Software Engineering Department
ORT Braude College

Capstone Project Phase A

SIGNiT

Real Time Translation of the Israeli Sign Language Letters into Text

Project No. 23-1-D-3

Supervisor: Dr. Reuven Cohen

Name	ID	E-mail
Kobi Mor Yosef	206076580	kobimor15@gmail.com
Tal Langer	205718588	123tallanger@gmail.com

Contents

Figures

Abstract

Many Israeli people that suffer from hearing loss or hearing impairment use the Hebrew sign language as their mean of communication .

The technologies and tools that exist today for translating sign language to text do not support Hebrew.

In this work, we will develop an application that will translate letters of Israeli sign language into Hebrew text. We'll use the MediaPipe tool, which is an open-source platform that offers customizable machine learning solutions for live and streaming media.

The application will enable users to express complete words and sentences by spelling the words in the Israeli sign language. It will also enable to automatically complete the letters into words, make it easier to use and shorten the spelling process.

1. Introduction:

Hearing loss or hearing impairment is a partial or complete inability to hear. It may occur in one or both ears, and may be present at birth, due to birth complications or genetic predisposition, or being acquired at a later time due to aging, infection, exposure to unreasonable noise and more. "Deaf" people are generally people who have little or no hearing at all. [9]

In Israel, there are about 700,000 deaf and hearing-impaired people, which is about 8% of the population. About 20,000 people in Israel speak sign language at some level, and for about 15,000 of them it is the first and main language.

Sign language consists of hand signs that express whole words and letter signs that allow spelling. Spelling is an integral part of the language and is used to spell names of people, countries, cities, places, phrases. It is also used for words that have no recognized representation in sign language, or in case that the speaking person does not know the representation.

In each country there is a different type of sign language. For example, American and Israeli Sign Language are different and are translated differently.

Today there are some technologies that translate written text into sign language representation, even in Hebrew, but technologies and tools that translate in real time some sign language into written text do not support Hebrew.

The goal of our project is to develop an application that allows users to translate sign language spelling into Hebrew text, in order to help sign language speakers pronounce any word in Hebrew - even words that have no representation in Israeli sign language.

The user will use a webcam and speak by spelling in sign language. The application will process the images from the webcam in real-time, determine which letter the user spells, and display it on the screen. In addition, the system will offer the user complete words based on these letters. The user will be able to choose the word by signing the “approve” sign or to keep spelling the word. In this way, the user will be able to generate full words and full sentences.

The application will enable deaf people to communicate in a natural way, without feeling frustrated and uncomfortable because they are not understood.

2. Background and Related Work:

2.1 Sign language translation:

Spelling in sign language is based on palm gestures, including the fingers. In [7], we can see the definition of gesture, the type of gesture people usually use, what gesture includes, etc. Allowing an application to recognize signs and gestures requires using various methodologies such as statistical and probabilistic modeling, pattern recognition, image processing, computer vision, etc. There are a variety of gesture recognition tools and approaches that use those models (see [7]). These tools are mainly based on two methods: sensor-based and vision-based.

There are many applications that translate written text into sign language representation, by displaying each word's sign language representation (even in Hebrew). In contrast, translating sign language into written text is more challenging so there are less tools that do so, and they also do not support Hebrew text.

Existing solutions for translating sign language into text:

2.1.1 Vision-based recognition tools to recognize sign language:

- **SignAll -** A system that uses four cameras and can translate both the speaker and the signer into chat text. The hearing user speaks, and his speech is shown in the chat, while the deaf user signs toward the cameras, and his signs are processed and displayed as text in the chat. The system can detect and follow body movement, body position, facial expressions, and hand or finger shapes. more info in [2].
- **Sign.mt -** A web-based application that provides translation from written text to sign language and from sign language (using a camera) to written text in several languages. The application supports both desktop and mobile platforms. It was developed according to the model presented in [6], that is generally based on identifying the user's pose landmarks, following his movements, and recognizing the word he expressed.

2.1.2 Sensor-based recognition tools to recognize sign language:

Most of the sensor-based recognition tools consist of wearable gloves, as they have been found to be the most effective in detecting the movements of the joints and fingers of the hand.

Data Gloves - According to the approach shown in [3], a deaf person wears a glove that detects his hand gesture. It sends the detected input to a microcontroller that processes it, and by using an existing database, the microcontroller determines which sign was executed and displays it on the screen.

2.2 Word suggestions:

In order to improve typing efficiency, many systems and tools use word suggestion algorithms. As claimed in [8], as these tools' accuracy is quite good, they can improve typing speed, mainly on tablets and phones.

Existing solutions for word suggestions:

- **Gmail Smart Compose -** This tool generates real-time interactive suggestions in Gmail that help users write emails by reducing retyping. The suggested shown words are based on the letters or words written so far, and if the user wants to add it to his sentence, he can click on the TAB key, and the sentence will be added to his text. More information can be found in [1].

- **SwiftKey -** A mobile keyboard that contains a word suggestion bar at the top of the keyboard, allowing the user to choose between 3 suggested words that he might use at the current state of the typing (considering the most used words or sentences). The keyboard learns new words, email addresses, names, etc. It also considers cases where the user accidentally types a wrong letter in a complete word, allowing him to fix the mistake by clicking on the main suggestion at the word suggestion bar.

2.3 Object detection and image processing:

Processing video is usually done by breaking it into images and processing each image separately.

2.3.1 MediaPipe:

MediaPipe is an open-source framework developed by Google that allows developers to build and deploy multi models of machine learning pipelines on a wide range of devices. These pipelines can be used for a variety of tasks, such as object detection, pose estimation, and hand tracking. [4]

One of its key features is its modular design, which allows developers to easily use pre-built modules. These modules can be used to perform tasks such as video input and output, image processing, machine learning inference, and data visualization.

MediaPipe also offers a few other benefits. It has a low latency and high frame rate, making it suitable for real-time applications such as hand tracking.

figure 1: one of MediaPipe's modules for body pose tracking

2.3.2 TensorFlow:

TensorFlow is an open-source machine learning framework developed by Google that is widely used for training and deploying machine learning models. TensorFlow was designed to be flexible and efficient, and it allows developers to build and deploy machine learning models for a wide range of applications, including image and speech recognition, natural language processing, and predictive modeling.

TensorFlow supports transfer learning, which is a machine learning technique that involves using a pre-trained model as a starting point to solve a new task. In TensorFlow, transfer learning can be implemented by using a pre-trained model as a starting point and fine-tuning the model on a new dataset for the new task. This can be done by using the pre-trained model's weights as initial values for the new model, and then training the new model using the new dataset.

3. Expected Achievements:

The goals we expect to achieve in the project are:

a. Create a real-time translation tool that translates sign language spelling into Hebrew text.
b. Allow the user to compose complete sentences by hand signs only and without physical contact.
c. Facilitate the use of the application by using a word suggestion tool that supports Hebrew.

3.1 Create a real-time tool that translates sign language spelling into Hebrew text:

In order to achieve this goal, we have to achieve many sub-goals:

a. Process the video: find the palm in a specific frame and identify the position of the palm. This step has to be done very fast because the palm is almost always in motion, and there are a lot of frames that need to be processed.
b. Create a dataset of Hebrew sign language letters to let the system learn and recognize the various Hebrew signs.
c. Create and train a model: using the dataset we created, we will create a model that learns the dataset of each letter.
d. Decide which letter or sign and put it as text: the system needs to figure out what letter or sign the user spelled and display it as text. It also needs to ignore any 'unwanted gestures'.

We would like to reach about 85% success in identifying the words and signs at the first time the user signs.

3.2 Allow the user to compose complete sentences by hand signs only:

We will also add non letter signs that are not necessarily commonly used in sign language, like 'space', 'dot', 'comma', question mark and exclamation mark.

In cases of mistakes, the user will have a delete sign to enable him to delete the wrong letter.

The system will also infer when the user finishes to spell a word.

3.3 Using a word suggestion tool that supports Hebrew:

We will look for a word suggestion tool that supports Hebrew, and can be integrated in our system.

For that purpose we'll suggest a special sign to let the user approve the suggested word.

The suggested words should be updated after each spelled letter.

3.4 Success criteria of the project:

The main criteria that we define for a successful project are:

- The system is able to recognize the Hebrew sign language letters in real-time.
- The letters displayed on the screen are the letters the user has chosen to appear.
- The user can create full sentences without touching any device.
- The user can delete letters, add ‘space’, ‘dot’, ‘comma’, and other signs that help to write sentences.
- The word suggestion recognizes Hebrew letters and suggests suitable words.

4. Research / Engineering Process:

4.1 Process:

4.1.1 Stages in software and research development process:

The process of developing software and research involves several stages, including:

a. Performing research about Sign Language in general, and specifically in Hebrew.
b. Conducting market research on existing products and assessing their benefits and drawbacks.
c. Interviewing deaf people about the use of spelling and well-known signs.
d. Determining the requirements for the system.
e. Search and read articles about relevant technologies and solutions.
f. Choosing the technologies and tools to be used during implementation.
g. Designing the system architecture.
h. Selecting a model development approach.
i. Design the testing and the evaluation of the system.

4.1.2 The preliminary work for the development process:

When we read about sign language, we discovered that it is not an international language and it differs from region to region. Israeli sign language was greatly influenced by the German sign language, mainly because of the influence of the immigrant community from Germany and its surrounding countries. In Israel, the Hebrew sign language is almost identical to the Arabic sign language because of the common population in the region. As written in the introduction, a large number of deaf and hearing-impaired people live in Israel, and for some of them, this is the primary language they use.

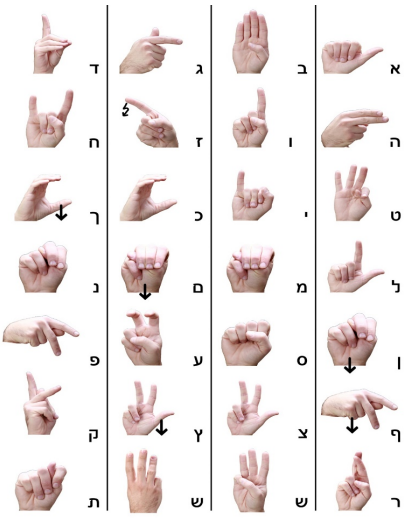

figure 2: the representation of Hebrew letters in Sign Language

With the understanding of the Sign Language subject, we conducted research on the sign language translation products available today. As we have not found any products that translate sign language spelling into Hebrew text, so we searched for tools that do it for American Sign Language (Known as ASL).

In the next step, we chose to interview Israeli deaf people to understand their needs and to find how they would like such a tool to function. This information helped us to write the requirements for our system.

Then, we read articles describing different ways to translate sign language into text, and learned about current technologies that might help with the task.

4.1.3 Choosing the technologies:

After reviewing several technologies, we decided that the way that would best suit our project is to use vision-based methods, because except for a computer and a webcam, it does not require the use of external electronic products that need to be developed or purchased and the results of using these methods are excellent.

MediaPipe:

The Hand Tracking module in MediaPipe can be used to detect and track hands in real-time video streams. The module uses a combination of techniques such as color filtering, contour detection, and skin detection to pre-process the video frames and improve the accuracy of the hand detection model. The module also uses a machine learning model, such as CNN, to classify the detected hands and track them as they move.

The Hand Tracking module is designed to be easy to use and it can be integrated into a MediaPipe pipeline using a simple API. It can be configured to detect and track hands in different scenarios, such as single-hand tracking or multi-hand tracking.

In the MediaPipe Hands module, a machine learning pipeline is used to detect and track the pose of a hand in an image or video. This pipeline consists of multiple models working together: a palm detection model and a hand landmark model. The palm detection model processes the full image and returns a bounding box that encloses the hand. The hand landmark model then operates on the cropped image region defined by the bounding box and returns key points coordinates in 3D that accurately represent the pose of the hand.

In order to be able to identify the letters accurately, we need a high level accuracy in identifying the joints of the palm. MediaPipe divides the palm into 21 parts, and provides a high level identification of the position of each part.

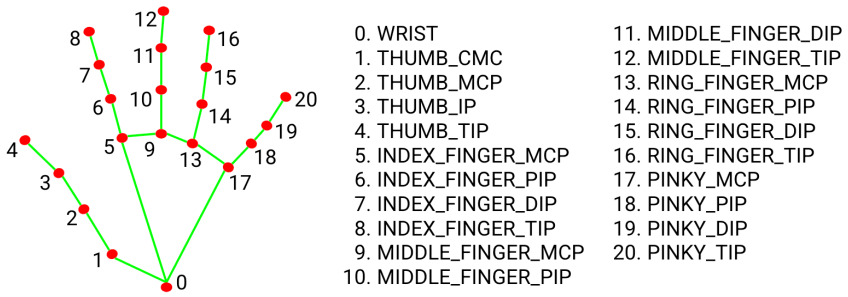

figure 3: 21 hand landmarks used in MediaPipe

Python:

We found that Python is a convenient language to implement a project that uses MediaPipe and includes image processing.

Google Teachable Machine:

We will use the Google Teachable Machine, which enables us to create and train our machine learning models without using our hardware. The model we will use is based on MobileNet architecture, which was developed by Google and has been widely used for tasks such as object detection and classification.

4.1.4 Challenges:

We will face several challenges that we will have to resolved:

- **Dataset -** Our model should handle 22 Hebrew letters, 10 digits, and 6 additional signs. We will need to create this dataset because we did not find available datasets for the Hebrew letters.

- **Letters that require hand movement -** There are several letters whose representation in sign language demand movement of the hand (like final letters, or letters with punctuation). We will try to implement them without the movement, and check the correction of the results. We will ignore final letters while the user signs, and when he finishes signing a word by signing ‘space’, ‘dot’, ‘comma’ and other signs that indicate the end of a word, we will check if the last letter was a letter with final letter representation and switch it automatically.

- **Hardware -** In order to train the model, we will need a powerful CPU (or also GPU). This can take a long time, and in case of errors, we might need to start again.

  We are going to use Google Teachable Machine, which allows us to create and train a model without using our hardware.

- **Distinguish between unintentional gestures** - We don’t want the user to approve each letter that he represents. After consulting deaf people, we decided that the

user will show his gesture to the camera, and will wait with his hand steady for up to 1 second.

- **Identity word endings** - The user will close his hand (like a fist) or take his hand off the screen for 1.5 seconds.
- **Creating a full sentence** - The text will appear in a text box, and we will provide the user signs that are used in sentences, like 'dot', 'comma', and more. We will also provide the user a special sign to delete a character from the text box. This will ensure that the user will not need to touch any device (like a keyboard or mouse) to write his text.
- **Working with word suggestions** - The suggested word will appear above the text box, and we will create a special sign that allows the user to 'accept' the word.

  We will use an existing tool that we will integrate into our application. We found LightKey [5], which is a free text prediction software that uses artificial intelligence to predict the next word or phrase that a user is likely to type. It is designed to help users type faster and more accurately by providing suggestions for the next word or phrase as they are typing. It is also able to learn from the user's writing style and the context of the document to provide more accurate predictions.

  We thought about an option to enable the user to use his second hand to approve the word, in order to make the process faster and easier.

4.1.5 The development method:

According to the way the final project is structured, we found the Waterfall method the most suitable for us because after performing phase 'A' of the project we have the system's requirements, design and testing plan, and at the next phase, we will work on the implementation and verification. Working on each part separately helps us to be sure that we have finished each part, and can continue to the next one.

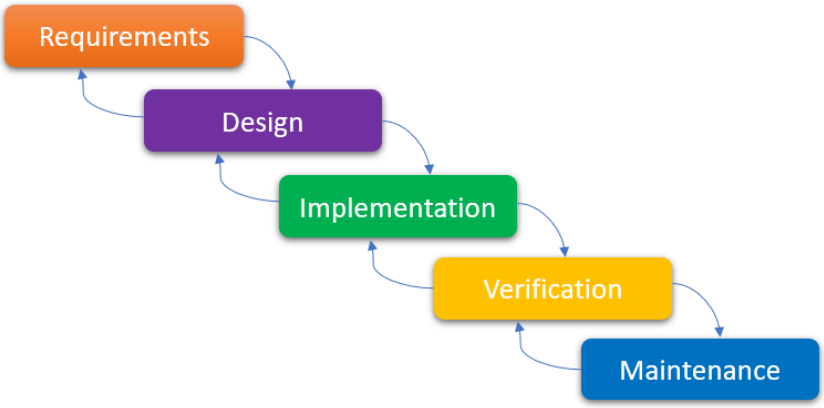

figure 4: The Waterfall model

4.2 Product:

4.2.1 Software Architecture Diagram

Part 1 - The machine learning process:

This diagram presents the machine learning process, including the creation of the dataset:

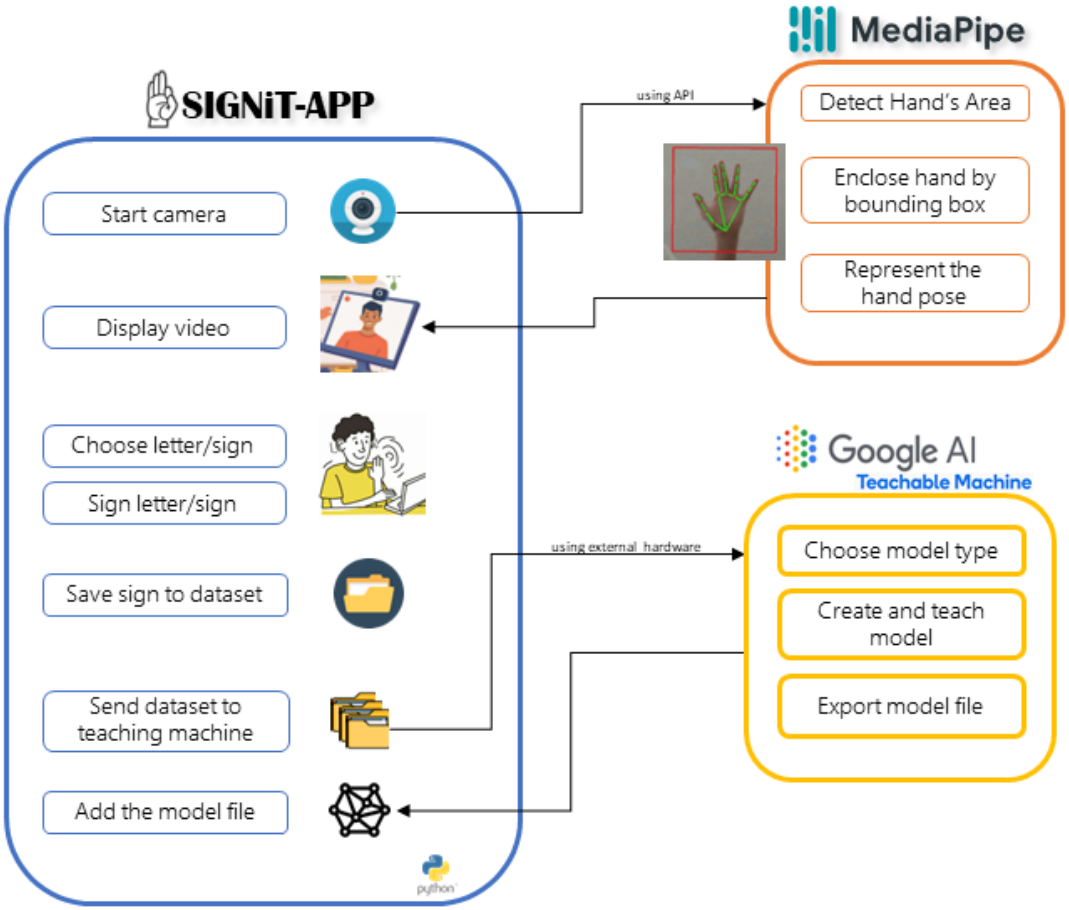

figure 5: Software Architecture Diagram - the machine learning process

In order to create the dataset, the user needs to choose a camera or use the default, specify the letter to be learned, and update the image files path. Then, he can start the machine learning process and see on the screen a video of his hand including the joints and the 21 landmarks given by MediaPipe. In order to add an image to the dataset, there is a need to click the ‘s’ (save) key when it is ready.

After creating the dataset, the user needs to open the Google Teachable Machine on his web browser, to choose the correct model with suitable properties, given the data set as input to the model, train the model and export the model file.

Part 2 - The user application:

This diagram presents a general look at how the application will work:

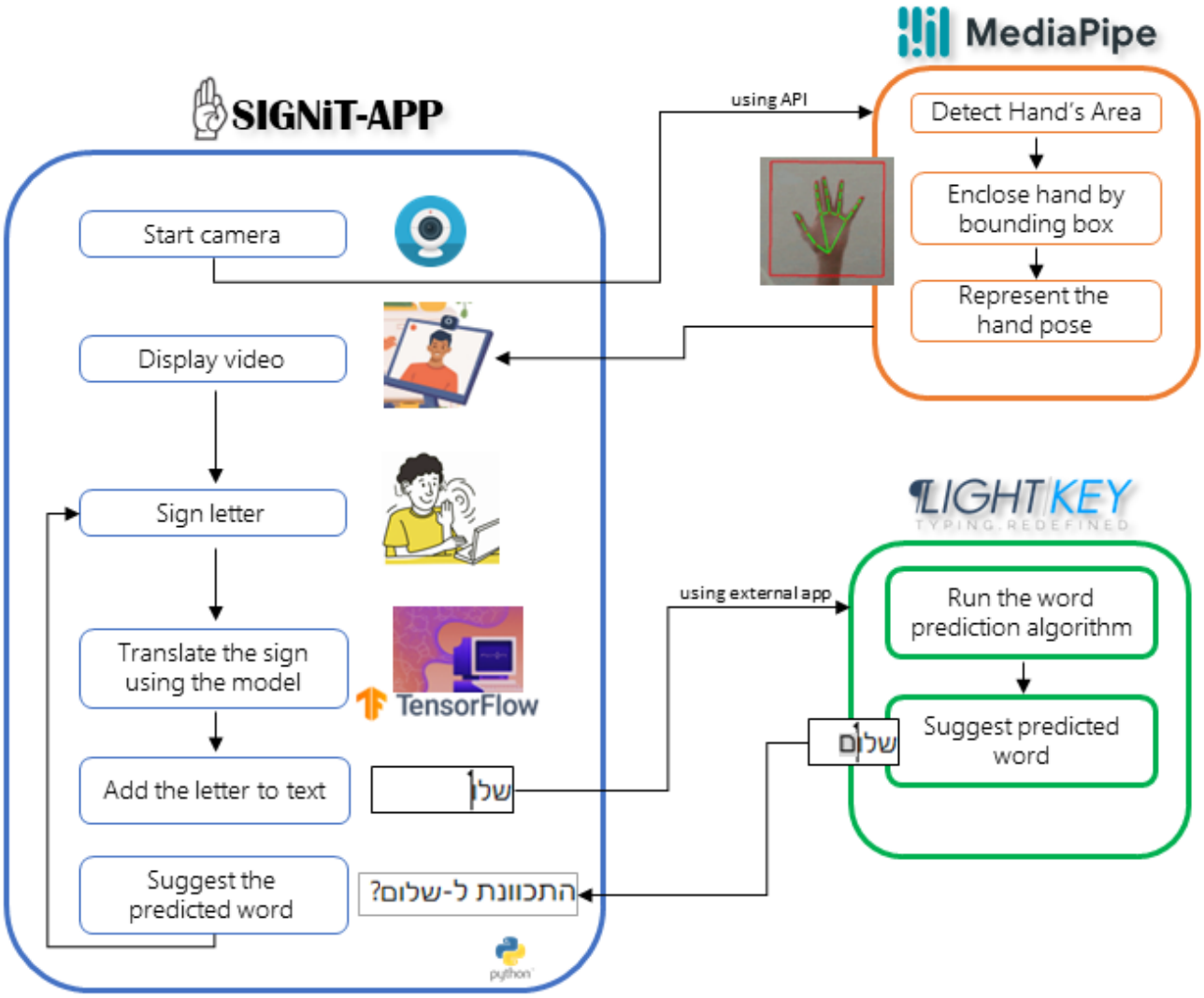

figure 6: Software Architecture Diagram - the user application

In order to use the app, the user needs to choose a camera or use the default. The camera output is processed by MediaPipe that puts a bounding box on the user's palm and represents the hand pose. The system displays the video to the user, and he can now sign letters. The system identifies the sign, translates it using the model and puts it on the screen. Then, LightKey reads the letter and runs the word prediction algorithm, finds a word to suggest, and shows it to the user. The user can choose to use this word or keep spelling.

4.2.2 Activity Diagram

This diagram shows the main activities that the user performs in the system, and how the system shall react to these activities. We can refer to this diagram as the main flow chart of the application.

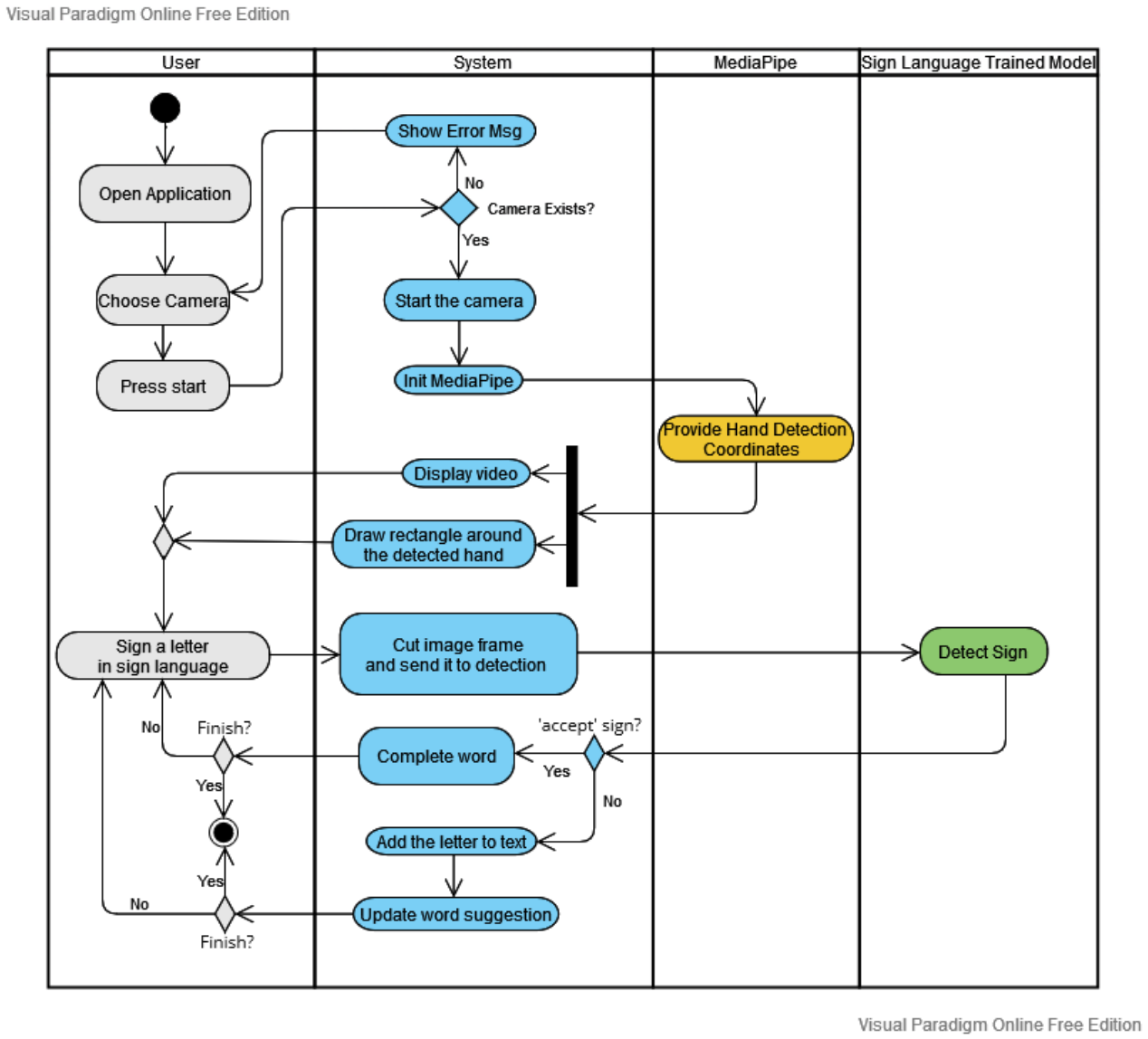

figure 7: Activity Diagram

4.2.3 User Interface

We have added images that describe a concise design of the appearance and elements in the user interface:

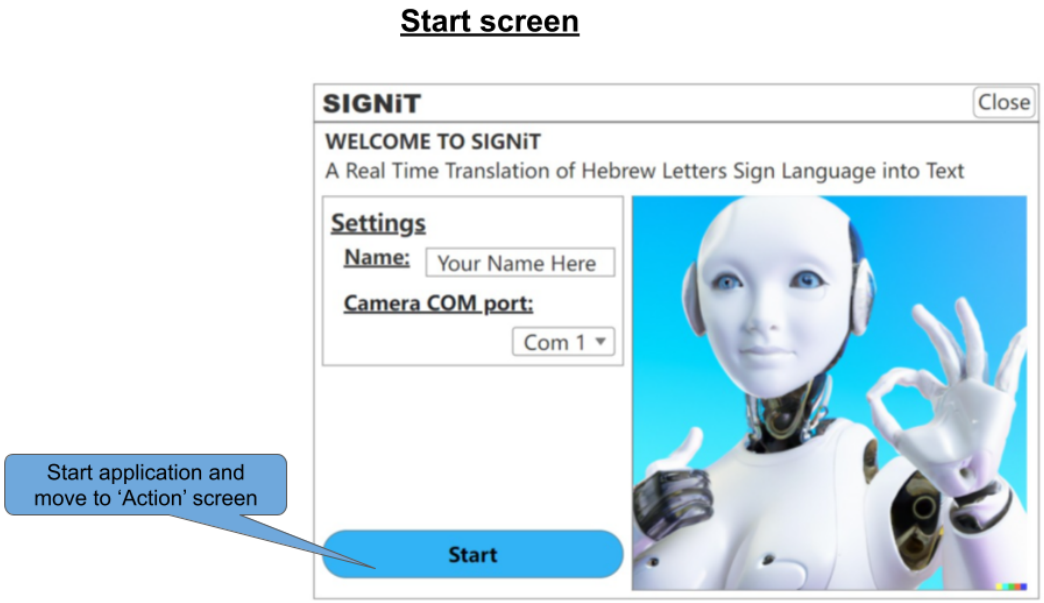

figure 8: Main page in the user interface

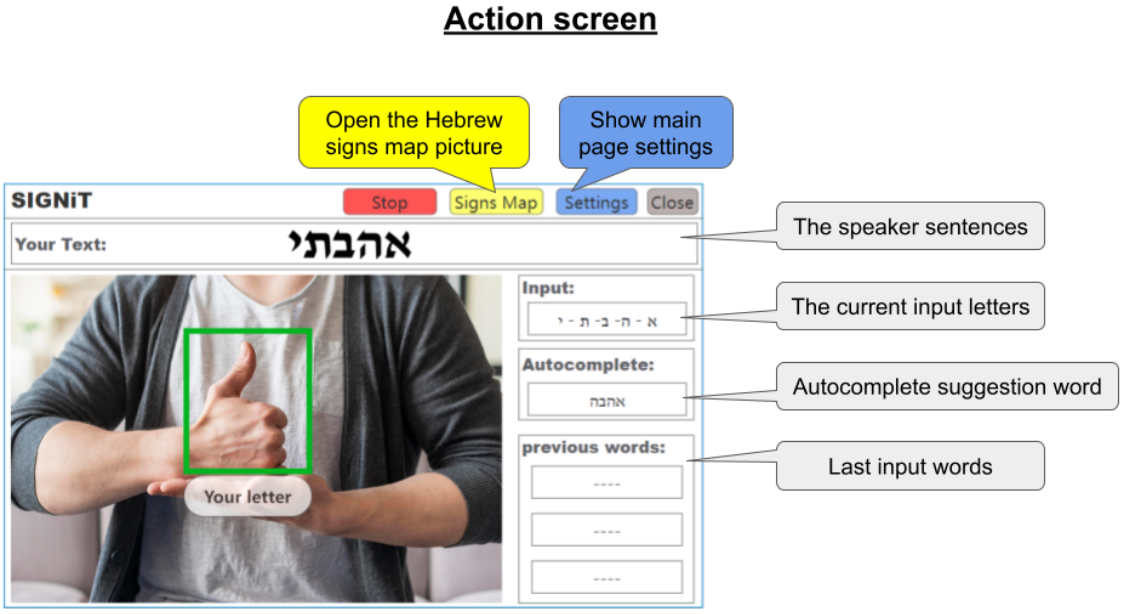

figure 9: Translate page in the user interface

'Your Text' textbox, contains all the sentences that the user entered. The user enters letter after letter, and creates whole sentences or paragraphs.

4.2.4 Class Diagram

This diagram presents an overview of the main classes and the main methods of the system, and the relationships between them:

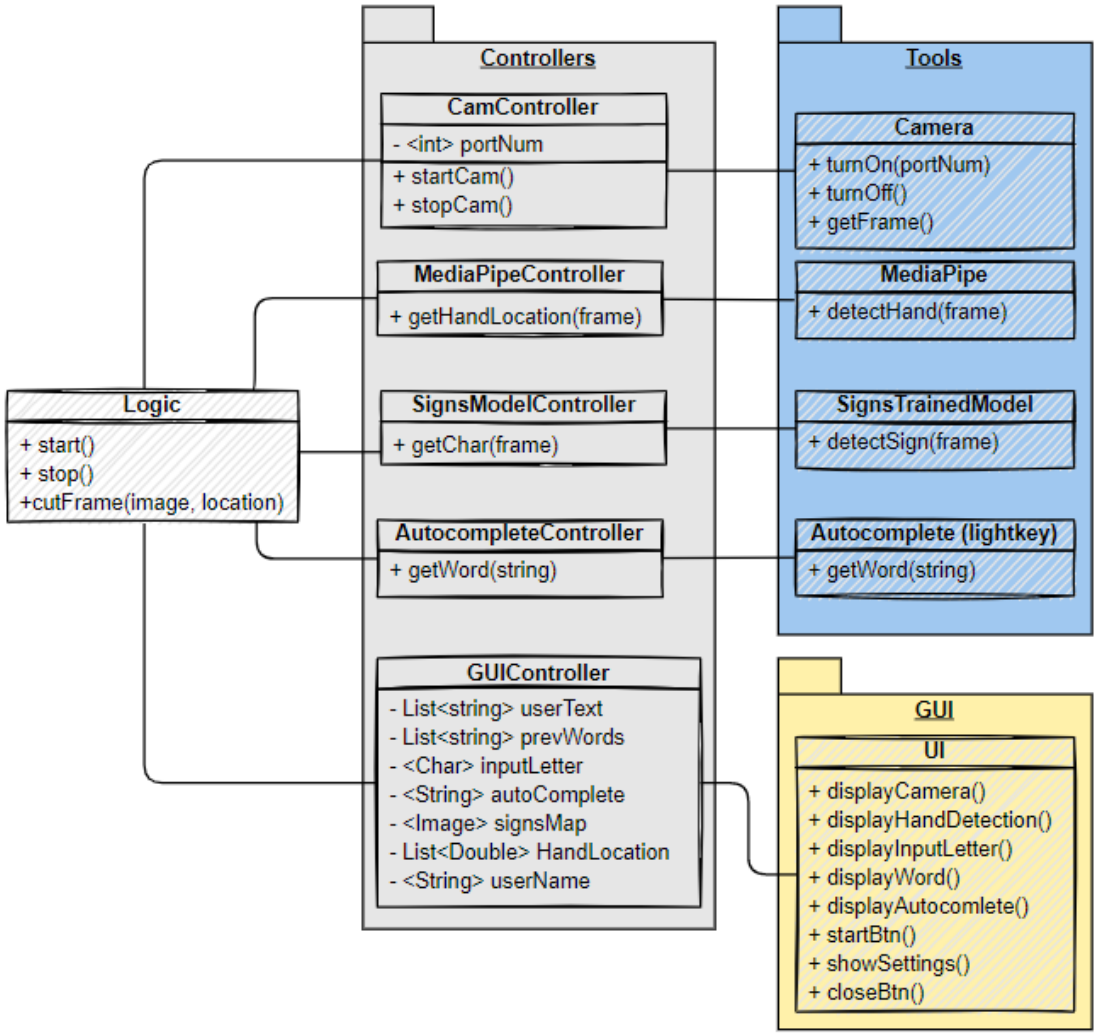

figure 10: Class Diagram

Diagram explanations:

Logic:

The main class of the application.

- start() function: will init all controllers and start the action.
- stop() function: will dispose all controllers and stop the action.
- cutFrame(image, location) function: will return a frame (cropped frame)

CamController:

The camera controller.

- startCam() function: will init the camera using the portNum (from the settings)
    - provide real time image to the logic class
    - throw exception if com port does not exist

MediaPipeController:

- getHandLocation(frame) function: will get a frame image from the logic function
    - call mediaPipe hand detection model
    - return the hand location to the logic class

SignsModelController:

- getChar(frame) function:
    - call signs trained model with the frame
    - return predicted letter

AutocompleteController:

- getWord(string) function:
    - call the ‘Light key’ service with the input string
    - return predicted word

GUIController:

Hold all GUI data provided by ‘logic’ class

5. Evaluation / Verification Plan:

5.1 System requirements:

ID	Requirement
1	The application will allow the user to set the speaker's name
2	The application will allow the user to set the camera Com port
3	When pressing 'Start':
3.1	The application will show the live camera image
3.2	The application will start detecting the user hands
3.2.1	The application will draw a rectangle around each detected hand
4	The application will detect the most probable sign
4.1	The application will show the detected letter under the detected hand
5	The application will add the detected letter into the word
6	The application will allow the user to undo (Remove the last letter from the word)
7	The application will provide autocomplete option (based on the input letters)
7.1	The application will allow the user to choose to autocomplete the word (by special gesture)
8	The application will move to the next word after (~1.5 seconds from the last input)
9	The application will detect end of sentence (by special gesture)
10	The application will display the input word
11	The application will display the full sentence on the screen
12	When pressing 'Signs Map': The application will display the Hebrew signs (as picture)
13	When pressing 'Stop': The application will stop all activities and go the startup page
14	When pressing 'Close': The application will be closed

5.2 Testing Plan:

1	Hand tracking model
1.1	Test detecting hands in real time:
1.1.1	Test different angles of camera
1.1.2	Test on diverse backgrounds (White, Black, Over the body, Over the face, colorful)
1.1.3	Test different hands colors and size
1.1.4	Test crop detected hand (the frame of the detected hand)
2	Signs detection
2.1	detection accuracy on set of images [Premade set from the internet]
2.2	detection accuracy on images cropped by the 'Hand tracking model'
2.3	detection of 'Autocomplete' sign
2.4	detection of 'End of word' sign
3	Autocomplete (Light key)
3.1	Test accuracy of predictions in Hebrew
3.2	Test integration with our system (using Light key as a service)
4	Application
4.1	Sanity test (test each requirement)
4.2	Integration test (test the integration of all the models together)
4.3	UX test (using Hebrew signs language speakers)

5.3 Validation process:

After the system development, we want to search for deaf or hearing-impaired people to check the system and tell us about their experience.

To make it short and useful, We would like to focus on this main criteria:

- Sign Language Translation: What percentage of recognition do you think the system achieves? Did you find a sign that the system was hard to recognize?

- UI: Did we miss something in the system’s screens? Was it clear enough? Something you think we should add/remove?
- Word Suggestion: What do you think about it? Did you use it? How accurate was it? Was the ‘accept’ sign good enough? Adding a second hand to accept would be helpful?
- In general: Do you find this kind of system useful? Do you have any recommendations for us? Something you want to tell the developers?

6. Summary:

In this work we found a possible solution to help deaf and hearing-impaired people communicate in the language which they communicate with on a daily basis, without being limited and fitting themselves to other people.

After researching for existing solutions, we have focused on spelling in the Israeli Sign Language because we have not found a solution that translates sign language to Hebrew that also takes the spelling into account.

We decided to create an app that translates letters in sign language into Hebrew text in real-time, using image processing and machine learning.

Considering that writing a text by spelling can be quite challenging, we decided to add a word suggestion tool that supports Hebrew, hoping that predicting the word that the user is trying to write will make it easier for him. We thought about providing the user the option to use his second hand to 'accept' the suggested word.

The work was done with the great hope that it could contribute to the integration of the deaf and hearing-impaired population in society in a better way, and with the hope that existing software for chats, video calls and conference calls will use or develop such a tool - which may lead to an increase in the number of their users and at the same time make the world a better place.

7. References:

[1] Chen, M. X., Lee, B. N., Bansal, G., Cao, Y., Zhang, S., Lu, J., ... & Wu, Y. (2019, July). Gmail smart compose: Real-time assisted writing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2287-2295). https://dl.acm.org/doi/abs/10.1145/3292500.3330723

[2] Hagen Marc. SignAll - They Translate Sign Language. Automatically. In *Closing The Gap.* https://www.closingthegap.com/signall-they-translate-sign-language-automatically/

[3] Lokhande, P., Prajapati, R., & Pansare, S. (2015). Data gloves for sign language recognition system. *International Journal of Computer Applications*, *975*, 8887. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=97fc603a799842630748089e090b1e9b97e5b489

[4] Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., ... & Grundmann, M. (2019). Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*. https://arxiv.org/abs/1906.08172

[5] Lightkey.io. *Lightkey - text prediction software for Windows*. https://www.lightkey.io/

[6] Moryossef, A., Tsochantaridis, I., Aharoni, R., Ebling, S., & Narayanan, S. (2020, August). Real-time sign language detection using human pose estimation. In *European Conference on Computer Vision* (pp. 237-248). Springer, Cham. https://link.springer.com/chapter/10.1007/978-3-030-66096-3_17

[7] Nanivadekar, P. A., & Kulkarni, V. (2013). Gesture recognition: a revolutionary tool. *International Journal of Technological Advancement and Research*, *3*(3). https://www.academia.edu/6699796/Gesture_Recognition_A_RevolutionaryTool

[8] Roy, Q., Berlioux, S., Casiez, G., & Vogel, D. (2021, May). Typing Efficiency and Suggestion Accuracy Influence the Benefits and Adoption of Word Suggestions. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (pp. 1-13). https://dl.acm.org/doi/abs/10.1145/3411764.3445725

[9] Wikipedia. *Hearing loss*. https://en.wikipedia.org/wiki/Hearing_loss

Link to our GIT repository