Aalto Interface Metrics (AIM): A Service and Codebase for Computational GUI Evaluation

Antti Oulasvirta[1], Samuli De Pascale[1], Janin Koch[1], Thomas Langerak[1], Jussi Jokinen[1], Kashyap Todi[1], Markku Laine[1], Manoj Kristhombuge[1], Yuxi Zhu[1], Aliaksei Miniukovich[2], Gregorio Palmas[3], Tino Weinkauf[3]
[1]Aalto University, [2]University of Trento, [3]KTH Royal Institute of Technology

ABSTRACT

Aalto Interface Metrics (AIM) pools several empirically validated models and metrics of user perception and attention into an easy-to-use online service for the evaluation of graphical user interface (GUI) designs. Users input a GUI design via URL, and select from a list of 17 different metrics covering aspects ranging from visual clutter to visual learnability. AIM presents detailed breakdowns, visualizations, and statistical comparisons, enabling designers and practitioners to detect shortcomings and possible improvements. The web service and code repository are available at interfacemetrics.aalto.fi.

INTRODUCTION

AIM is an online service and an open code repository for computational evaluation of graphical user interface (GUI) designs. AIM pools several previously published metrics and models, which have been empirically shown to be predictive of how users perceive, search, and aesthetically experience a design. These metrics range from design heuristics like symmetry to metrics and full-fledged models such as saliency and visual clutter. The source code is open-sourced, inviting contributions from researchers and practitioners. A well-documented Python API enables the system to be easily extended with new metrics.

The prime goal of AIM is to facilitate the use and appropriation of computational methods in design practice. Typically, evaluation in interface and interaction design practice relies on personal experience and empirical testing, and less so on computational modeling. While some previous papers (e.g. [8, 15, 20]) have applied models and metrics to assist designers, they do not offer explanations and automated evaluations. On the other hand, previous work on automated evaluation e.g. [1, 5, 19, 22]) has had limited scope (in terms of number of metrics) or have not been easily extendable. With AIM, we

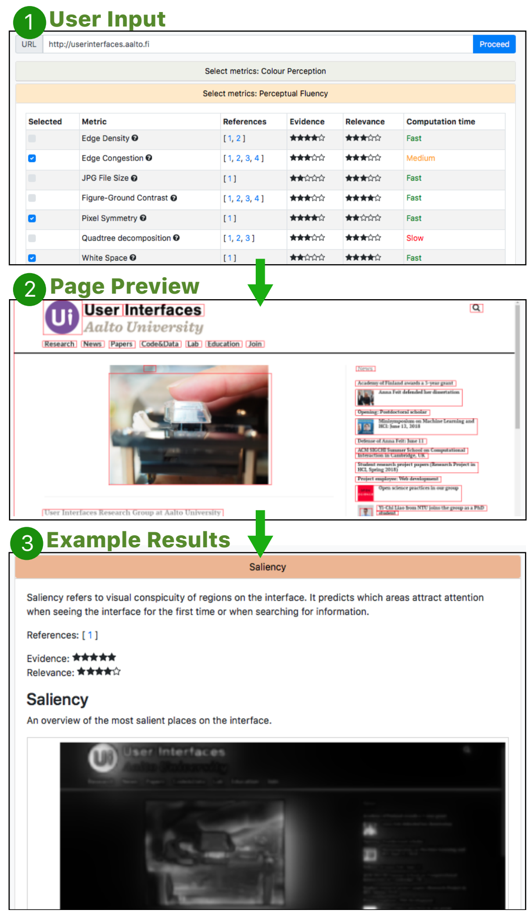

Figure 1. AIM is an online service and an open codebase for automated evaluation of GUI designs. (1) User enters URL; (2) AIM segments the image; (3) AIM presents detailed results per metric. It gives an overview of what the metric does, and an indicator of empirical evidence shown for its predictive power. A histogram offers comparison of the design to other commonly-found designs.

explore a large range of interface metrics, covering various aspects related to usability and performance, and provide a flexible system that can be easily extended to address additional aspects. An overview of the web user interface is given in Figure 1.

A secondary goal of AIM is to facilitate research efforts centered around computational models of human-

UIST '18 Adjunct October 14–17, 2018, Berlin, Germany

ACM ISBN 978-1-4503-5949-8/18/10.
DOI: https://doi.org/10.1145/3266037.3266087

computer interaction. Existing research on computational metrics has been fragmented across disciplines, UI types, data formats, and research groups. Implementing an existing model is often a significant undertaking. By providing a common platform, where models can be plugged in and implemented, we offer the means to unify efforts in investigating models and metrics.

The key features of AIM are:

- **Coverage:** The service covers a significant number of metrics and models, including both state-of-the-art topics in research as well as factors shown empirically to be relevant for GUI design.
- **Evidence-based evaluation:** All metrics are provided with a summary of the main principle, reference to scientific article, and empirical evidence. All scores are provided with histograms relating the current design to others in the domain.
- **Open source:** The codebase is published for anyone to download and extend. We invite contributions from the community.
- **Uniform API:** Inputs and outputs are consistent as much as possible, making it easy to adopt them in Python code.

METRICS AND MODELS IN AIM

To cover a wide range of criteria important for UI design, we selected 17 metrics and models (listed in Table 1), and implemented these in AIM. They cover four categories:

1. **Color Perception†:** These cover different aspects related to the colorfulness of the design, and how this influences perception and usability.
2. **Perceptual Fluency‡:** These estimate the ease with which the visible information is perceived and processed visually and aesthetically.
3. **Visual Guidance§:** These predict visual search performance while navigating the design.
4. **Accessibility⊕:** This estimates whether the design meets relevant accessibility requirements.

Metric	**Description**	**Comp time***	**Ref**
File size†	The file size (JPEG & PNG) of the image in bytes	0.000 (0.000)	[13]
Color Variability†	The amount of different colors in RGB, HSV, and LAB color spaces	1.946 (0.400)	[4, 12] [13]
Static Color Clusters†	Number of bins with ¿5 px. Bins are 32*32*32 px (in RGB)	2.307 (0.671)	[12, 13]
Dynamic Clusters†	Number of bins with >5 pixels, based on distance between pixels	42.435 (41.101)	[12, 13]
Colorfulness†	The standard deviation of pixels in the RGB color space	3.065 (0.228)	[4]
Luminance†	Standard deviation of luminance corrected for display perception	5.579 (0.674)	[12]
Color Harmony†	The sum of the distance of all pixels to a color scheme.	71.516 (59.396)	[2]
Edge Density‡	Ratio of edge pixels to all pixels	0.115 (0.091)	[12, 18]
Contour Congestion‡	Ratio of congested edge pixels to all edge pixels	11.165 (2.696)	[10, 12] [21]
Figure-Ground Contrast‡	The discriminability of the foreground from the background based on contrast.	0.206 (0.214)	[3, 12] [16]
Symmetry‡	Ratio of edges that are mirrored either horizontal, vertical, or diagonal	3.516 (2.197)	[12]
Visual Complexity‡	Balance, symmetry, and equilibrium based on quadtree decomposition	17.349 (6.600)	[14, 17] [23]
Grid Quality‡	Alignment to grids	8.020 (0.683)	[13]
White Space‡	Proportion of non-covered space on the website	0.005 (0.007)	[13]
Itti-Koch Saliency§	The degree to which a pixel stands out	0.897 (0.149)	[6, 9]
Visual Search Performance§	Visual search time for page elements	1.534 (0.954)	[7]
Color Blindness⊕	Images as seen by the three common color blindness types	13.369 (2.916)	[11]

†Color Perception, ‡Perceptual Fluency,
§Visual Guidance, ⊕Accessibility
* Avg time (and SD) per screenshot (in seconds), computed using top 10 sites in the Alexa Top 500.

Table 1. Metrics and models in AIM.

IMPLEMENTATION

AIM is implemented as a web application, consisting of two separate components: frontend and backend. The frontend handles the web user interface, including the metrics selection form and the presentation of results. This is implemented using the Vue.js JavaScript framework. The backend handles the evaluation of metrics, and is implemented using the Python-based Tornado web framework. In addition, the backend contains two subcomponents: metrics library and segmentation script. Both the metrics included in the library and the segmentation script are implemented in Python, excluding *visual search* and *grid quality* metrics which are implemented in Common Lisp and MATLAB, respectively.

When a user enters an URL and selects which metrics to run a request is made to the backend. Next, the backend captures a screenshot of the target website using Headless Chrome and runs the segmentation script against it to generate a list of visible elements (for *segmentation-based metrics* only). Each of these elements contain the following properties: 1. Identifier; 2. Absolute Position (x, y); 3. Size (width, height); and 4. Base64-encoded image data. The selected metrics are then computed with the base64-encoded representation of the website and the list of segmentation elements as input arguments. The metrics are independent from each other, and therefore can be run in parallel to increase total performance of the server. Finally, the results from the metrics are pushed one by one to the frontend via WebSocket as and when they become available.

ACCESS AND EXTENSIBILITY

The web service and code repository of AIM are fully open-sourced, and available at `interfacemetrics.aalto.fi`. AIM has been designed from the ground-up with extensibility in mind. As a result, new metrics can be added with relatively small effort using a uniform API. In practice, a new metric is defined in a separate Python file. It takes the screenshot or segmented page as input, and should return numerical scores, or an image, as output. It can be plugged in to the system by registering it in the front- and back-end.

REFERENCES

1. Michael D. Byrne, Scott D. Wood, Noi Sukaviriya, James D. Foley, and David E. Kieras. 1994. Automating interface evaluation. In *CHI Conference Companion*.

2. Daniel Cohen-Or, Olga Sorkine, Ran Gal, Tommer Leyvand, and Ying-Qing Xu. 2006. Color harmonization. In *ACM Transactions on Graphics (TOG)*, Vol. 25. ACM, 624–630.

3. Richard H Hall and Patrick Hanna. 2004. The impact of web page text-background colour combinations on readability, retention, aesthetics and behavioural intention. *Behaviour & information technology* 23, 3 (2004), 183–195.

4. David Hasler and Sabine E Suesstrunk. 2003. Measuring colorfulness in natural images. In *Human vision and electronic imaging VIII*, Vol. 5007. International Society for Optics and Photonics, 87–96.

5. Scott E. Hudson, Bonnie E. John, Keith Knudsen, and Michael D. Byrne. 1999. A Tool for Creating Predictive Performance Models from User Interface Demonstrations. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology (UIST '99)*. ACM, New York, NY, USA, 93–102. DOI: http://dx.doi.org/10.1145/320719.322590

6. Laurent Itti and Christof Koch. 2000. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision research* 40, 10-12 (2000), 1489–1506.

7. Jussi PP Jokinen, Sayan Sarcar, Antti Oulasvirta, Chaklam Silpasuwanchai, Zhenxin Wang, and Xiangshi Ren. 2017. Modelling Learning of New Keyboard Layouts. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 4203–4215.

8. Won Chul Kim and James D. Foley. 1990. DON: User Interface Presentation Design Assistant. In *Proceedings of the 3rd Annual ACM SIGGRAPH Symposium on User Interface Software and Technology (UIST '90)*. ACM, New York, NY, USA, 10–20. DOI: http://dx.doi.org/10.1145/97924.97926

9. Akisato Kimura. 2014. pySaliencyMap. GitHub. (4 May 2014). Retrieved July, 2074 from https://github.com/akisato-/pySaliencyMa.

10. Dennis M Levi. 2008. CrowdingAn essential bottleneck for object recognition: A mini-review. *Vision research* 48, 5 (2008), 635–654.

11. Gustavo M Machado, Manuel M Oliveira, and Leandro AF Fernandes. 2009. A physiologically-based model for simulation of color vision deficiency. *IEEE Transactions on Visualization and Computer Graphics* 15, 6 (2009), 1291–1298.

12. Aliaksei Miniukovich and Antonella De Angeli. 2014. Quantification of interface visual complexity. In *Proceedings of the 2014 international working conference on advanced visual interfaces*. ACM, 153–160.

13. Aliaksei Miniukovich and Antonella De Angeli. 2015. Computation of interface aesthetics. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 1163–1172.

14. David Chek Ling Ngo, Lian Seng Teo, and John G Byrne. 2003. Modelling interface aesthetics. *Information Sciences* 152 (2003), 25–46.

15. Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2015. DesignScape: Design with Interactive Layout Suggestions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 1221–1224. DOI: http://dx.doi.org/10.1145/2702123.2702149

16. Rolf Reber, Pascal Wurtz, and Thomas D Zimmermann. 2004. Exploring fringe consciousness: The subjective experience of perceptual fluency and its objective bases. *Consciousness and cognition* 13, 1 (2004), 47–60.

17. Katharina Reinecke, Tom Yeh, Luke Miratrix, Rahmatri Mardiko, Yuechen Zhao, Jenny Liu, and Krzysztof Z Gajos. 2013. Predicting users' first impressions of website aesthetics with a quantification of perceived visual complexity and colorfulness. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2049–2058.

18. Ruth Rosenholtz, Yuanzhen Li, and Lisa Nakano. 2007. Measuring visual clutter. *Journal of vision* 7, 2 (2007), 17–17.

19. Andrew Sears. 1993. AIDE: A tool to assist in the design and evaluation of user interfaces. *Interactive Systems Research Center, Baltimore* (1993).

20. Kashyap Todi, Daryl Weir, and Antti Oulasvirta. 2016. Sketchplore: Sketch and Explore with a Layout Optimiser. In *Proceedings of the 2016 ACM*

Conference on Designing Interactive Systems (DIS '16). ACM, New York, NY, USA, 543–555. DOI: http://dx.doi.org/10.1145/2901790.2901817

21. Ronald Van den Berg, Frans W Cornelissen, and Jos BTM Roerdink. 2009. A crowding model of visual clutter. *Journal of Vision* 9, 4 (2009), 24–24.

22. Mathieu Zen and Jean Vanderdonckt. 2014. Towards an evaluation of graphical user interfaces aesthetics based on metrics. In *Research Challenges in Information Science (RCIS), 2014 IEEE Eighth International Conference on*. IEEE, 1–12.

23. Xianjun Sam Zheng, Ishani Chakraborty, James Jeng-Weei Lin, and Robert Rauschenberger. 2009. Correlating low-level image statistics with users-rapid aesthetic and affective judgments of web pages. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1–10.