).next_joint.to_float_array())
P10 = np.array(frame.hands[0].fingers[2].bone(2).next_joint.to_float_array())
P11 = np.array(frame.hands[0].fingers[2].bone(3).next_joint.to_float_array())
# ring
P12 = np.array(frame.hands[0].fingers[3].bone(0).next_joint.to_float_array())
P13 = np.array(frame.hands[0].fingers[3].bone(1).next_joint.to_float_array())
P14 = np.array(frame.hands[0].fingers[3].bone(2).next_joint.to_float_array())
P15 = np.array(frame.hands[0].fingers[3].bone(3).next_joint.to_float_array())
# pinky
P16 = np.array(frame.hands[0].fingers[4].bone(0).next_joint.to_float_array())
P17 = np.array(frame.hands[0].fingers[4].bone(1).next_joint.to_float_array())
P18 = np.array(frame.hands[0].fingers[4].bone(2).next_joint.to_float_array())
P19 = np.array(frame.hands[0].fingers[4].bone(3).next_joint.to_float_array())
```

Figure 13 Code block that allows each finger node to be assigned to an array

```
# A5,6 on thumb
A = get_cos_angle_2vec((P2-P1), (P1-P0))
angles.append(A)
A = get_cos_angle_2vec((P3-P2), (P2-P1))
angles.append(A)

# A7-9 on index
A = get_cos_angle_2vec((P5-P4), (P4-P0))
angles.append(A)
A = get_cos_angle_2vec((P6-P5), (P5-P4))
angles.append(A)
A = get_cos_angle_2vec((P7-P6), (P6-P5))
angles.append(A)

# A10-12 on middle
A = get_cos_angle_2vec((P9-P8), (P8-P0))
angles.append(A)
A = get_cos_angle_2vec((P10-P9), (P9-P8))
angles.append(A)
A = get_cos_angle_2vec((P11-P10), (P10-P9))
angles.append(A)
```

Figure 14 Code block that allows the cosines of angles to be assigned to array named angles.

A label set was created in correspondence of the received data. Each of the data corresponds to a label on the same row. Then the dataset was divided into 30% test and 70% train. Random Forest classifier which is explained in 3rd chapter, implemented in the end to recognize PSL gestures.

Model's prediction score was calculated using Sklearn score() function. Score was 0.982 as shown with the code block in Figure 15.

```
model = RandomForestClassifier(random_state=42, n_jobs=-1)

model.fit(X_train,y_train)
RandomForestClassifier(n_jobs=-1, random_state=42)

model
RandomForestClassifier(n_jobs=-1, random_state=42)

model.score(X_test, y_test)
0.9820455274126323
```

Figure 15 Code blocks shows scoring of model.

After these steps, a method had to be chosen to move on to coding. Python was chosen as the programming language because it was easy to learn, comfortable to use and very suitable for this project with its rich libraries of machine learning. Later, it was decided to use Tkinter framework for the interface. Tkinter is a performance and lightweight framework, despite the old fashion look. Although lightness is not necessary for this project, it's aimed to be upgraded in the future so tkinter framework has seen appropriate for efficiency.

Every time the program is run, the model is trained by reading the csv files using Pandas library's read_csv() function, RandomForestClassifier() and fit() functions using sklearn library as shown on Figure 16.

```
xx = pd.read_csv('dataset/feature_angles.csv')
yy = pd.read_csv('dataset/labels.csv')
x=xx
y = yy['sign_id']
model = RandomForestClassifier(random_state=42)
model.fit(x,y)
```

Figure 16 Code block allows reading dataset and label from the file and training the model

Fitting means training. Fit function is used for training the model using data samples. Fit function adjusts the weights according to data values so that better accuracy can be obtained.The model can be used for estimates using the .predict () method call after training.

When separating the data we have into train, test, a percentage is determined. The RandomForestClassifier function divides this data in different places each time. If a random_state value is specified, it is divided by that value each time. That is, it has been tested with the same test data. 42 is inspired by the number calculated by the supercomputer at the end of 7.5 million years in The Hitchhiker's Guide to the Galaxy.

Firstly, the basic functions taken from the Leap Motion API documentation were written. As seen on Figure 17, Leap Motion API function has been used. on_init()

function confirms that the controller object is initialized. On_connect function confirms that controller is connected. on_disconnect() and on_frame functions are also same functionality with previous ones.

Desired codes can be written in these functions. For example, the code written in the on_disconnect function will work when the connection to the Leap Motion Controller is disconnected. Likewise, in the on_frame function, when the new frame of each tracking data is available, the codes we write into it will work.

```
class LeapMotionListener(Leap.Listener):

    def on_init(self,controller):
        print "Loaded."

    def on_connect(self,controller):
        self.i=0
        print "Leap Motion Connected."


    def on_disconnect(self,controller):
        print "Leap Motion disconnected."


    def on_frame(self,controller):
        self.i+=1
        frame=controller.frame()
```

Figure 17 Code block that includes Leap Motion API functions

The main job would be done in the on_frame function. First, it is checked whether the hand is right or left. If it is right, the data received from the controller is processed and assigned to an array and a guess is made. The code block that allows the data to be processed according to the established algorithm is shown in Figure 18.

```
for i,hand in enumerate(frame.hands):
    typeHand=0 if hand.is_left else 1

    if(typeHand==0 and i==0):#left 1. Hand
        pass

    elif (typeHand==1 and i==0): #right 1 hand

        dataSample =getAngles(frame)
        dataToSave = dataSample
        predicted = model.predict([dataSample])
```

Figure 18 Code blocks allows control which hand user uses and predict demonstrated sign.

If the gesture shown on the interface is the same as the gesture shown by the user, the next one will appear on application. Prediction process repeats in every 3 seconds because 3 seconds is enough time to change a hand movement. The individual and other data that the user predicted correctly is saved for future processing.

4.1 Creating dataset

In this study, the dataset was created using the right hand. Right hand data taken from two people and data corresponding to 20 fixed separate letters were transferred to the computer environment with the help of Leap Motion Controller.

There are 22 coordinates showing the different positions of the palm, thumb, index finger, middle finger, ring finger and little finger with different points. Each data points to a location in space. The dataset is created by taking each position every 3 seconds - so the user can change the position of their hand within that time.

While data is being collected, a certain number of data at the beginning and at the end have been deleted due to possible errors in the Leap Motion Controller device. A total of 9450 points per gesture were collected from each user, with a total of 450 samples per sign. 70% of this data was used for education and 30% was used for testing. When training data, it is of great importance to divide the data into training set and test set in order to avoid overfitting. First, the model model is trained and the test set is used to calculate accuracy. According to experimental studies, the best results are obtained if we use 20-30% of the data for testing and the remaining 70-80% for training. Accordingly, a ratio of 70:30 has been determined for this dataset. (Gholamy, Kreinovich, & Kosheleva, 2018)

Sample datas showed on Figure 19. The values shown here are created by processing the data received from the Leap Motion Controller. In the previous chapters, how the data is processed has been explained, but if we will explain in a single paragraph: The coordinates showing the position of the hand's 20 points (palm and knuckles) in space were taken from the Leap Motion Controller. The cosines of each node were calculated by subtracting one from the previous one in the received coordinates. In total, 22 values shown in Figure 19 were assigned to the array and transferred to the angles.csv file.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
-0.0624	0.787127	0.632883	0.854206	-0.06807	0.856356	0.615365	0.999651	0.994771	0.112547	0.26647	0.867681	-0.05384	0.346819	0.796375	-0.04444	0.293438	0.857369	0.268937	-0.99843	0.976013	0.973049
-0.05331	0.742713	0.63633	0.860751	0.014214	0.850674	0.489183	0.999373	0.995004	-0.01546	0.1519	0.885218	-0.10356	0.087681	0.858582	-0.03075	0.043142	0.725476	0.295611	-0.98218	0.973756	0.985746
-0.01952	0.750659	0.636127	0.859495	-0.02142	0.850702	0.521388	0.999018	0.992624	-0.01824	0.19786	0.878477	-0.1065	0.157492	0.837673	-0.04461	0.109254	0.751287	0.341123	-0.98816	0.973534	0.983946
-0.04685	0.712643	0.640974	0.86435	-0.00237	0.850687	0.328621	0.996968	0.995004	-0.02564	0.22873	0.870269	-0.09555	0.184501	0.836184	-0.02455	0.161225	0.768168	0.410313	-0.96772	0.973342	0.977916
0.003389	0.775479	0.626905	0.857799	-0.16428	0.862879	0.624072	0.999155	0.995488	0.167015	0.029502	0.868261	0.097301	0.045596	0.871282	0.409464	0.326555	0.848373	0.286238	-0.96366	0.966383	0.644361
0.006976	0.775143	0.632532	0.857349	-0.13693	0.917377	0.638039	0.996636	0.998759	0.286982	0.086649	0.837512	0.252452	0.209463	0.796172	0.50141	0.33303	0.864591	0.293915	-0.94818	0.945914	0.609864
0.107153	0.774606	0.623918	0.864362	-0.07263	0.847209	-0.11604	0.374116	0.788283	0.350283	0.414502	0.851671	0.300906	0.544984	0.987591	0.287291	0.775355	0.995194	-0.41081	0.844159	0.759602	0.69754
0.167868	0.776276	0.636533	0.860338	-0.22782	0.916099	-0.23861	0.125232	0.683955	0.052085	-0.12121	1	0.201081	-0.1779	0.862175	0.315491	0.862116	0.952572	-0.9728	0.631779	0.913182	-0.17341
0.295744	0.776492	0.634734	0.861741	-0.44952	0.650806	-0.0968	-0.04734	0.932551	0.087771	-0.0178	0.849897	0.256563	0.087894	0.741891	0.239251	0.191717	0.813363	-0.28293	0.995629	0.972322	0.885354
0.227626	0.742212	0.639728	0.860838	-0.17494	0.812708	0.701181	0.996781	0.995004	0.331128	0.00621	0.799149	0.325849	0.172993	0.804464	0.19128	0.323471	0.796755	0.277817	-0.96537	0.942217	0.959644

Figure 19 Calculated data from coordinates received from Leap Motion Controller.

At the very beginning, before the data was completed, a total of 5400 data / 12 letters were prepared to be tested. Gestures were shown by a person with the right hand.

Leap.dll: The system file required for Leap Motion to work on the project

Leap.lib: The file containing the Leap Motion Library.

Leap.py: The file containing the python code added by the Leap Motion developers.

LeapMotionApp.py: Python file containing codes that take data from Leap Motion, process and save them, and include the interface and make predictions.

4.2 Training and testing of dataset

Before modelling or predicting, data set must be break down into a training and testing. Thus, it will be possible to test the training set using the test set.

The training set is the set that is created to train the machine learning algorithm, to recognize the data and make its predictions through these data. In short, this can be called a dataset that allows the future data to be biased in certain ways.

The test set is the dataset that was previously set aside to test how accurately the algorithm created using the training set.

In this project, the dataset was divided into two, 30% test and 70% train set. Before modelling, trainings were made with the PANDAS library to see the score of the data and test results were obtained. According to these results, the estimation score of the dataset was determined as 0.994.

5 users used the application while testing the application. They demonstrated each of the 21 signs, one by one, with their right hand. the application correctly predicted 21 marks, except for 8 marks, in the first guess. When users made the wrong sign, they did not detect it correctly.

In the early stages of designing the application, the model was retrained each time a guess was made. In other words, model training took place every 3 seconds. In this case, even if the user demonstrate the sign close to the correct but incorrectly, the wrong sign could be guessed correctly, as the decision trees gave different results each time. Therefore, the model application was trained when it was first opened.

Gestures are divided into fixed and moving. Moving hand gestures are also assumed to be fixed, as PSLI is designed to recognize fixed gestures only. Problems were experienced in the representation of the letters E, H, I, K, O, S, R and U. These problems are described below:

The letter E is perceived differently each time by the Leap Motion Controller due to the overlapping of the fingers. Therefore, the coordinates received and processed from

the device are not in full harmony. How Leap Motion Controller detects hand showed on Figure 20 and Figure 21. The correct representation of gesture has showed on Figure 22.

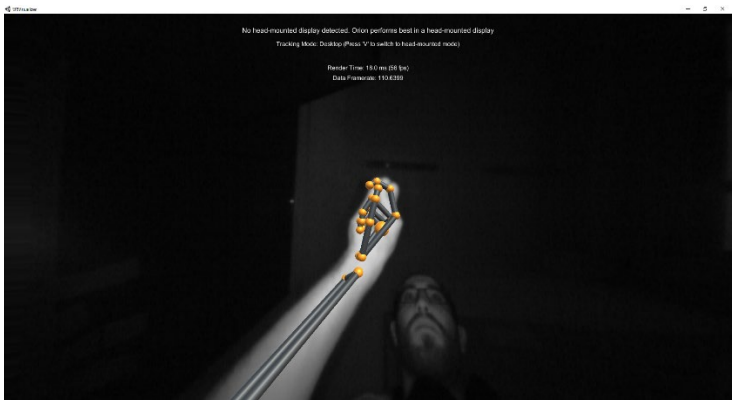

Figure 20 Screenshot of inaccurate visual detection of E letter demonstration.

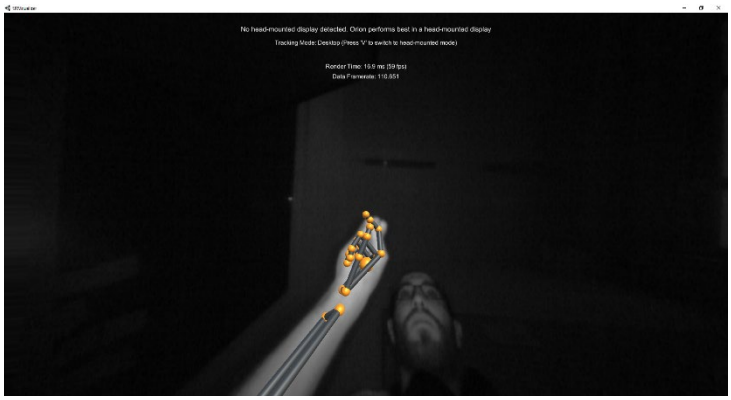

Figure 21 Screenshot of inaccurate visual detection of E letter demonstration.

Figure 22 Letter E demonstration. (Innowacji, 2020)

The problem for the letter H is that the number of data collected for the gesture is not enough. The gesture is shown in Figure 23. It is perceived correctly when it is displayed in a V shape.

Figure 23 Letter H demonstration. (Innowacji, 2020)

It is also perceived correctly when the thumb and pinky finger is raised while the letter I is demonstrated. Again, the problem will be solved with more and more accurate data collection. The sign is shown in the Figure 24.

Figure 24 Letter I demonstration. (Innowacji, 2020)

The problem with the letter K and the letter U is that they look the same. The letters are shown in Figure 25 and Figure 26.

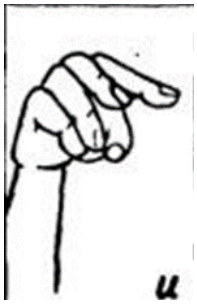

Figure 25 Letter U demonstration. (Innowacji, 2020)

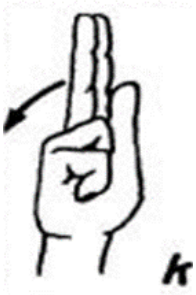

Figure 26 Letter K demonstration. (Innowacji, 2020)

Likewise, there is a very small difference between the letters O and S. Therefore, they are confused with each other while making predictions. Although wrong predictions are made most of the time, there are also predictions that turn out to be correct at first.The letters are shown in figure 27 and figure 28.

Figure 27 Letter O demonstration. (Innowacji, 2020)

Figure 28 Letter S demonstration. (Innowacji, 2020)

Since the fingers overlap while showing the letter R, the Leap Motion Controller confuses the fingers, and therefore, it can be predicted correctly even though the fingers

are shown without overlapping. The correct representation of the letter is shown in Figure 29.

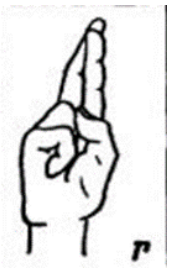

Figure 29 Letter R demonstration. (Innowacji, 2020)

If we accept the letters shown here as unsuccessful guesses, a total of 8 letters are wrong among the 21 signs.

In the test conducted with users, the result was calculated as 62%. It was decided that it was sufficient for this study. Since both results -model.score() result and test with users's result- were sufficient score for the research, the dataset creation method was continued. When the letters reached the planned number, the process was repeated again and the result was taken as "0.982". This was a result showing that the study was successful.

4.3 User Interface of Application

As explained in previous parts, Leap Motion API provides to get every data we need. Using Leap Motion library, 22 points of hand was collected. Code can be divided into two part: first is application part and the second is data collecting part.

When the application is launched, the interface appears on the front side. Signs are displayed to the user in the interface. If the user brings his hand towards the Leap Motion Controller as described and performs the action shown on the interface correctly, the other sign is switched. On the back side, the data received from Leap Motion Controller is used for estimation and this data is saved in the data.csv file.

On the other hand, if the sign is made correctly, it is saved in the collectedData.csv file together with the id and points of the sign made by the user and our program develops with each use.

Whether the sign language is done correctly in the interface is determined as follows; The data received from Leap Motion is sent to be estimated and the result of the estimation is compared with the ID of the letter that is currently shown, if the letter is

correct, the photo changes. PSLI recognizes user's movement and display in Leap Motion Visualizer as shown a user’s sample on Figure 30 and Figure 31.

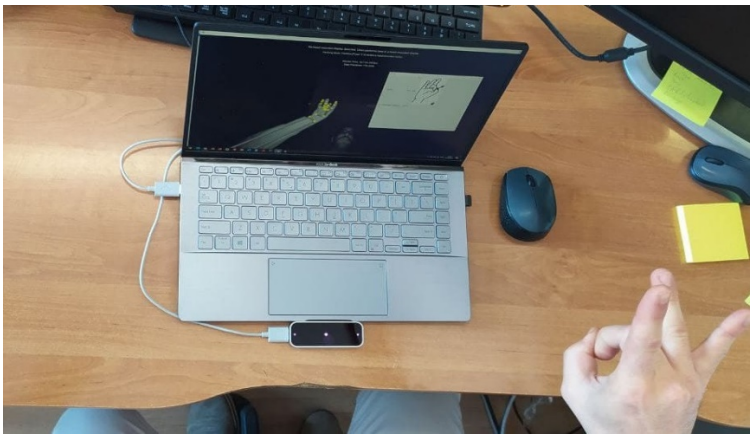

Figure 30 PSLI recognizes user's movement and display in Leap Motion Visualizer

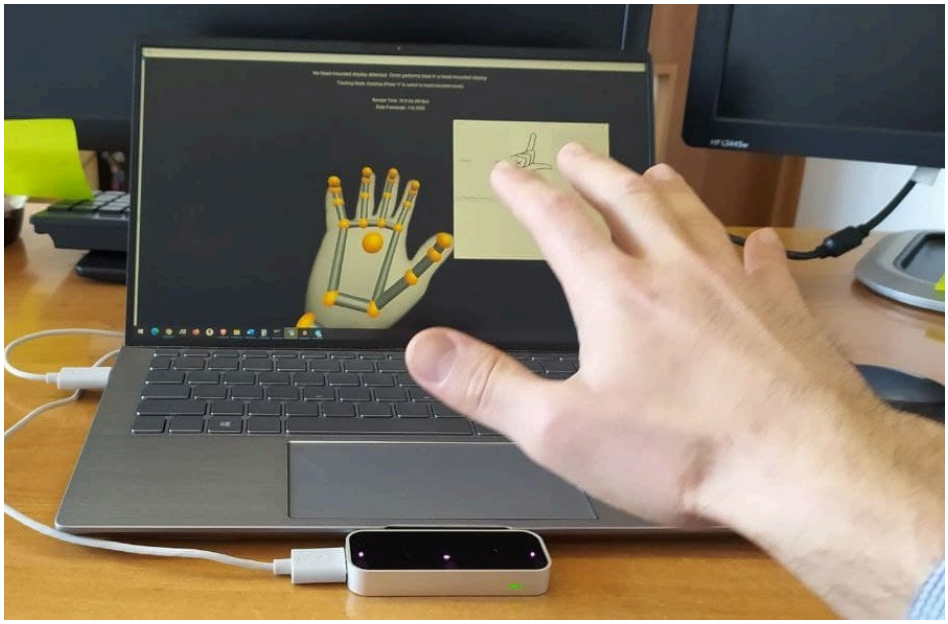

Figure 31 PSLI recognizes user's movement and display in Leap Motion Visualizer

When application started user give permission to save data to use future works. It’s shown on Figure 32.

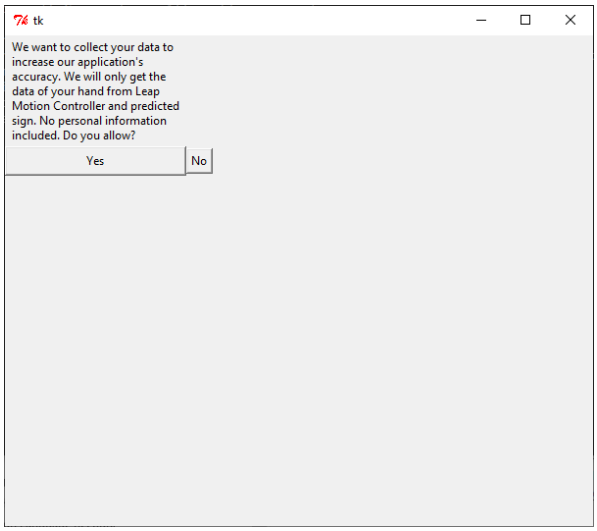

Figure 32 Giving permission page.

Then user will see the signs to make. If user make the sign correctly sign will change. Sample has shown on Figure 33.

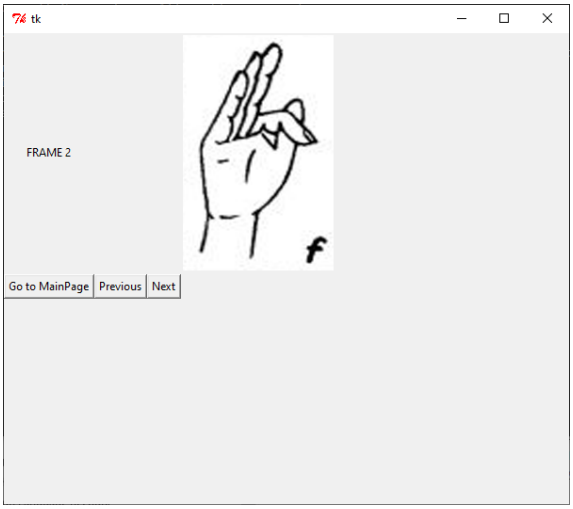

Figure 33 Showing signs page.

When the last gesture is reached, the program ends. By pressing the Previous button, other gestures can be displayed. PSLI continues to be developed and it is aimed to make this interface more interesting by adding a point system in the first stage.

Conclusion

In this project, it is aimed to design a system as a solution to the communication difficulty between the society and the hearing impaired individuals. It's decided to develop a functional system for recognizing fixed hand gestures using the Leap Motion Controller. Based on previous studies and relatively simple approaches to recognizing fixed hand gestures, a machine learning approach was applied.

It is hoped that future quantitative studies will improve the accuracy of the predictions. Future versions of PSLI may include a machine learning approach to distinguish between fixed and motion-based gestures if work is done on the recognition of gesture-based hand gestures. The results of the study showed a strong indication that variation in samples is the dominant factor that increases accuracy, meaning that different individuals should train the PSLI.

A dataset containing 9450 points showing 21 fixed letters was created to be used in the testing and training phases of the system. 5 users used the application while testing. They demonstrated each of the 21 signs, one by one, with their right hand. the application correctly predicted 21 marks, except for 8 marks, in the first guess. When users made the wrong sign, PSLI did not detect it correctly. The accuracy of the dataset was calculated as 97% using Random Forest Classifier score function and the result was calculated based on successful and unsuccessful predictions as 81.25% in the test conducted with users.

It was determined that the most important factor affecting the success was the mistakes made by the Leap Motion Controller while detecting the positions of the hand in space. Because when one finger is above the other, the device cannot fully perceive it and errors occur. This is a recurring theme in most articles that consider Sign Language recognition using the Leap Motion Controller, and the results of the mentioned studies are that the greater scope for improvement lies solely on the accuracy of the Leap API.

These shortcomings can be rectified with additional hardware (Kinect or other sensors) or future upgrades to Leap systems to improve the accuracy of gestural hands in sensor view. The system in its current state could possibly be applied to teach new learners to use, but with certain limitations. However, PSLI cannot be recommended for further learning or for sign language training of children. However, the most obvious lack of functionality is the support for movable and two-handed signs, which should be explored in future versions of PSLI.

It may be considered to introduce a scoring system in order to encourage the use of the PSLI by further developing it in the future. In addition to this function, a spaced repetition system can allow the student to practice more on poorly learned signs.

Additionally, context in Sign Languages is as important as in spoken language. Greater accuracy can be achieved by extending the PSLI to take into account previously recognized marks. If the new input is likely to be part of a sentence, the corresponding probabilities using text classification techniques used in machine learning can be used to increase the prediction accuracy.

As a result, PSLI is capable of recognizing fixed signs with good but limited proficiency. While currently unable to recognize the full scope of signs due to Leap API and implementation shortcomings, a continuous effort to improve learning aids has the potential to significantly improve sign language education.

Abstrakt

Liczba osób, które nie znają języka migowego w naszym społeczeństwie, jest wysoka. Sytuacja ta ogranicza komunikację między osobami z dysfunkcją słuchu a społeczeństwem, a nawet powoduje izolację osób niedosłyszących od społeczeństwa. Niniejsze badanie ma na celu przeszkolenie osób, które chcą się uczyć języka migowego, w celu zapewnienia komunikacji między osobami znającymi język migowy a osobami, które nie znają języka migowego, oraz przybliżenia tych osób niepełnosprawnych do społeczeństwa. Losowa klasyfikacja Forest zastosowana do sklasyfikowania 21 liter polskiego języka migowego przy użyciu funkcji pochodnych z danych Leap Motion Controller. . Wynik eksperymentu pokazuje, że najwyższy średni współczynnik klasyfikacji wynoszący 97% osiągnął odpowiednio Random Forest. Rezultatem jest rozpoznawanie polskiego języka migowego przy użyciu Leap Motion, zdolne do rozpoznawania stałych znaków z dobrą, ale ograniczoną biegłością. Chociaż obecnie nie jesteśmy w stanie rozpoznać pełnego zakresu znaków z powodu Leap API i niedociągnięć we wdrażaniu, ciągłe wysiłki na rzecz ulepszania pomocy dydaktycznych mogą znacznie poprawić edukację w języku migowym.

Słowa kluczowe: uczenie maszynowe, ruch skokowy, las losowy, punkty, przewidywanie, język migowy

Abstract

The number of people who do not know sign language in our society is high. This situation reduces the communication between the hearing-impaired people and the society and even causes the hearing-impaired people to be isolated from the society. With this study, it is aimed to training people who wants to learn Sign Language to provide communication between people who know sign language and people who do not know sign language and to bring these disabled people to the society. Random Forest Classification applied to classify the 21 letters of the Polish Sign Language using the derived features from the Leap Motion Controller data. The experiment result shows that the highest average classification score 0.982 was achieved by Random Forest respectively. The result is a Polish Sign Language Recognition using Leap Motion capable of recognizing fixed signs with good but limited proficiency. While currently unable to recognize the full scope of signs due to Leap API and implementation shortcomings, a continuous effort to improve learning aids has the potential to significantly improve Sign Language education.

Keywords: Machine Learning, leap motion, random forest, points, predict, sign language

Bibliography

1. (n.d.). Retrieved from roboticsbd: https://store.roboticsbd.com/707-large_default/leap-motion-controller-robotics-bangladesh.jpg
2. Chuan, G. (2014). American Sign Language Recognition Using Leap Motion Sensor. *Proceedings of the 13th International Conference on Machine Learning and Applications.*
3. Dobosz, K. (2018, January 1). *Observed area of the Leap Motion controller.* Retrieved from researchgate.net: https://www.researchgate.net/figure/Observed-area-of-the-Leap-Motion-controller-6_fig1_319940978
4. Gholamy, A., Kreinovich, V., & Kosheleva, O. (2018). *Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation.* Departmental Technical Reports (CS).
5. Innowacji, K. S. (2020). *Podstawy języka migowego.* Retrieved from ksiszkola.pl: https://www.ksiszkola.pl/products/podstawy-j%C4%99zyka-migowego/
6. *Introducing LeapUVC: A New API for Education, Robotics and More.* (2018, December 18). Retrieved from blog.leapmotion.com: https://blog.leapmotion.com/leapuvc/
7. Koehrsen, W. (2017, December 27). *Random Forest Simple Explanation.* Retrieved from williamkoehrsen.medium.com: https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d
8. Kulkarni, D. N. (2016). Real time sign language recognition using the leap motion controller. *International Conference on Inventive Computation Technologies (ICICT).* Coimbatore, India.
9. Mohandes, M. (2014). Arabic sign language recognition using the leap motion controller. *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE).* Istanbul.
10. Motion, L. (2015, 08 27). *How Does the Leap Motion Controller Work?* Retrieved from Medium: https://medium.com/@LeapMotion/how-does-the-leap-motion-controller-work-9503124bfa04
11. Motion, L. (n.d.). *Gestures.* Retrieved from developer-archive.leapmotion.com: https://developer-archive.leapmotion.com/documentation/csharp/devguide/Leap_Gestures.html

12. Oszust, M., & Wysocki, M. J. (2013). Polish sign language words recognition with Kinect. *6th International Conference on Human System Interaction (HSI'2013).*
13. Samir, E. T. (2014). Arabic Sign Language Recognition Using Leap Motion Sensor. *The 2014 9th International Conference on Computer Engineering & Systems.*
14. Sitnik, M. (2014). *smartniej*. Retrieved from Leap Reader – translator języka migowego wyświetlający tekst : https://smartniej.pl/hardware/leap-reader-translator-jezyka-migowego-wyswietlajacy-tekst/
15. Smolyakov, V. (2017, August 22). *Ensemble Learning to Improve Machine Learning Results*. Retrieved from statsbot.co: https://blog.statsbot.co/ensemble-learning-d1dcd548e936
16. Telli, E. (2020, 11 20). *Figure 5 - uploaded by Emrah Telli.* Retrieved from researchgate: https://www.researchgate.net/figure/Internal-structure-of-Leap-Motion-controller-11_fig5_348621013
17. tutorialspoint. (n.d.). *Scikit Learn - Introduction*. Retrieved from tutorialspoint: https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm
18. Ultraleap. (2020, January). Retrieved from ultraleap.com: https://cms.ultraleap.com/app/uploads/2020/01/Leap-3-quarter-icon.png
19. Villarreal, M. R. (2007, 01 06). *File:Scheme human hand bones*. Retrieved from wikimedia: https://commons.wikimedia.org/wiki/File:Scheme_human_hand_bones-en.svg