Baruch College | CUNY

ECE
ÉCOLE D'INGÉNIEURS
ENGINEERING SCHOOL

ASLTD

PROJECT PROPOSAL

In this document, we will describe in detail the project target, what we want to achieve, the project setup, the components we will use, the subsystem we will build, and the subtasks each team member will work.

"*Progress is impossible without change, and those who cannot change their minds cannot change anything.*" - George Bernard Shaw

TABLE OF CONTENTS

INTRODUCTION – THE VISION

The American Sign Language (ASL), a natural language which arose from a need to communicate from non-hearing individuals is a sophisticated language which uses signs in vice of spoken words. While it was birthed from the need to bridge the communication barrier between people, one not familiar with this language often feels at a loss when confronted with it, and more so the poor soul attempting to communicate using ASL. By appearing, ASL and all sign languages collectively, have created a neo-barrier which is exactly what our project aims to solve.

The American Sign Language Direct Translator (ASLTD) is a state-of-the-art solution that translates ASL into text. It has been designed to help people who are deaf or mute to be understood more easily in society, whatever the situation and the place. This app will not only translate simple motionless signs that can be made with one hand into text. It will also be able to translate more sophisticated words and phrases, requiring hand tracking and understanding of hand movements.

Thanks to our knowledge of Python code, our motivation, our teamwork and our hardworking nature, we're going to do everything we can to make this project as complete as possible, as we'll describe in detail below.

DESCRIPTION – THE OBJECTIVES

The ASLDT project is decomposed in four main functions which enables the team to solely focus on the task at hand. The four objectives are as follow:

- Read a video stream from the camera
- Detect whether an individual is in frame
- Identify any ASL signs or movements
- Display the translated words on-screen

See Preliminary System Design for a compact view of ASLDT's inner workings

Identifying the important objectives and creating sub-tasks (as described in TASK DIVISION) enables a smooth workflow and modular implementation of functionality and, most importantly, avoids side-tracking and half-implements of certain tasks.

OBTAINING A VIDEO STREAM

A video stream is a sequence of images, each composed of a two-dimensional matrix of pixels (exaggerated example above). Pixels each have, in standard dynamic range, 255 possible values of green, red, and blue. Obtaining this sequence of images is the first step to using ASLDT. Thanks to cameras present on practically all modern smart devices, this crucial first objective is accomplished using the OpenCV python library (see PROJECT SETUP for a comprehensive look at all the technologies used).

PICTURE TAKEN ON PRINCETON.EDU

OBJECT DETECTION

Once the video feed is obtained, the second task to tackle is detecting whether an individual is currently within frame of the camera. Achieving this saves processing power, as the AI will not try to translate signs if no person – or better yet, hand – is in front of it. It also increases user experience, since no output is given without the prerequisite of there being a hand, a user finds it easier to interface with ASLDT, especially concerning less technologically inclined users, who can easily be overloaded with information on screen.

The technicality of detecting a hand is explained in Training of ASLDT's AI along with Identifying signs .

IDENTIFYING SIGNS

LATER IN ASL (LIFEPRINT.COM)

Sign Identification is the core point and objective of ASLDT. Once signs are translated, all following operations are only to beautify the solution and make it more user-friendly. To identify signs in real life, we have the flexibility of the human mind. Although prone to errors, the human mind has the ability to adapt and learn on the fly. For example, ASL is usually spoken using the right hand as the dominant hand. Some speakers use the left hand. A light difference in our eyes, but a major focus point when developing a computer powered solution. A few other issues one could overlook are:

- Movements in signs
- Signs resembling closely to each other
- Connotation through the speed or rigidity of the sign
- Facial expressions and head movements

These light insinuations and differences in meaning do have a way to be dealt with through machines in a modern way: machine learning and artificial intelligence. At the core, AI – aside being a group of words being thrown out many times over by many big tech companies – is all about replicating a human mind, with the advantages of a computer's speed. This is done by simulating a *neural network*. The human brain is composed of dozens of billions of neurons, analog to a computer's transistor. The advantage of these neurons is their ability to grow stronger or weaker, thus allowing us to make quicker connections to things we already know. Without delving into neuroscience, this effect is replicable in a computer through the use of a neural network.

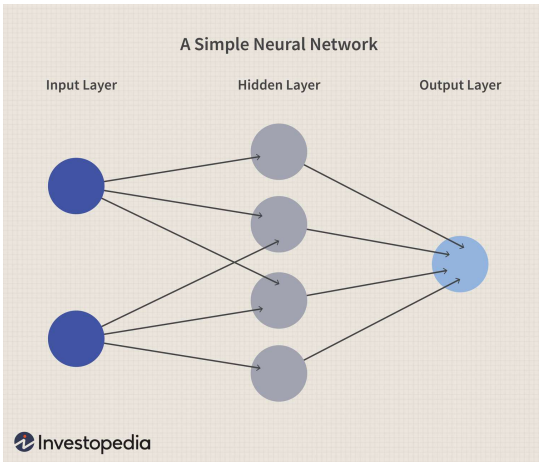
DISPLAYING THE OUTPUT

Once the data has been obtained, analyzed, and translated, it needs to be provided to the user in a simple and efficient format. Using *Pygame* as a graphics library to format and improve simplicity is a great solution to handle both controls and graphics at the same time. It allows for ASLDT to be bundled in a unique, space efficient operating system independent package.

TRAINING OF ASLDT'S AI

As mentioned in IDENTIFYING SIGNS, ASLDT's sign recognition is powered with AI. To start, the AI must be trained, so it can be "taught" how to interpret the data and in our case, the data is a video stream. Going back to the principle of a neural network (see image on next page), a neural network is composed of three layers.

- First is the input layer, where the AI is provided data. One can call this layer the "senses" in a humanized analogy.
- Second is the hidden layer, the brain of the neural network. This is where millions of *nodes* (the four grey circles) represent a decision. This layer is composed of many rows of nodes in a multidimensional array – the size of which is dependent on a computer's capabilities.
- Last is the output layer, representing the conclusion of the AI's "thoughts". In simple speak, this is where we retrieve the sign ASLDT has recognized.



SIMPLE REPRESENTATION OF A NEURAL NETWORK (INVESTOPEDIA.COM)

The AI starts off as a "baby", not knowing how to act or react when confronted with new sensory information. The lines between nodes, called *synapses* represent a path to a decision. When beginning, these are all equal, meaning **the decision-making process is completely random**, or pseudo-random in our case. For the AI to know how to evolve and build its brain, it needs a sort of reward when it finds the correct answer – an artificial way of making the network evolve towards a certain outcome (in our case: correctly identifying ASL). This is called *reinforcement learning*.

To efficiently and quickly train and grow the AI, we rely on the use of *datasets*, large collections of data which not only include the input layer's data, but also the correct answer. Having the answer along with the input makes it much easier to automate the training, as once the AI produces an answer when confronted with one of the images or videos, it can directly compare it with the expected answer, and grow accordingly.

The way the AI is trained is done through two methods and, once again, these can be observed in human evolution. The first is through updating the *weights* of the synapses. As mentioned above, when the AI first is born, its synapses are equal in weight. This means that each decision is just as likely to be taken as another. As the AI progresses in its training, it learns that certain decisions more often give the right output than others, so it gives more weight to certain synapses since these led it to the correct answer. The closer the answer, the higher the reward, and the more weight will be applied to certain synapses. Training only through *Synapse weight updates* has a major drawback, however: once a node has been identified as usually working, the AI will, if the dataset isn't varied enough, **nearly always pick this decision**. This further strengthens the synapse, and creates a vicious cycle, until the AI is mostly wrong, but will still pick the same decisions to achieve an answer. This problem is also called *overfitting*.

To address this issue, the principle of *mutation* is introduced. Much like human genetics, a random selection of synapses is changed to certain instances of the AI. This has the effect of enabling the AI to pull itself free from overfitting to a certain case and find a better way to get the answer from any scenario.

PROJET SETUP – THE TECHNOLOGIES

LANGUAGES & SOFTWARE

ASLDT is a Python based application, designed to run on a standard personal computer. To lead this project to a successful end, we will need the workflow offered by different powerful software solutions developed for coders and engineers as a whole: (see REFERENCES for links)

- Visual Studio Code – a versatile code editor with thousands of useful extensions
- GitHub – to neatly version, organize, split, and store code
- GitHub Pages – a web-page hosting option offered by GitHub
- Jupyter Notebook – a standard app in AI development to annotate, comment and run python code
- Trello – a powerful task management solution to keep progress organized

In addition to this software we use to develop ASLDT, we will offload AI training computation to AWS SageMaker as this is a highly graphics intensive task, usually not quite accessible at scale to individuals.

The project will also take advantage of some additional software, like cbFiles to host large files, LucidChart for charts, Smartsheet for progress tracking, and more.

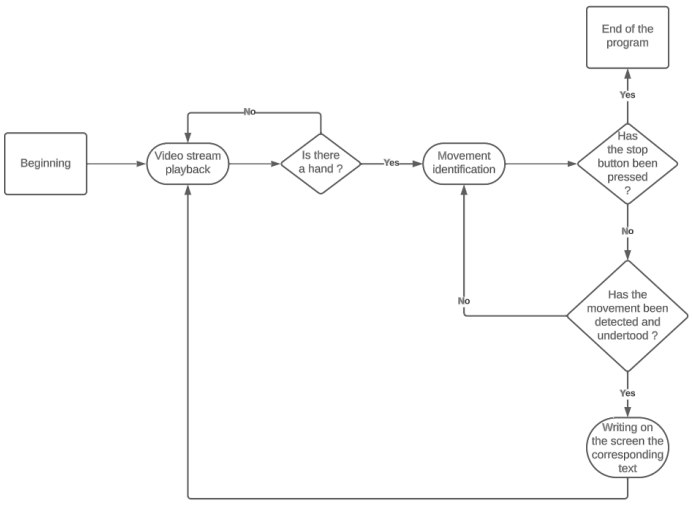
LIBRARIES

ASLDT is a complex application which leverages many different libraries. A non-exhaustive list of the principal ones follows.

- OpenCV – a computer vision focused python Library
- MatPlotLib – a graphing library (useful for visualizing potential optimizations for example)
- Pygame – a 2D game python library to handle graphics and controls
- Tensorflow – a machine learning open-source library developed by Google

PRELIMINARY SYSTEM DESIGN – INSIDE THE PROGRAM

Designing a plan of action and studying a problem before establishing a solution is the most sure-fire way to succeed in any endeavor. ASLDT being no exception, solving the issue at hand was extensively studied and itemized. The following flowchart helps visualize the internal workings of ASLDT at a general level. It follows the four main objectives mentioned in DESCRIPTION: obtaining the video stream, detecting appropriate shapes, identifying the signs, outputting the corresponding translation. This rough sketch is a good, concise explanation for ASLDT's inner workings.



FLOWCHART OF ASLDT'S FUNCTIONALITIES

COMPONENTS – THE HARDWARE

As ASLDT does not have heavy hardware requirements, it can be executed on many different devices with different architectures, the only necessary components being a functional camera and a display to output results.

TASK DIVISION – THE CONDUCT

For the smooth running of this project, we try to work mainly in groups. But to move faster on certain points, we have decided to divide the tasks according to our strengths:

MEMBERS OF THE GROUP	TASKS
[redacted]	Project Lead; Acquisition of datasets; TensorFlow dev; Report detailing
[redacted]	Data visualization; Python wrapper development; Project timeline maintainer
[redacted]	Computer vision dev; Web dev; Blog maintaining;

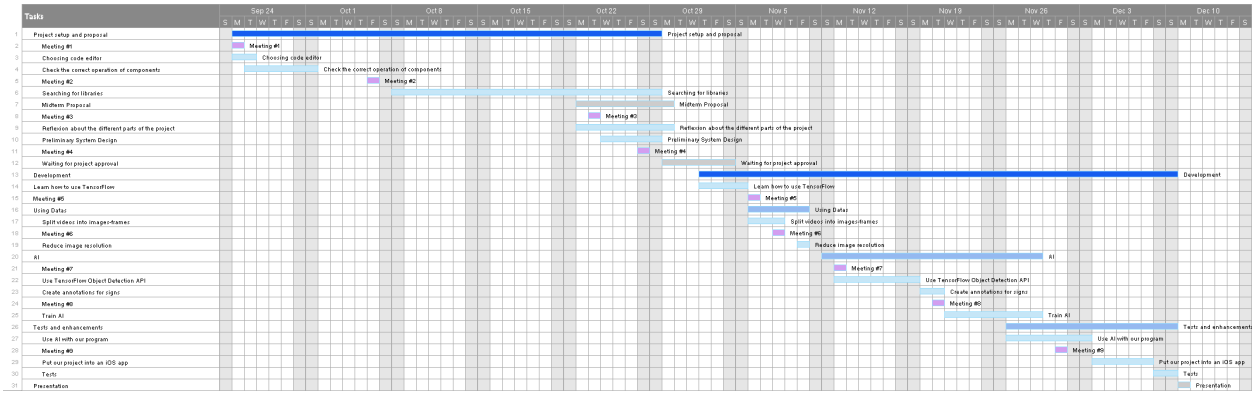
In addition to these split tasks, we keep certain tasks reserved to the group, such as AI development and design thinking.

PROJECT TIMELINE – NOW AND BEYOND

A clearly established timeline helps with time management and avoiding unexpected twists and turns.

CHART

Our project timeline can be described by this Gantt chart:



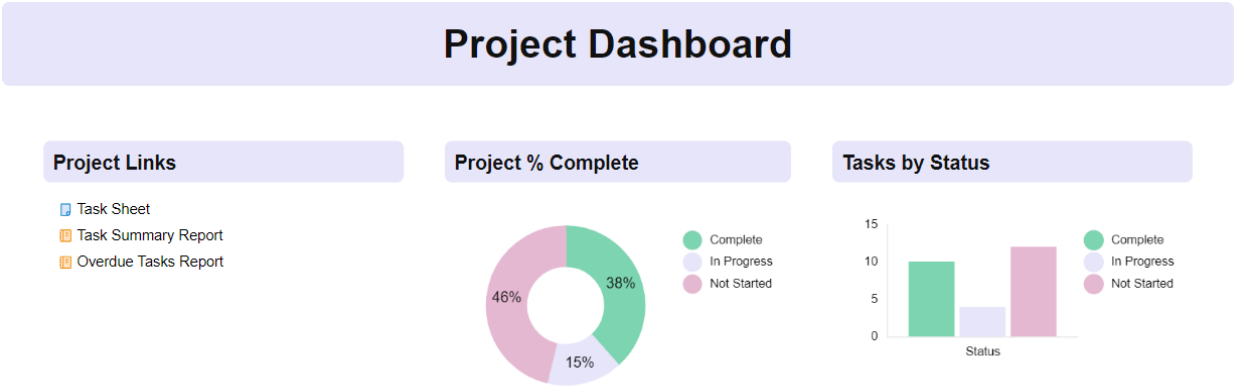
GANTT CHART OF THE PROJECT

Establishing a Gantt chart from the get-go allows easy visualization of meetings, tasks, and progress (along with the DASHBOARD). Any good project requires adequate planning, and that ideology is clearly present in this chart. Additionally, every interaction concerning ASLDT is not represented as regular communication is normal and expected.

The dark blue bars indicate the two general tasks: Setup & Proposal and Development and cornflower blue bars show subtasks. As the project advances, and we become more familiar with the project, the deadlines will become more concrete, allowing us to better develop individual tasks, clearly labeled in light blue.

DASHBOARD

For now, the reflection about the different important part of the project has been well done. But the project is not finished at all. Like we can see in the project dashboard below, we have a lot to implement and think about.



THE PROJECT DASHBOARD

To closely follow our progress, we will use Trello to have a greater view of the different tasks, but also Smartsheet to see our advancement. Having more insight into the well-being of the project is vital to keep steady progress and avoid last minute rushes to get everything back on schedule.

BEYOND THE PROJECT – THE EXTRA

Achieving the objectives set in DESCRIPTION is the first and foremost priority. With that in mind, additional bonus objectives have been thought out in the event implementing them is a possibility.

"Shoot for the moon. Even if you miss, you'll land among the stars" – Normal Vincent Peale

The first objective is translating complete sentences. ASL is a complex natural language in which expressions are common. For example, the literal translation of *"him"*, *"show"*, and *"car",* translate to *"He is showing a car"*. Such a complex endeavor cannot be considered as part of the core project, thus placing it in the extra objectives.

The second extra goal is having text-to-speech take advantage of most devices' speakers to automatically speak translated words (or sentences). This second objective is a lot more accessible, but **very** much extra, when compared to the main objectives.

Finally, the last point involves accessibility. Most individuals are not inclined to download a GitHub repository and run a python program, even if it solves such an important language barrier. As such, a regular application is a must for wider use. Developing an iOS app would be a perfect achievement for ASLDT. Leveraging the power of the A series chips and the more recent neural engine, would truly be a perfect match with the AI capabilities of ASLDT.

REFERENCES – THE SOURCES

Questions and guidance	ChatGPT
To get some ideas of the approach to have – Nicholas Renotte	YouTube
Flow chart creation	LuciChart
Knowledge database	Wikipedia
Python	Python
Python graphics library	Pygame
Versatile Code editor	VSCode
Code versioning and organizing	GitHub
Web hosting solution	GitHub Pages
Cloud AI training	AWS SageMaker
Python AI code editor and annotator	Jupyter Notebook
Python computer vision library	OpenCV
Python graphing library	MatPlotLib
Python machine learning library	TensorFlow
Self-hosted file sharing solution	cbFiles
Project task management	Trello