

STFEEG-Tool: A Spatial-Temporal-Frequency EEG Analysis Tool for Motor Imagery Brain-Computer Interfaces

Rui Zhang¹, Weidong Zhou^{1,2}, Yong Wang¹, Guoyang Liu^{1,2}

¹School of Integrated Circuits, Shandong University²Shenzhen Research Institute of Shandong University

✉ Corresponding Authors : Yong Wang <yongw@sdu.edu.cn>, Guoyang Liu <gyliu@sdu.edu.cn>

Citation

Zhang, R., Zhou, W., Wang, Y., Liu, G. STFEEG-Tool: A Spatial-Temporal-Frequency EEG Analysis Tool for Motor Imagery Brain-Computer Interfaces. *J. Vis. Exp.* (), e70425, doi:10.3791/70425. (2026).

Abstract

Motor imagery-brain-computer interfaces (MI-BCIs) have demonstrated significant potential for neurorehabilitation and cognitive neuroscience. However, a standardized and reproducible MI-EEG workflow for configurable spatial-temporal-frequency feature analysis remains limited, and many pipelines require complex configuration and parameter tuning with limited interpretability, hindering practical deployment and generalization. To address these challenges, an STFEEG-Tool was developed to provide a user-friendly, standardized, and interpretable workflow for EEG decoding in MI paradigms. STFEEG-Tool enables fine-grained configuration of temporal, frequency-band, and spatial segmentation, allowing the extraction of multiscale MI features. The toolbox integrates multiple feature extraction algorithms, including common spatial patterns (CSP) and divergence-based CSP (div-CSP), along with various classifiers, such as support vector machines (SVMs), Ridge Regression Classifier, and Lasso Classifier. A dynamic time-frequency scalp topographical map is provided to summarize spatial patterns across time-frequency segments and support interpretation of decoding results. Overall, STFEEG-Tool serves as a reproducible and extensible platform for fine-grained MI-EEG analysis, facilitating the translation of fine-grained decoding pipelines into practical, user-oriented applications.

Introduction

Motor imagery brain-computer interfaces (MI-BCIs) enable direct communication between the human brain and external devices without requiring peripheral muscle activity¹. In recent years, MI-BCI technology has been widely applied in neurorehabilitation², assistive control³, and human-computer interaction⁴. These applications highlight the need for accessible and reproducible analysis workflows that can be reliably adopted by researchers and clinicians, including users with limited programming expertise.

Conventional MI-EEG decoding approaches are predominantly based on spatial filtering and linear classification frameworks. Methods such as Common Spatial Pattern⁵ (CSP) and divergence-based CSP⁶ (div-CSP) have been widely used to extract

discriminative spatial features from multi-channel EEG signals. To further extract features across distinct frequency components, Filter Bank CSP⁷ (FBCSP) and Sub-band CSP^{8,9} (SBCSP) have been extensively explored. In addition, temporal segmentation has been integrated into the feature extraction process by dividing each trial into multiple time intervals, allowing the capture of evolving discriminative patterns and further enhancing decoding performance^{10,11,12}. To enhance the discriminability of extracted features, various feature selection strategies have been introduced to reduce redundancy and mitigate overfitting^{13,14,15}. For classification, traditional machine learning models such as support vector machines¹⁶ and regularized regression classifiers^{17,18,19} continue to be widely used because of their interpretability and computational efficiency.

Despite their effectiveness, these approaches are typically implemented in customized, dataset-specific codebases, which complicates systematic reuse and independent verification. Several graphical user interface (GUI)-based software platforms have been developed to support EEG analysis and BCI research. For example, EEGLAB²⁰ offers a comprehensive suite of functions for EEG preprocessing, time-frequency analysis, feature extraction, classification, and advanced visualization. Its highly modular and extensible architecture has enabled broad adoption within the EEG research field. BCILAB²¹, in integration with EEGLAB, incorporates over 100 signal processing and machine learning algorithms spanning the entire decoding pipeline, from preprocessing to classification. Similarly, OpenVIBE²² offers a visual programming environment for designing real-time BCI workflows, in which users can assemble signal-processing pipelines by graphically linking modular components. The platform is particularly suited for online EEG acquisition and real-time stimulus-response paradigms, and has been widely used in the development and testing of interactive BCI applications. In addition, PyNoetic, a modular Python framework for no-code EEG brain-computer interface development, enables end-to-end BCI workflow construction through an integrated GUI supporting data acquisition, preprocessing, feature extraction, and classification²³. Unlike earlier tools that require substantial manual configuration or scripting, PyNoetic offers a higher level of abstraction for pipeline design, enabling flexible assembly and reconfiguration of processing stages without low-level programming. In many MI-EEG studies, these platforms can already support routine preprocessing, conventional feature extraction, and baseline decoding and evaluation.

Several GUI-based platforms support MI-EEG analysis through standardized preprocessing, feature extraction, and baseline decoding (Table 1). However, they lack optimization for fine-grained spatial-frequency-time decomposition, including user-defined spatial partitioning and cross-domain visualization. To address these methodological needs, a fine-grained spatial-temporal-frequency decoding framework for MI-EEG²⁴ was previously proposed, jointly modeling spatial electrode groups, frequency sub-bands, and temporal segments to improve decoding performance and enhance multi-domain interpretability. Building on this framework, the Spatial-Temporal-Frequency EEG analysis tool (STFEED-Tool) was developed as a graphical user interface (GUI)-based toolbox that standardizes fine-grained MI-EEG decoding within a configurable and reproducible workflow while remaining compatible with existing EEG toolchains.

STFEED-Tool integrates data preparation, preprocessing, spatial-temporal-frequency segmentation, feature extraction, wrapper-based feature selection, classification, and visualization into a unified pipeline. Rather than replacing established EEG toolboxes, it complements them through transparent parameter specification,

consistent pipeline execution, and model-informed visualization to support fine-grained analysis.

The main contributions of this work are threefold. First, STFEED-Tool provides a user-friendly, end-to-end GUI workflow with standardized parameter configuration for fine-grained MI-EEG decoding. Second, it enables multiscale spatial-temporal-frequency segmentation, including user-defined spatial decomposition, which is not consistently available on general-purpose platforms. Third, it enhances interpretability through time-frequency topographical maps and electrode-group significance maps that summarize discriminative contributions and reveal subject-specific MI-relevant patterns across domains.

The toolbox is open-source and available at: https://github.com/CissyZR/Spatial-Frequency-Temporal_EEG_Analysis_Tool

Protocol

The data used in this protocol were obtained from publicly available, de-identified datasets. The original data collection procedures were conducted in accordance with the ethical standards of the respective institutions; therefore, no additional institutional approval was required for this secondary analysis.

Figure 1 illustrates the overall workflow of the Spatial-Temporal-Frequency Electroencephalography Analysis Tool. Follow the steps below to configure each module and reproduce the fine-grained spatial-frequency-time decoding protocol. The software used in this study is listed in the **Table of Materials**.

1. Preparation of the software and STFEED-Tool package

1. Install a numerical computing environment that supports MAT-file processing and GUI execution (see Table of Materials).
2. Download **STFEED - Tool** from the online repository, unzip it to a local folder, and locate the installer file **STFEED-Tool.mlappinstall**.
3. Double-click on **STFEED - Tool.mlappinstall**, and follow the on-screen instructions to complete the installation.
4. Open the App tab, confirm that STFEED-Tool appears in the installed apps list (**Figure 2**), and click on it to launch the GUI (**Figure 3**).

2. Preparation of the input data MAT-file

1. Create a **MAT - file (.mat)** containing four variables named *TrainData*, *TrainLabel*, *TestData*, and *TestLabel*.
 1. Store *TrainData* and *TestData* as 3-D arrays of size $N_{\text{trials}} \times N_{\text{channels}} \times N_{\text{timepoints}}$

2. Store *TrainLabel* and *TestLabel* as $N_{\text{trials}} \times 1$ column vectors of integer class labels, using a consistent encoding across training and test sets.

NOTE: Construct *TrainData* and *TestData* from non-overlapping sessions or recording blocks to prevent information leakage.

2. Ensure that *TrainData* and *TestData* share the same channel order and sampling frequency (Hz).
3. (Optional) Download the provided example MAT-file (*Data.mat*) based on BCI Competition IV 2a from the public repository to quickly test the software.

3. Launching the GUI and set paths

1. Launch the **STFEEG - Tool** GUI to open the MI-EEG Analysis window (**Figure 3**).
2. In the *Data Loading* panel, click on **Browser** next to *Data Path* and select the folder that contains the prepared MAT-file.
3. Confirm that the sampling frequency F_s displayed in the GUI matches the dataset sampling frequency (Hz).
4. Click on **Browser** next to *Save Path* and select an output folder to store results (e.g., configuration files, model files, and predicted labels).
5. (Optional) Specify *Model Path* based on the intended use.
 1. To save a model during training, click on **Browser** next to *Model Path* and select a destination folder; if *Model Path* is left empty, the software automatically saves the model in *Save Path*.
 2. To load a pretrained model for testing, click on **Browser** next to *Model Path* and select the folder that contains the saved model files.
6. Click on **Load** in the *Data Loading* panel to import the MAT-file. The software automatically checks the required variables and selected paths, and loads the dataset if no errors are reported.

4. Configuring the preprocessing (band-pass filtering)

1. Use the default band-pass filter settings shown in the GUI ($Order = 5$, $Fc1 = 5$ Hz, $Fc2 = 35$ Hz, filter type = *IIR*) unless a different preprocessing configuration is required.
2. (Optional) Enter the filter *Order* to override the default value.
3. (Optional) Enter the lower and upper cutoff frequencies ($Fc1$ and $Fc2$) in Hz to override the default values.
4. (Optional) Select the filter type from the drop-down menu to override the default option.

5. Click on **OK** to confirm the preprocessing configuration. If no changes are made, the software applies the default settings shown in the panel.

5. Configuring the fine-grained segmentation strategy

1. In the *Segmentation Strategy* panel of the main GUI, click on **Edit** to open the *Segmentation* configuration window.
2. Configure temporal windows.
 1. Select the *Temporal Windows* tab.
 2. Review the existing window list shown in the table (**Figure 4**).
 3. (Optional) To add windows, enter the *Start*, *End*, and *Interval* values, and click on **Add** to append the generated windows to the list.
 4. (Optional) To delete an existing window, right-click on the corresponding row in the table and select **Delete**.
3. Configure filter-bank frequency bands.
 1. Select the *Frequency Bands* tab (**Figure 5**).
 2. Review the existing frequency-band list shown in the table.
 3. (Optional) To add bands, enter the *Start* (Hz), *End* (Hz), and *Interval* values, and click on **Add** to append the generated bands to the list.
 4. (Optional) To delete an existing band, right-click on the corresponding row in the table and select **Delete**.
4. Configure spatial segmentation by channel groups (**Figure 6**).
 1. Select the *Spatial Segmentation* tab.
 2. Review the channel-group assignments in the *Groups of Channel Number* list.
 3. (Optional) To replace the current channel-group configuration, click on **Browser** next to *Channel Groups* to select a channel-group definition MAT-file.
 4. (Optional) To enable scalp visualizations, click on **Browser** next to *Channel Location* to select the channel-location file for the dataset channels.

NOTE: The public repository provides *savedChCell.mat* (channel-group definitions) and *channel_locations_22ch.txt* (22-channel scalp coordinates for BCI Competition IV 2a), which can be used directly with the included example data.

 5. (Optional) Click on **Replace** to load the selected channel-group file and update the *Groups of Channel Number* list. If a channel-location file has been selected in *Channel*

Location, the tool also loads it for subsequent scalp visualizations (**Figure 7**).

5. Apply and verify the segmentation settings.
 1. After completing the settings in all tabs, click on **OK** at the bottom of the *Segmentation* window to apply the updated temporal windows, frequency bands, and spatial segmentation.
 2. Return to the main GUI and confirm that the updated time bands, frequency bands, and channel groups are reflected in the *Segmentation Strategy* panel.

6. Configuring the feature extraction, classifiers, and cross-validation

1. In the *Classification Configuration* panel, select the *Feature Extraction* method from the drop-down menu.

NOTE: Detailed descriptions of CSP and div-CSP are provided in **Supplementary File 1**.

2. Set *nCSP* to select CSP features from both ends of the feature vector. The tool retains the first *nCSP* and the last *nCSP* components ($2 \times nCSP$ features in total).
3. Select the classifier(s) by checking the corresponding boxes (SVM, Ridge-Optimal, Lasso-Optimal).

NOTE: Descriptions of the three classifiers are provided in **Supplementary File 2**.

4. Set *Selection Bands Number* to specify how many fine-grained segments/features are retained in wrapper-based selection.
5. Set *Cross-Validation* to specify the number of folds for k-fold cross-validation on *TrainData/TrainLabel*.
6. (Optional) Enable *Parallel* to accelerate computation when multiple segments and classifiers are evaluated, if sufficient computational resources are available.
7. Click on **Cross - Validation** to perform k-fold cross-validation on *TrainData/TrainLabel* and train the final model, then test the final model on *TestData/TestLabel*.

NOTE: If multiple classifiers are selected, the tool automatically selects and saves the model with the highest cross-validation accuracy, then uses it for test-set evaluation.

7. Viewing the results and visualizations

1. After clicking on **Cross - Validation** in Step 6, review the performance summary displayed in the *Results* panel, including *Validation Accuracy/Kappa*, *Test Accuracy/Kappa*, and the *Per-fold Training Time* and *Per-fold Testing Time*. Record these values for reporting.

NOTE: *Validation Accuracy/Kappa* is reported as the mean k-fold cross-validation accuracy/kappa²⁵ computed on

TrainData/TrainLabel. *Test Accuracy/Kappa* is reported as the mean test-set accuracy/kappa²⁵ obtained by evaluating the best model from each fold on *TestData/TestLabel* and averaging the test results across folds.

2. Inspect the visualization outputs displayed in the *Results* panel.
 1. Review the *time-frequency topographical map* to identify discriminative patterns across time, frequency, and scalp regions.
 2. Review the *Electrode Group Significance Map* to identify channel groups that contribute most to decoding.

NOTE: Detailed definitions, construction procedures, and interpretation guidelines for both maps are provided in **Supplementary File 3**.

3. (Optional) Run test-only evaluation using a previously trained model.
 1. Click on **Test Load Model** to load a pretrained model from *Model Path* and evaluate it on the current input *TestData/TestLabel*.
 2. Select the *Model Path* folder saved during training (Step 3.5) and keep its directory structure unchanged. Ensure that *model.mat* is in the selected folder and *param.mat* is in the parent folder, as both are required to load the model and reproduce the visualizations.
 3. Confirm that, in this mode, the GUI reports *Test Accuracy* and *Per-fold Testing Time* for the loaded model, and the visualization panels update accordingly.
 4. (Optional) Locate the exported results in the save directory, which include predicted and true labels for further statistical testing.

Results

The STFEED-Tool was validated on both the BCI Competition IV 2a²⁶ and the BCI Competition III 3a²⁷, and the system successfully executed the complete decoding pipeline in both cases. The fixed cross-session protocol of BCI Competition IV 2a was followed: training on Session 1 and evaluation on the held-out Session 2, recorded on two separate days. Under the same configuration used in the previously reported fine-grained spatial-temporal-frequency framework²⁴, STFEED-Tool reproduced the original performance on BCI Competition IV 2a, achieving a mean test accuracy of 83.68%²⁴.

As shown in the results panel on the right in **Figure 8**, the validation and test accuracies and kappa were reported to provide a quantitative summary of model performance. Below these metrics, the time-frequency topographical map illustrated the spatial distribution of discriminative EEG patterns across

data-driven frequency and temporal segments. These maps were computed by averaging the time-frequency score (TFS) values within each segment, thereby revealing subject-specific activation characteristics across the time-frequency domain. The resulting topographical patterns indicated the scalp regions that contributed most strongly to decoding. Examination of the frequency axis identified the sub-bands that were most informative at different time intervals, whereas examination of the time axis

indicated when each sub-band contributed most to discrimination. In parallel, the Electrode Group Significance Map quantified the contribution of predefined electrode groups, providing a complementary summary of the relative importance of different spatial regions. Further benchmarking and validation details of the underlying fine-grained framework were described in previous work²⁴.

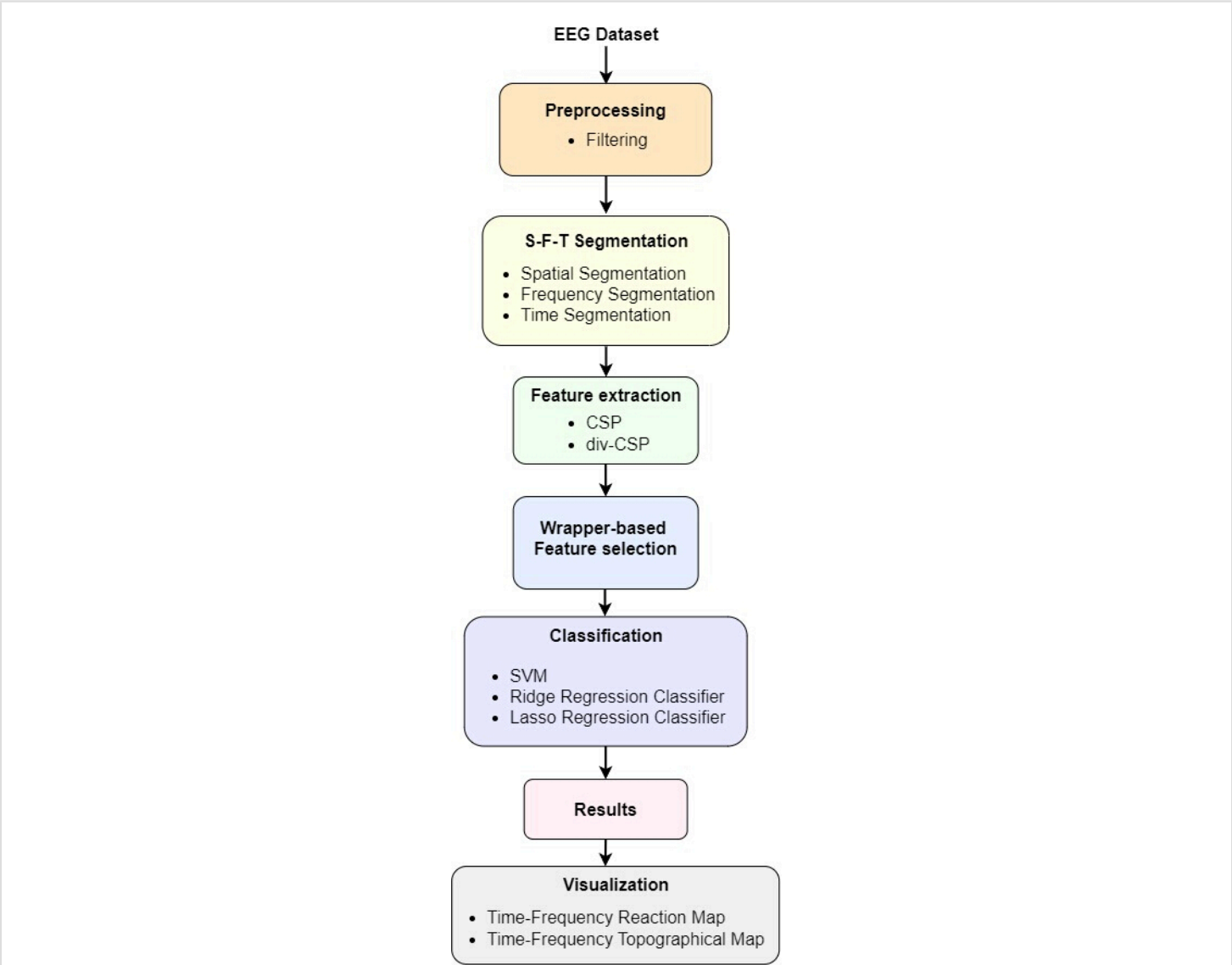


Figure 1: The overview of the STFEEG-Tool architecture. Illustrating the overall processing pipeline of the STFEEG-Tool from EEG data input to spatial-frequency-temporal analysis, classification, and result visualization. [Please click here to view a larger version of this figure.](#)

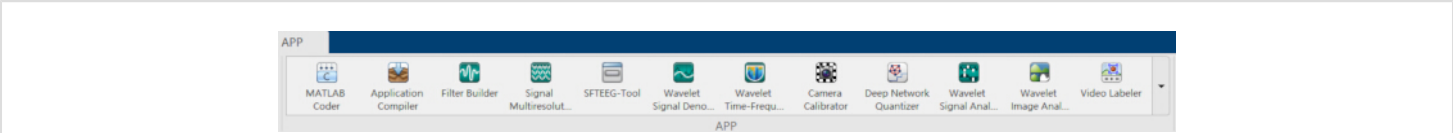


Figure 2: STFEEG-Tool icon in the Apps toolbar. The STFEEG-Tool icon appears in the Apps toolbar after installation. [Please click here to view a larger version of this figure.](#)

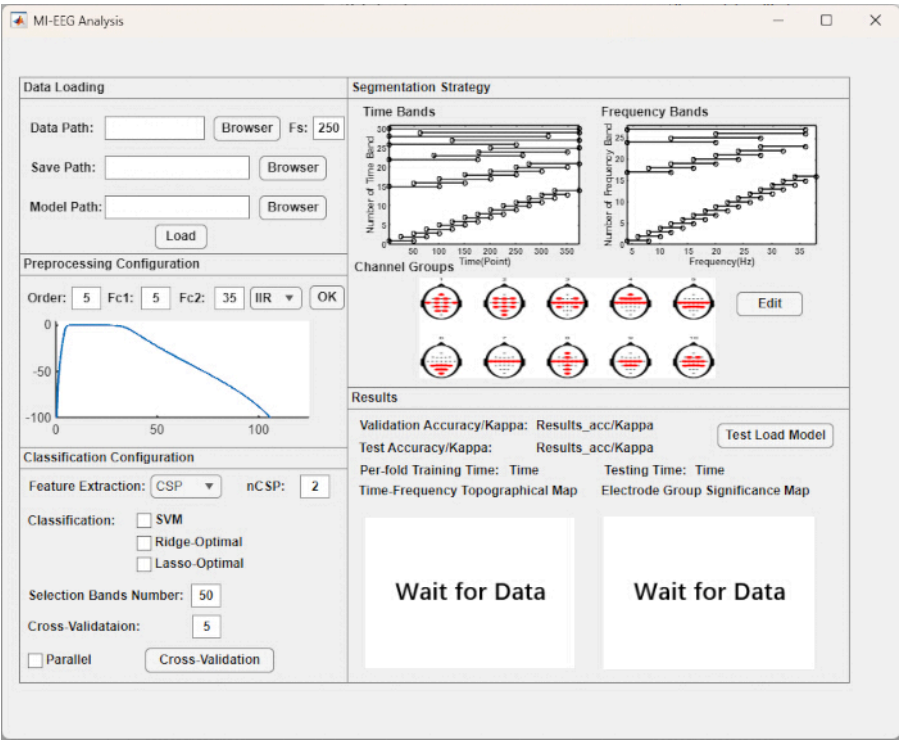


Figure 3: The GUI of STFEED-Tool. The main graphical user interface (GUI) integrated data loading, preprocessing, segmentation, classification, and results visualization, and was used to configure parameters and visualize decoding results. [Please click here to view a larger version of this figure.](#)

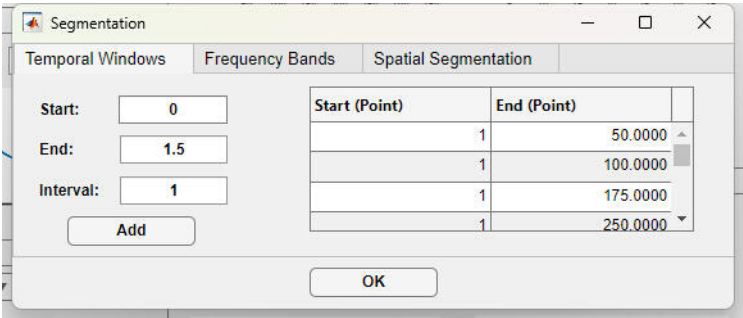


Figure 4: Temporal windows configuration panel. The panel displayed the list of temporal windows used for segmentation. [Please click here to view a larger version of this figure.](#)

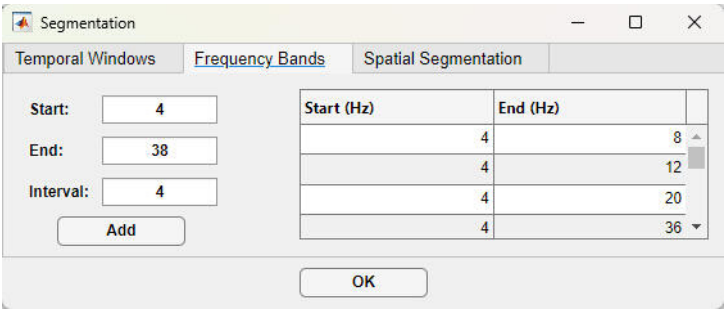


Figure 5: Frequency bands configuration panel. The panel displayed the frequency-band list used for segmentation. [Please click here to view a larger version of this figure.](#)

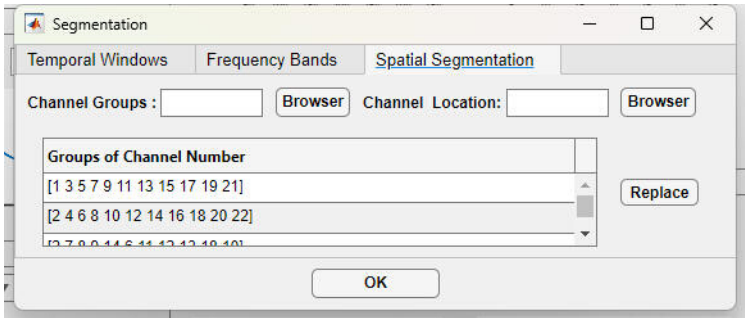


Figure 6: Spatial segmentation configuration panel. The panel displayed the channel-group definition used for spatial segmentation. [Please click here to view a larger version of this figure.](#)

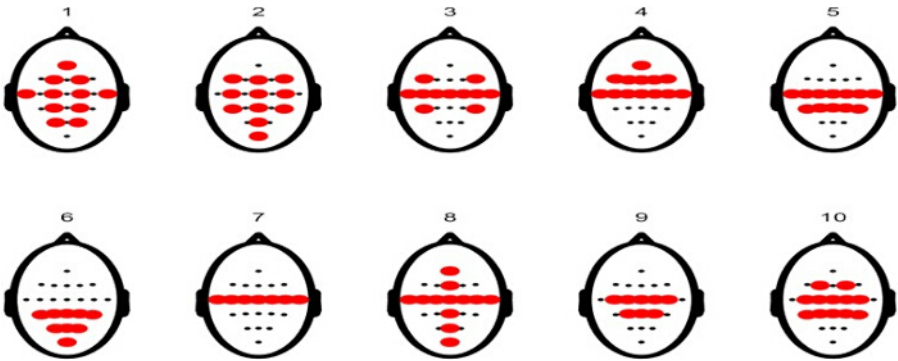


Figure 7: Visualization of default channel group configuration. The default channel configuration divided EEG channels into 10 groups according to the BCI Competition IV 2a dataset. [Please click here to view a larger version of this figure.](#)

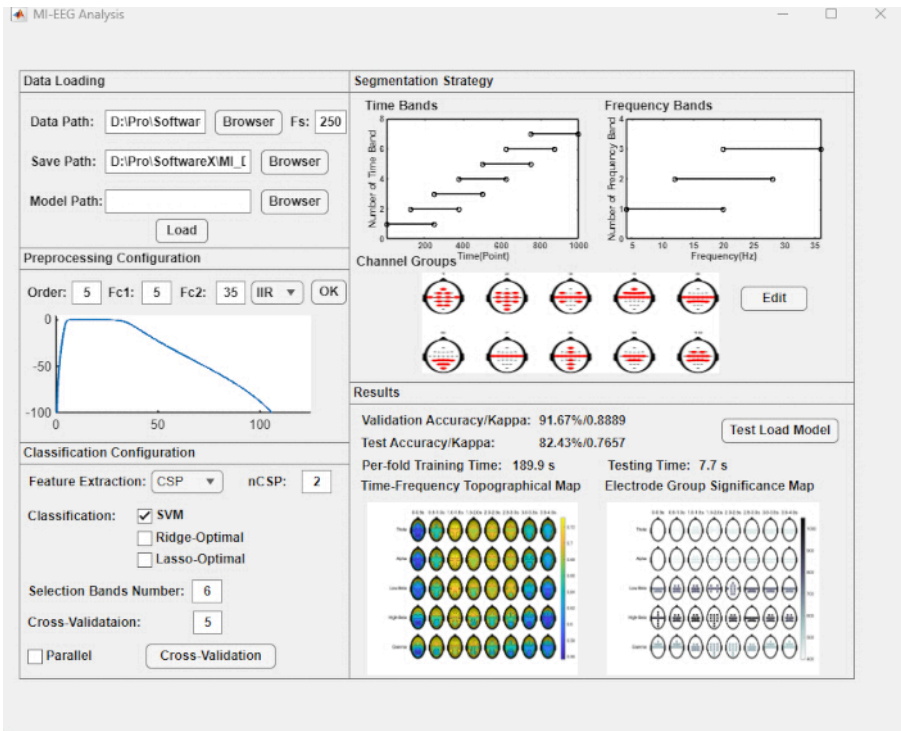


Figure 8: The GUI of STFEED-Tool with illustrative results. The Results module displayed a time-frequency topographical map and an Electrode Group Significance Map. [Please click here to view a larger version of this figure.](#)

Platform	Spatial-Frequency-Temporal Segmentation	Feature Selection	Classifier Support	Spatial-Frequency-Temporal Visualization	GUI	Open Source
EEGLAB ²²	Partial	Yes	Yes	Yes	Yes	Yes
BCILAB ²³	Partial	Yes	Yes	Partial	Yes	Yes
OpenViBE ²⁴	Partial	Partial	Yes	Partial	Yes	Yes
MNE-Python ²⁸	Partial	Partial	Partial	Partial	No	Yes
NeuroPy ²⁹	Partial	Yes	Partial	Yes	Yes	No
PyNoetic ²⁵	Partial	Yes	Yes	Partial	Yes	Yes
STFEED-Tool	Yes	Yes	Yes	Yes	Yes	Yes

Table 1: The comparison of different EEG platforms. This table compares representative EEG analysis platforms across spatial-frequency-temporal segmentation, feature selection, classifier support, graphical user interface, and open-source availability.

Supplementary File 1: Mathematical formulation of CSP and div-CSP. Formal definitions and optimization procedures for CSP and divergence-based CSP (div-CSP), including covariance normalization, generalized Rayleigh quotient formulation, KL-divergence objective function, and manifold-based optimization for spatial filter estimation. [Please click here to download this File.](#)

Supplementary File 2: Optimization objectives for classifiers. Mathematical formulations of the SVM, Ridge-regularized, and Lasso-regularized classifiers, including regularization terms and cross-validation-based selection of the regularization parameter. [Please click here to download this File.](#)

Supplementary File 3: Visualization Methods for Time-Frequency Topographical Maps and Electrode Group Significance Maps. To facilitate interpretable analysis of MI-EEG decoding, the proposed system integrates two complementary visualization modules. These visual representations summarize the discriminative contributions of spatial-frequency-temporal segments (SFTSs), enabling users to identify effective temporal-frequency patterns and subject-specific spatial relevance. [Please click here to download this File.](#)

Discussion

STFEEG-Tool is designed as an MI-EEG-specific toolbox that complements established EEG ecosystems such as EEGLAB²⁰, BCILAB²¹, OpenViBE²², PyNoetic²³. While these platforms provide flexible components for MI decoding, fine-grained spatial-temporal-frequency configuration and model-informed interpretation may involve additional integration across processing steps. This is particularly relevant when explicit electrode-group decomposition is included. Accordingly, STFEEG-Tool provides a unified GUI workflow with standardized outputs and integrated visualization for fine-grained, multi-scale MI-EEG decoding.

Several steps are critical for obtaining valid offline results. First, ensure that the input MAT-file follows the required data format and that key metadata (channel order and sampling frequency) are consistent between training and testing sets (Step 2.1 and Step 2.2). To prevent data leakage, training and test data should be constructed from strictly non-overlapping sessions or blocks. Second, STFEEG-Tool supports fine-grained multiscale spatial-temporal-frequency segmentation (Step 5), enabling both reproduction of the segmentation scheme described in the previous framework²⁴, and user-defined customization with explicitly recorded configurations. In particular, users may adjust temporal windows, channel-group definitions, and frequency-band partitioning, including the number and boundaries of sub-bands, to accommodate subject-specific rhythms.

When decoding performance is suboptimal, parameter adjustments, such as the band-pass filter configuration (Step 4),

temporal window length (Step 5.2), channel-group granularity (Step 5.4), and nCSP/classifier settings (Steps 6.1-6.3), can be used to adapt the pipeline to different datasets. To manage computational cost, users may reduce the number of temporal windows (Step 5.2), frequency bands (Step 5.3), or channel groups (Step 5.4), or enable parallel execution (Step 6.4). For interpretation, the time-frequency scalp topographical map and the electrode-group significance map provide complementary views of discriminative spectral-temporal components and spatial patterns (Step 7).

The main limitations of the current release stem from its offline analysis setting and fine-grained model selection. STFEEG-Tool does not currently support real-time data acquisition or closed-loop operation. In addition, computational demands increase with the number of segmentation units and the use of feature selection. Therefore, practical configurations should balance performance against computational cost. Future developments will focus on integrating real-time streaming interfaces, adding online adaptation modules, and extending the library of more efficient feature extraction and selection approaches. Beyond decoding performance, the segment-wise maps generated by STFEEG-Tool may facilitate exploratory neuroscience analyses and inform experimental design and individualized rehabilitation protocols.

Disclosures

The authors have no conflicts of interest to declare.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (No.62401342 and No.62271291), the Natural Science Foundation of Shandong Province (No.ZR2024QF092, ZR2024LZH007, and ZR2025ZD24), the Guangdong Basic and Applied Basic Research Foundation (No.2025A1515011826), the Shenzhen Fundamental Research Program (No.JCYJ20250604124702003), the Sichuan Provincial Key Laboratory of Philosophy and Social Science for Language Intelligence in Special Education (No.YYZN-2025-11), the National Key Research and Development Program of China (No.2024YFC2418300 and 2024YFC2418303), and the Key Laboratory of Social Computing and Cognitive Intelligence (Dalian University of Technology), Ministry of Education (No. SCCI2025YB02).

References

1. Guger, C., Ramoser, H., Pfurtscheller, G. Real-time EEG analysis with subject-specific spatial patterns for a brain-computer interface (BCI). *IEEE Trans Rehabil Eng.* **8** (4), 447-456 (2000).

2. Chaudhary, U., Birbaumer, N., Ramos-Murguialday, A. Brain-computer interfaces for communication and rehabilitation. *Nat Rev Neurol.* **12** (9), 513-525 (2016).
3. Khan, M. A., Das, R., Iversen, H. K., Puthusserypady, S. Review on motor imagery-based BCI systems for upper limb post-stroke neurorehabilitation: From designing to application. *Comput Biol Med.* **123**, 103843 (2020).
4. Pfurtscheller, G., Neuper, C. Motor imagery and direct brain-computer communication. *Proc IEEE.* **89** (7), 1123-1134 (2001).
5. Jiang, A. et al. Efficient CSP algorithm with spatio-temporal filtering for motor imagery classification. *IEEE Trans Neural Syst Rehabil Eng.* **28** (4), 1006-1016 (2020).
6. Samek, W., Kawanabe, M., Müller, K.-R. Divergence-based framework for common spatial patterns algorithms. *IEEE Rev Biomed Eng.* **7**, 50-72 (2013).
7. Ang, K. K., Chin, Z. Y., Zhang, H., Guan, C. Filter bank common spatial pattern (FBCSP) in motor imagery BCI. *Proc IEEE Int Joint Conf Neural Netw. (IJCNN)*, Hong Kong. 2390-2397 (2008).
8. Kumar, S., Sharma, A., Tsunoda, T. An improved discriminative filter bank selection approach for motor imagery EEG signal classification using mutual information. *BMC Bioinformatics.* **18** (16), 545 (2017).
9. Zhang, X., She, Q., Chen, Y., Kong, W., Mei, C. Sub-band target alignment common spatial pattern in brain-computer interface. *Comput Methods Programs Biomed.* **207**, 106150 (2021).
10. Yang, J., Ma, Z., Shen, T. Multi-time and multi-band CSP motor imagery EEG feature classification algorithm. *Appl Sci.* **11** (21), 10294 (2021).
11. Hwang, J., Park, S., Chi, J. Improving multi-class motor imagery EEG classification using overlapping sliding window and deep learning model. *Electronics.* **12** (5), 1186 (2023).
12. Liu, G., Tian, L., Zhou, W. Multiscale time-frequency method for multiclass motor imagery brain computer interface. *Comput Biol Med.* **143**, 105299 (2022).
13. Liu, A. et al. Feature selection for motor imagery EEG classification based on the firefly algorithm and learning automata. *Sensors.* **17** (11), 2576 (2017).
14. Kabir, M. H. et al. Investigating feature selection techniques to enhance the performance of EEG-based motor imagery tasks classification. *Mathematics.* **11** (8), 1921 (2023).
15. Wang, A., An, N., Chen, G., Li, L., Alterovitz, G. Accelerating wrapper-based feature selection with k-nearest-neighbor. *Knowl Based Syst.* **83**, 81-91 (2015).
16. Andrews, S., Tsochantaridis, I., Hofmann, T. Support vector machines for multiple-instance learning. *Adv Neural Inf Process Syst.* **15**, 577-584 (2002).
17. You, Y., Chen, W., Zhang, T. Motor imagery EEG classification based on flexible analytic wavelet transform. *Biomed Signal Process Control.* **62**, 102069 (2020).
18. Jin, J. et al. Correlation-based channel selection and regularized feature optimization for MI-based BCI. *Neural Netw.* **118**, 262-270 (2019).
19. Chen, S., Li, N., Kong, X., Huang, D., Zhang, T. Electroencephalography-based motor imagery classification using multi-scale feature fusion and adaptive lasso. *Big Data and Cogn Comput.* **8** (12), 169 (2024).
20. Swartz Center for Computational Neuroscience. *What is EEGLAB?*. SCCN, University of California San Diego. Available at: <https://sccn.ucsd.edu/eeglab/index.php> Accessed January (2025).
21. Zander, T. O. *BCILAB Documentation*. Available at: <https://github.com/sccn/BCILAB/wiki/BCILAB-documentation> (2020).
22. Tarara, P., Przybył, I., Schöning, J., Gunia, A. Motor imagery-based brain-computer interfaces: An exploration of multiclass motor imagery-based control for Emotiv EPOC X. *Front Neuroinform.* **19**, 1625279 (2025).
23. Singh, G., Chharia, A., Upadhyay, R., Kumar, V., Longo, L. Pynoetic: A modular Python framework for no-code development of EEG brain-computer interfaces. *PLoS ONE.* **20** (1), e0327791 (2025).
24. Liu, G., Zhang, R., Tian, L., Zhou, W. Fine-grained spatial-frequency-time framework for motor imagery brain-computer interface. *IEEE J Biomed Health Inform.* **29**, 4121-4133 (2025).
25. Allison, B. Z., Dunne, S., Leeb, R., Millán, J. D. R., Nijholt, A. *Towards Practical Brain-Computer Interfaces: Bridging the Gap from Research to Real-World Applications*. Springer Berlin (2012).
26. Ang, K. K., Chin, Z. Y., Wang, C., Guan, C., Zhang, H. Filter bank common spatial pattern algorithm on BCI competition IV datasets 2a and 2b. *Front Neurosci.* **6**, 39 (2012).
27. Dornhege, G., Blankertz, B., Curio, G., Müller, K.-R. Boosting bit rates in noninvasive EEG single-trial classifications by feature combination and multiclass paradigms. *IEEE Trans Biomed Eng.* **51** (6), 993-1002 (2004).
28. Das, R. K., Martin, A., Zuraes, T., Dowling, D., Khan, A. A survey on EEG data analysis software. *Sci.* **5** (2), 23 (2023).
29. Omar, L. VT2: A Python Pipeline for EEG Processing. Master's thesis, ZHAW - Zurich University of Applied Sciences, Available at: https://www.zhaw.ch/storage/engineering/institute-zentren/cai/studentische_arbeiten/Spring_2024/VT2_MSE_24_PyThon_pipeline_lone.pdf (2024).