Speaker Diarisation: Combining Pyannote With Directional Methods

Chris Lienaerts
AI Researcher, SpraakLab B.V.

March 2026

Abstract

Speaker diarisation aims to determine “who spoke when” in multi-speaker audio recordings. Conventional diarisation systems rely solely on speech characteristics derived from the audio signal, whereas directional methods exploit microphone array geometry to infer speaker position. In this work, we compare acoustics-based diarisation using pyannote with two directional approaches developed on the Shure MXA910 and miniDSP UMA-16 v2 microphone arrays. We further propose a hybrid method that integrates multi-channel energy vectors into the clustering stage of the pyannote pipeline. Experiments on controlled multi-speaker recordings derived from the IFADV corpus demonstrate that directional methods can achieve substantially lower speaker error rates compared to pyannote. The proposed hybrid approach consistently improves performance over the standard pyannote pipeline.

Contents

1 Introduction

1.1 What is Diarisation?

Automatic Speech Recognition (ASR) extracts valuable information from speech by producing a transcription of every word uttered. However, transcriptions alone may leave one wondering which words were spoken by whom in a multi-speaker recording. This is the task of speaker diarisation, which links each segment of speech to its respective speaker. When performing diarisation on an audio recording we are not trying to transcribe what was said, but rather who was speaking at which time. We define diarisation as follows:

> *The process of segmenting an audio stream into sections according to the identity of the speaker.*

A commonly used phrase when describing diarisation is *"Who spoke when?"*. Although this might imply that diarisation also encompasses speaker recognition, it should be understood that speaker diarisation is *not* the same as speaker recognition, nor does speaker diarisation result in speaker recognition. When performing diarisation, we only segment the audio into sections attributed to an arbitrary speaker identifier. Speaker recognition can be achieved using speaker verification models not mentioned here, provided that a labelled speaker model-created from an audio sample, labelled with the speaker's identity-is available.

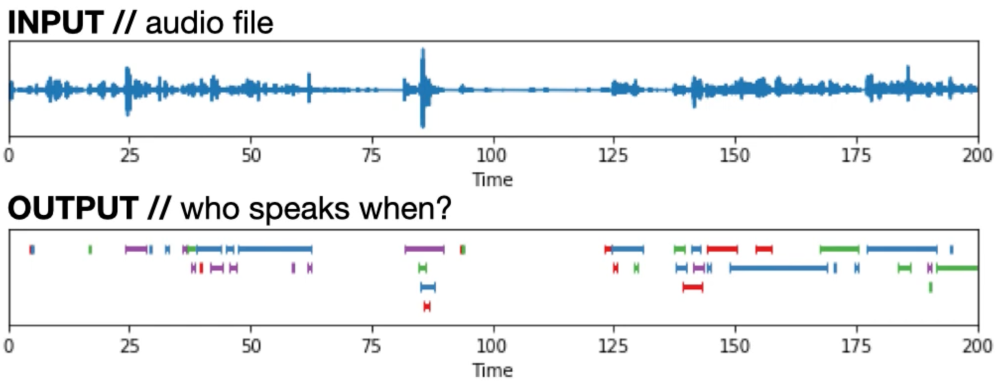


Figure 1: *Visual Representation of Diarisation Task: Audio is Segmented According to Speaker Identities [5]*

2 Background

In this section, some fundamental principles relevant to the diarisation methods presented in this work are discussed. This is not intended to be a comprehensive explanation of diarisation systems as a whole, but rather an overview of the specific terminology necessary to explain and compare the methods described here. Other diarisation approaches may employ a different combination of these techniques or entirely different methods.

We also make a distinction between two main categories of speaker diarisation methods:

1. Acoustic methods: These methods rely solely on the audio signal, leveraging differences in voice characteristics between speakers to perform diarisation. Diarisation systems of this kind are widely available and most are fairly easy to start using, without the need for expert knowledge or training data. A lot of open-source options exist, usually built with Python. We highlight pyannote as an example of an acoustic diarisation system in Section 4.1.

2. Directional methods: These methods determine the direction of an audio source by utilising the directional properties of the mix of sound waves, as registered by multiple microphones. These methods are not widely available and require expert knowledge to develop. They also need more controlled environments but are able of providing far higher accuracy. We highlight our own efforts in developing these methods in Section 4.2.

2.1 Sliding Window Technique

Audio is usually not processed in its full length for diarisation, but rather in segments. This is achieved by employing a sliding window as seen in Figure 2, which divides the audio into small, usually overlapping, segments or *windows*.

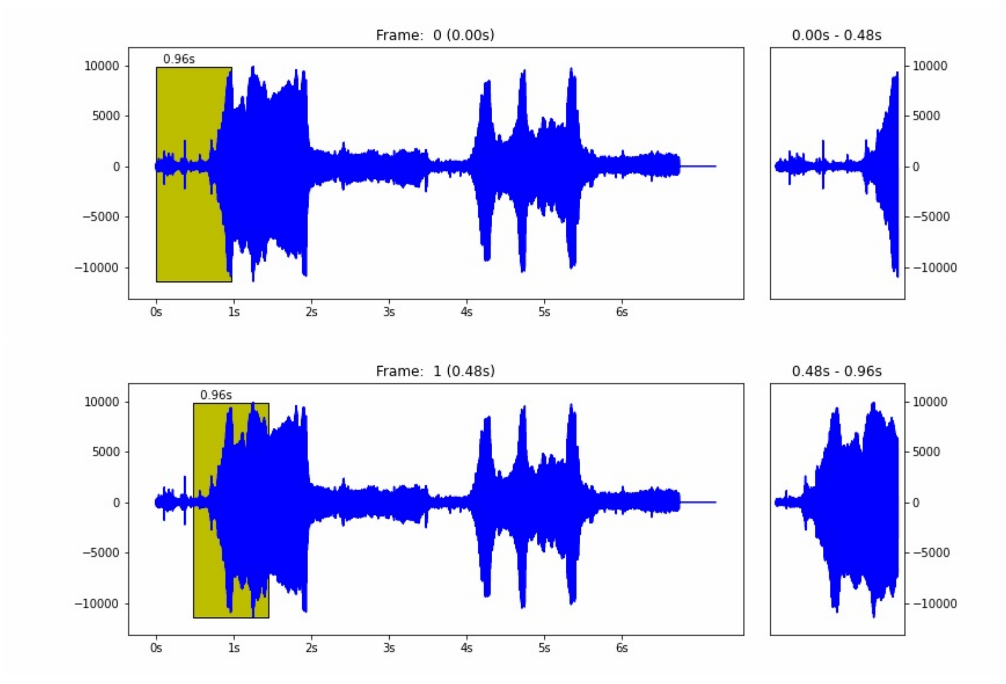


Figure 2: *Sliding Window Technique Used on an Audio Signal [10]*

Two main parameters can be manipulated: the window length and step size (the amount by which the window is shifted forward with every iteration).

A longer window length reduces the noise in the computed embeddings, but increases the risk of capturing more than one speaker in a window which can reduce the accuracy of speaker change annotation. Conversely, if the window length is too short, it may capture too little audio to reliably reflect distinct voice characteristics.

A large step size decreases the number of windows needed, and thus computational complexity, but risks sacrificing granularity in the diarisation results. The step size is usually smaller than the window length to include some overlap with the previous window.

This technique is used in many applications. For example, by iterating through the entire audio recording in this manner, an embedding of each segment can be compared to every speaker embedding to determine where a specific speaker is present in the recording. This technique is also employed in directionality-based diarisation methods, as described in section 4.2, to determine an audio source location for each segment, hereby creating audio source location data points across the entire recording.

2.2 Embeddings

Embeddings are a numerical representation of speech characteristics in the form of a vector. They are created by a neural network and can be computed on any given audio sample. The aim of this embedding model is to make embeddings of the same speaker similar, while embeddings of different speakers or audio segments are different. Speaker embeddings are most commonly used to compare known samples of a given speaker with other audio segments, either to differentiate between speakers or to determine where a particular speaker is speaking in a recording. Similarity is typically measured using cosine similarity, a general mathematical metric that quantifies the similarity between the directions of two vectors.

2.3 Determining Audio Source Direction

2.3.1 Microphone Arrays and TDoA

Microphone arrays consist of multiple microphones, typically arranged in a grid or circular configuration. Figure 3 shows an example of such an array. By recording sound simultaneously across several microphones, microphone arrays enable the comparison and combination of the individual signals to infer the direction of an audio source and enhance audio captured from a specific direction.

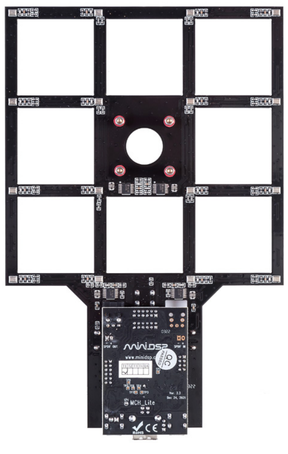


Figure 3: *miniDSP UMA-16 v2 Microphone Array, Comprising 16 Microphones*

Because the microphones in an array are spaced apart, a sound source produces slightly different signals at each microphone, differing in arrival time and amplitude due to variations in the distance between the source and each microphone. These differences in arrival time, known as Time Delay of Arrival (TDoA), are utilised to estimate the direction of an audio source relative to the array.

2.3.2 Beamforming

Beamforming is a signal processing technique used to direct the reception or transmission of sound waves in a specific direction. It leverages multiple microphones to achieve spatial selectivity, enhancing sound coming from a particular direction while filtering out unwanted noise and interference from other directions.

This technique is beneficial for both ASR and diarisation: By reducing background noise and focusing on a particular direction of audio input, speech can be captured more clearly, thereby increasing transcription accuracy [8] and, when using acoustic methods, diarisation performance.

Figure 4 provides a schematic of a simple form of beamforming; a delay & sum beamformer. It shows a linear microphone array consisting of four microphones. It can be seen that the signals of the microphones one through three are shifted by specific values, denoted as $\Delta\tau_1$, $\Delta\tau_2$ and $\Delta\tau_3$, so that the source signals align with the source signal on channel zero, which serves as the reference channel (meaning $\Delta\tau_0 = 0$). The four signals are then added together and divided by the number of microphones, to preserve the original amplitude of the source signal while effectively suppressing the noise signal. This is also known as constructive interference.

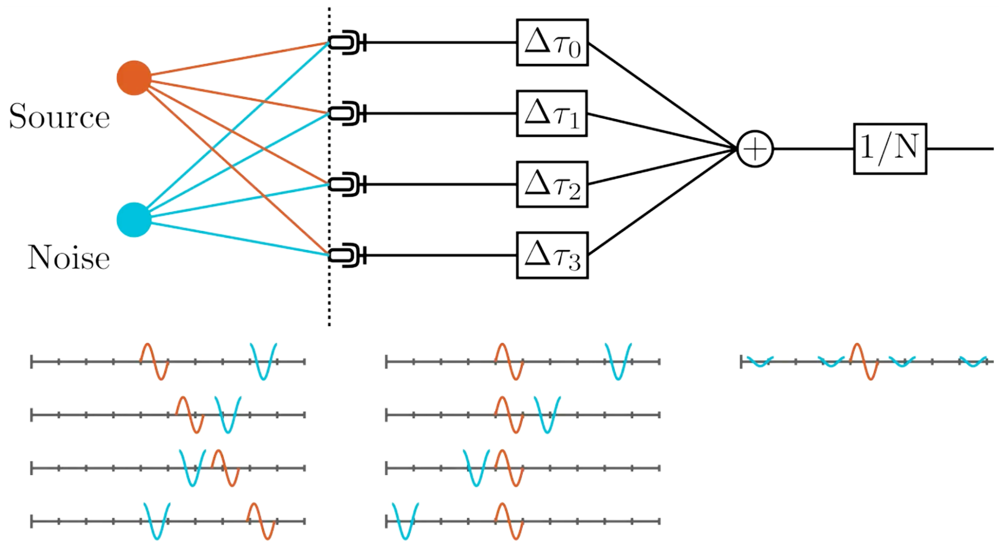


Figure 4: *Schematic of a Delay & Sum Beamformer [2]*

BeamformIt [1] is an open-source acoustic beamforming tool implemented by Xavier Anguera that utilises the filter & sum beamforming technique. This is a slightly more advanced version of the delay & sum technique used in Figure 4, as it applies an independent weight to each of the channels before summing them. It calculates the optimal delay, or shift, needed to align each input channel with the reference channel for a given segment of audio. This is applied to the full duration of an audio recording using the sliding window technique (section 2.1). The delay values used are stored and an optimised mono version of the original multi-channel recording, originating from a microphone array, is created. These stored delay values contain information about the angle of the audio source relative to the array for each window. They serve as features for the diarisation method detailed in section 4.2.2.

2.4 Clustering

Clustering is necessary to group segment-level, or *local*, speaker representations, such as embeddings or directional data, into global classes that correspond to distinct speakers present in the recording.

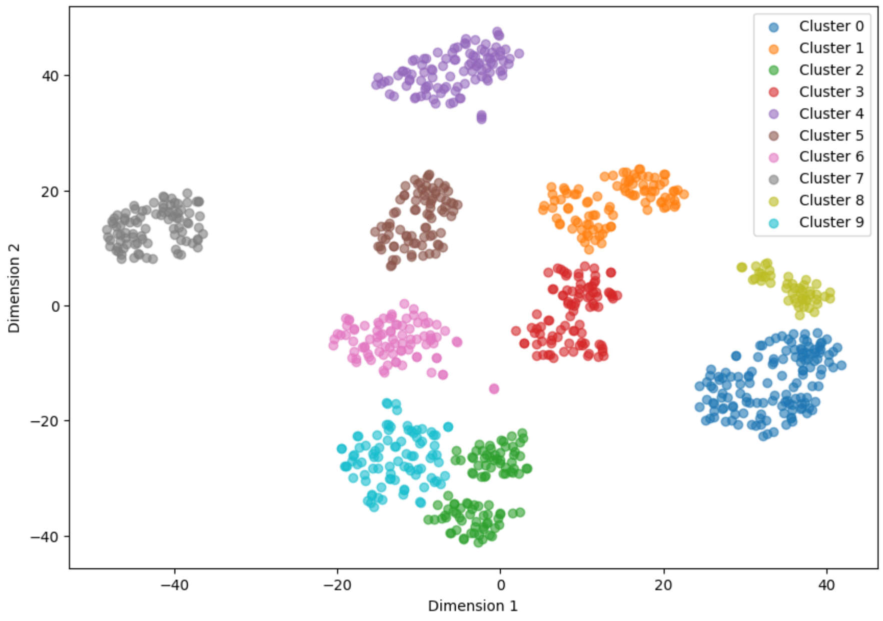


Figure 5: *Illustrative Example of Clustering in a Two-Dimensional Feature Space [13]*

Common clustering algorithms used in diarisation systems include agglomerative hierarchical clustering and k-means clustering.

Agglomerative clustering is a so-called 'bottom-up' clustering approach, in which each data point initially forms its own cluster [9]. Clusters are iteratively merged based on a similarity or distance measure until a stopping criterion is met. This can be either when the distance between the closest pair of clusters exceeds a predefined threshold, or when a desired number of clusters has been reached.

K-means clustering, in contrast, starts with a predefined number of randomly initialised clusters. Data points are iteratively assigned to the nearest cluster centroid, after which the centroids are recomputed, and this process continues until convergence. While agglomerative clustering does not require the number of clusters to be specified in advance, k-means clustering relies on a fixed number of clusters as an input parameter.

3 Data

The data used to evaluate the different diarisation methods is derived from the IFADV corpus [14], which comprises 20 dialogues, each lasting 15 minutes, between two (out of 34) well-acquainted participants, resulting in a total of five hours of speech. For each speaker, a separate audio channel was recorded using head-mounted microphones. These individual channels were used to simulate conversations involving two, four, and six speakers by playing the audio channels

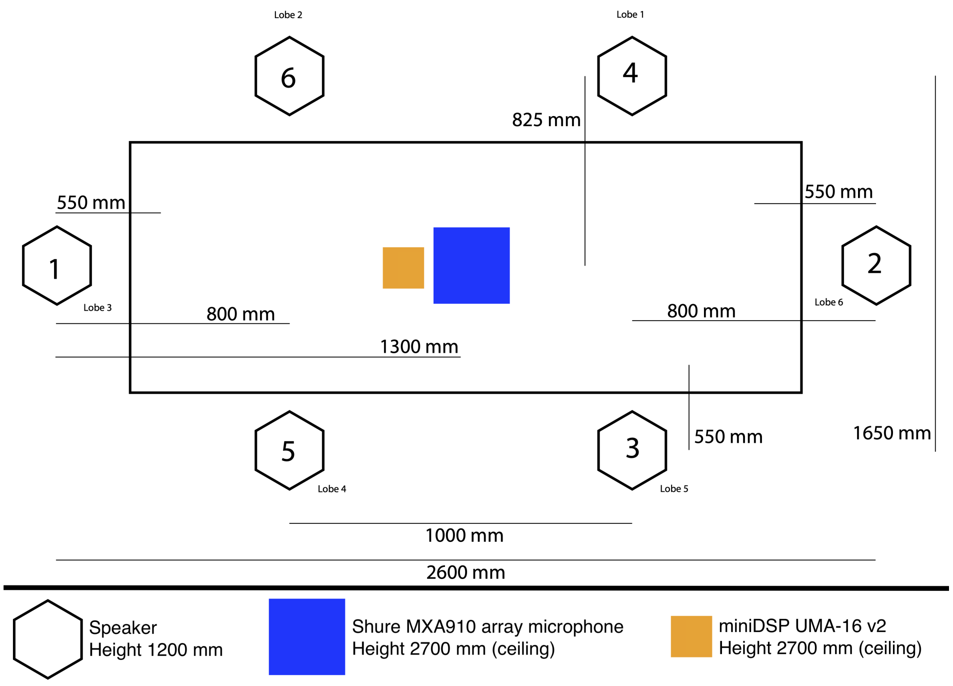


Figure 6: *Overview of the Setup Used For Recording the Dataset*

through different speakers in the recording setup, as depicted in Figure 6. This was carried out under two conditions:

1. Natural conversations: The original five hours of natural conversations between two speakers from the IFADV corpus [14], played through separate speakers-one for each speaker channel-to represent speaker locations.

2. Simulated conversations: Additional audio data was produced by cutting and recombining speaker turns from the original recordings. This process generated 20 hours of audio, consisting of simulated conversations between two, four, and six speakers. The two-speaker simulated conversations contained turns from speakers as paired in the original recordings, but with overlapping speech removed. The four-speaker simulations combined speaker turns from two natural conversations, and the six-speaker simulations included speaker turns from three natural conversations.

It should be noted that this data represents a controlled simulation of real-world conversations. In this setup, speakers have fixed positions at optimal distances from one another, whereas real-world scenarios might involve speakers moving during conversation or being in closer proximity. Furthermore, there was minimal background noise and the simulated conversations (condition 2) did not include overlapping speech, which typically is present in natural conversations.

4 Methods

4.1 Pyannote

Pyannote [12, 6] (pronounced like the French verb 'pianoter' [4]) is an open-source toolkit for speaker diarisation built on the PyTorch machine learning framework, consisting of a set of trainable end-to-end neural building blocks that are combined and jointly optimized [5]. It is one of the most commonly used diarisation systems, most notably utilized by WhisperX [3] for both diarisation and Voice Activity Detection (VAD).

It is an example of an acoustics-based diarisation system, utilising solely the audio signal to perform speaker diarisation in three main steps: segmentation, embedding and clustering [6]. First, a speaker segmentation model is applied using a sliding window. This model outputs frame-level speaker activity probabilities at a fine temporal resolution, for up to three speakers in a window. These probabilities are transformed into binary speaker segmentation information; temporal information of when one (or more) speakers are active in the current window. Based on this, speaker embeddings are computed on the audio segments where exactly one speaker is active. Agglomerative clustering is then applied to bundle speaker embeddings across windows and assign them to a distinct speaker.

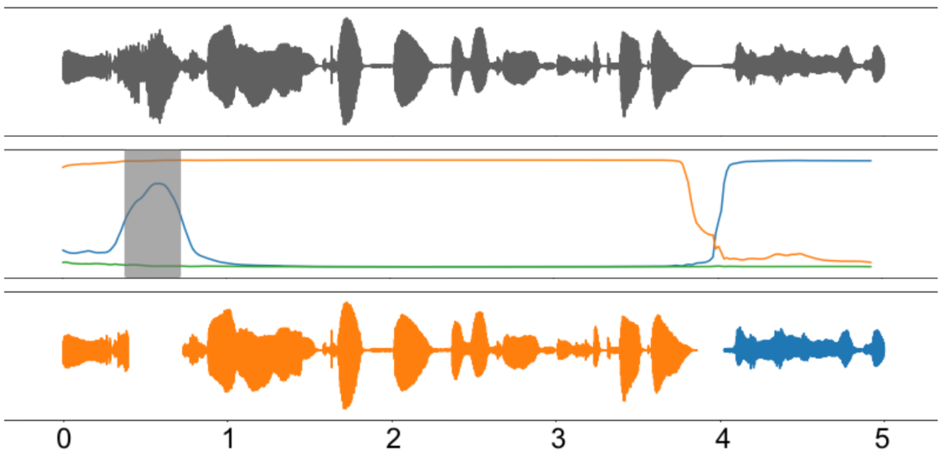


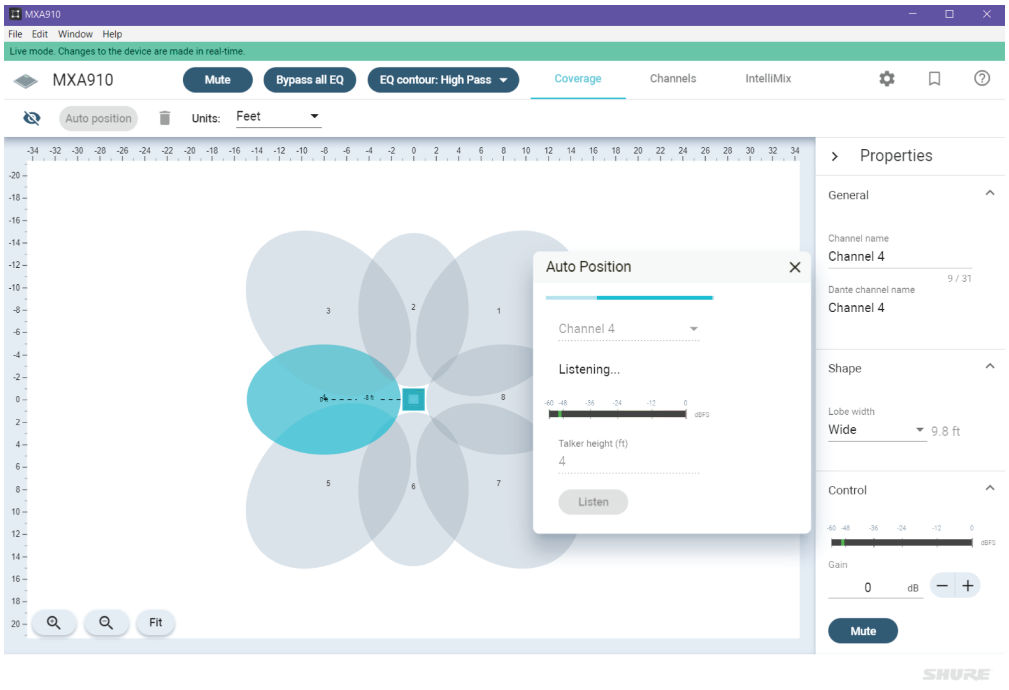
Figure 7: *Schematic of Local Speaker Embedding Extraction in the Pyannote Pipeline [6]. Speaker Embeddings are Computed Only on Segments Where a Single Speaker is Active, Based on the Speaker Segmentation Model Output Shown in the Middle Row.*

4.2 Directional Methods

As discussed in Section 2.3, microphone arrays can be employed to localise audio sources, and this spatial information can subsequently be used for speaker diarisation. As part of the GDAS project commissioned by Politie Nederland, we developed several directionality-based diarisation methods using different microphone arrays. Both the methods and corresponding microphone arrays differ in technique, cost and performance. We discuss two of these methods here.

4.2.1 Shure MXA910 Microphone Array

The Shure MXA910 is a ceiling-mounted microphone array that includes software allowing the user to define *lobes*: designated coverage areas designed to create optimized, beamformed channels for up to eight individual speakers, as shown in Figure 8. Each lobe produces a distinct audio channel that aims to clearly capture speech from speakers within the coverage area while attenuating speech or noise from outside it. Additionally, the system produces a mono channel comprising an optimised combination of all lobes.



Units: Feet

Channel: Channel 4

Talker height (ft): 4

Channel name: Channel 4 (9 / 31)

Dante channel name: Channel 4

Lobe width: Wide 9.8 ft

Gain: 0 dB

Figure 8: *Lobe Customisation in the Shure Software [11]*

Although this functionality was originally designed for digital conferencing, it can also be leveraged for speaker diarisation. In the recording setup (Figure 6), each speaker was assigned a lobe, and the resulting audio channels were used as features for performing diarisation; using a sliding window, the lobe with the most amount of energy is assigned as the active speaker for that window. Following this, smoothing and noise reduction is applied to produce a diarisation result.

As shown in Table 1, this diarisation method achieves excellent performance and yields the best overall results. However, the Shure MXA910 microphone array is expensive (approximately €6000). Because of this, we also developed a generalised diarisation algorithm suitable for more affordable microphone arrays. We demonstrate that speaker diarisation of nearly the same quality can be achieved at a fraction of the cost using, for example, the miniDSP UMA-16 v2 microphone array shown in Figure 3.

4.2.2 MiniDSP UMA-16 v2 Microphone Array

The miniDSP UMA-16 v2 does not include any proprietary software, and only delivers raw data in the form of 16 audio channels, one for each microphone in the array. Beamforming can be applied to these channels resulting in an optimised mono channel, but more importantly for diarisation, BeamformIt [1] will yield delay values for every channel through a sliding window, as mentioned earlier in section 2.3.2. These delay estimates are vectorised and subsequently clustered using k-means clustering to assign a speaker label to each time window. Following this, smoothing and noise reduction is applied to produce a diarisation result.

This diarisation method closely approaches the accuracy achievable with the Shure array but at a fraction of the cost, with the miniDSP UMA-16 v2 priced at approximately €260.

4.3 Adapting Pyannote: Adding Multi-Channel Energy Vectors as Clustering Input

Pyannote provides robust speaker diarisation performance for a fairly wide range of recording conditions. However, utilising energy differences between multi-channel recordings for diarisation can yield much higher accuracy under the right circumstances, as can be seen in Table 1. Therefore we propose to combine the two methods, while taking advantage of both their distinct advantages. We adhere to the original pyannote diarisation pipeline as closely as possible, but add functionality to integrate channel-level energy information in clustering, supplementing the information already encoded in the default pyannote embeddings with multi-channel energy vectors.

4.3.1 Shure Lobe Energy Vectors

To exploit the differences between the individual lobes recorded by the Shure microphone array for diarisation, we compute the average signal energy for each lobe over a given time segment. This average energy is expressed in decibels and computed as $e_\ell = 10 \log_{10}\left(\frac{1}{N}\sum_{n=1}^{N} x_\ell[n]^2\right)$, where $x_\ell[n]$ denotes the discrete-time signal of lobe ℓ over a segment of N samples.

The resulting values are stored in what we refer to as a *Shure lobe energy vector*, e.g., $(-44.56, -46.51, -47.37, -48.47, -45.31, -38.37)$. Each component of this vector corresponds to the average energy observed in a single lobe during a set time segment. As the data used in this work (see Section 3) was recorded using six lobes, all Shure lobe energy vectors are six-dimensional, with one component per lobe.

This representation is inspired by the diarisation approach described in Section 4.2.1, which assigns the lobe with the highest energy as the active speaker for a given time segment. However, that type of assignment discards potentially informative structure present in the relative energy distribution across lobes. By utilizing the full energy vector, thus preserving inter-lobe energy differences, this approach encodes additional information that can be exploited in the clustering stage of the diarisation pipeline.

To incorporate this method into the pyannote diarisation pipeline, the general structure of the pipeline remains unchanged. Following segmentation, local speaker embeddings are computed on binary speaker-active segments [6]. This segmentation step defines the first two dimensions of the resulting embedding tensors, namely the number of audio segments across the whole recording and the maximum number of individual speakers in a segment (three, as discussed in Section 4.1).

Shure lobe energy vectors are computed on the same segments, producing an embedding tensor that matches the pyannote embedding tensor in these first two dimensions and differs only in embedding dimensionality. Whereas the pyannote embeddings are 256-dimensional, the Shure energy vectors have a dimensionality equal to the number of lobes (six in this work). Z-score normalisation is applied to the energy vectors prior to clustering, to ensure that their components are of comparable magnitude to those of the pyannote embeddings.

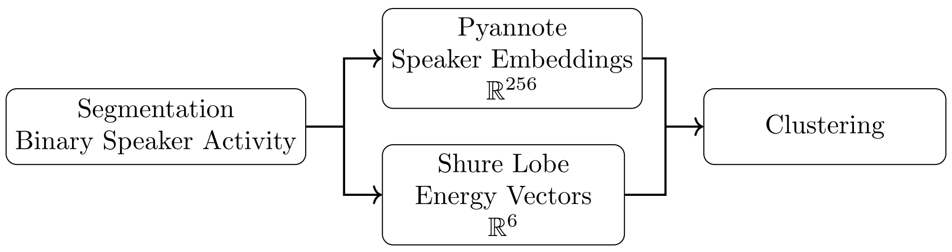


Figure 9: *Schematic of Shure Lobe Energy Vectors Integration Into the Pyannote Diarisation Pipeline*

4.3.2 Mixed Clustering

The clustering step of the pyannote diarisation pipeline is adapted to incorporate both the default pyannote speaker embeddings and the Shure lobe energy vectors. In the standard pyannote diarisation pipeline, agglomerative hierarchical clustering operates directly on local speaker embeddings. Pairwise distances between these embeddings are computed internally by the clustering algorithm, and merging decisions are driven solely by similarity in the speaker embedding space.

In our adapted pipeline, the clustering stage is reformulated to operate on a precomputed, mixed distance matrix that combines speaker embedding similarity and spatial energy information. Pairwise distances are computed separately for the pyannote speaker embeddings and for the Shure lobe energy vectors. These distance matrices are then combined into a single mixed distance matrix using a weighting factor, allowing controlled integration of speaker embedding similarity and spatial energy information. Agglomerative clustering is subsequently applied to this mixed distance matrix.

By integrating the two clustering inputs on a distance level, we enable joint use of heterogeneous representations, i.e., high-dimensional speaker embeddings and low-dimensional lobe energy vectors, without requiring them to share a common embedding space, and preserve compatibility with the original pipeline.

5 Results

This section provides an overview of performance statistics of the different diarisation methods discussed, applied to the data as detailed in section 3.

The Diarisation Error Rate (DER), as defined by the National Institute of Standards and Technology (NIST) [7], is the most commonly used metric for evaluating speaker diarisation systems. DER represents the percentage of incorrectly attributed speech time, encompassing falsely detected speech, missed speech detections, and incorrectly attributed speech, relative to the total amount of speech time. A DER of zero signifies perfect performance, while higher scores indicate poorer performance. Since the false alarm and missed speech components of DER depend on Speech Activity Detection (SAD), and our diarisation systems are currently designed only to add speaker information to a transcription without employing SAD, we compare speaker error time; the amount of time for which the system predicts an incorrect speaker during speech.

Additionally, we introduce our own evaluation metric: the attributed ASR score. This measure reflects the proportion of words in the ASR that are attributed to an incorrect speaker, offering a clear indication of the quality of a potential transcription enriched with speaker identities.

Pyannote diarisation was applied to the optimised combined channel of the Shure MXA910 microphone array. This ensures a fair comparison with the pyannote-Shure combination system, which also uses the combined channel to compute the pyannote embeddings, while the Shure lobe energy vectors are derived from the individual lobe recordings.

Table 1 shows the overall performance on all the available data. For comparing the performance between the two conditions described in section 3, see the Appendix (section 8).

System	Speaker error time	Attributed ASR
pyannote	33.7%	29.5%
Shure	2.6%	3.4%
miniDSP	4.2%	5.8%
pyannote Shure combination	12.0%	13.4%

Table 1: *Performance of the Different Diarisation Systems*

6 Conclusion

The choice of which diarisation technique to employ depends heavily on situation-specific variables.

Directional methods have the potential to offer far higher diarisation accuracy, but that performance is highly dependent on certain physical conditions: A microphone array is required and should be positioned centrally within the speakers locations. Speakers should be evenly dispersed across a space, but not too far from the array and they cannot move around. When these conditions are met, these methods are able to deliver up to tenfold greater diarisation accuracy compared to acoustic methods, as shown in Table 1. They do require additional hardware investment over acoustic methods, with prices of microphone arrays that can range from several dozen euros to thousands. Also, more storage space is needed to store the separate audio channels.

The primary advantages of acoustic methods lie in their ability to be applied retrospectively to any existing (mono) audio recording, without the need for specialised hardware. Although quite limited in absolute accuracy, they can provide fairly robust performance across a wide range of recording situations, with varying microphone type and placement. For example, acoustic methods are insensitive to speakers moving around or all coming up to the same microphone to speak, whereas these variables are fundamental determinants for the performance of directional methods. Since only audio characteristics are available to distinguish speakers with acoustic methods, there is a higher risk of confusion between similarly sounding speakers. Additionally, a speaker's voice characteristics may vary over time and between recordings due to factors such as emotional expression and microphone type.

Combining the two methods and expanding the pyannote diarisation pipeline to utilize both audio characteristics and spatial information provides a reliable increase in performance over the original system. This seems to indicate that acoustic and spatial cues provide complementary information to the clustering stage. At the same time, preserving the original pyannote structure means that segmentation errors are not mitigated; the clustering stage is enhanced, but the upstream segmentation model remains unchanged. This design choice represents both a strength and a limitation. The segmentation model provides robust Speech Activity Detection (SAD), something which the spatial localisation methods lack. On the other hand, it acts as an upper bound on performance in applications where SAD is not required, as any SAD will inherently introduce some missed speech.

The results detailed in the Appendix (section 8) further illustrate the interaction between pyannote's existing pipeline and the added Shure energy vectors. When pyannote already performs reasonably well (e.g., on natural conversations, Table 2), incorporating Shure lobe energy vectors brings performance close to that of directional methods, while still benefitting from the robustness that comes from the pipeline and its constituent parts. When baseline pyannote performance is poor (e.g., on simulated conversations, Table 3), the mixed approach still provides a substantial (over twofold) improvement over the acoustic-only system, although absolute performance remains hampered by the pyannote embeddings.

Importantly, the controlled recording conditions used in this study favour directional methods. In real-world scenarios, where speakers may move or speak at the same time, purely directional approaches may show severely degraded performance. In such conditions, the mixed system is expected to generalise more effectively by leveraging both spatial and acoustic information. Furthermore, the weighting factor between embedding and energy-based distances allows the system to be tuned towards either modality, offering additional flexibility.

In summary, directional methods have the potential to be incredibly accurate diarisation systems but require tightly controlled circumstances and additional hardware. We demonstrate that we can successfully integrate their strengths into pyannote, while retaining the benefits from the existing pipeline.

7 Acknowledgements

We thank Martijn Bentum of Radboud University and Dennis Bergveld of Politie Nederland for their efforts in compiling and recording the data and providing a schematic of the recording setup. Further gratitude is extended to Politie Nederland for allowing the use of this data in this publication. The data collection was conducted as part of the GDAS project; an initiative by Politie Nederland focused on speech-to-text solutions, in which SpraakLab was consulted to research and develop diarisation techniques.

References

[1] Xavier Anguera, Chuck Wooters, and Javier Hernando. “Acoustic beamforming for speaker diarization of meetings”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.7 (2007), pp. 2011–2022.

[2] AudioLabsErlangen. *A gentle introduction to beamforming.* URL: `https://www.youtube.com/watch?v=etTcru7CrWU`.

[3] Max Bain et al. “WhisperX: Time-Accurate Speech Transcription of Long-Form Audio”. In: *INTERSPEECH 2023* (2023).

[4] Hervé Bredin. *Frequently asked questions.* URL: `https://github.com/pyannote/pyannote-audio/blob/develop/FAQ.md`.

[5] Hervé Bredin. *pyannote audio: neural building blocks for speaker diarization.* URL: `https://www.youtube.com/watch?v=37R_R82lfwA`.

[6] Hervé Bredin. “pyannote.audio 2.1 speaker diarization pipeline: principle, benchmark, and recipe”. In: *Proc. INTERSPEECH 2023.* 2023.

[7] Jonathan G Fiscus, Jerome Ajot, and John S Garofolo. “The rich transcription 2007 meeting recognition evaluation”. In: *International Evaluation Workshop on Rich Transcription.* Springer. 2007, pp. 373–389.

[8] Takuya Higuchi et al. “Robust MVDR beamforming using time-frequency masks for online/offline ASR in noise”. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE. 2016, pp. 5210–5214.

[9] Marijn Anthonius Henricus Huijbregts. "Segmentation, diarization and speech transcription: surprise data unraveled". PhD thesis. University of Twente, 2008.

[10] Luiz GUStavo Martins. *Transfer Learning for Audio Data with YAMNet.* URL: https://blog.tensorflow.org/2021/03/transfer-learning-for-audio-data-with-yamnet.html.

[11] *MXA910, MXA910-60CM, MXA910W-A, MXA910W-US user guide.* URL: https://www.shure.com/en-US/docs/guide/MXA910.

[12] Alexis Plaquet and Hervé Bredin. "Powerset multi-class cross entropy loss for neural speaker diarization". In: *Proc. INTERSPEECH 2023.* 2023.

[13] Manav Sarkar. *Exploring Different Clustering Techniques for Sentence Embeddings.* URL: https://medium.com/@manavsarkar/exploring-different-clustering-techniques-for-sentence-embeddings-5b344e441c5f.

[14] R. van Son et al. "The IFADV corpus: A free dialog video corpus". In: *Proceedings of LREC 2008.* 2008, p. DVD.

8 Appendix

System	Speaker error time	Attributed ASR score
pyannote	17.2%	14.9%
Shure	1.6%	7.5%
miniDSP	3.1%	8.0%
pyannote Shure combination	3.7%	8.3%

Table 2: *Performance of the Different Diarisation Systems on Natural Conversations*

System	Speaker error time	Attributed ASR score
pyannote	37.9%	34.1%
Shure	0.5%	2.2%
miniDSP	2.0%	5.1%
pyannote Shure combination	14.0%	15.0%

Table 3: *Performance of the Different Diarisation Systems on Simulated Conversations*