ASL Detection – Using Facial Recognition Algorithms in a Novel Way

QEA Project 2: Faces Project Technical Report

Pranavi Boyalakuntla
Partner: Gail Romer

04/08/19

Abstract

A significant portion of the U.S. population communicates using ASL, American Sign Language. There is a large gap in communication between people that speak ASL and people that don't speak ASL. This gap can be bridged by using image processing and computer vision algorithms. This paper covers one possible algorithm, the Eigenfaces algorithm, that could be used to determine what ASL letter sign a person could be signing. The accuracy of the algorithm for each letter of the alphabet against the training data set was found to be 74.9±13.5. This is comparatively high and could be improved greatly by increasing the number of images in the training set, diversity of hands, and employing basic feature recognition algorithms.

Introduction

ASL ranks as anywhere between the 4th and 6th most used language in the U.S [1]. As this number inevitably grows, the importance of being able to communicate with ASL signers becomes more and more important. As it is impractical to hope that everyone will be able to learn ASL, utilizing image processing and computer vision algorithms appears to be the solution that is the most practical and efficient to enable communication between individuals.

Image processing algorithms have already been developed for facial recognition. Looking at these algorithms in a new context could lead to novel ways to detect ASL. To have proof of concept, we zero in on recognizing ASL signs for the alphabet.

When handling people's communication, their words could be misrepresented as potentially harmful or offensive. This can act as a detriment to interpersonal relationships, especially if someone doesn't verify the words with the original signer. In addition, if the sentiment of a sentence isn't conveyed properly, the intention behind words could be misunderstood and interpersonal relationships could be impacted.

In another sense, the data that is gathered from ASL recognition could be used to both improve the algorithm, and track people's conversations which can invade people's privacy and be used for targeted advertisement. Tracking what people are saying can also lead to unnecessary police alerts, especially if the software is incorrect in identifying what people are saying.

Lastly, as this is an image recognition software, it could have racial bias which could lead to minorities or people of color having lower accuracy in translation. As a result, it is very important to have diversity in the initial ASL training set.

Methods

To identify ASL letters, we used the Eigenfaces algorithm [2]. To begin, a set of images that can be used to 'train' the algorithm is required. By train, we mean that these are images that the algorithm will try to match the 'test' images to. If the testing image, or image we are trying to identify, is a close enough match to the training image, we can then assume that the training image and testing image belong to the same category (in our case: 'a', 'b', 'c', etc.).A data set of 100 images of the sign for each letter was used.

The training data needs to be normalized such that the number of pixels and varying brightness of the images don't heavily impact how well the algorithm works. This can be done by subtracting the mean intensity of the entire image from every intensity value from the same image and dividing every intensity value by the square root of the total number of pixels. Rather than subtracting the mean image, as the Eigenfaces algorithm suggests [2], we subtract out the mean of each image because the training images we used vary a lot in brightness. The images need to be reshaped such that a larger matrix holds each image as a column vector (refer to equation 1).

$$I_{NormalizedTrain} = \frac{1}{\sqrt{N}} \cdot \begin{bmatrix} I_{Train1} - \mu_1 & I_{Train2} - \mu_2 & I_{Train3} - \mu_3 \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}$$

Equation 1: Let $I_{NormalizedTrain}$ equal the normalized training image intensity matrix, N equal the number of pixels in the image, μ equal the mean intensity for each column vector and I_{Train} equal the column vector of intensities of the training images.

This algorithm utilizes a set of 'Eigenfaces', or eigenvectors that describe the pixel-by-pixel covariance between the training set images [2]. When these eigenvectors are reshaped into the correct image size, an image that visually explains the areas of largest variation in the data set is produced. The first eigenvector explains the most variance, the second eigenvector explains the second most variation, and so on. In order to arrive at this matrix, the method of singular value decomposition, or SVD, is used.

$$I_{NormalizedTrain} = U \Sigma V^T$$

Equation 2: This shows the singular value decomposition of $I_{NormalizedTrain}$. Let U be the pixel-by-pixel covariance matrix of $I_{NormalizedTrain}$, Σ be the eigenvalues squared in the diagonal of the matrix, and V be the image by image covariance of $I_{NormalizedTrain}$.

$I_{NormalizedTrain}$ has the singular value decomposition as shown in Equation 2. For the purposes of the algorithm the matrix U is the one that we will use as it will provide the eigenvectors for pixel-by-pixel covariance. U is a matrix where the columns are the eigenvectors of $I_{NormalizedTrain} * I_{NormalizedTrain}^T$.

In order to only use the eigenvectors that show relevant eigenvalues, we will only use the first few eigenvectors provided in U. In the case of recognizing ASL letter, 150 eigenvectors proved to result in the highest final test accuracy. Let this matrix be called $I_{Eigenimage}$.

$$I_{TrainWeights} = I_{NormalizedTrain}{}^T \cdot I_{Eigenimage}$$

Equation 3: Let $I_{TrainWeights}$ equal the matrix of linear combination coefficients, or weights, that need to be multiplied by the 150 Eigenimage vectors in $I_{Eigenimage}$ in order to equal the image column vectors in $I_{NormalizedTrain}$.

Once this matrix of eigenvectors is found, the linear combination of the first 150 eigenvectors that results in the test images needs to be found. This can be found using the expression in Equation 3. Now that we have the weights associated with the training images, we need to find the weights associated with the test images in order to compare the two.

In the case of ASL recognition, the training set should change each time. To feasibly test this algorithm without using live camera techniques, a test database where 2900 images of each letter sign was used. We asked the user for a string of letters to recognize. Then, to simulate different lighting and positions as if live camera was used, a different image from each letter database was used each time the letter was called. This was done by a random number generator to load a different number of image each time. These images then have to be reformatted and normalized the same way that the training images were (refer to equation 1).

$$I_{TestWeights} = I_{NormalizedTest}{}^T \cdot I_{Eigenimage}$$

Equation 4: Lets $I_{TestWeights}$ equal the matrix of linear combination coefficients, or weights, that needs to be multiplied by the 150 Eigenimage vectors in $I_{Eigenimage}$ in order to equal the image column vectors in $I_{NormalizedTest}$.

The weight vectors for the testing image need to be found. This can be done using the method shown in Equation 4. As stated before, the weight vectors that are the most similar to each other will point to the two images that are theoretically the same. This can be done by finding the Euclidean distance between the test and train image vectors and selecting the smallest distance to be the best match.

$$I_{distance} = \sum_{i} (I_{TrainWeights} - I_{TestWeights}) \circ (I_{TrainWeights} - I_{TestWeights})$$

Equation 5: This equation creates a new matrix with the differences between each of the weight values squared that is the same size as both $I_{TrainWeights}$ and $I_{TestWeights}$. Let $I_{distance}$ be a matrix of the column sums of the Hadamard product of the matrix of differences between train and test, and itself.

We evaluated the equation shown in Equation 5. The reason we are not taking the square root to find the Euclidean distance is that computers are generally worse at approximating square roots. It takes less cycles to just square things and therefore saves time. Then, to find the magnitude of the distance, we calculated the sum of each column in the resulting matrix. The smallest value in this matrix signifies the image that the test image is most similar to. The image that corresponds with the weights therefore match. Since we know what letter the training image is, we can then assign the same letter to the testing image and print the appropriate final string.

Results and Discussion

'A'	'B'	'C'	'D'	'E'	'F'	'G'	'H'	'I'	'J'
76.233%	81.4%	86.267%	80.53%	71.33%	74.0%	80.167%	85.9%	76.167%	83.3%

'K'	'L'	'M'	'N'	'O'	'P'	'Q'	'R'	'S'	'T'
76.067%	80.03%	75.9%	82.267%	64.17%	84.47%	88.4%	68.93%	70.03%	79.5%

'U'	'V'	'W'	'X'	'Y'	'Z'
61.4%	64.0%	67.37%	73.63%	79.13%	85.4%

Table 1: These values show the percent of images from our testing data set that are accurately identified as the ASL letter sign that they correspond to.

The percent accuracy for each image is shown in table 1. As it can be seen the accuracy for each image is relatively good, with the lowest value being 61.4% accuracy for the letter U and highest value being 88.4% for the letter Q.

We also created the code that can take in user input, randomly select images for each letter, and output the final recognized string. This is shown in figures 6 and 7.

```
What word do you want to recognize?robot

wordrecog =

    'ROBOT'
```

Figure 1: This image shows the user inputting the word 'robot' and the algorithm correctly recognizing the word from the ASL signs.

```
What word do you want to recognize?quantitative

wordrecog =

    'QWANTIADSIVS'

>> ASLRecognition
What word do you want to recognize?engineering

wordrecog =

    'ANGINIERSNG'

>> ASLRecognition
What word do you want to recognize?analysis

wordrecog =

    'ENALYSKS'
```

Figure 2: This image shows the user inputting the words quantitative, engineering, and analysis separately and the algorithm incorrectly recognizing the word every time.

```
>> ASLRecognition
What word do you want to recognize?faces

wordrecog =

    'FACES'

>> ASLRecognition
What word do you want to recognize?faces

wordrecog =

    'FACES'

>> ASLRecognition
What word do you want to recognize?faces

wordrecog =

    'EKDBS'
```

Figure 3: This image shows the algorithm correctly identifying the word faces twice and incorrectly identifying the same word on the third trial through.

As seen in figure 3, the algorithm answer changes every time it is run. This demonstrates how the test images change every time and how this can affect accuracy.

The individual recognition accuracies of each letter shows how training data sets need to be more flushed out. In addition, the training and testing data sets were created by the same person [4]. Therefore, this algorithm currently only works for this one individual.

If this algorithm was applied to more people, the skin color and hand shape, among other factors, can affect the ASL to written English translation. This can negatively impact people of backgrounds and disabilities as their ASL may be harder to recognize accurately. This also shows how sometimes the results can be accurate while other times it may result in a

completely inaccurate answer. This can lead people to believe that the program works and take it for granted. This can damage interpersonal relationships if the word is translated into something that can be perceived as offensive.

A simple way to improve the system that could fix some of these problems is to have a more diverse and wide training set. By including more people and types of people in the training set, the chance that the test image matches one of these will increase. In addition, using a more accurate algorithm like Fisherfaces can increase the general accuracy of the system as a whole.

Conclusions

After utilizing the Eigenfaces algorithm to recognize ASL letters, there is definitely a proof of concept that facial recognition algorithms can be applied to other uses including identifying ASL letters. There was a relatively high percentage of images that were correctly identified with the mean accuracy being 76.77%. These accuracies ranged from 61.4% to 88.4%. If the training set was more fleshed out, or some basic feature recognition techniques were applied, this percent of accurately identified images could become even higher. In the grand scheme of things, with a larger training set in diversity of hands, and more images and words, image recognition techniques can be applied to recognizing ASL as a language.

References

1. "Ranking of American Sign Language as a 'Spoken Language' by SignGenius." Accessed April 4, 2019. https://www.signgenius.com/sign-language/ranking-of-asl-as-spoken-language.shtml.

2. Turk, Matthew, and Alex Pentland. "Eigenfaces for Recognition." *Journal of Cognitive Neuroscience* 3, no. 1 (January 1991): 71–86. https://doi.org/10.1162/jocn.1991.3.1.71.

3. "ASL Alphabet | Kaggle." Accessed April 5, 2019. https://www.kaggle.com/grassknoted/asl-alphabet#asl_alphabet_train.zip.