Processing Kinematics of Nonmanual Markers in R

Jan Bulla, Vadim Kimmelman
University of Bergen
Bergen, Norway
{jan.bulla, vadim.kimmelman}@uib.no

Abstract

Nonmanual markers, such as head and eyebrow movements, eye blinks, and mouth shapes, are an important part of natural languages, both spoken and signed. Recent developments in computer vision have made it possible to extract facial and body landmark positions, as well as head-rotation measures, from 2D video recordings, which can be further processed to analyse the kinematics of nonmanual articulators. In this paper, we present an R-based workflow for processing raw outputs of computer vision toolkits with the goal of producing reliable and interpretable kinematic measurements of nonmanual articulators.

Keywords: nonmanual markers, kinematics, peak detection, smoothing

1. Introduction

Natural languages (both signed and spoken) employ nonmanual articulators (the head, the body, and facial features) to produce linguistically meaningful and relevant information. For sign languages, nonmanual markers have been shown to play a crucial role at all levels of language, including phonological, morphological, lexical, syntactic, semantic, and prosodic functions (Pfau and Quer, 2010). For spoken languages, co-speech gestures also include an important nonmanual component, such as headshakes, nods, tilts, eye blinks, and eyebrow movements, which are used to express meaning and regulate conversations (see e.g. Wagner et al., 2014; Nota et al., 2021).

Until a few years ago, measuring nonmanual markers in signed and spoken languages involved using a motion-capture system (Hadar et al., 1983, 1985; Wagner et al., 2014; Puupponen et al., 2015). Motion-capture systems require expensive equipment, and output large amounts of data which require extensive post-processing and statistical analysis (Puupponen et al., 2015), which, in practice, limits the length of recordings that can be made and analysed using this approach. Furthermore, most existing datasets of natural sign languages consist of 2D video recordings.

However, recently computer vision tools such as OpenPose (Cao et al., 2018), OpenFace (Baltrusaitis et al., 2018), and MediaPipe (Lugaresi et al., 2019) have made it possible to extract facial and body landmarks and head-rotation estimates from 2D video, enabling analysis of the kinematics of head movements (and other manual and nonmanual movements). Several papers describing methods for using these computer vision tools for measuring kinematics have been published, both for sign languages (Börstell, 2023) and co-speech gestures (Trujillo et al., 2019). The EnvisionBox project (https://envisionbox.org) is an especially useful resource, producing tutorials on leveraging various computer vision tools for linguistic kinematic analysis, see specifically Trujillo and Pouw (2021); Pouw (2024).

While some studies have also been published using these tools specifically for analysing nonmanual markers (Kimmelman et al., 2020; Bauer et al., 2024), several methodological challenges remain. Specifically, we identify:

Dealing with noisy measurements. Any measurement instrument produces noise, and this is especially true for computer vision tools that are not developed specifically for kinematic analysis of sign languages or gestures. Consequently, it is necessary to separate the noise from the relevant motion in the outputs of these models. For instance, in Figure 1 we can compare the raw outputs of OpenFace yaw measurements for a single headshake in Brazilian Sign Language (Libras) with an inferred smooth motion of the head.

Peak detection. Many nonmanual movements are periodic (such as headshakes), or have relevant peaks and plateaus that researchers need to locate and measure (such as head nods). For instance, in Figure 1 the direct interpretation of local minima and maxima as peaks produces a large number of peaks that do not reflect real head turns.

Calculating measurements. Numerous kinematic measures can be calculated for any nonmanual marker (e.g. range of motion, velocity, frequency, etc.). Furthermore, these measures can be summarized in different ways for individual motion events. Figure 1 shows that the measures derived from the data crucially depend on the smoothing and peak detection steps. For example, range of motion in the raw data equals 8.7 degrees, and in the smoothed data it is 6.2 degrees. In addition, the number of peaks – and thus the derived peak frequency – differs markedly.

Proceedings of the 12th Workshop on the Representation and Processing of Sign Languages, pages 61–70
16 May 2026.

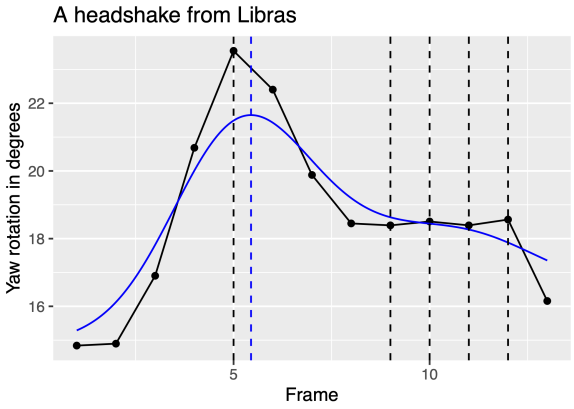

Figure 1: Visualization of yaw (in degrees) for a single headshake from Libras, see https://osf.io/krejq for the video. Black dots and line: raw measurement; blue line: inferred movement after smoothing using kernel regression; black dashed lines: raw local maxima and minima; blue dashed line: inferred peak after smoothing

In this paper, we present a solution addressing these challenges, implemented as a small set of R scripts. These are openly available at https://doi.org/10.17605/OSF.IO/QFG98, together with detailed documentation, a demo script, and the data used for our analyses. The demo script introduces the functions presented in Section 2 and may serve as a template for adapting the workflow to other datasets. The scripts require a table with raw outputs of measurement instruments as input. First, they apply smoothing to address the noisy measurement problem; next, they apply a peak-detection algorithm that identifies and classifies peaks into various categories; finally, they produce a large number of kinematic measures for each motion event in the original dataset. We have tested this approach using outputs of OpenFace (Baltrusaitis et al., 2018) and MediaPipe (Lugaresi et al., 2019), but it is compatible with any instrument outputting per-frame measurements of an articulator.

In Section 2 we present our scripts and briefly explain the underlying statistical theory of the developed functions. In Section 3 we present a case study applying this approach to the study of kinematics of negative headshakes in Libras. Section 4 discusses limitations of the approach and concludes the paper.

2. Presentation of the Script

This section introduces the three R (R Core Team, 2025) functions `smooth.curves.est`, `smooth.curves.classif`, and `smooth.curves.res.prep` that together allow users to analyse head-movement time-series data. The functions are applied sequentially as follows.

Step 1: The function `smooth.curves.est` estimates a smoothing curve for a given sequence of observations and identifies the resulting peaks. It provides a graphical display of the fitted curve, the peak locations, and (optionally) the first and second derivatives of the smoothing curve.

Step 2: The function `smooth.curves.classif` classifies the inferred peaks from Step 1 according to user-defined criteria. It can also display the classification graphically (optional). In addition, `smooth.curves.classif` generates various quantitative measures based on the inferred peaks as well as basic statistical measures of the underlying data.

Step 3: The function `smooth.curves.res.prep` consolidates the output from the function `smooth.curves.classif` for one or several sequences of observations into a format that is more suitable for downstream analysis and visualisation.

Note that the computationally most demanding components occur in Step 1. In the following, we provide a description of each function, including a brief summary of the statistical foundations required for Step 1.

2.1. Smoothing

The first step carries out the estimation of smoothing curves for a set of input values provided by the user. In principle, these input values can consist of a comparatively general set of pairs of x- and y-values. In the application considered, we assume that the x-values correspond to time index measured in frames, and the y-values measure an angle of movement or a distance between articulators (e.g. the eyebrows and the eye line). We implemented two different approaches for estimating the smoothing curve: kernel regression and smoothing splines.

Kernel regression belongs to the class of non-parametric statistical methods; the underlying theory for continuous data dates back to Nadaraya (1965) and Watson (1964). It estimates the relationship between variables by calculating a weighted average of nearby data points, using a so-called kernel function to determine the weights. This allows kernel regression to model complex, non-linear patterns without assuming a specific functional form, thus constituting a more flexible alternative to linear regression for smoothing data and making predictions. Roughly speaking, kernel regression is similar to a moving average but uses data-driven weighting by placing more emphasis on points closer to the prediction location. The weighting process is

controlled by the crucial bandwidth parameter (usually denoted by h). What follows is a brief technical formulation of the kernel regression estimator. Let $m(x) = \mathbb{E}[Y \mid X = x]$ denote the (unknown) regression function. A standard formulation is the Nadaraya-Watson (local constant) estimator, which at a target point x takes the form

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} K_h(x - X_i) Y_i}{\sum_{i=1}^{n} K_h(x - X_i)} = \sum_{i=1}^{n} w_i(x; h) Y_i,$$

$$\text{with } w_i(x; h) = \frac{K_h(x - X_i)}{\sum_{j=1}^{n} K_h(x - X_j)}$$

$$\text{and } K_h(u) = \frac{1}{h} K\left(\frac{u}{h}\right).$$

Here, the kernel K is a weight function and $w_i(x; h)$ are the normalised kernel weights. The latter are non-negative, sum to one, and allocate larger weight to observations X_i closer to x (as determined by the kernel K and bandwidth $h > 0$). Then, $m(x)$ corresponds to the kernel estimator, i.e., the estimated conditional mean of Y at x. The bandwidth parameter $h > 0$ controls how rapidly weights decay with distance from x. This parameter determines the degree of smoothing and controls what is known as the bias-variance trade-off. More precisely, the bandwidth h determines how far the kernel's influence extends, thus balancing local data fitting (small h leading to low bias and high variance) with overall smoothness (large h leading to high bias and low variance). The aim of bandwidth selection is to find an optimal balance of bias and variance for the regression curve.

For carrying out kernel regression in R, we rely on the function `npreg` from the package `np` (Hayfield and Racine, 2008). Bandwidth selection is implemented in the function `npregbw`, which allows choosing between a local linear and a local constant (i.e. Nadaraya-Watson) estimator. Moreover, `npregbw` carries out cross-validation to assess the optimal bandwidth via two criteria: Kullback-Leibler or least-squares. We refer the reader to Li and Racine (2004) and Racine and Li (2004) and the references therein for further details on the theoretical background of both the available estimators and bandwidth selection.

For our data, we observed that both cross-validation criteria led to relatively strong smoothing, which turned out to be slightly weaker for the Kullback-Leibler criterion. Therefore, we selected this criterion for our bandwidth selection. Regarding the choice of estimator, we could not determine a clear preference between the local linear or the local constant version: depending on the data analysed, sometimes one or the other showed favourable results. Nevertheless, we observed that occasionally one, sometimes both estimators resulted in more or less severe over-smoothing (i.e., h tends towards infinity). Hence, the function `smooth.curves.est` internally carries out bandwidth selection with both estimators and selects the smaller of the two bandwidths. This constitutes a pragmatic approach, because we did not observe under-smoothing in any of our test datasets. Finally, it is noteworthy that the function `npregbw` possesses several additional arguments that enable fine-tuning of the bandwidth selection process. For our applications, we did not find any relevant effect of the kernel type and therefore relied on the default Gaussian kernel. Moreover, we set the number of sets of random initial points for starting the process of finding extrema of the cross-validation function to 25 for numerical stability – although a lower value might suffice if computational power is scarce.

Smoothing splines belong to the class of non-parametric regression methods and provide a flexible approach to estimating smooth functional relationships between variables. Their theoretical foundations date back to early work on spline functions and penalised least squares, notably by Reinsch (1967) and Wahba (1990). Smoothing splines estimate the regression function by fitting a smooth curve that balances fidelity to the observed data with a penalty on excessive curvature. This balance allows smoothing splines to capture complex, non-linear patterns without requiring the specification of a particular functional form, making them a powerful alternative to parametric regression techniques for smoothing and prediction.

Conceptually, smoothing splines differ from local averaging methods such as kernel regression in that they are global smoothers. Rather than computing predictions based on local neighbourhoods, smoothing splines determine a single smooth function over the entire domain of the predictor. The degree of smoothness is governed by a smoothing parameter, which controls the trade-off between goodness of fit and smoothness of the estimated curve. More formally, the smoothing spline $\hat{f}$ is defined as the function that minimises

$$\hat{f} = \arg\min_{f} \underbrace{\sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2}_{\text{data-fit term}} + \lambda \underbrace{\int \left(f^{(3)}(t)\right)^2 dt}_{\text{roughness penalty}},$$

i.e. $\hat{f}$ denotes the estimated regression function (the optimiser of this criterion). The data-fit term measures how closely the fitted function matches the observed responses at the sampled covariate values, while the roughness penalty quantifies the rate of change of curvature (wiggliness) of f through the integrated squared third derivative. The scalar $\lambda \geq 0$ is the smoothing (regularization) parameter that weights the roughness penalty relative to the data-fit term: larger λ yields smoother fits; smaller λ prioritizes fidelity to the data.

In R, we implement smoothing splines using the function `ss` from the package `npreg` (Helwig, 2020, 2024). In our case, this function fits a quintic smoothing spline via penalised least squares, where roughness is quantified by $\int (f^{(3)}(t))^2 dt$ and the smoothing parameter λ weights this penalty relative to the data-fit term. By default, λ is chosen by generalised cross-validation (GCV) to balance fit and smoothness in a data-driven manner. Unlike standard leave-one-out cross-validation, GCV is an analytical approximation that exploits the linearity of smoothing splines: instead of repeatedly refitting the model, it adjusts the residual sum of squares using the smoother's effective degrees of freedom to estimate predictive performance efficiently. An alternative approach is to supply a fixed λ or to target an effective degrees of freedom value for more direct control over smoothness. In our applications, after comparing several selection methods for λ, we relied on the default GCV selection because we found that it yielded stable and interpretable fits without requiring manual tuning. During our selection process, we also identified the Bayesian Information Criterion (BIC) as second-best performing method. Overall, we observed occasional under-smoothing regardless of the selection method.

We also evaluated various alternative nonparametric approaches for estimating smooth functions in the context of our applications. These included local polynomial regression via base R's `loess` function, which implements LOESS (Cleveland and Devlin, 1988). We selected the span via a custom implementation of k-fold cross-validation (with $k = 5$ and $k = 10$, respectively, in testing), using polynomial degree two, and relied on locally weighted least squares. We ultimately did not pursue this approach further due to a tendency to under-smooth on our datasets. We additionally considered an alternative classical kernel smoothing via base R's `ksmooth` function, fixing bandwidths manually across a grid to assess sensitivity. However, no single bandwidth yielded consistently satisfactory results across our test datasets. Finally, we studied smoothing within the generalised additive model (GAM) framework: (i) with the `gam` function from the `gam` package (Hastie, 2025), following the original GAM formulation (Hastie and Tibshirani, 2017); and (ii) with the similarly named gam function from the `mgcv` package, which fits penalised regression splines to balance flexibility and smoothness (Wood, 2004, 2017). In our experiments, the `gam` package used its default smoothing spline terms, while `mgcv` was evaluated with both thin plate regression splines and P-splines. Neither of the GAM implementations was ultimately pursued due to a tendency to over-smooth for our applications.

After estimating the smoothing curve, we identify the locations of its peaks (i.e., local minima and maxima in the mathematical sense) by numerically finding the roots of the approximate first derivative. The classification of each peak is determined by the sign of the approximate second derivative at that point. We obtain both derivatives by numerical approximation, more precisely, using the classical backward finite-difference method, where the user can control the approximation accuracy. After inferring the extrema, the function `smooth.curves.est` optionally produces a plot of the input data overlaid with the estimated smoothing curve and the inferred peaks. In addition, the first- and second-derivative curves can be added to the plot by the user. Figure 2 shows the estimated smoothing curves for our single headshake from Libras, using kernel regression (top panel) and smoothing splines (bottom panel).

Finally, the function returns the estimation results of the kernel regression or smoothing spline, respectively, along with additional quantities relevant for further analysis. Note that various consistency checks are performed before estimating the smoothing curve: (i) the number of frames must match the number of movement angles; (ii) the number of observations must satisfy $n \geq 5$. In addition, the presence of missing observations is recorded – identified by non-equidistant (gapped) frame indices. Sequences failing checks (i)–(ii) are excluded from fitting. However, missingness is merely recorded and does not constitute a failure.

2.2. Peak Detection

The function `smooth.curves.classif`, implemented for Step 2, requires similar input to Step 1: (1) paired x- and y-values, corresponding to frames and movement angles, respectively; (2) the estimation-results list from Step 1. The estimation results also contain certain control parameters from Step 1. These have to be supplied again because some computations are repeated. While this is not computationally the most efficient approach, it significantly reduces the storage size of the results object produced by Step 1. Core functionality of Step 2 is the classification of the inferred peaks from Step 1 according to user-defined criteria. This classification is based on the principle that the label assigned to a specific peak depends on the magnitude of movement since the previous peak. Depending on this movement magnitude, the peak falls into the category 'relevant' for a sufficiently large enough movement, 'hold' for a sufficiently small movement, and 'not classified' for the remaining cases. The necessary amount of movement can be quantified either in absolute (e.g. degree) units or in relative terms, i.e., as a proportion of the total movement range of all observations in the sample. Example: suppose the peak under consid-

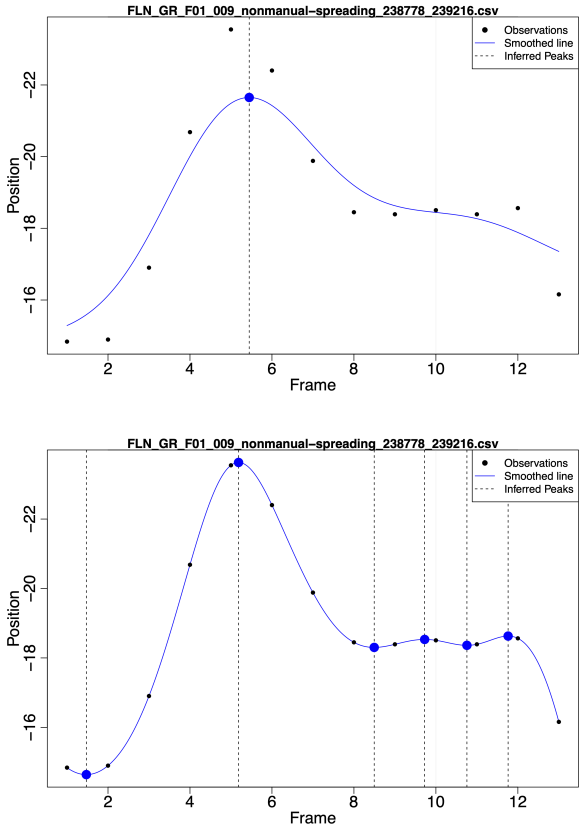

Figure 2: Smoothing example using kernel regression (top) and smoothing splines (bottom).

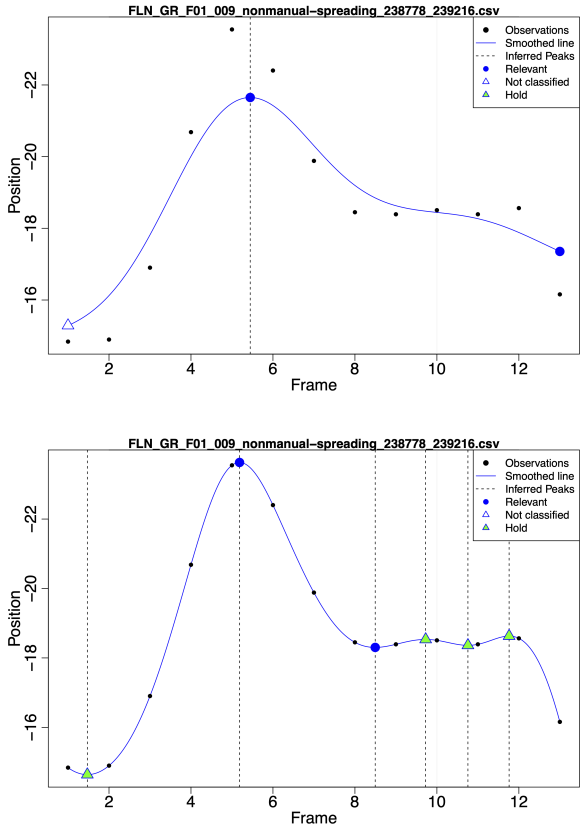

Figure 3: Peak detection example using kernel regression (top) and smoothing splines (bottom).

eration is a maximum. Then the previous peak is a minimum, and we consider the difference between these two peaks for classifying the maximum. If this difference equals 15 degrees and the threshold for a relevant peak was set at 10 degrees, the maximum is classified as a 'relevant' peak. However, if the difference equals only 2 degrees and the threshold for a hold equals 5 degrees, we classify the maximum as a 'hold'. All observed movements between 5 and 10 degrees result in a 'not classified' maximum.

The first inferred peak and the border points require special attention. For the first inferred peak, no previous peak is present by definition. Therefore, the first available observation at the left border serves as the reference point for assigning a label. Moreover, border points, i.e., the first and last observed values of the sequence under consideration, can be set to be peaks manually by the user via the argument `border.check`. This argument allows labelling a border point as a peak if no other inferred peak is available within a user-defined range (set via the argument `border.frames`) following the first or preceding the last observation, respectively. By default, a border peak at the very beginning of a sequence of observations falls into the 'not classified' category, because no previous movement data are available. This behaviour can, however, be overridden by the user (via the argument `border.classif.first`). When enabled, the first border peak is classified based on the movement magnitude up to the first subsequent peak. Consequently, the border peak at the first observation and that subsequent peak then receive identical labels, as both classifications are based on the same movement magnitude. On the other hand, the classification of a border peak at the end of an observation sequence follows the same logic as for all other peaks inferred from the smoothing curve. While border-point treatment may be of reduced importance for very long sequences of observations, it may be important when only a few observations are available.

Figure 3 illustrates the peak classification for the two previously estimated smoothing curves. In the top panel (kernel regression), only one peak was inferred via first-derivative roots, near Frame 6. In the absence of other roots near the boundaries, both the first and last observations are treated as border peaks. Following the logic described above, the peak at Frame 1 is labelled 'not classified' by default. If the override is enabled, this peak would be labelled 'relevant' because the movement from Frame 1 to the subsequent peak

near Frame 6 exceeds the relevance threshold. The second peak in the middle is classified as 'relevant' because of a sufficiently large magnitude of movement since the first peak. The same is true for the third peak at the right border. In the bottom panel (smoothing splines), no border peaks are inserted. Then, after a 'hold' at the beginning, two 'relevant' peaks follow. Afterwards, a period with three holds occur between approximately Frames 9 and 12.

Similar to Step 1, the function `smooth.curves.classif` displays the classification results as a figure. Moreover, it returns a list object with various quantities, which are mostly linked to the peaks in the sample but also include basic descriptive statistics of the underlying data (see following section). If no estimation results are available, only the descriptive measures are returned.

2.3. Measurements

We detail the output of the function `smooth.curves.classif` in the following. The returned quantities are heterogeneous and can be grouped into six categories. A brief description of each category follows.

A. Basic information: the series name and, when available, the frame rate.

B. Peak-based measures: quantities related to peaks inferred by the smoothing curve and border peaks (e.g., number and location of peaks, movement range between adjacent peaks).

C. Movement-based measures - standard: descriptive measures related to the observed movement. These are computed twice, from a) the originally observed positions and b) the corresponding positions inferred by the smoothing curve (e.g., average position, range, maximum frame-to-frame velocity).

D. Movement-based measures – robust: statistically robust counterparts to the measures presented under Item C, again computed from both the observations and the smoothing-curve positions (e.g., median position, 80%/90% interquantile range, 90% quantile of the absolute frame-to-frame velocity).

E. Other measures: measures mainly relying on either the observations or the inferred smoothing curve, that do not fit Items C-D (e.g., number of missing observations, predicted position values).

F. Control measures: quantities related to script execution and the estimation procedure.

We provide a detailed per-quantity description in the function documentation available in the repository at `https://doi.org/10.17605/OSF.IO/QFG98`.

The total number of quantities (of varying dimension) returned by the function `smooth.curves.classif` exceeds 50. When multiple sequences of observations are processed, it is useful to post-process the resulting output into a more user-friendly format. This third step is performed by the function `smooth.curves.res.prep`, which takes as input a list with output from Step 2 as list elements. The function returns a list with six elements. We describe these elements below.

1. `res.mat.obs`: a matrix collecting all single-valued descriptors (one row per input sequence; one column per quantity).
2. `res.mat.peaks`: a matrix with peak-related quantities for all series. It includes quantities derived from all peaks and, separately, from 'relevant'-labelled peaks.
3. `res.list.peaks`: a list of matrices, where each matrix contains peak-specific quantities resulting from one of the sequences of observations analysed.
4. `res.list.peaks.rel`: analogous to Item 3, this element consists of a list of matrices. However, it is based only on 'relevant' peaks.
5. `res.list.pred.obs`: a list of numeric vectors with predicted positions from the inferred smoothing curve at each frame of the original sequence.
6. `res.fail`: a logical vector indicating whether Step 1 (estimation) failed for a given sequence.

3. A case study

In the previous section, we provided the details of our scripts. In this section, we provide a short but realistic example of how they can be used to analyse kinematic properties of negative headshakes in Brazilian Sign Language (Libras). The data come from Corpus de Libras (Quadros et al., 2020), and were annotated for another study (Figueiredo, 2026a,b)

The dataset contains 323 individual negative headshakes produced by 10 signers of Libras. The headshakes were annotated according to a number of linguistic variables, including *spreading*: whether they co-occur with only one manual sign or spread over at least two manual signs. The research question was whether spreading and non-spreading headshakes are kinematically different. We processed the video files with OpenFace (Baltrusaitis et al., 2018) and used `pose_Ry` converted to degrees to analyse headshake kinematics, i.e. yaw measurements. More specifically, we applied the scripts developed in the current study with default settings for kernel regression and smoothing

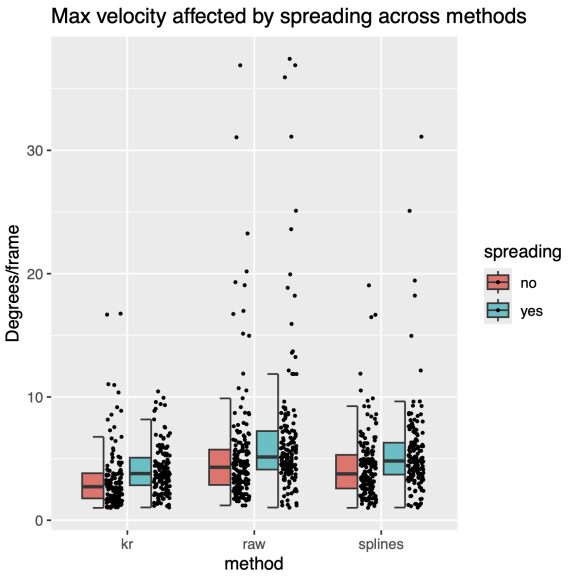

Figure 4: Maximal velocity by spreading status in raw data, spline-smoothed, and kernel-regression-smoothed data.

splines.

For the purposes of illustration, we focus on a small number of measurements. Figure 4 summarises how maximal velocity varies by spreading. We observe that spreading headshakes have a higher maximal velocity than non-spreading headshakes both in raw data, and in smoothed data (for splines and kernel regression). However, we also observe that the raw data exhibit the highest number of potential outliers, followed by splines, with kernel regression producing the fewest. This indicates that kernel regression produces, on average, smoother fits than spline smoothing, which is also what we observed for other datasets (not shown here). Thus, while the effect of spreading is observed with and without smoothing, it will be estimated more stably when smoothing is applied, because smoothing mitigates measurement noise and spurious extrema.

Figures 5 and 6 show measures that cannot be directly derived from the raw data, as discussed above, namely the number of peaks (reflecting the number of head turns) and the peak frequency (peaks per frame). We observe that both splines and kernel regression-based methods show that spreading headshakes exhibit a higher number of peaks but a lower frequency of peaks. This is not surprising, as spreading headshakes are typically longer and thus we expect the head to be able to turn more times. By contrast, for non-spreading headshakes – since they are relatively short – the frequency of peaks has to be higher in order for the head to be able to conduct perceivable turns. For these measures, we again observe that the spline-based method produces less smoothing, and thus a wider distribution of values.

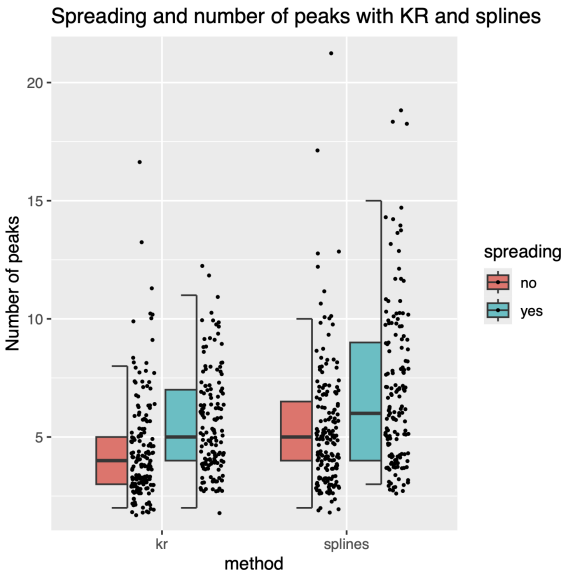

Figure 5: Number of peaks by spreading status in the data smoothed with splines, and with kernel regression, respectively.

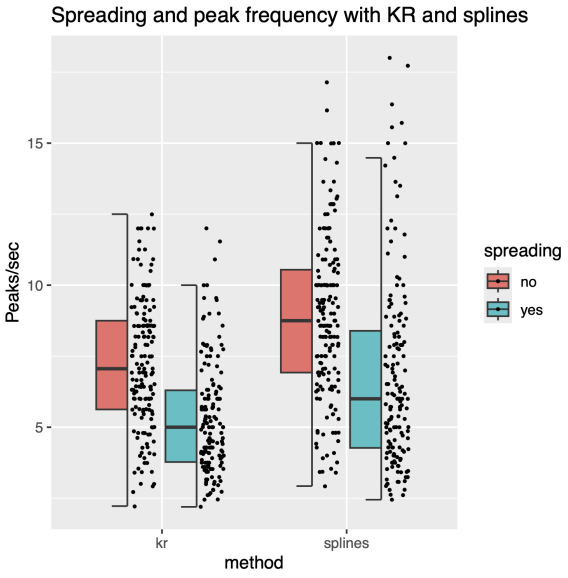

Figure 6: Frequency of peaks (peaks per second) by spreading status in the data smoothed with splines, and smoothed with kernel regression, respectively.

It is not possible to determine which smoothing approach performs better for this dataset, as no ground-truth measurements of head movements are available. Instead, visual inspection of individual movements with smoothing and peak detection should be used to ensure reasonable results, which

is why the scripts also output these visualizations. It seems, however, reasonable to choose spline smoothing if one believes that the outputs of the CV instrument used are stable and largely free of extremely faulty measurements, as this method preserves most of the variation present in the raw data. In contrast, if the CV instrument is relatively unreliable and noisy, kernel regression may produce superior results.

For other studies that fully employ the presented smoothing and peak-detection approach to negative headshakes, see Kimmelman et al. (2026). For an application to eye blinks, see Susman (2026).

4. Conclusions and Limitations

To sum up, in this paper we propose a workflow for processing outputs of computer vision tools applied to nonmanual markers in sign languages and co-speech gestures. The scripts we developed offer a solution to three crucial problems when processing such data, namely, smoothing, peak detection, and the calculation of measurements. The full pipeline is implemented in R (R Core Team, 2025) and is openly available and documented in the repository at https://doi.org/10.17605/OSF.IO/QFG98.

The proposed approach, however, has some clear limitations. First, while the goal is fully automated processing, it still requires expert input for calibration. For the smoothing part, we have tested the methods using several datasets of head movements, and decided on the best approaches using our experience in sign-language linguistics. However, these datasets are not representative of the breadth of signed and spoken languages, nor of the many different types of nonmanual movements. Nevertheless, it is possible – if needed – to modify the methods and to implement other methods (beyond the ones we tested) using the same general approach, which, in principle, should be applicable to other movement types of nonmanual articulators.

For the peak-detection part, expert input is even more crucial, as the peak classification into relevant peaks and holds bases entirely on expert-defined thresholds. In fact, in the published version of the script, we use a simple relative threshold, which does not perform perfectly even on all sequences belonging to the dataset that we tested. Hence, more complex approaches might be required for more robust peak classification in future.

We plan to explore automatic, data-driven peak classification in future work on this approach. However, a central challenge is the need for creating sufficiently large, well-annotated datasets in which the true movement patterns are known with high reliability. The construction of such datasets does not constitute a trivial task, especially for low amplitude movements.

Another limitation of this approach is that it was developed and tested on sequences of approximately 5-50 frames in length. It is not reliably applicable to shorter sequences due to mathematical and algorithmic constraints of some of the components; in particular, kernel regression is less forgiving than smoothing splines for very short sequences. For longer sequences, there is no inherent restriction, but we have not tested the robustness of the smoothing, and – especially for peak detection – additional calibration may be required.

5. Acknowledgments

Funded by the European Union (ERC, NONMANUAL, project number 101039378). Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

6. Author contributions

Vadim Kimmelman and Jan Bulla conceptualised the study. Jan Bulla developed the software, with input and testing from Vadim Kimmelman. Vadim Kimmelman conducted the case study. Both authors wrote the paper.

7. Data availability

The scripts described in this paper, together with a demo script and a test dataset, and the script used to produce the figures, are available at https://doi.org/10.17605/OSF.IO/QFG98.

8. Bibliographical References

Tadas Baltrusaitis, Amir Zadeh, Yao Chong Lim, and Louis-Philippe Morency. 2018. Openface 2.0: Facial behavior analysis toolkit. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 59–66. IEEE.

Anastasia Bauer, Anna Kuder, Marc Schulder, and Job Schepens. 2024. Phonetic differences between affirmative and feedback head nods in German Sign Language (DGS): A pose estimation study. *PLOS ONE*, 19(5):e0304040.

Carl Börstell. 2023. Extracting Sign Language Articulation from Videos with MediaPipe. In *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 169–178, Tórshavn, Faroe Islands. University of Tartu Library.

Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2018. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008*.

William S. Cleveland and Susan J. Devlin. 1988. Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting. *Journal of the American Statistical Association*, 83(403):596–610.

Lorena Figueiredo. 2026a. A corpus-based study on negation in Brazilian Sign Language.

Lorena Figueiredo. 2026b. Negative Nonmanual Markers in Brazilian Sign Language (Libras). Zenodo.

U. Hadar, T. J. Steiner, and F. Clifford Rose. 1985. Head movement during listening turns in conversation. *Journal of Nonverbal Behavior*, 9(4):214–228.

U. Hadar, T.J. Steiner, E.C. Grant, and F.Clifford Rose. 1983. Kinematics of head movements accompanying speech during conversation. *Human Movement Science*, 2(1-2):35–46.

T.J. Hastie and R.J. Tibshirani. 2017. *Generalized Additive Models*, 1 edition. Routledge.

Trevor Hastie. 2025. *gam: Generalized Additive Models*.

Tristen Hayfield and Jeffrey S. Racine. 2008. Nonparametric Econometrics: The **np** Package. *Journal of Statistical Software*, 27(5).

Nathaniel Helwig. 2020. Multiple and Generalized Nonparametric Regression. In *SAGE Research Methods Foundations*. SAGE Publications Ltd, 1 Oliver's Yard, 55 City Road, London EC1Y 1SP United Kingdom.

Nathaniel E. Helwig. 2024. *npreg: Nonparametric Regression via Smoothing Splines*.

Vadim Kimmelman, Anastasia Bauer, Carl Börstell, Jan Bulla, Bux Allah, Laurence Crettenand, Lorena Figueiredo, Anna Kuder, Hannah Lutzenberger, Marloes Oomen, and Roland Pfau. 2026. Kinematics of negative headshake in seven sign languages.

Vadim Kimmelman, Alfarabi Imashev, Medet Mukushev, and Anara Sandygulova. 2020. Eyebrow position in grammatical and emotional expressions in Kazakh-Russian Sign Language: A quantitative study. *PLOS ONE*, 15(6).

Qi Li and Jeffrey S. Racine. 2004. Cross-validated local linear nonparametric regression. *Statistica Sinica*, 14(2):485–512.

Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. 2019. MediaPipe: A Framework for Building Perception Pipelines.

É. A. Nadaraya. 1965. On Non-Parametric Estimates of Density Functions and Regression Curves. *Theory of Probability & Its Applications*, 10(1):186–190.

Naomi Nota, James P. Trujillo, and Judith Holler. 2021. Facial Signals and Social Actions in Multimodal Face-to-Face Interaction. *Brain Sciences*, 11(8):1017.

Roland Pfau and Josep Quer. 2010. Nonmanuals: their prosodic and grammatical roles. In Diane Brentari, editor, *Sign Languages*, pages 381–402. Cambridge University Press, Cambridge.

Wim Pouw. 2024. Wim Pouw's EnvisionBOX modules for social signal processing.

Anna Puupponen, Tuija Wainio, Birgitta Burger, and Tommi Jantunen. 2015. Head movements in Finnish Sign Language on the basis of Motion Capture data: A study of the form and function of nods, nodding, head thrusts, and head pulls. *Sign Language & Linguistics*, 18(1):41–89.

Ronice M. de Quadros, Deonísio Schmitt, Juliana T. Lohn, and Tarcísio de A. Leite. 2020. Corpus de Libras.

R Core Team. 2025. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Jeff Racine and Qi Li. 2004. Nonparametric estimation of regression functions with both categorical and continuous data. *Journal of Econometrics*, 119(1):99–130.

Christian H. Reinsch. 1967. Smoothing by spline functions. *Numerische Mathematik*, 10(3):177–183.

Margaux Susman. 2026. Phonetic properties of blinks in French Sign Language.

James P. Trujillo and Wim Pouw. 2021. Kinematic Feature Extraction for Motion Tracking Analysis.

James P. Trujillo, Julija Vaitonyte, Irina Simanova, and Asli Özyürek. 2019. Toward the markerless and automatic analysis of kinematic features: A toolkit for gesture and movement research. *Behavior Research Methods*, 51(2):769–777.

Petra Wagner, Zofia Malisz, and Stefan Kopp. 2014. Gesture and speech in interaction: An overview. *Speech Communication*, 57:209–232.

Grace Wahba. 1990. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics.

Geoffrey S. Watson. 1964. Smooth Regression Analysis. *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, 26(4):359–372.

Simon N Wood. 2004. Stable and Efficient Multiple Smoothing Parameter Estimation for Generalized Additive Models. *Journal of the American Statistical Association*, 99(467):673–686.

Simon N. Wood. 2017. *Generalized Additive Models: An Introduction with R*, 2 edition. Chapman and Hall/CRC.