ECE 6258 DIGITAL IMAGE PROCESSING: SIGN LANGUAGE TRANSLATION

Lucas GRUSS, Johann FOTSING

Georgia Institute of Technology (Lorraine)
Electrical and Computer Engineering
2 Rue Marconi, 57070 Metz, France

ABSTRACT

We present a sign language classifier that uses different methods from Machine Learning and Computer Vision to classify the ASL alphabet. We develop ad-hoc methods to differentiate moving and still images as an additional source of information towards classification.

1. INTRODUCTION

Human-machine interactions are getting more and more intricate. It is for instance possible to interact with computers through the use of physical devices (mouses, keyboards, trackpads...) and voice recognition. Gesture interaction can be part of interaction with machines, for instance in helping cobots understanding the context while they are used, or for impaired people who would not be able to use other kind of input.

This project is an attempt at real-time hand gesture recognition. We focused on American Sign Language (ASL), and the recognition of the alphabet. The alphabet in ASL is composed of 24 postures and 2 gestures (letters J and Z). The emphasis of the project is to allow for the sign recognition to happen in real time. There are Neural Networks architectures that are able to reconstruct hands in the 3-dimensional space (feature extraction). However processing even a single frame requires several seconds of computation, so this technique is not employed in our approach.

2. GOALS AND ASSUMPTIONS

It is important that we set a precise context for the work we have done during this project and the usage of the program we built.

Our main goal is to perform fast hand sign classification, using classical consumer electronics : a laptop and integrated webcam. The computer used to benchmark the system is a HP Core Notebook equipped with Intel i5 (7th generation), 8 GB of RAM and a Nvidia GTX 950M as GPU.

Our approach consists in segmenting hand in an image and then classifying it with machine learning tools. Since the segmentation is a colour based approach, it is important that the user stands in an appropriate background. Also, we limit the sign detection to **only one hand** in order to easily manage conflicts between the hand and the face.

3. SEGMENTATION OF THE HAND

There are numerous approaches to ASL translation in research and projects. A lot of them define a precise area in which the hand should be placed so that classification of the sign can be accomplished. We implement an approach that does not make such an assumption and users can sign virtually anywhere in the field of vision of the camera. Nevertheless, we constraint our program to work with only one hand in the frame.

3.1. Skin detection

In order to extract the hand from the image captured by the webcam, it was chosen to extract the hand based on the ratio between red and green in the colored image, as the red and green ratio lies in a specific interval for all skin colours (**Fig. 1**).

It should be noted that depending on the quality of the webcam used and the lighting conditions, the colour levels for the same individual might vary considerably. This adds a challenge to the detection of skin as the skin colour of the individual is not the only variation that has to be taken into account when segmenting the hand from the image. In order to circumvent those issues, we always perform a calibration prior to running the algorithm.

From the input image I we compute the mask M which is such that :

$$M[m,n] = \begin{cases} 1 \text{ if } r < \frac{I[m,n,R]}{I[m,n,G]} < R \\ 0 \text{ otherwise} \end{cases}$$

This mask is a mapping of all the skin coloured pixels in the image captured by the webcam, and is useful for background

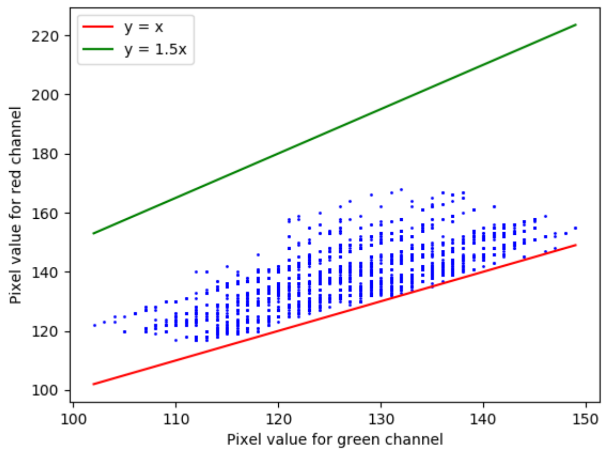

Fig. 1: Pixel value for skin on green and red channels

removal as well as identification of features in the images (mainly face and hands).

3.2. Noise filtering

It is likely that the mask is noisy and that pixels detected as skin coloured are actually part of the background, and skin coloured pixels are not detected. Specific lighting conditions and quality of the camera influence this type of noise. A computationally inexpensive approach to smooth out all the noisy pixels (salt noise on a mostly black image, and pepper noise on skin area) is to filter the image with a Gaussian filter with a kernel size of and then use a threshold value to put back pixels values to black or white (0 or 1). This helps removing both pepper and salt noise. This approach was preferred to a contra-harmonic filter, which requires more computation (Gaussian filtering with OpenCV is very fast).

3.3. Extraction of the hand

On the mask M, we are able to identify at most three regions filled with 1 : the face and the hands. At this point, the challenge is to know which of those regions is a hand. To work around this problem, we have taken advantage of some tools provided by the OpenCV library. The architecture of this solution can be described by the following image :

Contours extraction

Edges are easily detected in a black and white image and they correspond to the contours of white regions in the mask image. If the **calibration** has been performed properly, the largest detected contours correspond to the face and the hand. Finding the centers of the largest contours and their minimum enclosing circles, we can extract the hand and the face. A priori, the face contour should be longer than the hand contour,

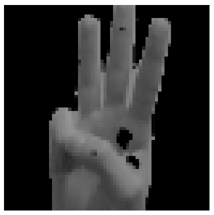

Fig. 2: Segmented hand from our protocol

but this strongly depends on the context. The area follows the same behaviour as the contour.

Hand recognition and segmentation

Given the two images of hand and face extracted previously, we need to determine which one is the hand we are interested in. We first relied on the first Hu moment $\mathbf{M_{00}}$ (equivalent to comparing the area) to separate the head from the hand, with the assumption that :

$$\mathbf{M_{00,hand}} < \mathbf{M_{00,head}}$$

This holds in most cases but robustness issues would happen when the hand was too close to the camera, thus making it appear bigger than the head. The use of the other Hu moments was considered but when signing, the shape of the hand changes hence the Hu moments too. We could also consider that the shape that has Hu moments that change the least is the face (the shape of the face stays consistent over time), but would require to track the position of the head. This was over complicating the problem.

OpenCV provides face classification with HaarCascade face and eye detectors, which is much more convenient and the technique is very robust. This classifier relies on Haar-like features in the image [1]. Thanks to this classifier, we can test whether a large skin coloured area is a face of a hand.

Now that we know which area is what, we extract the hand based on the minimum enclosing circle of the shape (c, r), where c is the center point of the circle and r its radius. We extract the hand in a image of $2r \times 2r$ centered on c and we apply the mask on this image to remove background.

4. MOTION DETECTION

We provide an approach to differentiating still images and motion. For computation saving, the approach does not make use of motion estimation, as it requires a large number of computation. We only care about binary classification of whether there is motion in the frame sequence or not. Based on the presence of motion or not, images are sent to the right sign language classifier (CNN).

The protocol is the following, to each frame a background extractor is applied. With the background extractor, all pixels belonging to background are coloured black and pixels associated with movement flare up. We store every fifth frame from the camera in a buffer (we only need to store one image) that we denote F_{-5} and we compare those frames by computing the Mean Square Error (MSE) :

$$\textbf{MSE} = \sum_{i=0}^{N} \sum_{j=0}^{M} (F[i,j] - F_{-5}[i,j])^2$$

The more movement (the faster the image moves), the "more different" successive images are. As such, for movements executed at different velocities $v_1 < v_2$, it holds generally that :

$$\textbf{MSE}_1 < \textbf{MSE}_2$$

where MSE_i is the mean square error between two frames of a movement executed at speed v_i.
Having a perfectly still image is impossible, and between consecutive frames the **MSE** is generally not equal to zero because of noise, variations of lighting and quality of the hardware used. We thus define a threshold T_{motion} that is used to determine the nature of the sign :

$$\begin{cases} \textbf{MSE} > T_{motion} & : \text{gesture} \\ \textbf{MSE} < T_{motion} & : \text{posture} \end{cases}$$

5. SIGN CLASSIFICATION

We perform sign classification using convolutional neural networks. Whether the sign is a gesture or a posture, images recorded by the webcan are fed to either of two CNNs, one for posture classification and one for gesture classification. The architecture is the same in both cases with the only difference being the number of classes it can classify from. We are able to use the same approach because our implementation to classifiy gestures is a naive one : we process single frames extracted from a whole gesture.

5.1. Posture Classification

We want to distinguish between 24 posture classes. We used an architecture that is quite recurrent in posture classification:

Training

The training set for postures consists of **24 classes**, each represented by 1,000 images. We augment the data set artificially by mirroring the images : this is to be able to classify signs performed by both hands (in sign language, signs can be executed with both hands, with some signs requiring the use of the other hand) and allows the model to find the right features in the image. The total augmented data set hence possesses around 52,000 images. We also have an additional data set used for validation which contains 9,000 images.

The training set for gestures consists of **2 classes**, each represented by 1,500 images. We perform the same data augmentation as before for a total data set containing 3,000 images. The validation set contains 1,000 images.

The best training curves recorded are the following :

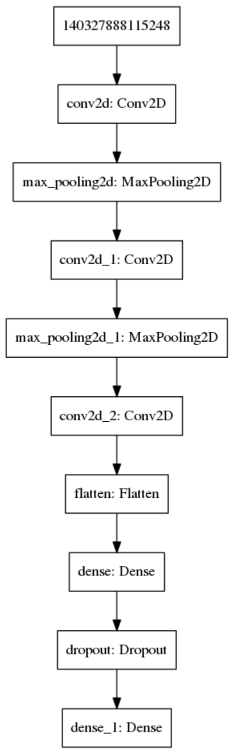

Fig. 3: Confusion matrix on the gesture classification

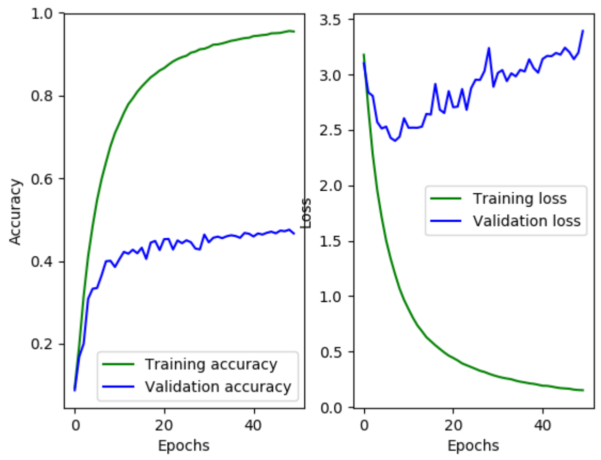

Fig. 4: Training curve for posture classification

Evaluation

We evaluate the CNN's performance directly in our sign classification program. Despite the satisfying accuracies observed during training, the real performance of the CNN is very poor.

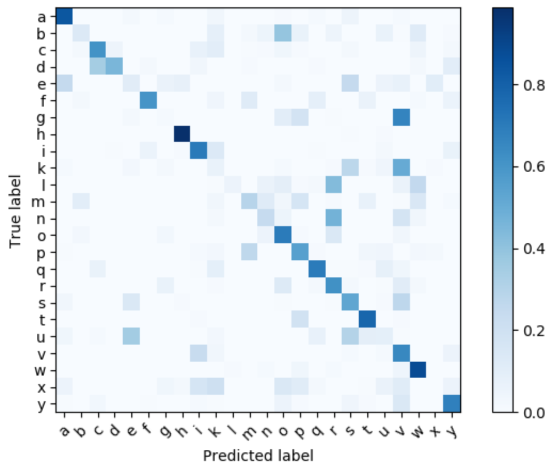

Fig. 5: Confusion matrix on the posture classification

6. CONCLUSION

We have seen all individual parts that compose our system. It should be noted that the real-time classification of sign language by our system does not work and the output from the CNN is almost never right. This is very frustrating because we were not able to fix this problem, and even though the classifiers (CNN) have accuracy which are not impressive, their inaccuracy when used in production is really surprising, given their result on validation data.

Our hypothesis for the issue with real time use is that lighting conditions have important consequences on the performance of the classification and reliability of the system, and are to be very similar to the conditions when recording data.

Gesture recognition gave results that were no better than random guessing. It could be improved by adopting another approach than the naive one, instead of feeding a single image from a gesture, a sequence of frames could be fed to the CNN. This could be particularly interesting to do with a larger number of gestures.

7. REFERENCES

[1] Paul Viola and Michael Jones, Robust Real-time Object Detection. Cambridge, 2001

[2] Xu and Pei, A Real-time Hand Gesture Recognition and Human-Computer Interaction System Department of Electrical and Computer Engineering, University of Minnesota, 2017

[3] Larsson and Fredrik, Shape Based Recognition, Department of Electrical Engineering, Linpking University, 2011

[4] Samer Hijazi, Rishi Kumar, Chris Rowen, IP Group and Cadence, Using Convolutional Neural Networks for Image Recognition, 2015

[5] Asanterabi Malima, Erol zgr, and Mjdat etin, A Fast Algorithm For Vision-based Hand Gesture Recognition For Robot Control