SoftwareX 31 (2025) 102316

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

PyFCS GUI: A new visualization and interaction framework for fuzzy color spaces

Rafael Vazquez-Conejo [a,c,*], Maria Tejada-Casado [a,c], Luis Javier Herrera [b,c], Razvan Ghinea [a,c], Jose Manuel Soto-Hidalgo [b,c]

[a] Department Optics, University of Granada, Spain
[b] Department Computer Engineering, Automation and Robotics, University of Granada, Spain
[c] Instituto de Investigación Biosanitari, ibs.Granada, 18011, Granada, Spain

ARTICLE INFO

Keywords:
Python
Fuzzy colors
Open-source

ABSTRACT

A new graphical user interface (GUI) has been developed as an extension to the open-source PyFCS library for constructing and analyzing fuzzy color spaces. This GUI improves usability and accessibility by allowing users to generate fuzzy color spaces from palettes or images, visualize them interactively in 3D, and perform advanced color mapping. Parametric controls enable users to explore various levels of generalization and specificity in the color modeling process. The resulting fuzzy color spaces can be exported in reusable formats to promote reproducibility. This extension significantly enhances the practical applicability of fuzzy color models, providing a versatile tool for both scientific research and artistic or perceptual analysis.

Code metadata

Current code version	v1.0.2
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-25-00412
Permanent link to Reproducible Capsule	
Legal Code License	MIT License
Code versioning system used	git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	Python ≥ 3.10 matplotlib==3.8.0 numpy==1.26.1 opencv-python==4.8.1.78 pillow==10.0.1 plotly==5.22.0 PyQt5==5.15.11 scikit-image==0.22.0 scikit-learn==1.3.2 scipy==1.11.3 PyQtWebEngine==5.15.7
If available Link to developer documentation/manual	https://github.com/RafaelConejo/PyFCS_GUI/tree/main/PyFCS_GUI_Manual
Support email for questions	rafaconejo@ugr.es

1. Motivation and significance

Color perception is a fundamental topic in vision science and plays a key role in various computational applications, including computer vision, image retrieval, and computational aesthetics [1–3]. However, bridging the gap between objective, numerical representations of color and the subjective, context-dependent ways in which humans perceive and describe color remains a persistent challenge [4,5]. This discrepancy — commonly referred to as the *semantic gap* — complicates tasks involving semantic interpretation of color [6].

Fuzzy logic has long provided a principled framework for modeling linguistic color categories as fuzzy subsets within perceptually grounded spaces [5,7–9]. Two main approaches have emerged in this domain: the first relies on exhaustive perceptual experiments to define membership functions [10–12], which, while precise, pose scalability and reproducibility challenges; the second is based on the theory of conceptual spaces [13,14], offering a geometric and cognitively

* Corresponding author.
E-mail addresses: rafaconejo@ugr.es (Rafael Vazquez-Conejo), mariatejadac@ugr.es (Maria Tejada-Casado), jherrera@ugr.es (Luis Javier Herrera), rghinea@ugr.es (Razvan Ghinea), jmsoto@ugr.es (Jose Manuel Soto-Hidalgo).

https://doi.org/10.1016/j.softx.2025.102316
Received 18 June 2025; Received in revised form 31 July 2025; Accepted 16 August 2025
Available online 3 September 2025

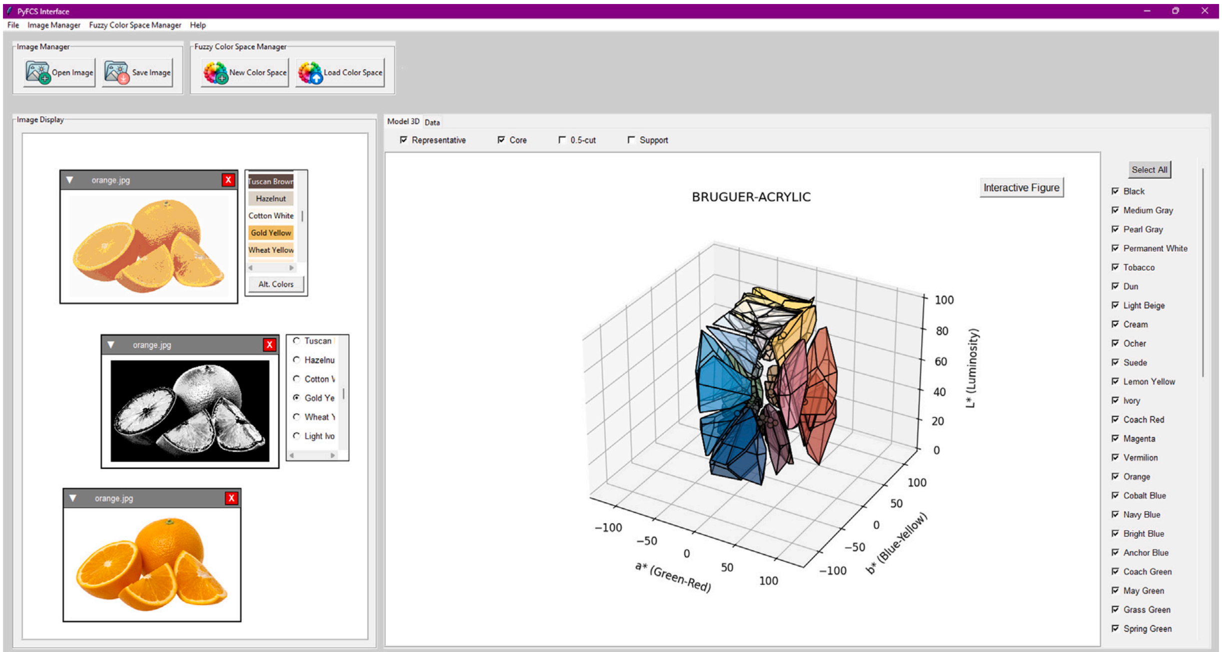


Fig. 1. Main interface of PyFCS GUI.

inspired foundation for representing categories through perceptual similarity. Unlike approaches that require exhaustive perceptual data collection, the conceptual spaces framework allows for the definition of color categories based on perceptual similarity, reducing the need for extensive fieldwork. Building on this latter approach, early tools such as JFCS [15] showed the feasibility of constructing fuzzy color spaces using conceptual spaces, but faced practical limitations related to computational efficiency, platform dependency, and restricted color spaces support.

To address these issues, the `PyFCS` library [16] was developed as an extensible and scalable solution, enabling the creation and analysis of fuzzy color spaces, and fully integrated within the Python scientific ecosystem. However, its command-line interface posed usability barriers for non-programmers, limiting exploratory and educational use. To overcome this, we present *PyFCS GUI*, a cross-platform graphical interface built with `Tkinter`, designed to make the creation, visualization, and application of fuzzy color spaces more accessible. This tool enhances interactivity, fosters reproducibility, and expands the applicability of fuzzy color modeling in both research and educational settings.

2. Software description

The PyFCS Graphical User Interface (PyFCS GUI) builds upon the foundational PyFCS library [16], whose modular architecture has enabled this new interface to emerge as a powerful and accessible tool for exploring fuzzy color spaces in a visual way.

PyFCS provides the computational core for fuzzy color modeling, which is based on the theory of conceptual spaces and a principled methodology for constructing fuzzy color categories [17]. In this approach, each fuzzy color is a concept represented in a conceptual space which is defined using a set of positive (fully representative of the concept) and negative (non-representative of the concept) prototypes within a perceptual color space such as CIELAB or RGB. A Voronoi tessellation of the space, centered on these prototypes, establishes a base region that is associated to the 0.5-cut of the fuzzy set. This region is then scaled inward and outward to define the core and support volumes of the fuzzy color, respectively. Membership degrees are computed through linear interpolation between these volumes, resulting in a continuous and cognitively grounded fuzzy representation.

This extensible core design has allowed PyFCS GUI to layer on interactive visualization and user-guided exploration without modifying the core logic.

2.1. Software architecture

The architecture of PyFCS GUI is structured around a tight integration with the PyFCS back-end. PyFCS GUI acts as a visual and interactive shell, mediating between user inputs and PyFCS's perceptual color modeling capabilities. Developed in Python and implemented using `Tkinter`, the GUI incorporates standard scientific libraries including NumPy, SciPy, Matplotlib, Plotly, PIL, and OpenCV. This modular ecosystem ensures cross-platform compatibility, facilitates educational use, and allows for future extensions and domain-specific adaptations.

The interface is organized into three main functional zones: (1) fuzzy color space management, (2) interactive visualization of perceptual color categories, and (3) application of color-based analysis to images. Fig. 1 illustrates the interface, which includes tools for palette-based and image-based category creation, real-time feedback visualizations, and intuitive access to image analysis operations.

2.2. Functional overview

The core functionalities of PyFCS GUI are designed to extend the capabilities of the underlying PyFCS library through an accessible graphical interface, while maintaining full interoperability with existing tools and file standards. The system supports the import and export of both `.fcs` files (fuzzy color space definitions) and `.cns` files (color naming schemes), ensuring seamless integration with other software and workflows for fuzzy modeling and perceptual analysis.

Specifically, PyFCS GUI supports four principal functional goals:

1. **Visualization of fuzzy color spaces:** Users can interactively explore the internal structure of fuzzy categories — including kernel, α-cuts, and support sets — within perceptually grounded color spaces. This feature aids in understanding the graded nature of category boundaries and the semantic overlap between hues.

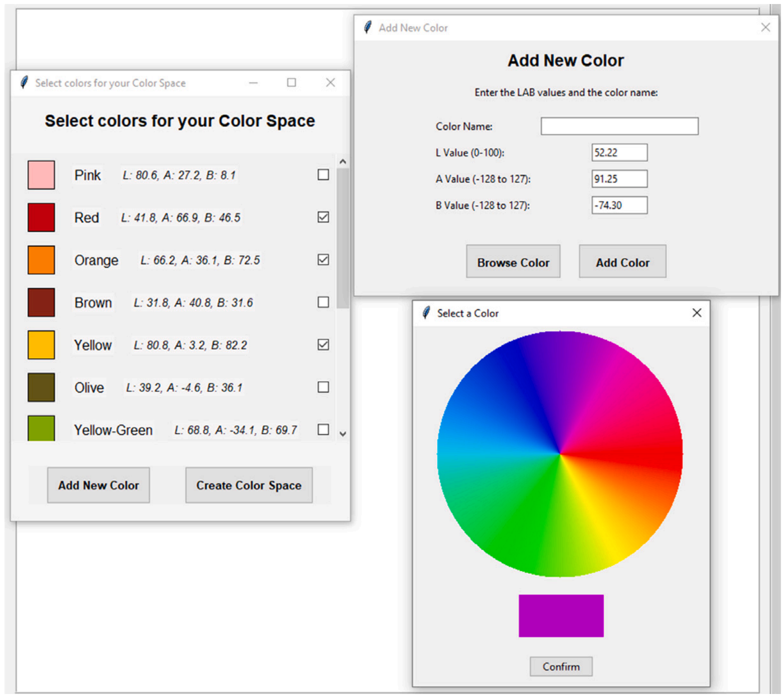


Fig. 2. Palette-based creation interface in PyFCS GUI. Users can manually select color prototypes, assign labels, and define fuzzy categories without relying on image clustering.

2. **Learning color spaces from images and palettes:** In addition to automatic extraction from image data using clustering algorithms such as DBSCAN [18], PyFCS GUI allows users to construct fuzzy color spaces interactively from color palettes. Users can intuitively select color samples as positive prototypes using a visual interface, in accordance with the principles of conceptual space theory. The image-based approach supports a threshold parameter to control the number and specificity of extracted clusters, while the palette-based method facilitates manual, fine-grained definition of perceptual categories.
3. **Fuzzy color mapping:** The system provides tools for assigning fuzzy category memberships to individual pixels in an image. The resulting chromatic maps, available in both grayscale and color-coded formats, visualize how perceptual regions are distributed and how they relate to the fuzzy category structure.
4. **Educational and didactic capabilities:** With an intuitive design and real-time visual feedback, PyFCS GUI serves as an effective teaching tool. It supports classroom demonstrations and student-driven exploration in domains such as fuzzy set theory, conceptual modeling, and linguistic categorization of perceptual stimuli.

To assess the usability and perceived quality of the PyFCS GUI, a user study was carried out using the standardized User Experience Questionnaire (UEQ) [19,20], which evaluates aspects such as attractiveness, efficiency, dependability, and clarity. The questionnaire was completed by a group of representative users after their interaction with the software. The results were overall positive, indicating a high level of user satisfaction and providing empirical support for the system's usability. The full questionnaire and results are also made available in the PyFCS GUI repository.

3. Illustrative examples

To demonstrate the core functionalities and practical use of the software, a representative case study is presented involving the creation, visualization, and application of fuzzy color spaces on an image containing several orange fruits against a white background. This example illustrates the workflow supported by the GUI, emphasizing three key capabilities: fuzzy color space generation from image data or palettes (Section 3.1), interactive perceptual visualization (Section 3.2), and fuzzy semantic mapping (Section 3.3). Notice that, although this case study uses the CIELAB color space due to its perceptual properties, PyFCS also supports other color spaces such as RGB, HSI, HSV as long as they can be defined as metric spaces. In addition to this illustrative example, Section 3.4 presents a performance benchmark of the GUI, along with a discussion of its broader applicability across domains such as digital art, user interface design, and cultural heritage restoration.

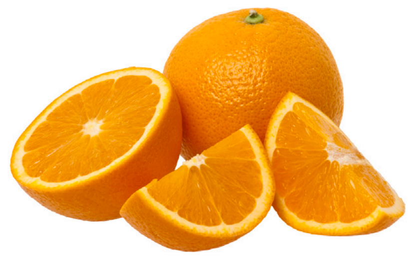
3.1. *Creating fuzzy color spaces*

The creation of fuzzy color spaces in PyFCS GUI can be performed through two complementary modes: image-based extraction and palette-based definition. Both approaches are grounded in the theory of conceptual spaces, where selected color samples act as positive prototypes that anchor the resulting fuzzy categories.

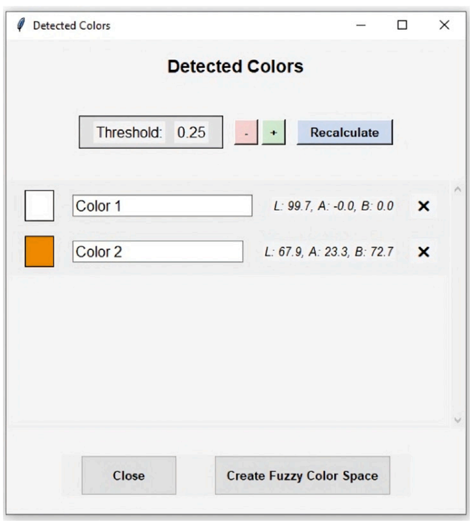
Image-based creation. This procedure begins by selecting the *Image-Based Creation* option from the *Create New Fuzzy Color Space* menu. Users can upload an arbitrary image — for example, one depicting orange fruits on a white background — and extract its dominant chromatic components through unsupervised clustering. PyFCS GUI employs the DBSCAN algorithm [18] to identify color clusters in a perceptually uniform space (e.g., CIELAB), taking into account the distribution and density of pixel values.

A user-adjustable threshold parameter governs the granularity of extraction. Lower values produce fewer, broader clusters, representing coarse chromatic groupings, while higher values yield more refined, narrowly bounded shades. The centroids of these clusters are interpreted as prototypes in the conceptual space, forming the basis of the fuzzy categories.

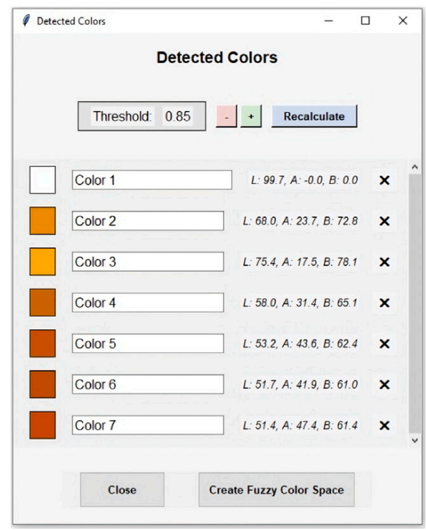
For instance, with a threshold of 0.25, the system may identify two principal color groups — orange and white — yielding a generalized fuzzy space $\widetilde{\Gamma}_{generic-orange}$ (Fig. 3(b)). At a higher threshold of 0.85, it may extract six distinct orange hues plus the background white, resulting in the more granular $\widetilde{\Gamma}_{specific-orange}$ (Fig. 3(c)).



(a) Base image used for chromatic clustering: primarily orange and white regions.



(b) Low threshold (0.25): two broad chromatic categories extracted—orange and white.



(c) High threshold (0.85): seven fine-grained categories identified, capturing nuanced orange variations and the white background.

Fig. 3. Effect of threshold parameter on image-based fuzzy color space creation. The central image is the source from which chromatic clusters are extracted. As the threshold increases, the number of extracted clusters grows, yielding a finer chromatic segmentation. These clusters can serve as positive prototypes for generating fuzzy categories in the conceptual space.

Palette-based creation. Alternatively, users may opt for the *Palette-Based Creation* mode, which enables manual specification of color prototypes through a visual color picker and sample editor (Fig. 2). In this mode, users can select and name individual colors from a palette to define the fuzzy categories directly. This approach is especially useful when working with standardized shade guides, institutional color codes, or perceptual labels.

Each selected color becomes a central prototype in the conceptual space, around which the fuzzy set is built. The graphical interface facilitates intuitive control over label assignment and prototype definition, ensuring flexibility and reproducibility in the generation of fuzzy color spaces.

Both creation modes — automatic and manual — are fully integrated within PyFCS GUI, and the resulting spaces can be saved in `.fcs` format for further use or interoperability with other tools.

3.2. Visualizing and navigating fuzzy color spaces

The resulting fuzzy color spaces are explored through the 3D visualization tool, which enables inspection of color sets within the CIELAB perceptual space. Users can interactively rotate, zoom, and inspect structural components such as the core, α-cut (at $\alpha = 0.5$), and support regions.

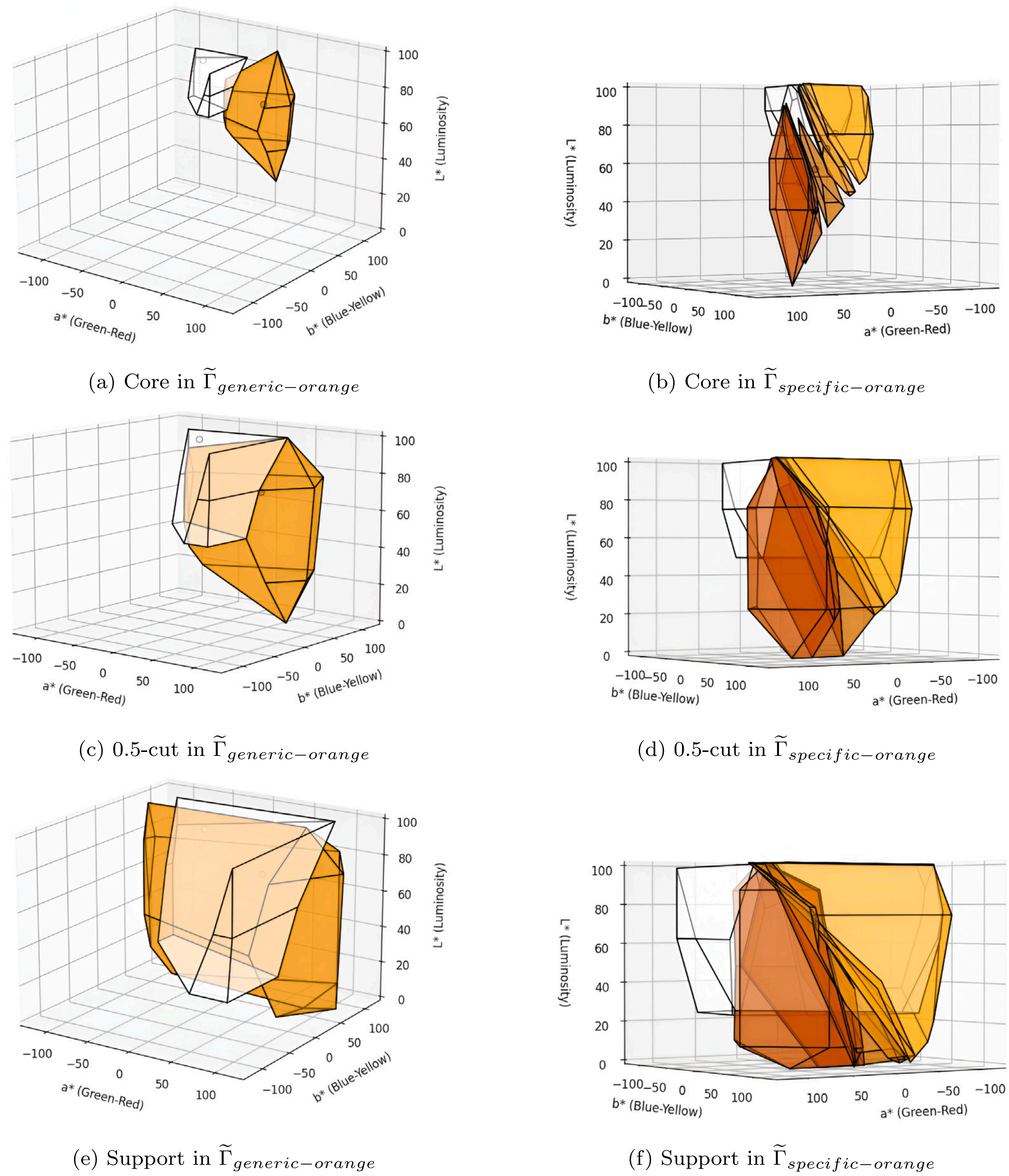
The two-category space $\widetilde{\Gamma}_{generic-orange}$ reveals broad, smoothly transitioning regions (Figs. 4(a), 4(c), and 4(e)). In contrast, $\widetilde{\Gamma}_{specific-orange}$ exhibits a more complex structure, where the six orange hues are distinctly distributed (Figs. 4(b), 4(d), and 4(f)). These visualizations facilitate intuitive understanding of the spatial distribution of fuzzy concepts and are aligned with the conceptual spaces framework [17].

3.3. Fuzzy color mapping for semantic analysis

To evaluate the semantic applicability of the fuzzy models, the *Color Mapping* function is applied to the original image. This feature computes the degree of membership of each pixel to a selected fuzzy color and displays the result in grayscale, highlighting perceptual regions.

Using $\widetilde{\Gamma}_{generic-orange}$ (Fig. 5), the mapping shows clear separation between fruit and background, suitable for coarse-level segmentation. The more detailed $\widetilde{\Gamma}_{specific-orange}$ (Fig. 6) reveals fine-grained variations among orange tones, making it well-suited for nuanced tasks such as texture differentiation or lighting effect analysis.

This illustrative workflow highlights the flexibility of the PyFCS GUI in supporting both generalized and fine-tuned modeling, while



(a) Core in $\widetilde{\Gamma}_{generic-orange}$

(b) Core in $\widetilde{\Gamma}_{specific-orange}$

(c) 0.5-cut in $\widetilde{\Gamma}_{generic-orange}$

(d) 0.5-cut in $\widetilde{\Gamma}_{specific-orange}$

(e) Support in $\widetilde{\Gamma}_{generic-orange}$

(f) Support in $\widetilde{\Gamma}_{specific-orange}$

Fig. 4. Visualization of the core, 0.5-cut, and support surfaces in the fuzzy color spaces: (a,c, (e) $\widetilde{\Gamma}_{generic-orange}$ and (b,d, (f) $\widetilde{\Gamma}_{specific-orange}$.

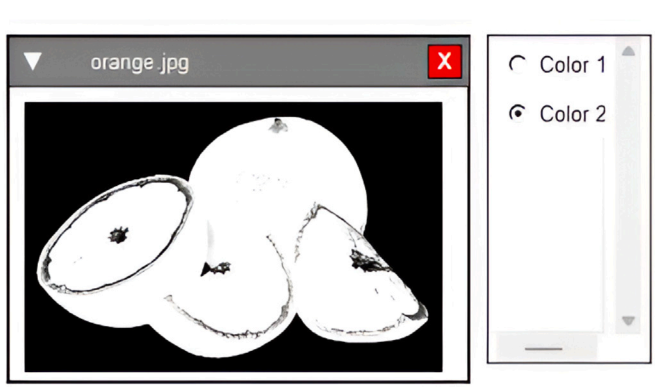


Fig. 5. Color mapping with $\widetilde{\Gamma}_{generic-orange}$.

providing interpretable and perceptually grounded outputs throughout the process.

3.4. Performance benchmark and extended applications

In addition to the illustrative example, we conducted a performance benchmark to evaluate the computational efficiency of the PyFCS GUI during the construction of several fuzzy color spaces (random colors and the three ISCC-NBS based fuzzy color spaces [21]). The evaluation measured execution time, CPU usage, and memory consumption, and included a direct comparison with JFCS, its Java-based predecessor. Results indicate that PyFCS GUI achieves substantially lower memory usage (up to 35% less) and faster execution times (with an average speedup of 1.8×across tested scenarios), while maintaining a stable CPU load. These improvements highlight the efficiency of the Python-based implementation and confirm its suitability for interactive use in research and educational contexts. Full details of the benchmark, including code and reproducible scripts, are available in the project's GitHub repository, ensuring transparency and replicability.

Beyond the orange fruit-based example presented as illustrative example, the PyFCS GUI demonstrates a wide potential for application in domains where nuanced and flexible treatment of color is essential. For example, in digital art and artistic palette design, the tool can extract

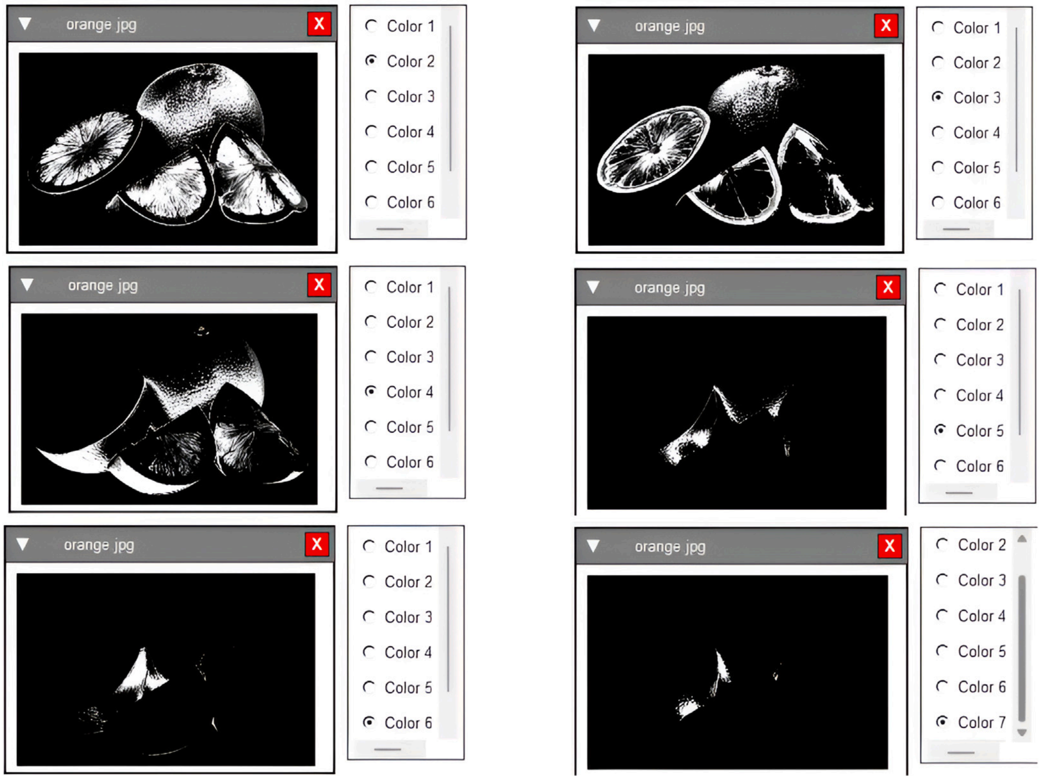


Fig. 6. Color mapping with $\widetilde{\Gamma}_{specific-orange}$ for individual fuzzy colors.

dominant tones from paintings or design prototypes and generate fuzzy color spaces that capture perceptual transitions between hues. In user interface design, it enables the creation of adaptive and accessible color themes, such as color-blind-friendly palettes, by clustering dominant shades and mapping them into fuzzy categories. Similarly, in cultural heritage restoration, the GUI can support the comparison of faded pigments with plausible original tones, allowing for perceptual mappings that incorporate uncertainty in chromatic reconstruction. These examples illustrate how the software can serve as a versatile platform, extending its utility beyond academic demonstration into practical, real-world applications. Additional case studies are made available in the project's GitHub repository to further showcase this versatility.

4. Impact

The PyFCS GUI represents a significant advance in the usability and dissemination of fuzzy color modeling tools, particularly in perceptual and computational domains where color semantics play a central role. By abstracting away the complexity of direct programming interaction, the GUI lowers the barrier to entry for researchers and practitioners from non-technical backgrounds. This usability enhancement enables users to intuitively construct, visualize, and apply fuzzy color spaces, adjust granularity and fuzziness parameters on the fly, and immediately interpret the resulting α-cuts and membership structures. Such capabilities not only improve reproducibility, but also reduce calibration effort and modeling bias.

Beyond usability, the system fosters new research directions. Its ability to generate fuzzy color spaces directly from visual data creates opportunities to study phenomena such as perceptual similarity, context-sensitive categorization, and interpretable color segmentation within a fuzzy logic framework. These functionalities are especially relevant in cognitive modeling [1], computational analysis of art, and cultural semantics, where color meanings must be inferred under conditions of ambiguity [22–24]. Furthermore, the support for both image-based and palette-based creation of fuzzy categories allows researchers to explore how fuzzy color concepts emerge from real-world visual data and how linguistic and cultural contexts modulate perceptual prototypes [12,25].

The software's compatibility with standard formats (.fcs for fuzzy spaces and .cns for color naming schemes) ensures interoperability with existing tools, allowing PyFCS GUI to function not just as a standalone environment but also as part of broader analysis pipelines. Since its release, the PyFCS GUI has been made openly available as part of the PyFCS library and is already attracting interest within the research and educational communities. Early feedback highlights its utility in graduate-level instruction and exploratory research in areas such as fuzzy perception, dental shade analysis, and explainable AI. Its accessible and modular design makes it well-suited for integration into interdisciplinary workflows and applied contexts such as clinical diagnostics or digital heritage restoration. By offering an open-source, platform-independent, and pedagogically oriented tool, PyFCS GUI lays the groundwork for broader adoption and innovation in perceptual modeling, semantic computing, and visual reasoning under uncertainty.

5. Conclusions

This work presents a graphical user interface (GUI) extension for the PyFCS library that significantly enhances its usability, accessibility, and analytical reach. By enabling intuitive interaction with fuzzy color spaces — through visual creation, interactive exploration, and application to image analysis — the GUI makes the theoretical foundations of fuzzy color semantics more accessible to a broader audience. It thus bridges the gap between complex computational modeling and practical, user-centered exploration.

Developed in Python and built upon a modular architecture, PyFCS GUI supports diverse use cases, from cognitive modeling and perceptual research to education and artistic analysis. Its implementation in an open and widely adopted ecosystem encourages community involvement, customization, and seamless integration into broader research workflows.

CRediT authorship contribution statement

Rafael Vazquez-Conejo: Writing – original draft, Visualization, Software, Investigation. **Maria Tejada-Casado:** Writing – review & editing, Supervision, Project administration. **Luis Javier Herrera:** Writing – review & editing, Supervision, Project administration. **Razvan Ghinea:** Writing – review & editing, Supervision, Conceptualization. **Jose Manuel Soto-Hidalgo:** Writing – original draft, Supervision, Methodology, Investigation, Conceptualization.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used GPT-4 in order to improve language and readability. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research has been supported by the Project PID2022-142151 OB-I00 funded by MICIU/AEI/10.13039/501100011033 and ERDF/EU.

References

[1] Palmer SE. Vision science - photons to phenomenology. The MIT Press; 1999.

[2] Itten J. The art of color: the subjective experience and objective rationale of color. John Wiley; 1974.

[3] Kim H-S, Kim EJ, Kim J. The optimal color space for realistic color reproduction in virtual reality content design. Electronics 2023;12(22).

[4] Rosch E. Cognitive representations of semantic categories. J Exp Psychol [Gen] 1975;104:192–233.

[5] Rosch E. Prototype classification and logical classification: The two systems. In: Scholnik E, editor. New trends in cognitive representation: challenges to Piaget's theory. Hillsdale, NJ: Lawrence Erlbaum Associates; 1978, p. 73–86.

[6] Smeulders A, Worring M, Santini S, Gupta A, Jain R. Content-based image retrieval at the end of the early years. IEEE Trans Pattern Anal Mach Intell 2000;22(12):1349–80.

[7] Berlin B, Kay P. Basic color terms: Their universality and evolution. In: Anthropology, linguistics, psychology, University of California Press; 1991.

[8] Oh A-S, Lho T-J, Kwon J-W, Kim K-B. Color preference and personality modeling using fuzzy reasoning rule. In: Gavrilova ML, Gervasi O, Kumar V, Tan CJK, Taniar D, Laganá A, Mun Y, Choo H, editors. Computational science and its applications - ICCSA 2006. Berlin, Heidelberg: Springer Berlin Heidelberg; 2006, p. 887–94.

[9] Shamoi P, Inoue A, Kawanaka H. Application aspects of fuzzy color modeling. In: 2019 IEEE international conference on systems, man and cybernetics. 2019, p. 1130–5.

[10] Benavente R, Vanrell M, Baldrich R. Parametric fuzzy sets for automatic color naming. J Opt Soc Am 2008;25(10):2582–93.

[11] Menegaz G, Le Troter A, Sequeira J, Boï J. A discrete model for color naming. EURASIP J Adv Signal Process 2007;2:1–10.

[12] Shamoi P, Muratbekova M, Izbassar A, Inoue A, Kawanaka H. Towards a universal understanding of color harmony: Fuzzy approach. In: Fuzzy systems and data mining IX. IOS Press; 2023.

[13] Gärdenfors P. Conceptual spaces: The geometry of thought. A bradford book, MIT Press; 2004.

[14] Gärdenfors P. The geometry of meaning: Semantics based on conceptual spaces. MIT Press; 2014.

[15] Soto-Hidalgo J, Martínez-Jiménez P, Chamorro-Martínez J, Sánchez D. JFCS: A color modeling java software based on fuzzy color spaces. IEEE Comput Intell Mag 2016;11(2):16–28.

[16] Conejo RV, Tejada-Casado M, Herrera L, Guinea R, Soto-Hidalgo JM. PyFCS: A new python library to create and manipulate fuzzy color spaces. In: Proceedings of the 2024 IEEE international conference on fuzzy systems. 2024.

[17] Chamorro-Martínez J, Soto-Hidalgo J, Martínez-Jiménez P, Sánchez D. Fuzzy color spaces: A conceptual approach to color vision. IEEE Trans Fuzzy Syst 2016.

[18] Deng D. DBSCAN clustering algorithm based on density. In: 2020 7th international forum on electrical engineering and automation. 2020, p. 949–53.

[19] Hinderks A, Schrepp M, Domínguez Mayo FJ, Escalona MJ, Thomaschewski J. Developing a UX KPI based on the user experience questionnaire. Comput Stand Interfaces 2019;65:38–44.

[20] Schrepp M, Hinderks A, Thomaschewski J. Construction of a benchmark for the user experience questionnaire (UEQ). Int J Interact Multimed Artif Intell 2017;4(4):40–4.

[21] Kelly K, Judd D, Inter-Society Color Council and United StatesNational Bureau of Standards. The ISCC-NBS method of designating colors and a dictionary of color names: Supplement. Circular of the national bureau of standards, U.S. Department of Commerce, National Bureau of Standards; 1955.

[22] Kay P, McDaniel C. The linguistic significance of the meanings of basic color terms. Language 1978;54(3):610–46.

[23] Amante J, Fonseca M. Fuzzy color space segmentation to identify the same dominant colors as users. In: 18th international conference on distributed multimedia systems. 2012, p. 48–53.

[24] Jonauskaite D, Wicker J, Mohr C, Dael N, Havelka J, Papadatou-Pastou M, et al. A machine learning approach to quantify the specificity of colour–emotion associations and their cultural differences. R Soc Open Sci 2019;6(9):190741.

[25] Muratbekova M, Shamoi P. Color-emotion associations in art: Fuzzy approach. IEEE Access 2024;12:37937–56.