Strategies for Automatic Generation of Information Processing Pathway Maps

Anirudh Lakra[1*], Cai Wingfield[2,3], Chao Zhang[4,5,6], Andrew Thwaites[2,4]

[1]Department of Computer Science, University College London, London, UK

[2]MRC Cognition and Brain Sciences Unit, Cambridge, UK

[3]Institute for Data and AI, University of Birmingham, Birmingham, UK

[4]Department for Speech Hearing and Phonetic Sciences, University College London, London, UK

[4]Department of Electronic Engineering, Tsinghua University, Beijing, China

[6]Shanghai Artificial Intelligence Laboratory

Corresponding authors: *Andrew Thwaites*

**Anirudh Lakra is now at Amazon.com, Inc, and his position at Amazon is unaffiliated with this work.*

1. Abstract

Information Processing Pathway Maps (IPPMs) are a concise way to represent the evidence for the transformation of information as it travels around the brain. However, their construction currently relies on hand-drawn maps from electrophysical recordings such as magnetoencephalography (MEG) and electroencephalography (EEG). This is both inefficient and contains an element of subjectivity. A better approach would be to automatically generate IPPMs from the data and objectively evaluate their accuracy.

In this work, we propose a range of possible strategies and compare them to select the best. To this end, we a) provide a test dataset against which automatic IPPM creation procedures can be evaluated; b) suggest two novel evaluation metrics—*causality violation* and *transform recall*—from which these proposed procedures can be evaluated, and c) propose and evaluate a selection of different IPPM creation procedures. Our results suggest that the *max pooling* approach gives the best results on these metrics. We conclude with a discussion of the limitations of this framework, and possible future directions.

(164 words)

2. Introduction

Functional brain mapping is the data-driven process of associating specific brain regions with critical functions such as vision, sensation, movement, and language. Gaining a clear understanding of where and how the brain performs these functions has wide-ranging implications. Accurate functional mapping, for instance, is essential for advancing neurotechnology performance and safety, as well as making neurosurgery more precise and reliable. Furthermore, it lays the foundation for exploring higher-order cognitive functions like memory, attention, and learning. Consequently, functional brain mapping remains a key focus in current research.

A recent development in this field is the creation of *Information Processing Pathway Maps* (IPPMs) from electrophysiological neural recordings (Thwaites et al., 2025). IPPMs represent the sequences of mathematical transformations that describe how sensory information is processed as it travels through the nervous system and cortex (Figure 1). These maps have significant potential across various applications, including Brain–Computer Interfaces (BCIs), human prosthetics, and clinical interventions. To date, IPPMs have been developed for processes such as loudness processing (Thwaites et al., 2015; 2017), color processing (Thwaites et al., 2018), visual motion processing (Wingfield et al., 2025), and tactile processing (Thwaites et al., 2025).

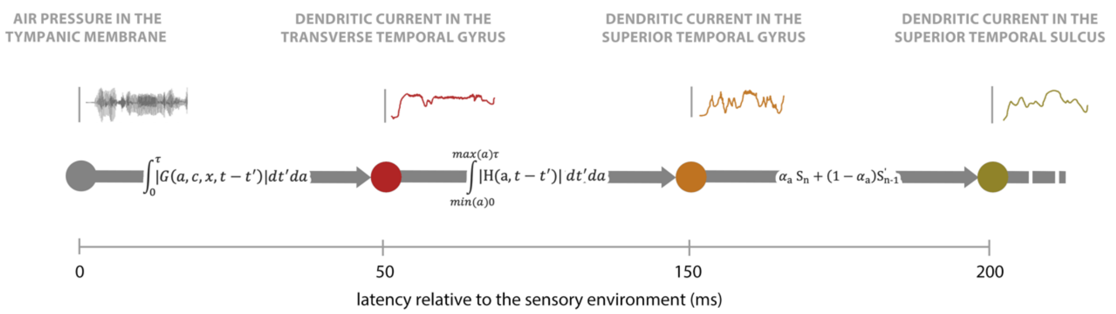

Figure 1: Example IPPM. Sound hitting the tympanic membrane undergoes a sequence of mathematical transformations as it travels up the auditory pathway, with the outputs of these transformations being entrained to neuronal activity in different locations in the nervous system and cortex. The position of each node on the x-axis denotes the latency at which the outputs of these transforms are entrained. Adapted from Thwaites et al., (2025).

Before discussing the construction of IPPMs, it is important to define some key terms. A *transform* refers to any mathematical function that hypothesizes how stimulus properties correspond to measured cortical activity in the human brain. Given the inherent spatial accuracy of our neuroimaging methodology, we resolve results into small, tessellating hexagons around 3mm in diameter, referred to as *hexels*. *Expression* refers to the situation where the output of a particular transform correlates with the observed cortical activity in a specific hexel.

IPPMs can be created using any time-varying measure of neural activity, including electroencephalographic (EEG) and magnetoencephalographic (MEG) recordings. The process involves two main stages (Figure 2).

In Stage 1, the researcher starts with time-varying neural activity data from a large number of hexels, recorded while participants engaged in tasks like listening to a podcast or watching a movie. The researcher also begins with a *candidate transform list* (CTL) which includes potential transforms believed to occur in the nervous system. The stimulus is processed through each transform in the list, generating precise predictions of cortical activity. These predictions are then compared to the actual neural activity across various latencies. After passing through a model-selection procedure (Thwaites et al., 2017), a transform *expression map* is created. This result is often visualized as an *expression plot* (see Stage 1 of Figure 2, for example).

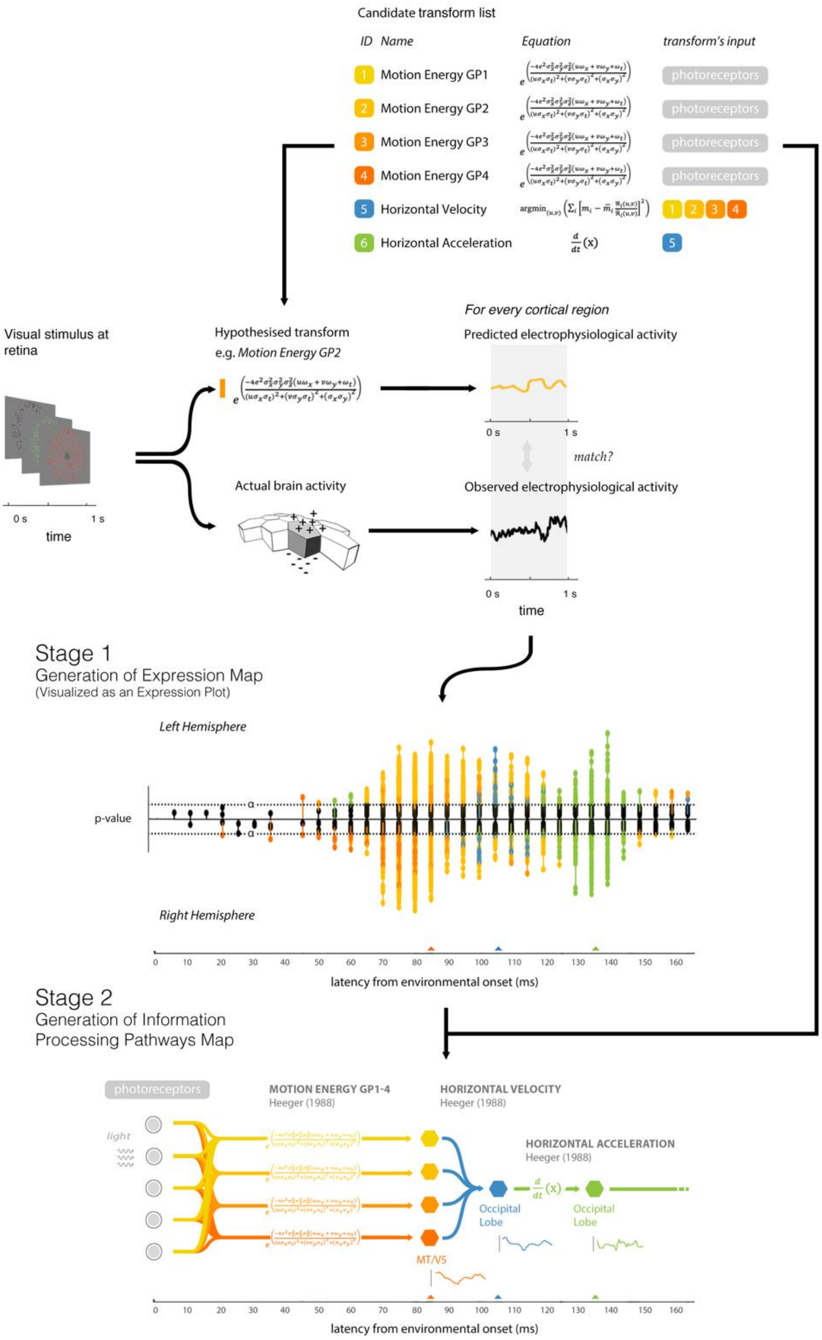

Figure 2. The main steps involved in IPPM creation. Reproduced from Thwaites et al. (2025). In the first stage, expression data generation, the cross correlations between the outputs predicted by hypothesized transforms, model selection and significance testing applied, which results in a map of cortical expression (visualized here as an *expression plot*, where the stems show the latency at which each of the hexels for each hemisphere best matched the output of the tested transform, with the y-axis shows the evidence supporting the match at this latency; if any of the hexels have evidence, at their best latency, indicated by a *p*-value lower than α*, the match is significant and the stems are colored, depending on the transform). In the second stage, IPPM generation, the expression data and candidate transform list are added as a constraint to infer the processing pathways map. This second stage is currently done by hand. The final IPPM describes the transforms underlying human visual motion processing (Wingfield et al., 2025).

During Stage 1, the tested transforms are *input-stream-to-hexel* transforms, representing the relationship between the input stream (e.g., auditory or visual stimuli) and hexel activations. By contrast, IPPMs are constructed using *hexel-to-hexel* transforms, representing the relationships between different nodes within the IPPM. While Stage 1 does not explicitly test these *hexel-to-hexel* transforms, they can be inferred from the definitions in the CTL, and in Stage 2, the researcher uses these definitions, along with the expression data, to infer the IPPM.

Despite the critical role of Stage 2 in IPPM creation, it can currently only be performed manually. This is due to the "blurred" nature of expression data, resulting from the inherent difficulty in source localization in EEG and MEG (Grave de Peralta-Menendez et al., 1996; Grave de Peralta-Menendez and Gonzalez-Andino, 1998). This difficulty leads to a phenomenon known as *point spread*, where responses bleed into neighboring cortical sources (Hauk et al., 2011). Consequently, hundreds of hexels may appear significantly entrained to given transforms, creating clusters of expression spikes in the expression plot (Figure 3). Resolving these spikes into separable effects can be challenging. Where one researcher might interpret them as a single focused effect and select the most significant spike as representative, another may see two temporally overlapping effects. Distinguishing between a single temporally distributed effect and multiple shorter effects with some overlap requires the researcher’s judgment, guided by prior knowledge from the literature.

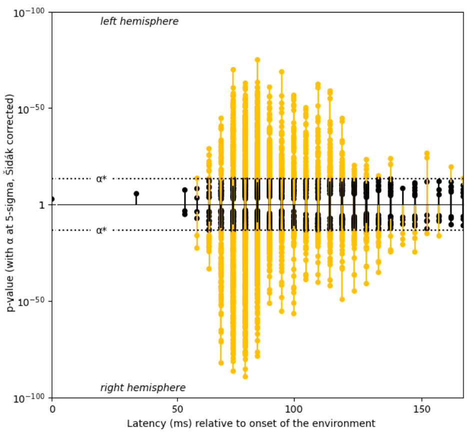

Figure 3. The expression plot for Heeger horizontal ME GP2 This plot highlights the bleeding of expression into nearby cortical regions. It can be difficult to distinguish whether these are

multiple responses close in time or part of the same response. Adapted from Wingfield et al., (2025).

While IPPMs are effective tools for data visualization, their manual creation has several inherent drawbacks. First, the process is subjective: there are no formalized rules for IPPM generation, which can lead to variations between researchers. Second, this subjectivity might tempt researchers to interpret the expression plot in ways that align with their study's goals. Third, being forced to rely on prior literature to determine where distinct effects "should be" detracts from the objectivity of the process. Finally, the manual creation of IPPMs can be labor-intensive, especially when mapping a large number of transforms. For IPPMs to be effectively used in clinical, diagnostic, or BCI contexts (Thwaites et al., 2025), they need to be fully data driven.

This paper develops an automated IPPM generation system that implements the logic of an objective researcher. To evaluate this system, we propose a framework using two IPPM baselines—auditory loudness processing and visual motion processing—along with two metrics: *Causality Violation* (CV) and *Transform Recall* (TR). While handcrafted IPPMs might serve as a plausible gold standard, we avoid using them as benchmarks due to the risk of inherent errors. Instead, we focus on evaluation metrics that assess qualities necessary for a true IPPM. This framework aims to set the stage for future advancements in automatic IPPM generation.

3. Problem formalization

An IPPM can formally be defined as a Directed Acyclic Graphs (DAGs) whose nodes are significant hexels, and whose edges connect serially composed transforms. Given (1) expression data and (2) a candidate transform list (CTL) that includes information about the relationship between *input-stream-to-hexel* transforms and *hexel-to-hexel* transforms, we wish to create a DAG with these qualities. In particular, this information allows us to make inferences about whether transforms are taking place serially or in parallel (Figure 4).

Figure 4. Some of scenarios that the IPPM generator might face. A The candidate transform list (CTL) allows us to convert from input-to-node transforms (present in the expression map) to node-to-node transforms represented in the resultant IPPM. **A.** In this example, all transformations in the CTL are input-to-node, resulting in the above estimation of the IPPM. **B.** In this scenario, *transform_B()* has been replaced with *transform_C(),* which has the output *transform_A()* as its input. This alters the resultant IPPM accordingly, placing the new transform in a sequence. **C.** This scenario is the same as Scenario 2, except that the expression map has a second hexel that is entrained to the output of *transform_C().* This alters the resultant IPPM, adding an 'empty' transform which copies the information (unchanged) to another hexel. This is commonly known as the *null()* or *identity()* transform. **D.** This scenario is the same as Scenario 1, except that the CTL has a third transform D which accepts the output of A and B as input, and the expression map has a hexel that is entrained to it. This alters IPPM, creating a sequence of three transforms between hexels. This list of scenarios is not exhaustive but aims to give a sense of the situations an automatic IPPM generator needs to be able to handle.

To mitigate point spread, a clustering subsystem is applied before the graph generator to estimate which clusters of significant spikes derive from the same underlying effect. The output of the clusterer is a much smaller set of significant hexels representing the temporal foci of separable effects. Given the clustered hexels and latencies, the graph generator then applies a candidate transform list to generate the required graph.

4. IPPM Evaluation Framework

As noted, handcrafted IPPMs are not a suitable comparator for the evaluation of automatic IPPM generation, due to the inherent subjectivity present in their production. But there are other ways to evaluate IPPM accuracy. In particular, all accurate IPPMs have fundamental properties about them that must be true, and we can assess an automated IPPM's accuracy by estimating to what extent they comply with these properties.

We propose two such metrics. The first is **Causality Violation** (CV), which is based on the premise that information cannot travel backwards in time. Suppose we have a transform, $B(\cdot)$, which has the parent transform, $A(\cdot)$. Such a relationship suggests that information should travel forwards in time from parent to child, from $A(\cdot)$ to $B(\cdot)$. In the generated IPPM, this would be reflected by the nodes for A being placed earlier to those for B, with the arrows of causation leading from A to B. We define CV as the number of edges facing backwards divided by the number of edges in total:

$$\text{causality\_violation(ippm)} = \frac{\#\{\text{backwards-facing edges}\}}{\#\{\text{edges}\}}$$

Our second metric, **Transform Recall** (TR), focuses on ensuring that the IPPM retains as much useful information as possible. During clustering stage, the clustering algorithm may mislabel significant expression spikes as anomalous and exclude them from the clusters, leading to missing transforms in the final IPPM. We argue that the clustering algorithm should avoid doing this unless it is highly certain that the expression it is excluding is indeed an anomaly. Consequently, TR is defined as the proportion of detectable transforms in the candidate transform list that appear in the generated IPPM (where a detectable transform is one that shows significant evidence in the expression plot). This approach was motivated by the observation that an automatic IPPM generator cannot detect a transform missing from the expression data, so we should not penalize a candidate generator for missing it:

$$\text{transform\_recall(ippm, dataset)} = \frac{\#\{\text{Unique transforms which label IPPM nodes}\}}{\#\{\text{Transforms with evidence of expression}\}}$$

CV and TR are, to some extent, complementary; each measures a different aspect of IPPMs that are often in tension with one another. While CV focuses on the correctness of the location of the nodes, TR evaluates the system sensitivity. By reducing the number of nodes, CV will tend to improve, since with less nodes it becomes less likely for a node to precede its parent, leading to less violations. However, this comes at the expense of TR as if the threshold for a non-anomalous cluster is too high, we can mislabel significant spikes as anomalies, resulting in discarded nodes. Thus, CV prioritizes correct IPPMs with a minimal complexity while TR prioritizes IPPMs that capture the greatest quantity of salient information from the expression plot. Through both metrics, we can locate the model with the optimal fit but also with the greatest parsimony.

Poor CV or TR may arise from either an erroneous expression plot, inaccurate or incomplete CTL, or faulty clustering algorithm. In this paper, we are concerned with the last source of error—the clustering error. To identify the ideal generator, we need to isolate the clustering error, which requires controlling for other sources of error. One can achieve this by fixing a CTL established by prior literature. We analyze this assumption and its consequences in greater detail later in the Discussion section.

With TR and CV established, we can state our evaluation framework: each IPPM generation strategy will be tuned and assessed using TR and CV on two datasets related to *Loudness* and *Motion* (Thwaites et al., 2017; Wingfield et al, 2025).

In practice, the proposed strategies require the estimation of suitable hyperparameters. Since CV and TR are in tension with one another, we select the hyperparameters by performing a grid-search over a range of values, plotting the Pareto frontier, and identifying points that first maximise TR, then CV. Our rationale for prioritising TR is that it signifies that all the salient information in the expression plot has been captured, so all information required to create the perfect IPPM is there but was not attained due to one of the sources of error.

5. **Strategies for Automatic IPPM Generation**

In addition to setting out an IPPM-accuracy framework, we test a range of solutions that aim to automatically generate IPPMs. Automatic IPPM generation requires two steps, *Clustering,* which attempts to negate the effects of point spread, and *The IPPM Builder,* which creates a suitable DAG from the CTL and the clustered expression map. The rules that underpin the second stage IPPM Builder can become quite complex but are relatively straightforward to justify (see section 5.2). The biggest issue in accurate IPPM generation is the initial clustering step, which can make a big difference to the accuracy of the final IPPM. As a result, we have chosen to use the same IPPM Builder rules with all clustering strategies tested in this paper.

5.1 Clustering

As noted, EMEG source reconstruction blurs the activity estimated on the cortex. "Clustering", or "denoising", refers to the process of trying to compensate for a mixing of a signal with a source of noise (in our case, the inherent spatial imprecision of EEG and MEG, as well as other sources of experimental noise). We investigated a variety of state-of-the-art denoising techniques. The hyperparameter configuration for each denoiser was evaluated based on how well its results matched priors from literature. Therefore, we use the same configuration across transforms for interpretability of results; however, they could be further improved by leveraging transform-specific hyperparameters. In the following sections, when we describe the time complexity for each system, N_H, is the number of hexels on the cortical surface (equal to 10,242 per hemisphere for our data), and N_T is equal to the number of transforms.

The algorithm powering the clustering stage operates by preprocessing the expression plot, then running one of the clustering algorithms below on the preprocessed data, and, finally, postprocessing the clustering output. The dimensions of the expression plot are latency on the *x*-axis and the surprisal (i.e. *-log(p)*) on the *y*-axis. From empirical experiments, our recommended preprocessing consists of removing insignificant spikes, shuffling the expression plot to remove spurious correlations, discarding the surprisal dimension and focusing solely on the density of points in time, merging the hemispheres to double the number of datapoints, and, lastly, scaling the latency to be unit length. Next, the selected clustering algorithm assigns each of the significant spikes to a cluster, or tags them as anomalies. Finally, in postprocessing, we remove any anomalous points, tag the most significant spike per cluster as the focus, and discard the rest of the spikes. The resulting expression plot contains only temporal foci, which can be used to draw an IPPM.

One could argue that it is better to select the temporal foci as the average centroid per cluster, rather than taking the most significant point. The primary advantage afforded by this strategy is that the average is more representative of the underlying cluster than the most significant spike. Unfortunately, the issue with this approach is that the average is a virtual datapoint, i.e., it does not correspond to a real hexel and expression. Consequently, IPPMs, which are designed for interpretability, become more obscure. Moreover, the most significant spike in a cluster is the spike that displays the strongest evidence for a match. Therefore, it is the most likely location within a cluster where the transform appears.

5.1.1 Max Pooler

Max Pooler (MP) is a latency-focused clustering algorithm that partitions latency into fixed bin size, b, width bins and tags bins with more than threshold, θ, spikes as clusters. b and θ constitute the hyperparameters for MP. MP takes $O(N_T \cdot N_H)$ time to cluster because it requires one loop to assign each spike to a bin, repeated for N_T transforms. And example of this clustering is shown in Fig 5.

Like the other clustering algorithms, MP assumes certain properties that it believes the clusters to satisfy. Specifically, it assumes that every cluster has the same size: b. Additionally, clusters must contain at least θ points, so isolated spikes with high magnitude but low density in latency are discarded as anomalies. Notably, for practical purposes, MP does not consider the surprisal of spikes.

MP, on the one hand, provides easy to interpret results, has low implementation overhead, and is not resource intensive. On the other hand, it makes inflexible assumptions about the nature of the clusters, particularly regarding cluster size.

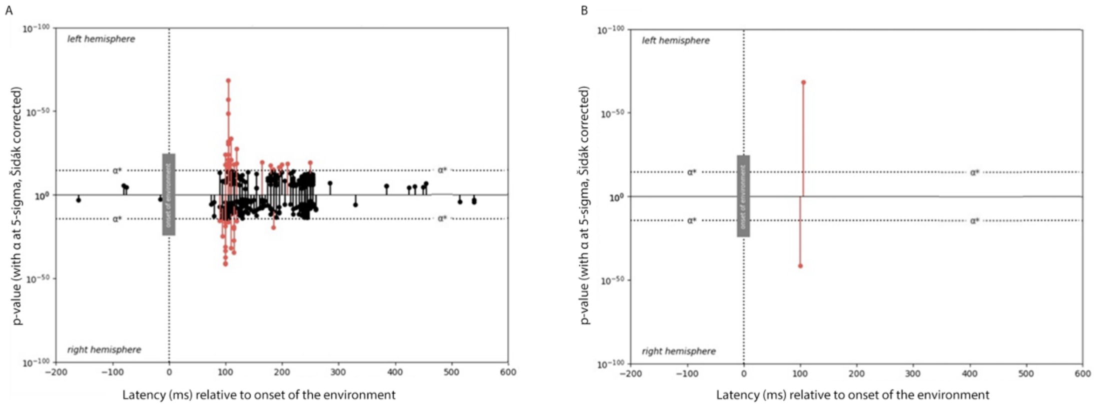

Figure 5. Illustration of 'denoising' Expression Data. A. Original expression data for the transform 'Heeger Horizontal velocity'. B. The expression data for 'Heeger Horizontal velocity' following the clustering strategy of Max Pooling; the resulting map has been cleared of all.

5.1.2 Adaptive Max Pooler

Adaptive Max Pooler (AMP) is an extension of MP that relaxes the assumption that each cluster has the same width. AMP takes the output of MP and merges adjacent clusters recursively, resulting in variable-width clusters. Instead of a fixed bin size, AMP uses a minimum bin width b; otherwise, AMP uses the same heuristics as MP.

Although there is an additional time penalty for looping through each bin and merging each bin, the time complexity remains $O(N_T \cdot N_H)$, the same as MP. Hence, AMP achieves greater generalization power with virtually no increase in complexity.

AMP is a "Goldilocks" solution: it is a well-rounded algorithm that balances adaptability and efficiency. However, its performance is sensitive to the choice of minimum bin width: setting it too high risks merging distinct clusters, while a value that is too low may fragment single clusters into multiple parts, particularly in cases with minor variations in the data.

5.1.3 Gaussian Mixture Model

Gaussian Mixture Modelling (GMM) (Pearson, 1894; Dempster et al, 1977) is a generative machine learning approach that models expression plots as multimodal Gaussian distributions. The optimization is performed using the *Expectation-Maximization*

(EM) algorithm (Dempster et al, 1977), which is commonly used in the context of latent variables. To determine the optimal number of Gaussians, our algorithm conducts a grid-search evaluated by AIC (Akaike, 1973). Additionally, GMM suffers from the *singularity* problem, the case where it fits a single Gaussian to a lone datapoint, leading to a singular covariance matrix and infinite likelihood. To mitigate this, GMM checks for singular covariance matrices after each fitting and reruns the fitting until it finds a non-singular matrix or reaches a maximum number of retries.

The time complexity for GMM is $O(N_T \cdot N_K \cdot I \cdot T \cdot N_H \cdot K \cdot D^2)$, where N_K is the number of Gaussians to grid-search up to, I is the number of initializations attempted, T is the maximum number of iterations, K is the number of clusters, and D is the number of dimensions. The $N_H \cdot K \cdot D^2$ arises from the M-step, which dominates the EM algorithm's time complexity due to the computation of the covariance matrix.

GMM assumes the underlying expression plot can be modelled by a multimodal Gaussian distribution, a reasonable assumption given the Gaussian blurring. While AIC performs best with large datasets due to its susceptibility to overfitting with smaller ones, using GMMs with fewer components helps address this by skewing the data-to-parameter ratio.

Due to the unconstrained covariance matrices, GMM can model elliptical clusters, making it more flexible than MP or AMP. Yet, such flexibility comes at the expense of computational complexity as GMM must be run numerous times per transform. Although, in practice, by keeping various hyperparameters, such as I, bounded, the computational overhead can be managed effectively.

5.1.4 Mean Shift

Mean Shift (Comaniciu, 2002) is a non-parametric, unsupervised machine learning algorithm, which models the expression plot using Kernel Density Estimation (KDE) (Rosenblatt, 1956; Parzen, 1962). Critically, Mean Shift does not require the number of clusters to be predetermined; it begins with the maximum number of clusters—one for each spike—and merges nearby clusters by exploiting the Capture Theorem (Bertsekas, 2016). The time complexity for Mean Shift is O($N_T \cdot T \cdot N_H^2$), where T is the maximum number of iterations.

Mean Shift is the most general algorithm encountered so far, as it imposes no *a priori* assumptions about the nature of the underlying ground-truth, enabling it to model an arbitrary number of clusters and shapes. However, the performance of mean shift is heavily dependent on the *bandwidth* hyperparameter, which defines the initial cluster radius. As the radius of the clusters increases, the fitting becomes increasingly smoother but also more prone to underfitting. Moreover, using the same bandwidth globally implies the feature space is homogeneous, which is plausible given the Gaussian blurring. Finally, Mean Shift computes the distance between kernels in each iteration, leading to quadratic complexity with respect to dataset size, making it the most computational prohibitive algorithm.

5.1.5 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN (Ester et al., 1996) is a density-based, hierarchical clustering algorithm. It locates clusters by identifying core points, defined as points with at least *a set number of* points in an ε-neighborhood, and then attempting to jump to nearby points from these core points. Every point that is reachable from a core point, regardless of whether that is one or multiple jumps, is associated as part of that cluster. The time complexity for DBSCAN is ($N_T \cdot N_H \cdot log\ N_H$).

DBSCAN uses density as a proxy for cluster plausibility and the same ε globally. Its generalization performance is akin to Mean Shift, as it places no assumptions on the shape or number of clusters, while also having slightly better time complexity. Furthermore, DBSCAN can be viewed as the next-generation approach to AMP. The primary difference between the two is that DBSCAN incorporates insignificant bins as part of a cluster if they are reachable from a significant bin. Consequently, it can overcome the fragmented cluster problem that affected AMP.

5.2 IPPM Builder

The *IPPM builder* constructs a DAG from the clustered expression map and the CTL. The CTL is provided as a dictionary, where child transforms are the keys, and each key is associated with a list of its parent transforms. The builder iterates through the transforms, starting with those that have no children. For each iteration, an edge is added from the final node of the parent transform to the initial node of the child transform. This loop continues, with the childless transforms being removed in each iteration, until no transforms remain, at which point the process terminates (see SI for more details). The final parent edges are connected to an input stream node, which has a latency of zero.

While this approach is robust enough to handle the scenarios illustrated in Figure 4, more complex cases involving intricate serial and parallel processes may require advanced DAG construction strategies. Future work could explore such strategies in greater detail.

6. Results

Each automatic IPPM generation strategy was evaluated using two expression datasets, one related to loudness processing (Thwaites et al., 2017) and one related to horizontal motion processing (Wingfield et al., 2025). In the Thwaites et al (2017) study, human loudness processing is hypothesized to comprise of eleven transforms which characterize the loudness of the auditory environment. In Wingfield et al. (2025), motion processing is modelled as six transforms. Both processing streams were chosen due to their well-studied and self-contained nature.

All strategies had their hyperparameters tuned, optimizing for CV and TR. The first result of interest is that, this hyperparameter tuning caused all the strategies to converge on adopting the same approach: that of global max pooling. This resulted in all the approaches giving the same result – these are shown in in Table 1.

Loudness IPPM Evaluation			**Motion IPPM Evaluation**		
Metric	Hemisphere	Algorithm Score	Metric	Hemisphere	Algorithm Score
TR	Left	1	TR	Left	1
	Right	1		Right	1

	Both	1		Both	1
CV	Left	0	CV	Left	0
	Right	0.353		Right	0.111
	Both	0.353		Both	0.111

Table 1. The TR and CV for both the Loudness and Motion IPPMs. The evaluation for the left, right and both hemispheres are shown. All strategies converged on the same result, so only one result is shown here.

As a visual aid, we also print out a comparison of the manually inferred IPPMs reproduced from both Thwaites et al (2017) and Wingfield et al. (2025), and their automatically generated counterparts (Fig 6.). The automatically generated versions are relatively close to the hand versions, although there are some obvious differences, particularly in the Loudness IPPM, where the identity transforms (where x=y), are missing, together with some of other nodes and edges. Potential reasons for this are covered in the discussion.

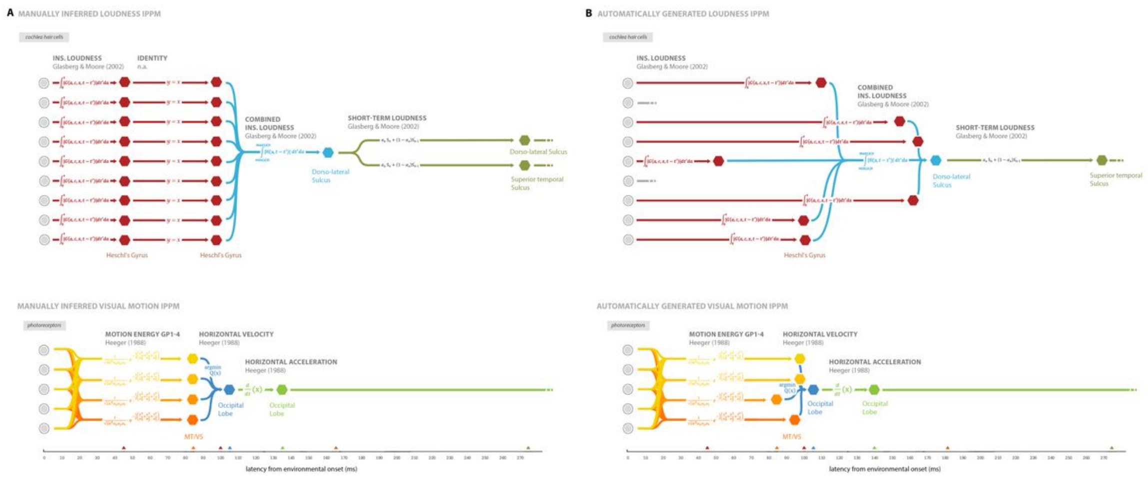

Fig 6. Comparison of the (A) manually inferred IPPMs created for (top) loudness processing (reproduced from Thwaites et al, 2017), and (bottom) motion processing (reproduced from Wingfield et al., 2025), and (B) their automatically generated counterparts. In both cases, the clusters from both hemispheres are used.

7. Discussion

7.1 Analysis of Clustering Strategies

Having compared several clustering strategies, we found that each converged to a global max-pooling strategy when optimizing on our evaluation metrics. It is likely that this convergence towards global max pooling stems from the fact that the evaluation metrics reward the strategies when they limit each transformation to one node. More specifically, TR is maximized after processing one node, so there is no benefit for the algorithm to incorporate additional clusters, and the addition of new nodes will likely deteriorate CV. As a result, both CV and TR pressure the generators to limit themselves to one node. This explains why the identity transforms are missing – multiple nodes for a transform are connected by null edges, representing the identity transform.

Although global max pooling optimizes both evaluation metrics, manual inspection of the expression maps do heavily imply that there should be multiple nodes loudness processing. To improve on this in future work one option would be to define an augmented TR: for example, GMMs can be performed prior to clustering, to identify the number of clusters, and an augmented TR could be defined as the average ratio of nodes to clusters over all transforms. This augmented TR would likely result in the retention of the identity transforms.

7.2 Analysis of Resultant IPPMs

All of the automatic generation strategies performed more poorly on the right hemisphere. Both Thwaites et al (2017), and Wingfield et al., (2025) assume that the hemispheres process the information in a symmetric fashion, so their IPPMs were expected to be consistent. The reason for this poor performance in the right atmosphere is not clear; it may be due to measurement error (both datasets were recorded in the same EMEG acquisition equipment), or due to random chance.

The loudness IPPM has a worse average CV than motion, suggesting that it's construction may be of slightly lower quality. Analysis of the stem plots and IPPMs reveals this is caused by two of the transforms for loudness appearing before their parent nodes. Although Motion also exhibits some causality violation, it has fewer transforms than Loudness, reducing the potential for such violations.

One notable point is the absence of TVL loudness channel 4 and 8 from the Loudness IPPM (marked as grey edges in figure 6B). Since the TR is maximized, this implies that the underlying expression data did not contain any significant spikes for channel 4 and 8. Indeed Thwaites et al (2017) note that they are missing in their own analysis, but they include these transforms in their manually inferred IPPM anyway; this underpins the importance of the current work removing such subjectivity from the IPPM generation procedure.

7.3 General Comments

A fundamental challenge with the methodology described here is the reliance on the same data for both training and testing, due to the small amount of data available. As a result, the clustering algorithms overfit, inflating the evaluating metric scores, and the ability of the approach to generalize across diverse circumstances is not known. This problem should be overcome as an increasing number of expression datasets become available.

A second challenge lies in the assumption that both the CTL and the expression data are ‘correct’. This assumption may not be true - transforms in the CTL are, by their nature, approximations of human sensory processing, and expression data is derived from relatively noisy EMEG data. Over time, it is likely that the accuracy of both will be improved, and this will affect the accuracy of the resultant IPPMs that are generated.

7.4 How could the results be improved?

There are three main ways we could improve the results: enriching the dataset quality, modifying the evaluation metrics, and modifying the clustering strategy.

Improving the Dataset

Improvements in the accuracy of the CTLs and expression maps would create narrower clusters, with higher peaks. This would make identifying distinct clusters easier.

Augmenting the expression data with the spatial location of each hexel may also improve performance. For example, the spatial proximity of two blurred clusters may hold valuable information that makes it easier to decide whether those blurred clusters are part of a single entity or disparate clusters. However, implementing this improvement is not trivial. Due to point spread error, expression clusters are spread across the folded cortical surface in ways that are rarely contiguous.

Modifying the evaluation metrics

There is likely scope to improve the evaluation metrics used here. As mentioned above, TR could be modified to better measure the amount of salient information retained by the clustering algorithm, particularly the modality of the number of clusters for a transform.

More generally, there may be aspects of 'good' IPPMs that are not adequately covered by TR and CV. One is the complex interplay between serial and parallel processing, especially when confronted with multiple nodes. A discussion on what these more advanced metrics might look like is beyond the scope of this work, but it may require a more nuanced view from the research community as to what 'good' IPPMs should look like.

Improving the Clustering Strategies

This study assumed that each transform exhibits the same pattern of noise, allowing the same set of hyperparameters to be applied for all transforms and leading to a transform-agnostic strategy. Hyperparameters could be set independently for each transform, but this is likely to yield only a modest improvement in performance at the cost of interpretability. Therefore, transform-agnostic strategies are preferred, even though they may result in slightly lower performance.

More generally however, considering all the cluster strategies converged on a single strategy, it seems likely that more example expression maps and CTLs will be a prerequisite to improving on their accuracy.

Conclusion

In this paper we presented a procedure which automates the process of building IPPM graphs from cortical expression maps in a data-driven, objective manner. Furthermore, we have presented two evaluation metrics by which researchers can compare competing methods for building IPPM graphs.

In testing a range of possible strategies for clustering expression maps, it was found that all approaches converged on a global max pooling strategy. However, this is likely due to a lack of data, and there may be many improvements that can be made to improve IPPM creation in the future.

IPPMs are a representation of cortical processing, and the ability for researchers working in neuroimaging to create them in a manner free of subjectivity is an important challenge in the field. This paper hopes to have provided an outline of some of the tools and metrics that will be needed for this task.

10. Acknowledgements

10. Funding Statement

This work was supported by a grant from Neuroverse FZCO to AT and CZ

References

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In B. N. Petrov & F. Csáki (Eds.), *2nd international symposium on information theory* (pp. 267–281). Budapest, Hungary: Akadémia Kiadó.

Bertsekas, D.P. (2016) Nonlinear Programming: 3rd Edition, *Athena Scientific*

Botev, Z.I., Grotowski, J.F., Kroese, D.P. (2010) Kernel density estimation via diffusion. *Annals of Statistics, 38*(5) 2916–2957. https://doi.org/10.1214/10-AOS799

Comaniciu, D., Meer, P. (2002) Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 603-619. https://doi.org/10.1109/34.1000236

Dempster, A.P.; Laird, N.M.; Rubin, D.B. (1977) "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of the Royal Statistical Society, Series B.* 39 (1): 1–38.

Ester, M., Kriegel, H.P., Sander, J., Xu, X. (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining,* 226-231.

Pearson, K. (1984) Contributions to the mathematical theory of evolution, *Philosophical Transactions of The Royal Society A* https://doi.org/10.1098/rsta.1894.0003

Grave de Peralta-Menendez, R., Gonzalez-Andino, S. L., and Lutkenhoner, B. (1996). Figures of merit to compare linear distributed inverse solutions. Brain. Topogr. 9:117–124. doi: 10.1007/BF01200711

Grave de Peralta-Menendez, R., and Gonzalez-Andino, S. L. (1998). A critical analysis of linear inverse solutions to the neuroelectromagnetic inverse problem. IEEE Trans. Biomed. Eng. 45, 440–8. doi: 10.1109/10.664200

Hauk O, Wakeman D. G., Henson R. (2011) Comparison of noise-normalized minimum norm estimates for MEG analysis using multiple resolution metrics, NeuroImage 54 (2011) 1966–1974 doi:10.1016/j.neuroimage.2010.09.053

Parzen, E. (1962) On estimation of a probability density function and mode. The Annals of Mathematical Statistics, 33(3):1065–1076

Pedregosa F, Varoquaux G., Gramfort, A, Michel, V., *et al.* (2011) Scikit-learn: Machine Learning in Python, *JMLR* 12, pp. 2825-2830

Rosenblatt, M. (1956). "Remarks on Some Nonparametric Estimates of a Density Function". *The Annals of Mathematical Statistics*. 27 (3): 832–837. doi:10.1214/aoms/1177728190.

Thwaites, A., Nimmo-Smith, I., Fonteneau, E., Patterson, R. D., Buttery, P., & Marslen-Wilson, W. D. (2015). Tracking cortical entrainment in neural activity: auditory processes in human temporal cortex. *Frontiers in computational neuroscience*, *9*, 5.

Thwaites, A., Wingfield, C., Wieser, E. Soltan, A., Marslen-Wilson, W.D., Nimmo-Smith, I. (2018) Tracking cortical entrainment to the CIECAM02 and CIELAB color appearance models in the human cortex. *Vision Research*. 145: 1–10 doi: 10.1016/j.visres.2018.01.011

Thwaites, A., Schlittenlacher, J., Nimmo-Smith, I., Marslen-Wilson W.D., Moore, B.C.J. (2017) Tonotopic representation of loudness in the human cortex. *Hearing Research*. 344: 244–254 doi:10.1016/j.heares.2016.11.015

Thwaites, A., Zhang, C., Woolgar, A. (2025) Information Processing Pathway Maps: — A Scalable Framework for Mapping Cortical Processing, *Preprint*, https://kymata.org/assets/preprints/intro_to_IPPM_preprint.pdf

Wingfield, C., Soltan, A., Nimmo-Smith, I., Marslen-Wilson, W.D., Thwaites, A. (2025) Tracking cortical entrainment to stages of optic-flow processing. *Vision Research* *226*(108523). doi:10.1016/j.visres.2024.108523

Supplementary information

Pareto Frontier for hyperparameter tuning

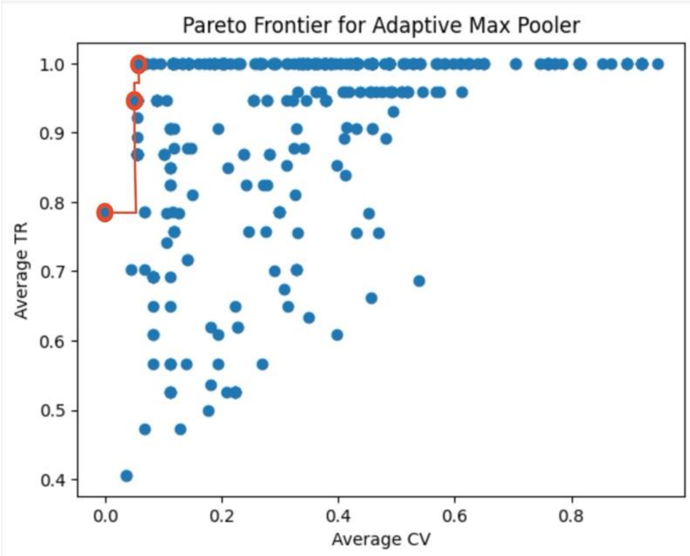

Figure SI1. The average TR and CV across hemispheres and datasets for different hyperparameter configurations. The red dots indicate settings of TR and CV that lie on the Pareto Frontier: the set of points where you cannot optimize one metric without deteriorating the performance of the other metric. To choose between these points, the one that maximises TR at a satisfactory CV was chosen.

IPPM Builder

The IPPM builder class takes the clustered expression plot and CTL as inputs and outputs a DAG. The CTL is passed as a dictionary containing transforms as keys and a list of parent transforms as the value. As such, the CTL contains all the information we need to define the set of nodes and edges. From the clustered expression plot, we retrieve the node size, which is proportional to the magnitude, and the location on the latency axis.

The builder algorithm iterates through the transforms, isolates the expression data for the selected transform, constructs nodes for each central temporal foci effect, and, finally, draws edges to any children transforms.

A top-level transform is defined as a transform that does not transmit information to other transforms, i.e., it is childless. The builder iterates through the transforms in a top-down fashion. To add an edge, we require knowledge of the final node for the parent and the initial node for the child. By generating the graph top-down, we initialize the information for the child before the parent, so when we create the parent, we have all the information we need to add an edge from the last parent node to the first child node. The builder continues this loop and removes top-level transforms in each iteration; eventually, there will be no more transforms left, so it will terminate.

Input streams are a special case of transforms since we do not observe any expression data for them. However, we know that the stimulus begins at latency = 0, so we can construct a node in the IPPM at latency = 0.

The time complexity is $O(n_T * n_H \log n_H + n_T^2)$. The $n_H \log n_H$ comes from sorting the central foci effects by latency, so we can rapidly add an edge from the last spike of a parent to the first spike of the child. Currently, the first term dominates the time complexity since n_H is about 10,024. As the number of transforms increases, specifically past 10,024, we expect the n_T^2 to dominate.