Sign Language ASL Recognition using Bag of Visual Words and Super Vector Machines

Samuel Ortega Cuadra[a] **and Laia Alexandra Sjöberg Cerezo**[b]

[a] *1669776*
[b] *1667894*

Abstract—This project focuses on developing a system for recognizing American Sign Language (ASL) alphabet and numbers. The proposed approach combines image preprocessing, feature extraction, and prediction models to accurately classify hand gestures. The dataset consists of images of hands representing ASL letters and numbers, which are processed by isolating the hand region, converting it in grayscale and applying a mask for feature extraction. Dense SIFT and a Bag of Words model are used to create feature representations, and a Support Vector Machine (SVM) classifier is trained and optimized through grid search for maximum accuracy.
The finished system can recognize hands by retrieving frames from video input, and translating ASL gesture sequences into coherent text.
This work lays a foundation for future developments in sign language recognition and improving accessibility for the deaf community by providing a tool for real-time translation of sign language to text.

Keywords—*a, b, c, d*

Contents

1. Introduction

This document presents the development of a system for recognizing American Sign Language (ASL) alphabet and numbers. It outlines the steps taken to achieve this goal, from the **dataset collection** and preprocessing to the implementation of a **Bag of Visual Words** to help builiding a **SVM model** for classification. The system is then extended to **process videos**, extract frames, identify ASL symbols, and **reconstruct** the words being spelled.

2. Finding The Dataset

When looking for a good ASL dataset, we found the American Sign Language Dataset dataset on Kaggle. This dataset contains 2515 images of 400x400 pixels, each showing a hand making an ASL letter or number. The dataset is divided into 36 folders, one for each letter of the alphabet and one for each of the numbers from 0 to 9. The images are in JPEG format and have a black background with a white hand. The dataset is well-organized and easy to use, making it a good choice for our project.

2.1. Dataset Preprocessing

The preprocessing of the dataset invloves the following steps:

1. **Dataset Organization**: The dataset is organized into folders, one for each letter of the alphabet and one for each number from 0 to 9. Each folder contains 65 to 70 images of hands making the corresponding ASL symbol. A *categories* dictionary was created to map each label to its corresponding folder name. The filenames of all images were collected and combined with their respective category labels to construct a DataFrame.
2. **Shuffiling the Dataset**: To avoid any bias in the training and testing data, the dataset was shuffled. This ensures that the data is randomly distributed and that the model will not be trained on any specific order of images.
3. **Image Loading**: Each image was loaded from its respective path and converted into a NumPy array. Providing a new variable with a structured representation of pixels called *pixel_ data*.
4. **Grayscale conversion**: The images were converted to grayscale to reduce the complexity of the model and improve training speed.

2.2. Masking the Hand

In order to find the best features we tried to mask the hand in the image using differents methods. The optimal resulting method was surprisingly the easiest. We took a look at the first 20 pixels of the image before converting it to grayscale and we found the most frequent color. Sequently, we applied a standard deviance and removed all of the background with the same colours, which will be really important when extracting the keypoints in the next chapter.

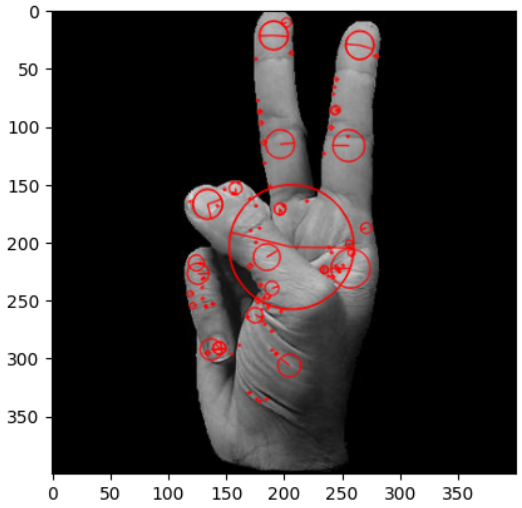
Information

We tried other methods such as brightness threshold, gaussian blur and several others from the computer vision library. They all ended up being less efficient than the one we finally used.

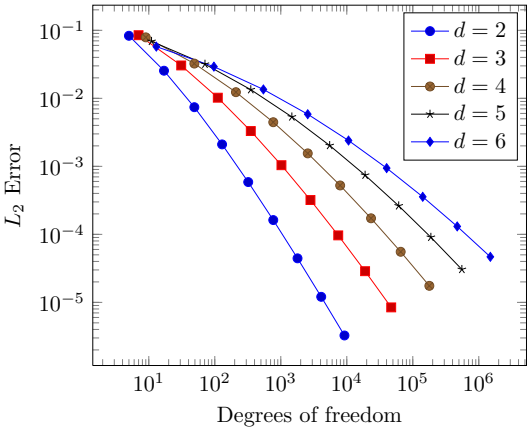
3. Feature extraction

3.1. Figures

Fig. 1 shows an example figure. Figures can be referenced using `\ref{ident}`, where `name` is a previously-defined `\label{name}` for any object in the document. You can actually also do this with sections (Section 3) or equations or other types of objects that will be seen below.



(a) Example left figure.



(b) Example right figure.

Figure 2. Example figure that covers the width of the page obtained from PGFPlots [1].

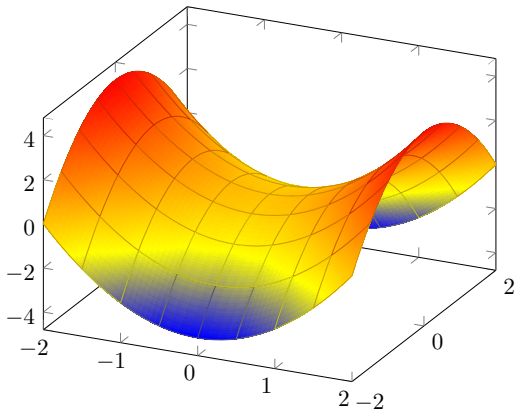


Figure 1. Example figure obtained from PGFPlots [1].

Fig. 2 shows an example of two figures that cover the width of the page. It can be placed at the top or bottom of the page. The space between the figures can also be changed using the `\hspace{Xpt}` command.

Information

Figures in LaTeXare known to be quite anarchic. You should always wait until the very end to place them well into the document. In any case, you should always reference the Figure and provide a Caption so that it can be contextualised even when it lies far away from the text that accompanies it.

Information

Avoid raster graphics (jpeg, png) for figures if you can. Always use vectorial formats for plots and diagrams. The preferred formats for vectorial images are SVGs converted to Encapsulated PostScript (.eps, you can use Inkscape to perform this conversion) or PDF files. Regardless of the image format you are using, generate reasonably high-resolution files (the usual is 300dpi, which means 300 pixels per squared inch). **Keep in mind that Matplotlib can be configured to generate plots with these settings.**

3.2. Tables

Table 1 shows an example table. The `\tabletext{}` is used to add notes to tables easily.

Information

This template uses the `booktabs` package by default. This package makes typesetting nice tables easier, but it imposes some general design guidelines. The most prominent is the distinction between middle and edge rules (horizontal lines) and the discouragement of vertical lines.

Information

You should **always** avoid typesetting tables by hand. Use tools such as this one to spare yourself some unnecessary pain.

Table 1. Small example table.

Column 1	Column 2
Data 1	Data 2
Data 3	Data 4

Note: I'm a table text for additional information.

4. Predicting using the codebook

4.1. Tauenvs

This template has its own environment package *tauenvs.sty* designed to enhance the presentation of the document. Among these custom environments are *tauenv*, *info* and *note*.

There are two environments which have a predefined title. These can be included by the command `\begin{note}` and `\begin{info}`. All the environments have the same style.

An example using the tau environment is shown below.

Environment with custom title

This is an example of the custom title environment. To add a title type `[frametitle=Your title]` next to the beginning of the environment (as shown in this example).

Tauenv is the only environment that you can customize its title. On the other hand, info and note adapt their title to Spanish automatically when this language package is defined.

4.2. Taubabel

In this new version, we have included a package called *taubabel*, which have all the commands that automatically translate from English to Spanish when this language package is defined.

By default, tau displays its content in English. However, at the beginning of the document you will find a recommendation when writing in Spanish.

Note: You may modify this package if you want to use other language than English or Spanish. This will make easier to translate the document without having to modify the class document.

5. Video Processing

Equation 1, shows the Schrödinger equation as an example.

$$\frac{\hbar^2}{2m}\nabla^2\Psi + V(\mathbf{r})\Psi = -i\hbar\frac{\partial\Psi}{\partial t} \tag{1}$$

The *amssymb* package was not necessary to include, because stix2 font incorporates mathematical symbols for writing quality equations. In case you choose another font, uncomment this package in tau-class/tau.cls/math packages.

If you want to change the values that adjust the spacing above and below the equations, play with `\setlength{\eqskip}{8pt}` value until the preferred spacing is set.

6. Hand Detection

This class[1] includes the *listings* package, which offers customized features for adding codes in LaTeX documents specifically for C, C++, LaTeX and Matlab.

You can customize the format in tau-class/tau.cls/listings style.

```
function fibonacci_sequence(num_terms)
    % Initialize the first two terms of the
    sequence
    fib_sequence = [0, 1];

    if num_terms < 1
        disp('Number of terms should be greater
    than or equal to 1.');
        return;
    elseif num_terms == 1
        fprintf('Fibonacci Sequence:\n%d\n',
    fib_sequence(1));
        return;
    elseif num_terms == 2
        fprintf('Fibonacci Sequence:\n%d\n%d\n',
     fib_sequence(1), fib_sequence(2));
        return;
    end

    % Calculate and display the Fibonacci
    sequence
    for i = 3:num_terms
        fib_sequence(i) = fib_sequence(i-1) +
    fib_sequence(i-2);
    end

    fprintf('Fibonacci Sequence:\n');
    disp(fib_sequence);
end
```

Code 1. Example of Matlab code.

If line numbering is defined at the beginning of the document, I recommend placing the command `\nolinenumbers` at the start and `\linenumbers` at the end of the code.

[1] Hello there! I am a footnote :)

This will temporarily remove line numbering and the code will look better as shown in Code 1.

7. Final Results

The default formatting for references follows the IEEE style. You can modify the style of your references, for that, go to tau-class/tau.cls/biblatex. See appendix for more information.

8. Future Improvements

8.1. Alternative title

You can make the following modification in tau-class/tau.cls/title preferences section to change the position of the title.

```
\newcommand{\titlepos}{\centering}
```

Code 2. Alternative title.

This will move the title to the center.

8.2. Info environment

An example of the info environment declared in the 'tauenvs.sty' package is shown below. Remember that *info* and *note* are the only packages that translate their title (English or Spanish).

Information

Small example of info environment.

8.3. Equation skip value

With the `\eqskip` command you can change the spacing for equations. The default *eqskip* value is 8pt.

```
\newlength{\eqskip}\setlength{\eqskip}{8pt}
  \expandafter\def\expandafter\normalsize\
    expandafter{%
    \normalsize%
    \setlength\abovedisplayskip{\eqskip}%
    \setlength\belowdisplayskip{\eqskip}%
    \setlength\abovedisplayshortskip{\eqskip-\
    baselineskip}%
    \setlength\belowdisplayshortskip{\eqskip}%
  }
```

Code 3. Equation skip code.

8.4. References

In case you require another reference style, you can go to tau-class/tau.cls/biblatex and modify the following.

```
\RequirePackage[
  backend=biber,
  style=ieee,
  sorting=ynt
]{biblatex}
```

Code 4. References style.

By default, *tau class* has its own .bib for this example, if you want to name your own bib file, change the *addbibresource*.

```
\addbibresource{tau.bib}
```

9. Acknowledgements

Tau LaTeXtemplate built by Guillermo Jimenez.

References

[1] *PGFPlots - A LaTeX package to create plots.* [Online]. Available: https://pgfplots.sourceforge.net/.