Vector-based Sign Language Recognition

Julien Baron, Guil Diyan, Guillaume Pires and Lucas Schackis
CESI Strasbourg
Emails: julien.baron@viacesi.fr , guil.diyan@viacesi.fr , guillaume.pires@viacesi.fr , lucas.schackis@viacesi.fr

***Abstract*—Sign language recognition (SLR) has been explored for many years and is still a challenging problem when it comes to real-time translation. While many algorithms are already meant to compute datasets and recognise diverse signs, there is still room for improvements. Indeed, cameras and algorithms are constantly improving, providing more depth and accuracy in both hand detection and sign recognition. Therefore, it is important for the data format to adapt as well, instead of using plain images, we could try using vectors representing the evolution of each movement to improve the accuracy of the recognition. This work will show our primary efforts on sign language recognition with vector-based datasets. Thus, the first experience were we used an home-made hand detection system based on 6 keypoints reached an accuracy of approximately 75% while our second experiment will show the impact of some features and the hands keypoints number on the efficiency of the SLR by reaching an approximate score of 81% success rate with the best sets of parameters on several attempts on a small dataset composed of 5 differents signs.**

1. Introduction

Wherever communities of deaf people exist, sign languages are developing. As with spoken languages, these vary from region to region and represent complete languages not limited in expressiveness. As for any research field, there are significant challenges and constraints in Sign Language Recognition (SLR), especially in real-time. One of the biggest would be the strong co-articulation between words. It means that the appearance of a sign depends on preceding and succeeding signs.

With the evolution of Artificial Intelligence and Machine Learning, we are, as of today capable of translating signs efficiently (With an approximate accuracy of 98%) featuring Convolutionnal Neural Networks (CNNs) for example. However, if many technical advances were made on diverse algorithms, only a few considered the importance of the structure of the datasets given to the latter, which could yet impact algorithm performances. Moreover, datasets rarely consider the spatial movement of the hand, which is clearly a key point in SLR because it can drastically change the deep meaning of a word, which can lead us to the question : How can we translate sign languages efficiently ? First, we shall present a method to recognize and track the hand and its fingers in an heterogeneous environment by placing keypoints on the hand, then, we'll extract all the movements vector of each keypoints to make a dataset to eventually propose some vector-based datasets structures to check the impact on classification algorithms and SLR by extension, which will indeed be the representations of spatial movements of the hand.

2. State of the art

Sign detection and SLR are the heart of concerns in the new technologies sector. Many methods both in image processing and classification fields were developed, we'll present them below. The main purpose will be to give a first glance on the field of study by looking at the main algorithms, methods and datasets used in SLR. Thereafter it will enable us to better understand the impact of each feature on the SLR efficiency and, therefore, will help us acquire some decent bases to contribute more efficiently and realistically to the field.

2.1. SLR Theory

To get a better understanding of how all SLR methods work and what features are chosen for datasets, there is a need to better understand the SLR fundamentals. Linguistic research in sign language has shown that signs mainly consist of four basic manual components : hand configuration, place of articulation, hand movement, and hand orientation, not to mention that Sign Language Recognition needs to consider facial expression and body gesture as well. [1] Usually, the main subject of the study is the "strong hand", i.e, the upper hand. Although sometimes we can study both hands if the sign represented by them is the same for both of them.

Moreover we can mention that after some researches we have discovered that the highest hand is the most interesting. Indeed, if a hand is used it is always the highest one. If the left hand is used, the videos are mirrored. Therefore, the model only needs to learn to process one side. [2]

According to a study [3], it would seem that when using the Kinect sensor, the sign recognition rate decreases when the subject is sitting. Indeed, from about 70% success rate when the subject is standing, the rate decreases down to 48% when sitting. Furthermore, it is only when

recognizing words, not even sentences, which would decrease the success rate even more. Nonetheless, due to this technology, it is possible to use the depth map, the user position on it and the joint map as data and form a usable dataset for Convolutional Neural Networks (CNN).

Multiple types of Sign Language Recognition systems exists. Some are visual based, some are sensor based, some are in real time, others aren't. Indeed, practical applications of SLR are mainly in real time, leading us to study in depth real-time SLR.

2.1.1. Real-time SLR systems. Real-time SLR systems are hard to put in place due to the resource-intensiveness when processing the images. The Multi-Layered Random Forest (MLRF) used for classification in [4], saves time on training the model and consumes little memory compared to the single random forest method. They also use an ensemble of shape function (ESF) descriptor. This method greatly improves recognition even when changing the hand position and can be computed in real-time due to the nature of the studied features (Depth, shape, etc.) based on translation-, rotation- and scale- invariant images. In their experiment, they needed 4000 seconds to train a tree with the first method on a quad core PC with 97.8% of accuracy but needed less than 1000 seconds on one core using their own method with 97.4% of accuracy.

Head gesture can also be used as a way to drive automatically a wheelchair in real-time [5]. To realize this solution, they used two algorithms, the Camshift algorithm to track the user's face and reduce the window size to only contain the face in it. Then, they used the Adaboost algorithm in this smaller window to get precise information of the face position and size, as well as head gesture.

2.1.2. Vision based approach. While conducting our researches we realised that the SLR was composed of three phases. In the capture phase, we basically gather the user sign. Then in the classification phase we classify each frame from the video to letters to eventually reconstruct them and display the most likely word from classification scores.

In order to distinguish the different signs, many studies [2], [6], [7] are based on the CNN or Convolutional Neural Networks algorithm. We can obtain results around 95 % of correct answers without background and other users in the field. If we decide to take these constraints into account, the processing time will be increased. Indeed, we are no longer on a 2D processing (gray scale) of the image as in the first study, but in a 3D approach with color.

Another way that has been seen in [8], is where they got the 3D depth information generated by the Kinect sensor. Thanks to it, we can obtain the region of the hand which is segmented thanks to a wristband. Having the hand region, a conditional random field is used to recognize the sign from the hand and then a Boostmap embedding method to verify the hand shape.

The Hidden Markov Model (HMM) is used to recognize signs from the color skin [9], where they experimented two viewpoints without the head gestures. The first one is a first-person viewpoint and the second one is a second-person viewpoint of a person sitting in a chair. The first-person viewpoint was more successful. Indeed, there was about 98% word accuracy with this viewpoint, but only 92% when it is the second-person viewpoint.

3. Contributions

We now have a concrete idea of the methods as well as the constraints that we must take into consideration. It seems important to us to propose an innovative solution that can abstract from the two main issues we have seen previously (background and user outfits can infer with the accuracy of signs recognition) while integrating these notions of movement in space. To do this, we hypothesize that placing points of interest on the hand and following their evolution in the space and time will be able to have convincing results on the prediction of the meaning of the sign in progress.

To confirm this hypothesis, we shall extract some vectors from keypoints placed on the strong hand and use those vectors to train some classifications models. If the accuracy of the laters is acceptable, then we could conclude that vector-based datasets can be viable for SLR. Moreover, it could be interesting to compare those scores with the accuracy of the same models with other types of datasets, to strengthen our hypothesis that vectors can be a reliable data in SLR.

3.1. Solution's foundation

Our solution is based on two major ideas about the vectored representation of the hand movement. As we said earlier, in SLR background variations can be a serious constraint that can severely alter the hand recognition, hence the sign recognition. Placing keypoints on the detected hand and calculate their movement vectors could bypass this issue. Furthermore, luminosity, skin tint and all types of color based inferences could also be avoided as vectors should be non-discriminating in terms of color.
Indeed, placing those points initially requires us to effectively detect the hand beforehand, implying that we need to resolve background and shape issues to at least place our keypoints and keeping track of them in the most optimal way possible.

This background dependency implies that the camera used to capture signs cannot move as any movement of the latter would falsify all the operations we made on background.

In this study we removed the presence of the head to focus on the hand movements and the vector-based datasets efficiency. For the sake of simplicity and efficiency we'll overlook the z coordinates of the points. Besides, the hand movements are from the French sign language (LSF)

3.2. Data Extraction

Having well-suited Hand-Detection and data-extraction methods is essential when we talk about performances in SLR, thus we need to find a decent way to detect the strong hand and extract data from it.

To determine movements vector of the hand, we first need to place key-points on it to track their movement. Several methods using Multiview Bootstrapping, Confidence maps, etc. already exists as ways to identify keypoints on hands efficiently [10]. For the sake of keeping our solution simple and easily maintainable, we're proposing a way to place 6 points on the hand and extract movements vectors afterwards.

The data extraction is performed alongside a real-time video capture from a computer webcam. First of all, each frame will undergo a background subtraction process to enlighten the hand boundaries and place the key-points. Then, vectors from each of those key-points will be computed to be usable as training data for our classification algorithms.

3.2.1. Background processing. This method consists of calculating the running average of the background for a given period. Then we add our hand in the frame and calculate the difference between the background and the actual image to get an image containing the foreground newly added object.

The running average is computed by the following formula :

$$dst(x, y) = (1 - \alpha) \cdot dst(x, y) + \alpha \cdot src(x, y) \qquad (1)$$

Where *dst(x,y)* is the destination/output image (32-bit or 64-bit floating point), *src(x,y)* is the input image which can be gray-scaled or colored (8-bit or 32-bit) and alpha which is the weight of the source/input image. Alpha decides the speed of updating (higher value means running average will be performed over a smaller set of previous frames).

A binary threshold is used on that difference image so as to get only our hand region.To do this we apply the same threshold value to each pixel. If the pixel value is below the threshold, it is set to 0, otherwise it is set to a maximum value in our case 1. This gives a black and white image as we can see below.

$$\textit{dst(x,y)} = \begin{cases} maxval & \text{if } src(x, y) > threshold \\ 0 & \text{otherwise.} \end{cases}$$

Where *dst(x,y)* is the destination/output image (32-bit or 64-bit floating point), *maxval* is the maximum value set, *src(x,y)* is the input image which can be gray-scaled or colored (8-bit or 32-bit) and *threshold* is the threshold value used.

Figure 1. Image of the hand's threshold

3.2.2. Identifying key points. From that image, the Open CV library provides us with an algorithm allowing to outline the hand boundaries.

Figure 2. Image of the hand and its real outline

Thanks to that, we will be able to define the convex hull, representing graphically a curve passing by the hand extremums. We talk about convexity defects. A convexity defect is an area positioning itself between the object segmentation outlines and the object real outlines.

With the convexity hull, the center of the hand can be determined with the following formula :

$$C_x = \frac{a+b}{2} \tag{2}$$

where a is the leftmost point and b is the rightmost point.

$$C_y = \frac{c+d}{2} \tag{3}$$

where c is the highest one point and d is the lowest point.

Figure 3. Image of the hand with outlines and its convexity hulls

To define the fingers from those informations, we take points in pairs from the convex hull and its opposite convexity defect to draw a triangle. The following cosine theorem is used to determine the cosine of the triangle.

$$c = \sqrt{a^2 + b^2 - 2ab\cos\gamma} \tag{4}$$

But to find c, gamma is needed. To find it, the following formula can be used :

$$\gamma = cos^{-1}(\frac{a^2 + b^2 - c^2}{2ab}) \tag{5}$$

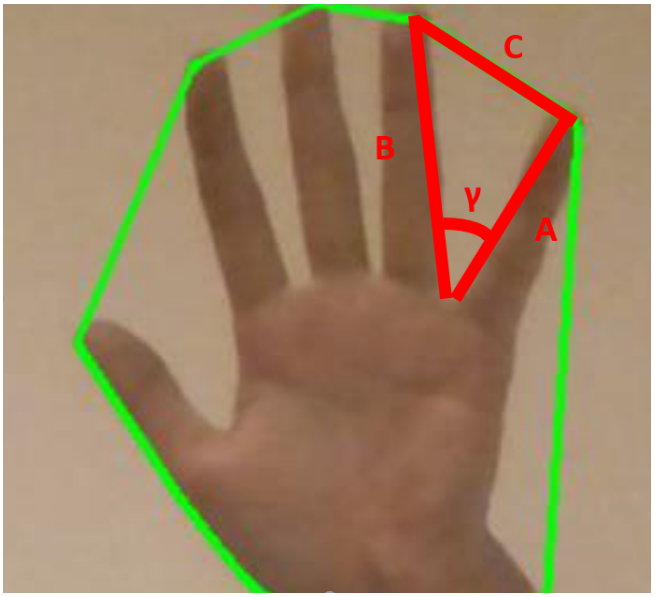

Figure 4. Hand and its outlines containing convexity hulls and the representation of the triangle

Once the angles are calculated, only the ones with a value lower than 90° are kept. Indeed, this range of values corresponds to the minimum and maximum between two fingers.

Now that the coordinates of the points associated to the finger are known, the vectors can be easily deducted. The following formula is used to create the vectors from the coordinates :

$$\vec{AP} = (A_x - P_x; A_y - P_y) \tag{6}$$

where A is the point of the actual frame while P is the point of the previous frame. The extracted data is then saved into our dataset.

Thanks to computed vectors, we can also deduce magnitudes(7) and directions(8) of them which corresponds to the velocity and the direction of each movement of each point.

$$|\vec{AP}| = \sqrt{(P_x - A_x)^2 + (P_y - A_y)^2} \tag{7}$$

$$\theta\vec{AP} = \arctan AP_y/AP_x \tag{8}$$

4. Signs classification methods

In this part we use the datasets previously generated by our algorithm. Their role is to train our chosen classification algorithms and thus allow them to guarantee better predictions of the hand signs. Multiple classification algorithms of supervised learning have been implemented. We opted for three distinct algorithms often used for this kind of problem which are the following :

Random forest classifier (RFC): It contains and use a number of decision trees on various subsets of the given

dataset and takes the average to improve the accuracy of the prediction for that dataset.

K Nearest Neighbors (KNN): The principle behind that algorithm is that from a given dataset, we can estimate the class of a new data from the majority class of the K nearest data neighbors, where K is the number of neighbors considered and the only parameter.

Support Vector Machine Classifier (SVM Classifier): The principle behind it is to draw the best line that can separate an n-dimensional space into classes so that a new data point can easily be put in the corresponding category. To that end, SVM chooses points/vectors that help in creating that separation.

5. Proposed Architecture

The proposed solution consists of extracting the coordinates as well as the vectors associated to the points of interest on the hand. Those data will be the learning base of the classification algorithms. Once trained, they will be capable of predicting what kind of sign the user performed.

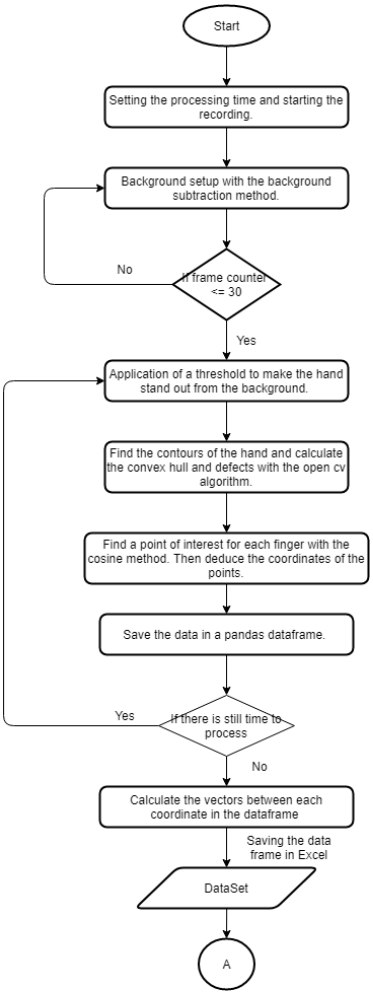

Figure 5. Dataset creation chart

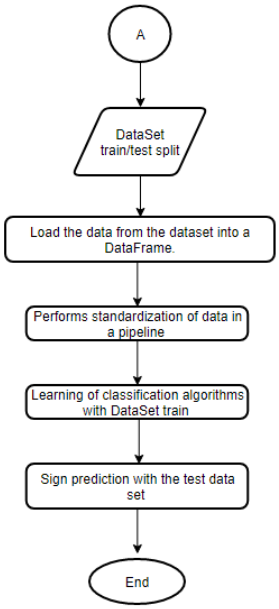

Figure 6. Chart representing the flow of the training and prediction of signs

First of all, the solution is based on background substraction. The first thirty frames are left so that the algorithm can preprocess the environment. Once that step completed, we can proceed to the extraction of the points of interest's coordinates as previously explicated. These data are subsequently saved into a buffer. We chose beforehand a capture time determining the stop that process. Henceforth, we can calculate the vectors from the buffer's data and then save it into an Excel file. It is now possible to import these data and let them be processed by Artificial Intelligence algorithms to let them learn how to recognize signs. For that purpose, we start preparing the dataset by standardizing it. With this data, we let those algorithms learn the characteristics of each sign. That way they should be able to recognize them later on.

6. Results

6.1. With Convex Hull identification technique

Classification Models Benchmark		
Tested Model	Error	Accuracy
Random Forest Classifier	33%	67%
K Nearest Neighbors	26%	74%
Support Vector Machine Classifier	46%	54%

We observe that the K Nearest Neighbors algorithm differs from other algorithms in its results. These results

seem to validate our previously stated hypothesis. And give us a good idea of what we can expect from this kind of motion based method.

Nevertheless we think that with more time we would be able to improve them. In fact, in order to reach good levels of prediction, our dataset must be quite large. To do so, it must be composed of millions of signs recorded by different people. We would then have an increased accuracy and a more reliable prediction.

6.2. With Confidence Map keypoints identifications technique

As mentioned before in [10], we tried to compare our homemade results with experiments made using the Confidence Map technique to identify keypoints on the hand. To be exhaustive in our analysis, we tried to modify multiple parameters such as the number of key points on the hand and the feature used to classify our data. We realised several attempts while changing parameters such as the number of keypoint and the feature used to classify to check if it would predict the good outcome based on 5 signs (The 5 firsts day of the week) and computed diverses values as for example the cross-validation scores to avoir overfitting our classifiers and their precision / recall scores to check their performances. Then we compared the average precision based on the number of keypoints and the feature selected. However, as mentionned before, due to the fact that our datasets are quite small, the high-success rate is also meant to be taken carefully. Still, it might be useful to conjecture that the number of keypoints and the feature used to classify do have a very noticeable impact on the classification efficiently and thus, the SLR efficiency.

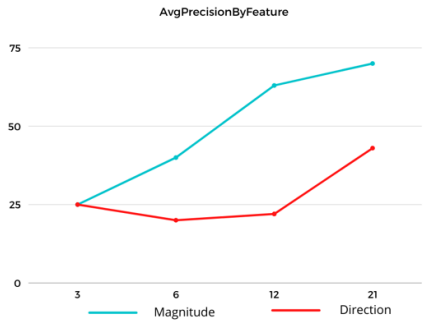

Figure 7. Curve of average precision by feature

6.2.1. Based on magnitude. We first used the magnitude of each vector as a way to classify our signs:

Average accuracy based on the number of keypoints			
Number of key-points	SGD	KNN	RandomForestClassifier
3	20%	25%	40%
6	55%	30%	30%
12	75%	50%	69%
21	81%	76%	47%

As we can see, it seems that the number of keypoints has decent a impact on the recognition accuracy.

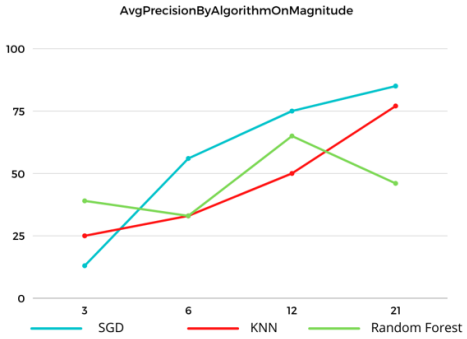

Figure 8. Curve of average precision by algorithms on magnitude

6.2.2. Based on directions. Here the results are not as efficient as before, but it still reinforce our idea that the number of key-points is quite influent in the result accuracy.

Average accuracy based on the number of keypoints			
Number of key-points	SGD	KNN	RandomForestClassifier
3	26%	22%	29%
6	25%	20%	23%
12	27%	60%	34%
21	47%	20%	73%

Here, the average accuracy is a bit lower than with the magnitude feature, but it still shows that in a general case, increasing the number of keypoints could be inferring in a significant way on the SLR and thus could be a important parameter when it comes to seek performance in sign recognition.

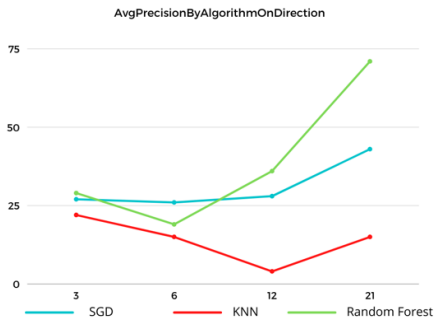

Figure 9. Curve of average precision by algorithms on direction

7. Conclusion and future works

7.1. Conclusion

We proposed the usage of movement vectors as a feature for our datasets, which we got by tracking the hands and the fingers. While testing this new type of dataset, we observed that some models had decent results, while totally perfectible. Nonetheless, it showed that vectors could potentially be a good feature for SLR. Due to the lack of time and ressources, we couldn't compare it's efficiency to other types of datasets, which would still be an interesting experiment. The main advantages of the usage of vectors are the non-discrimination of skin color and it can also increase the precision when used with a changing background, as we focus only on the movement itself. Moreover, thanks to our results we could conjecture that the number of keypoints is decisive in the efficiency of SLR. Due to our feature comparaison, we also can conjecture that the magnitude has a bigger impact than directions on the quality of the translation. Therefore, potentially, it could eventually be used in a more practical application like a translator or with IoT. There's no doubt that enhancing the SLR system performances and reliability are keypoints to enhance the global quality of communication between deaf people, mutes, etc. while increasing interactions in their daily lives.

The usage of our own tracking algorithm can also explain the difference between the precision obtained in our experiment and the precision in [1], [2], [3], [4], [6], [7], [8] and [9].

7.2. Future works

Indeed, our results should be taken with precaution as our tracking technique is still perfectible. There is still a need to better check the impact of the number of key-points identified on the classification accuracy. Instinctively, one might think that the more key points there are, the more precise the classification will be, but depending on the ways those key-points are placed, it could also negatively affect our sign-recognition, so further researches could be interesting to validate this conjecture. Moreover, it could be really interesting to evaluate the impact of each identified keypoints on sign recognition as some points tends to be more important than others. One way to do so would be to basically try enabling different keypoints on the hand and evaluate the precision scores afterwards. To push further our researches, we could take the depth of each key-point into account, as it is a key feature for SLR.

We could focus our work to enrich our vector-based dataset. We could also publish an application to collect the data of multiple Sign Language speakers to get a more robust dataset. Additionally, it would be worth to explore and optimize the different features potentially useful to enhance the accuracy of classification algorithms in SLR.

References

[1] P. Dreuw, "Speech recognition techniques for a sign language recognition system," *RWTH Aachen University*, 2007.

[2] L. Pigou, "Sign language recognition using convolutional neural networks (1)," *Ghent University, Belgium*, 2015.

[3] Z. Zafrulla, "American sign language recognition with the kinect," *SIAM Journal on Computing*, 2011.

[4] A. Kuznetsova, "Real-time sign language recognition using a consumer depth camera," *Institute fur Informationsverarbeitung, Leibniz University Hannover, Hannover*, 2013.

[5] P. Jia, "Head gesture recognition for hands-free control of an intelligent wheelchair," *Department of Computer Science, University of Essex*, 2007.

[6] N.Priyadharsini, "Sign language recognition using convolutional neural networks (2)," *dept of Computer Science and Engineering Sri Venkateswara College Of Engineering*, 2017.

[7] B. Garcia, "Real-time american sign language recognition with convolutional neural networks," *Stanford University*, 2016.

[8] H.-D. Yang, "Sign language recognition with the kinect sensor based on conditional random fields," *Chosun University, South Korea*, 2014.

[9] T. Starner, "Real-time american sign language recognition using desk and wearable computer based video," *The Media Laboratory Massachussets Institute of Technology, Cambridge, United-States*, 1998.

[10] T. Simon, "Hand keypoint detection in single images using multiview bootstrapping," *Carnegie Mellon University*, 2017.