2D-to-3D Lifting of Sign Language Body Poses with Recurrent Neural Networks

Jordi Aguilar
jordi.aguilar.larruy@estudiantat.upc.edu

Abstract

This paper aims at improving the quality of a dataset that contains multiple sequences of 3D poses extracted from American Sign Language videos. Each pose consists of 147 points with three coordinates each. We propose an algorithm able to correct missing points as well as to add some constraints such as the length of the bones. To prove the quality of the algorithm's outcome, we evaluate the task of lifting 2D to 3D poses with a deep learning model trained on raw data, and another one trained with the preprocessed data.

1. Introduction

Sign languages are languages that use the visual-manual modality to convey meaning. This means that they are expressed by movements of the hands, face or even small movements of the body. There are an estimated amount of 466 million people that are deaf or hard-of-hearing whose primary means of communication are sign languages [1]. Moreover, there are different sign languages all over the world just as there are different spoken languages, and they are not an ungrammatical form of the respective language, but a separate language with its own rules for pronunciation, word formation and word order.

Over the years, there has been a release of language models based on deep learning that enable communication between persons that speak or write with different languages [7, 12, 2, 8], but, for now, there are not consistent works that perform sign language translation or automatically generated sign language videos [15] good enough for the Deaf community to understand them.

As stated in [4], creating scalable automated systems that enable the continuous translation of speech into sign language is therefore a very important issue with high potential impact. Nonetheless, this is a difficult task that comprises hard problems such as speech recognition, continuous translation and sign language animation generation. Moreover, in order to build models that are able to achieve these tasks, large datasets that can feed data-driven methods are required independently if we tackle the problem in a supervised or self-supervised approach.

However, collecting large datasets is challenging in its most basic format of standard 2D video as it requires a recording studio with controlled conditions such as a solid background or fixed camera poses. The difficulty of such recordings grows exponentially when considering a dataset with 3D poses. For this reason, the present work is build over the How2Sign Dataset, a large-scale multimodal dataset in American Sign Language constructed to deal with the lack of datasets to properly implement sign language models. Nevertheless, finger joints are sometimes misplaced or even missing, and these errors are far from ideal when we use this data as ground truth. Moreover, hands play a central role in sign language understanding. Our main goal is to analyse this dataset and elaborate an algorithm able to correct sequences of 3D *keypoints*. This algorithm creates a new skeleton assuming fixed lengths for each bone, and then applies a gradient-descend-based method that mitigates too rapid movement between frames and undesirable bone positions.

In this paper we also describe a second task, consisting in 2D to 3D lifting. That is, given a 2D representation of the *keypoints* of a skeleton, predicting the depth of each keypoint. This task could serve as a tool to estimate the depth of any given 2D labelled sign language dataset, enriching the information contained by the skeleton poses [14]. To do so, we train a simple model composed of a LSTM and a fully connected layer. In this work, we are going to use this task to show the differences in performance, if any, between training the very same model with the data as it is and the data once the preprocessing has been applied. The evaluation will be made using cleaned data.

Our main contribution in this paper is the proposition of an algorithm designed to preprocess the How2Sign Dataset to avoid missing or misplaced coordinates. Thus, enhancing the quality of this data and making one step forward to contribute to the irruption of translation and synthesis models for Sign Language.

2. Related work

The main influence of this work is the How2Sign Dataset [5]. This dataset consists of a parallel corpus of 80 hours of instructional videos and their corresponding American Sign Language translation. Eleven signers participated on it, recording in two different sets: the green screen studio and the Panoptic Studio [6]. The first one was equipped with two HD cameras, one placed in front of the signer with a green screen as background, and the other at a lateral view. The Panoptic Studio is a system of hundreds of cameras placed at a geodesic dome, providing 3D skeletons poses of the interpreters. A total of 3 hours are recorded in the Panoptic Studio. This is the subset that we are going to be using for this work and will be better explained in the next section.

This work has also benefited from *2D to 3D body pose estimation for sign language with Deep Learning* [11], one of the first publications exploiting the 3D skeletons provided by the How2Sign dataset. This work tackled the 2D to 3D skeleton lifting, so it can be applied to annotate not only to rest of the How2Sign dataset, but any given 2D labelled sign language dataset. The approach taken was the construction of a model consisting of a Long Short-Term Memory layer and a fully connected layer. Furthermore, a normalization of each axes in the *[-1, 1]* range was applied, followed by a normalization by mean and standard deviation. Two different set ups were tested, a regression approach where the output was a continuous value and a classification approach where the output space was partitioned in bins. Best results were achieved using the regression approach, but the paper also highlights the difficulty to estimate the depth of the hands and the lack of improvement when using more complex models with more parameters. Our work mimics the architecture used in this paper to model the 2D to 3D lifting, but differs in the normalization applied to the data.

Finally it is worth mentioning the paper [17], a work dealing with text-to-video language synthesis in Czech Sign Language. The authors worked with Czech SL synthesis without relying on any explicit SL translation and developed a simple but robust feed-forward translator, but more importantly for our work, they designed a method for the skeletal keypoints correction that creates 3D skeletal models from the 2D models to achieve geometrical consistency. In particular, they construct a 3D skeleton while taking into account plausible bone lengths. Then, they use the projection of the constructed 3D skeleton to achieve a cleaned and robust 2D skeleton. This algorithm will be the baseline method for our preprocessing algorithm. We will follow the same ideas proposed in this paper and explained in section 4, adapting them to allow 3D skeletons input.

3. The Dataset

In this section we will present the main characteristics of the subset of the How2Sign dataset we are working with.

The How2Sign Panoptic Studio sub-dataset is structured in folders, each corresponding to a recording. It contains 24 videos where two signers translate instructional videos from the existing How2 dataset [13] to American Sign Language videos. The videos are recorded at 24 or 30 frames per second and their duration is about 290 seconds in total, resulting in 5 videos with a span of around 7000 frames and 19 videos with a span of around 8700 frames.

Inside each folder we can also find a JSON file for each frame of the videos. It contains the *keypoints* of all the persons that appear in that frame. In some cases we can find persons belonging to the organization that appear in the video, but we are just going to ignore their *keypoints* and focus on the two signers that appear in each of the videos.

The number of *keypoints* or joints for each individual is 137. They are divided in three parts: the body (25 *keypoints*) the hands (21 *keypoints* each, 42 in total) and the face (70 *keypoints*). Each keypoint has 3 coordinates which we call x, y and z, but it is important to recall that the orientation of the skeleton is arbitrary and depends on the camera perspective, which is not frontal in this case. Moreover, for each keypoint we have a confidence score in the interval [0, 1] that indicates the reliability of the set of coordinates, being 1 absolute certainty and 0 very bad precision. This latter aspect will be of high importance for our goal since we aim at having a more robust dataset without misplaced points and a consistent confidence for all the frames.

During the development of this project, we have taken special attention to the understanding and examination of the data. In particular, we focused on characterising the missing points to have an approximate idea of their amount. In table 1, we summarize two findings:

	Body	Face	Left hand	Right hand
a)	0.2%	0.005%	1.6%	1.8%
b)	0.47	0.69	0.69	0.68

Table 1: *a)* Percentage of frames with more than 20% of missing points among all the videos. *b)* Mean of the confidence score. Computed among all videos and interpreters

We can see how the percentage of missing points in the hands is higher than in the other parts. This is not surprising since they involve a lot of fast movements and are the part of the body that is most likely to suffer occlusion. However, the confidence scores are much lower in the body keypoints.

4. Skeletal model correction

In this section we are going to introduce the preprocessing algorithm to correct the data and achieve important improvements to its consistency. It is divided in two parts: the first one tries to obtain reasonable confidence scores for each frame by means of a weighted mean between the closest neighbours, and the second part makes use of an invariant factor such as the length of the bones to provide robustness to the skeletons while preserving smoothness between frames. It is applied to the whole skeleton at the same time.

This method was proposed in the paper *Words are our glosses* [17], but we have adapted it to fit 3D skeletons instead of 2D ones. The main difference is that in their algorithm they have to obtain a 3D skeleton in order to scale each bone to the corresponding length whereas there is no need to do this step in our algorithm.

Next, we are going to explain each one of the steps to implement this method.

4.1. Pruning

This step filters frames that are not reliable or extremely inconsistent. Skeletal models with a lot of missing or misplaced parts are replaced by uninitialized skeletal models. Fortunately, in the next section it is explained how they are going to be reconstructed. The filtering of a frame is done by putting a minimum threshold to the mean of the confidence score of a specific subset of keypoints. The keypoints selected are those from the torso and arms, since they are the root of the other parts of the skeleton and we can't afford to have an skeleton that will present global misplacing issues due to poor confidence in these key parts.

4.2. Interpolation

The interpolation step applies a filter to each keypoint in the temporal domain to avoid missing parts and to gain a bit of smoothness in the sequence. Previous to the implementation of the current method, we tried to apply well-known windows such as a flat window, the Hamming window or the Bartlett window [9]. The idea behind this strategy was to smooth the sequence by averaging each keypoint with its temporal neighbours. The flat window performs an arithmetic average between the neighbours at a maximum distance $window\_length/2$, while the other two can be interpreted as a weighted mean, where the further the distance to the current point, the lower the weight assigned to the neighbour. However, two problems arose: the first one being that good quality measurements were affected by poor quality neighbours or even by 0 values. The second one was when we had consecutive missing values wider than the window length, since the result of the filter in these sections was still unprofitable.

The solution proposed to fight this problem was to use variable window lengths based on the confidence score of the keypoints. For each joint we create a symmetric window long enough so that the sum of confidence scores of the neighbours is at least 1. This means that if a keypoint has a confidence score of 1, it will remain unaffected. Nevertheless, if there is a sequence of consecutive 0 values, the window will be long enough so that at least we are taking into consideration keypoints with enough confidence to sum 1. Moreover, once we have defined the window length, a weighted mean by the confidence score is applied, causing that the 0 values or the misplaced ones have almost no influence in the outcome of the filtering.

4.3. Reconstruction

In this phase a skeleton with fixed bone lengths is created. This allows us to have consistency in the skeleton poses and corrects points that may be misplaced. The first step is to get the length of each bone. To do so, we compute the distance between pairs of consecutive joints among all the frames of a video and we apply the median. It should be noted that we are considering symmetry between bones of the skeleton, meaning that symmetric bones from the left and right shoulders, arms, hands and legs must have the same length.

The main difference between the original paper comes in this section. To achieve equal bone length in 2D skeletons, they were required to work with a 3D skeleton. To do so, they relied on the vectors v_i between joints. Starting from the head keypoint k_0 in the coordinate $(0, 0, 0)$, the following keypoint k_1 was forced to be at the same distance as the corresponding bone length L. One can see that this problem has one or two possible solutions. If L is shorter than the norm of the vector between the joints, the following keypoint will be placed at $k_1 = k_0 + \frac{L}{||v_0||}v_0$. Note that the z coordinate will remain 0. If $L > ||v_0||$ we will assume that in order to meet the length constraint the point has to be lifted, so the coordinates of k_1 are going to be $x_1 = x_0 + v_0^x$, $y_1 = y_0 + v_0^y$ and z_1 will have the value that satisfies the bone length. Given that $L^2 = (x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2$, the solution for z_1 is:

$$z_1 = z_0 \pm \sqrt{L^2 - (x_1 - x_0)^2 - (y_1 - y_0)^2} \qquad (1)$$

Because it is not know which solution is more admissible, they chose the smaller one. This procedure is applied recursively through consecutive bones. The output is a 3D skeleton where the bone length requirements are met.

Our problem does not require to transform a 2D skeleton to a 3D skeleton given that we already have it. Thus, following the same recursive approach we are going to scale each vector accordingly to meet the bone length while preserving the angle between joints. If we call k_0 to an already fixed keypoint, k_1 to the next keypoint, L the length of the

corresponding bone and v_0 the vector from the point k_0 to k_1, the equation to find k_1 is as follows:

$$k_1 = k_0 + \frac{L}{||v_0||} v_0 \tag{2}$$

This procedure is also applied recursively through consecutive bones. The output is a 3D skeleton where each bone has its defined length.

4.4. Regularization

The skeleton obtained in the previous procedure is already helpful because the bone lengths are fixed, but taking this skeleton as the final outcome may be undesirable since we have relied on the correctness of the angles between joints, and this can lead to misplacement of some parts of the body and discontinuities between frames. To solve these issues, one final regularization step is done. Taking as input the scaled skeletons x and as target the skeleton after the interpolation has been done y, we build the following loss function based on MSE:

$$\mathcal{L}(x) = \sum_{t=1}^{T} \sum_{k=1}^{K} \frac{(y_t^k - x_t^k)^2}{t \cdot k} \tag{3}$$

Where T is the number of frames and K the number of keypoints. It is clear that 0 value of the MSE can be found by simply reaching the y coordinates, but we would find undesired bone lengths, so we are going to add two regularization factors in order to find a mid-point between both sets. Hence, our loss function will look like this:

$$\mathcal{L}(x) = \sum_{t=1}^{T} \sum_{k=1}^{K} \frac{(x_t^k - y_t^k)^2}{t \cdot k} + \alpha \cdot R_1 + \beta \cdot R_2 \tag{4}$$

Where T is the number of frames and K the number of keypoints, R_1 is the length of the bones and R_2 is the difference between keypoint coordinates of consecutive frames. α and β are weights for the respective regularizations. It is interesting to notice that R_1 affects the keypoints in the spatial domain K, while R_2 influences the keypoints in temporal domain T. After optimizing the loss function, the final result of the preprocessing is achieved.

5. Evaluation

The present task consists in building a system that takes a sequence of vectors as input, in which each vector contains the x and y coordinates of every keypoint in a given frame, and outputs another sequence of vectors, in which each vector contains the estimated z coordinate of every keypoint in the given frame.

The purpose of this task is not to beat the current state-of-the-art work in 3D generation or to find new mechanisms to tackle this task. Instead, we are interested in comparing the performance of the same model trained with two different sets of data. Data quality and abundance are two factors really well appreciated in machine learning and deep learning since they are the basis for any data-modelling approach. Moreover, one of the wills that numerous authors express when their experiments underperform or do not generalize properly is the need for more and cleaner data.

Given the preprocessing method described in the previous section, we are being presented with a good opportunity to check if the cleaned data generated by the algorithm has a positive impact in the 2D to 3D lifting task with respect to the very same model trained with the unprocessed or raw data.

Before beginning with the model introduction, the data preparation steps based on [3] are going to be explained. First, all the skeletons are scaled so that the length of the bone between the neck and the mid-hip is equal to 1. It is important to notice that the three coordinates are scaled with the same factor, so we are preserving the ratios between dimensions. Then the mid-hip joint is moved to the origin. Finally, we make sure that the line between the neck and the mid-hip is over the y axis and that the skeleton is facing along the z axis. This ensures that by predicting the z coordinate, we are predicting the depth of the skeleton.

5.1. Model

To achieve the proposed task, we are going to build a deep learning model similar to the one developed in [11]. The approach consists of a Long Short-Term memory layer followed by a feed-forward layer. A LSTM network is really useful when dealing with time series or other kinds of sequences. In each time step, it receives not only the corresponding input features but also feedback from previous time-steps. This *feedback* is formed by a hidden-state and a cell state. The first one is the output from the previous time step and is simply concatenated to the current input. However, the second one is the core idea behind LSTMs. The cell state is a vector that runs through the entire sequence that is updated in each time step by the input of this one. It is the memory of this architecture and the one that enables the computation of outputs based on previous time steps.

5.2. Training the model

The data are split into two sets: train and validation. The first one contains 19 videos while the second one contains 5 videos. This means that approximately 80% of the frames belong to the training set and 20% of the frames belong to the test set. This solution ensures a good generalization of the model and may be preferable than using other options such us taking always the last 20% fraction of each video or splitting randomly the videos because we may change the distributions of poses between both sets. For example,

there may be some movements specific to the end of videos such as putting the hands together that are less seen in other parts of the video. To fasten the process and keep it simple, only one interpreter from the two available has been chosen to train and test the model.

One of the limitations that training LSTM exhibits is the length of the sequence accepted by the network . Our videos contain thousands of frames, but the sequence length of LSTM must be a low value so that the gradient propagates along all the sequence. Otherwise, the gradient might achieve very high values, a problem known as exploiting gradient. If we cut each video in fragments of *seq_length*, the problem arises when the internal states of the LSTM is reset between batches. This means that in practice, each video is split in num_frames/seq_length clips, and no information is shared between consecutive clips. To solve this problem a special configuration has to be set: the videos must be carefully placed so that each clip is placed one after the other in different batches, but at the same batch position, as depicted in 1. Then, the internal states of the LSTM are not reset between batches and the next clip can exploit all the accumulated information. This feature is adopted both at training time and at test time.

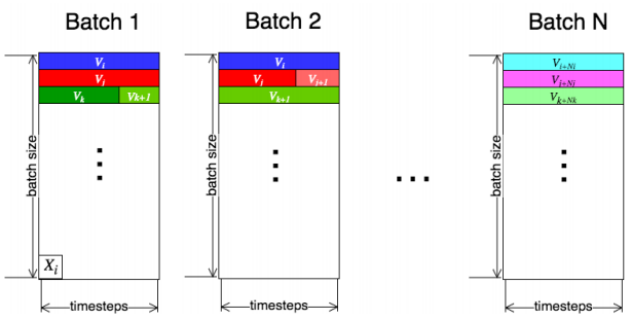


Figure 1: Graphical representation of how the data is sorted in batches. Image extracted from [10]

It is inevitable that the end of some videos will fall inside a batch. This could cause a problem since there won't be any relation between the last frame of a video and the first frame of the next video. This situations are also taken into account and the hidden state and the cell state are reset to 0 (its default value) inside a sequence whenever this happens. Luckily since we have a low number of videos this situation is not very frequent. The low amount of videos also implies that if we want to have a *batch size* bigger than the number of videos, some of them are going to be inevitably cut since we only have 22 of them.

The loss function used to train the network has been the Mean Absolute Error (MAE) $L = \frac{1}{n}\sum_{i=0}^{n}|y_i - h(x_i)|$ instead of the Mean Squared Error (MSE) $L = \frac{1}{n}\sum_{i=0}^{n}(y_i - h(x_i))^2$. These functions are also called L1 and L2 respectively. We can find two main differences between these methods. L1 loss is less sensitive to outliers which makes it more robust against them, but more importantly L1 loss gradients are the same throughout all the training process whereas gradients of L2 loss are big when the predictions are really bad but get smaller as the predictions get closer to the local minimum. This means two things: L1 loss takes longer to decrease but its gradients are larger for small prediction errors. Although this last fact may be harmful at the end of the training because there is a lack of precision, it has been seen in works working with keypoint coordinates from skeletons [16] that L2 yielded too small gradients due to the quadratic decrease, causing the models to struggle to achieve convergence.

6. Results

The results are divided in two parts. First we are going to evaluate the output of the preprocessing algorithm described in section 4. This will be done comparing qualitatively the output of the preprocessing algorithm when it is applied to some skeletons. Then we will present the results of the models trained to transform 2D skeletons to 3D ones. We will compare the accuracy of the models trained with unprocessed data versus the model trained after the data has been preprocessed.

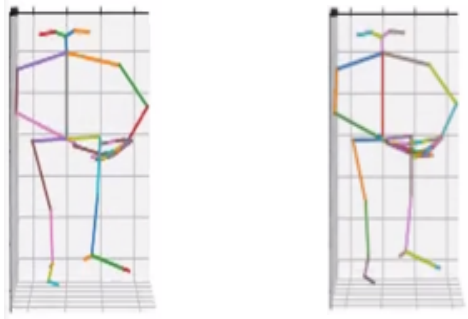
6.1. Qualitative results

The following pairs of figures show the differences obtained when preprocessing the data. All of them are focused on hands correction given that it is the part of the body that suffers more errors. The following examples have been hand-picked to better show situations where the algorithm has corrected missing points or inaccurate joint positions. Although there has been a meticulously evaluation of multiple pairs, these examples have been extracted from one unique video (*190611_asl2*) and one of the two signers. With this, we want to express that there may be plenty of other examples where our algorithm has an impact on the data and that the following examples are not remote.

In figure 2, we observe one of the main causes of missing points: the signer puts their hands together, making it really hard to predict the keypoints of the hand that it is being hidden. In this case, we are only detecting one finger of the right hand. The result of applying the algorithm is the reconstruction of the right hand, achieving a natural position.

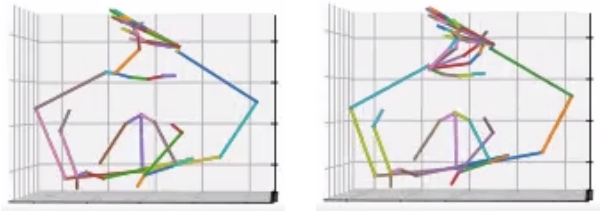
In figure 3, we are seeing an occlusion phenomena. The signer is performing a gesture that hides the left hand. This causes that three of the fingers are not properly retrieved and are missing in this particular frame. The algorithm outputs a reconstructed hand which fits with the existing fingers and with the other hand. We can appreciate how the hands are now perpendicular to each other.

Finally, in figure 4 we show an example of correction of a bone length. Since both hands are together, we have added



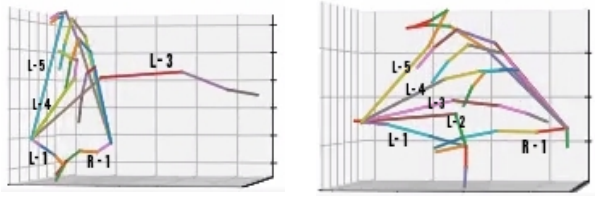
(a) Before preprocessing (b) After preprocessing

Figure 2: Example of skeleton seen frontally. Missing keypoints correction



(a) Before preprocessing (b) After preprocessing

Figure 3: Example of skeleton seen from above. Occlusion correction



(a) Before preprocessing (b) After preprocessing

Figure 4: Example of hand skeleton. Keypoint misplacement correction

a few labels to illustrate where the fingers are. *L* refers to left hand, *R* to right hand and the numbers are the ordered fingers where number 1 is the thumb. We can see that in the initial frame the second finger is missing and moreover the third finger is ridiculously long. When we apply the algorithm, the length of the third finger is reduced to an acceptable size, and we generate new keypoints for the second finger. In this case, the position of the second finger is a bit unnatural, but it is definitely desired against having missing keypoints.

6.2. Quantiative results

As described in section 5, we have trained a neural network with two different sets of data. The first model will be trained with the raw data and the second one will be trained with the preprocessed data. Both models have the same architecture and the same number of parameters. The evaluation will be done using a set of preprocessed data unseen by any of the models. This has been done to test them in skeletal models without missing keypoints, which are more admissible and close to the ground truth than a set with missing keypoints.

Despite having trained the models using the Mean Absolute Error loss function, we will compare them by means of a metric called *Percentage of Correct Key-points* (PCK). This metric considers a keypoint correct if the distance between the predicted and the true joint is within a certain threshold. The threshold can be arbitrarily chosen, but it is common to use a certain fraction of a given bone. In order to ease the interpretation of the results, we have taken the bone from the mid-hip to the neck which was scaled to have a length of 1, as stated at the beginning of section 5 . We have taken several values α to compute the fraction of this bone, starting from $\alpha = 0.01$ to $\alpha = 0.5$. There is no need to go higher since we will have a very decent accuracy within this thresholds.

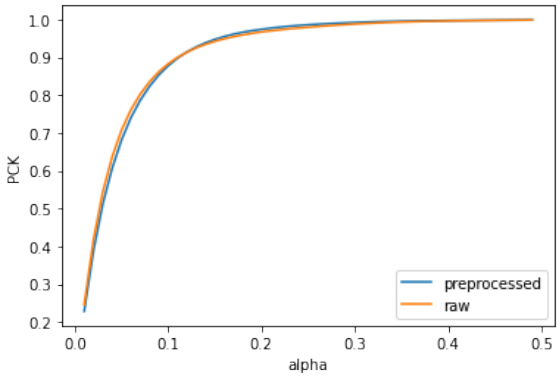


Figure 5: PCK computation of raw and preprocessed models for different values of alpha

α	0.02	0.05	0.1	0.2
Raw	**0.418**	**0.707**	**0.882**	0.967
Prep.	0.389	0.681	0.877	**0.974**

Table 2: PCK of both models for the values 0.02, 0.05, 0.1 and 0.2 of α

Figure 5 illustrates that both models have a very similar behaviour and the differences are minimal. At the end, they are the same model trained with almost the same data. When looking closely, we can observe that if the threshold is very low, the model trained with raw data performs slightly better than the model trained with preprocessed

data. However, if we keep increasing the threshold, there is a point where the model trained with cleaned data is slightly better. From this point, the preprocessed model always outperforms the raw model.

From these results we can extract two outcomes: 1) The model trained with the preprocessed data from our algorithm has less variance than the model trained with the raw data. This means that in general, most of the points predicted by the preprocessed model are close to the target even though for small thresholds, the model trained with raw data performs better. This makes sense since our algorithm brings misplaced or missing points close to their correct position to fulfill the restriction of the length of the bones. Hence, we are reducing the variance of the data and this feature is reflected in our experiment. 2) The model trained with raw data is still able to predict the depth of the coordinates without being influenced by raw input data. The reasons to explain this phenomena may be the low amount of outliers and their systematic "behaviour": when there is a missing point, the x, y and z coordinates have the very same value, so it is easy to predict the third one given the first two. Also, these keypoints are well differentiated from good keypoints (since all three coordinates are equal), so they don't perturb the data for this specific task because the model can guess that a given point is an outlier if x and y coordinates are already outliers. This is added to the fact that the model trained with preprocessed data do has points that have been reconstructed with an automatic algorithm and may not correspond to the nature of real bones.

7. Conclusions and future work

In this paper we perform an analysis of the How2Sign dataset focused on the sequences of skeleton coordinates. We show the distribution and amount of missing points as well as an explanation of the details and characteristics that this data contains. These mainly occur in the hands keypoints, specifically when they are put together or really close, causing occlusions that do not allow the prediction of a keypoint.

We present an implementation of a method for correcting skeleton poses and for interpolating missing skeleton parts when dealing with 3D data. It exhibits realistic results and reduces the number of outliers when applied to the How2Sign dataset. It works well to scale the bones, but it relies in the correctness of the angles between them. This may cause loss of homogeneity in some reconstructions, usually when there are large gaps with missing keypoints.

Finally we propose a deep learning model based on recurrent neural networks to predict the depth of 2D skeletons. This task is used to evaluate two identical models trained with different data. One is trained with raw data and the other one with clean data. The results show that the model trained with raw data does not perform worse than the model trained with clean data even when evaluating it with cleaned data. Nevertheless, the variance of the output is reduced when using the cleaned data. From these results we conclude that for this specific task, the neural network is able to generalize properly and disregard the missing and misplaced points without needing cleaned data.

7.1. Future work

This work can be followed in two different paths. The first one is to adopt the results obtained by the preprocessing algorithm to perform other tasks than 2D to 3D lifting that may be more sensible to missing keypoints. This could include skeletal models synthesis from text / speech, sign language translation or extraction of poses from rgb images. Another interesting area of work would be to pursuit a model to obtain a proper 2D to 3D lifting. This contribution could be very powerful since it would allow to enrich the 2D skeletons contained in the How2Sign dataset. Thus, enhancing the information contained in the sequences and rising the potential of the models trained with it.

8. Acknowledge

I would like to thank my two advisors, Dr Xavier Giró-i-Nieto and PhD Student Amanda Duarte for guiding and teaching me during this project. Their advice and enthusiasm has been very valuable to achieve the final result of this work.

References

[1] World Health Organization 2019. Deafness and hearing loss. https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[3] Dylan Drover, Rohith MV, Ching-Hang Chen, Amit Agrawal, Ambrish Tyagi, and Cong Phuoc Huynh. Can 3d pose be learned from 2d projections alone?, 2018.

[4] Amanda Duarte. Cross-modal neural sign language translation. In Jordi Torres and Xavier Giró i Nieto, editors, *Proceedings of the 27th ACM International Conference on Multimedia - Doctoral Symposium*, Nice, France, 10/2019 2019. ACM, ACM.

[5] Amanda Duarte, Shruti Palaskar, Deepti Ghadiyaram, Kenneth DeHaan, Florian Metze, Jordi Torres, and Xavier Giro i

Nieto. How2sign: A large-scale multimodal dataset for continuous american sign language, 2020.

[6] H. Joo, H. Liu, L. Tan, L. Gui, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3334–3342, 2015.

[7] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis, 2020.

[8] Liang Lu, Naoyuki Kanda, Jinyu Li, and Yifan Gong. Streaming end-to-end multi-talker speech recognition, 2020.

[9] B. Meddins. *Introduction to Digital Signal Processing*. Electronics & Electrical. Newnes, 2000.

[10] Alberto Montes, Amaia Salvador, Santiago Pascual, and Xavier Giro i Nieto. Temporal activity detection in untrimmed videos with recurrent neural networks, 2017.

[11] Pol Pérez-Granero. 2d to 3d body pose estimation for sign language with deep learning. Master's thesis, 2020.

[12] Kaizhi Qian, Zeyu Jin, Mark Hasegawa-Johnson, and Gautham J. Mysore. F0-consistent many-to-many non-parallel voice conversion via conditional autoencoder. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020.

[13] Ramon Sanabria, Ozan Caglayan, Shruti Palaskar, Desmond Elliott, Loïc Barrault, Lucia Specia, and Florian Metze. How2: A large-scale dataset for multimodal language understanding, 2018.

[14] Denis Tome, Chris Russell, and Lourdes Agapito. Lifting from the deep: Convolutional 3d pose estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[15] Lucas Ventura, Amanda Duarte, and Xavier Giro i Nieto. Can everybody sign now? exploring sign language video generation from 2d poses, 2021.

[16] Márton Véges and András Lőrincz. Absolute human pose estimation with depth prediction network, 2019.

[17] Jan Zelinka and Jakub Kanis. Neural sign language synthesis: Words are our glosses. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020.