Image Style Transferred to Graphical User Interfaces

Karim Hammoudi, Adnane Cabani, Halim Benhabiles, Mahmoud Melkemi

Image Style Transferred to Graphical User Interfaces

Karim Hammoudi[1,2], Adnane Cabani[3], Halim Benhabikes[4], and Mahmoud Melkemi[1,2]

[1] Université de Haute-Alsace, Department of Computer Science, IRIMAS, F-68100 Mulhouse, France
{karim.hammoudi,mahmoud.melkemi}@uha.fr
[2] Université de Strasbourg
[3] Normandie Univ, UNIROUEN, ESIGELEC, IRSEEM, 76000 Rouen, France
adnane.cabani@esigelec.fr
[4] JUNIA, CNRS-IEMN, Universite de Lille, Lille F-59000, France
halim.benhabiles@junia.com

Abstract. This paper presents an approach for permitting the restyling of an application by automatic analysis of a considered image. The approach relies on the analysis of color distribution from a query image and a color-to-component mapping in order to perform its style transfer to the GUI. No deep learning technique is used, e.g. making the approach applicable without investigating image style learning through training stages. We show how our approach can i) be used to directly and dynamically restyle GUI of applications, ii) be adopted as an end-user functionality towards the restyling of mobile app, desktop app, web app and so on, iii) serve to developers e.g. for facilitating the selection of appropriate reference graphic charter during the design stage of their applications by exploring GUI appearances from a list of considered query images. Experimental results show the efficiency of the approach and its high potential of generalization.

Keywords: Image Style Transfer · Visual Computing · GUI · Graphic Charter · Design Pattern · UX · HCI.

1 Introduction and motivation

Nowadays, the graphical user interface of majority of applications deployed over the world is static in the sense that the default designs cannot be changed or can sometimes be modified according to a predefined list of design templates. In this context, we propose a restyling approach which exploits an input image of our choice (already stored image, taken photo) for dynamically colorizing a graphical user interface.

The style transfer is a particularly active topic which is more and more investigated for automatically transferring style characteristics of a data [1–3] or a drawing technique [4–6] to another one towards facilitating styling tasks for a large spectrum of applications.

To the past, e.g. in [7], efficient statistical approaches have been developed to successfully transfer color characteristics from one image to another one.

In [8], authors proposes an approach which can change the appearance of photos to a different time of day. This approach exploits a database of time-lapse videos. By this way, a photo taken in daylight can be transformed into a realistic photo taken at night.

Recently, authors of [9] proposed an approach for permitting a photorealistic style transfer while limiting spatial distortions or unrealistic artifacts which should not happen in real photographs. The key ingredient of their approach is the use of wavelet transforms that naturally fits in deep networks. In [10], authors propose an approach which permits to insert the image of an object in the image of a painting and to automatically harmonize the overall. To this end, the style of the painting is transferred to the image of the inserted object making the resulting global composition appearing as it was originally painted. Besides, the deep learning approach which is presented in [11] can transfer in high quality the style of a cartoon face, a caricature face or an anime face to a real face picture. In [12], authors propose a framework that permits a style transfer without applying a direct use of an image, but only with a text description of the desired style. By this way, associating the term "fire" to an image permits to apply on a fire style on this latter.

Most of the recent style transfer approaches currently rely on the use of neural-related techniques but not all [13]; e.g. in reason of applicative contexts for which data training stages, data-to-data matching stages or deep architecture parametrization cannot be easily operated.

To the best of our knowledge, very few approaches have been proposed for transferring the style of an image to a graphical user interface. In [14], a deep learning restyling approach is presented to globally transfer the style of an image to the image of a considered graphical user interface. The obtained results are very interesting and promising. However, it seems that their quality of results is related to a well-choice of the images. On the other side, suggested training stages could be time-consuming which may be fastidious for the users. In the next section, we propose an approach which could be less time-consuming since no machine learning technique is used. On the other hand, our method evaluates how aesthetic and usable could be the interface restyled from a query image.

2 Proposed approach

Figure 1 shows the global dataflow diagram of the color-based approach which is proposed for restyling the graphical user interface of an application from a query image. This restyling workflow is applied to the GUI of a mobile application but other types of application such as web application or desktop application can be considered. In our process, the proposed application, which implements our approach, includes functionalities for loading a stored image or for taking a picture from a camera for its exploitation towards restyling the graphical user interface. Once the image loaded or taken, dominant colors are identified. To this

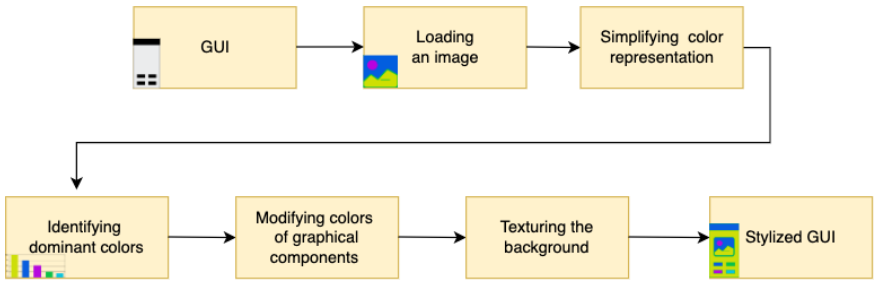


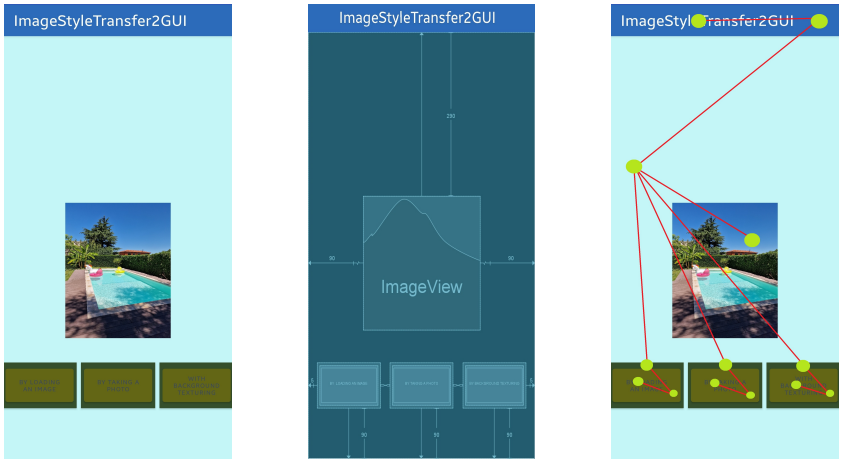
Fig. 1: Global dataflow diagram of the proposed approach for dynamically restyling a GUI of an application from an image.

end, an histogram of color distribution is computed and a classification of color intensities is operated [15] with respect to the number of component category which is present in the considered graphical user interface (e.g. text, buttons, background). Then the default color of graphical components is replaced by the identified dominant colors of the considered image. Additionally to the transfer of color, the analyzed image can also be used for texturing the background of the application according to the expectation of the user. The output shows an application which is restyled with the appearance of the considered image.

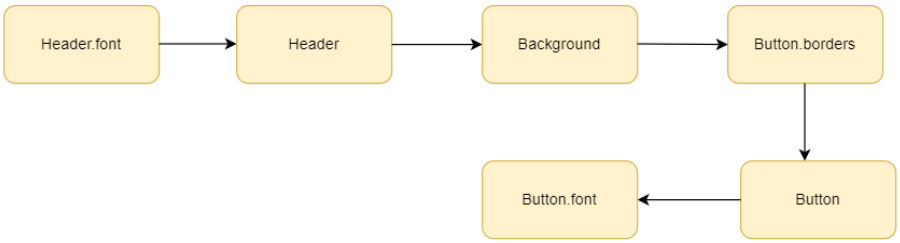
More specifically, the modification of default colors associated with the graphical components is performed from an image by previously producing a component adjacency graph. Indeed, considering the graphical user interface of an application (e.g. Figure 2a), the corresponding blueprint mode is activated. This permits to obtain a simplified and wireframe view of the GUI (see Figure 2b). The blueprint mode provides a wireframe and complementary view of the GUI which permits to focus on components of the design without the distraction of content.

Both conventional design and blueprint views are exploited in order to define an adjacency graph (see Figure 2c) where each region corresponds to a node [16]. This latter graph can then be represented as a component adjacency graph such as shown in Figure 2d. We can observe that the text of the header (font) is connected to the filled header. This latter is connected to the background which is connected to the border of buttons with a single edge as well as to the central image. Since this latter corresponds to a loaded picture which is static in this design, this component is not included in the graph. Button borders are then connected to filled buttons which are themselves connected to the button text.

More precisely, this defined component adjacency graph is then used as a fixed data structure towards simultaneously evaluating the aesthetic quality and the usability of the restyled graphical user interface. More precisely, a metric has been defined in order to globally measure the contrast in between components of



(a) Default application. (b) Blueprint view. (c) Region adjacency graph.



(d) Component adjacency graph.

Fig. 2: Workflow applied to produce a component adjacency graph.

the defined adjacency graph. It is assumed that a high contrast value will result in a visually pleasant GUI with a satisfactory level in component usability.

The components of the defined adjacency graph are encapsulated into one ordered vector $\mathbf{v}$.

$$\mathbf{v} = (C_1, ..., C_n)^T \tag{1}$$

where $C_{i_{(1 \leq i \leq n)}}$ corresponds to there retained adjacency components ($n \geq 2$). The constrast ratio is defined as follows:

$$r = \frac{\sum_{j=1}^{m} \Delta(C_j, C_{j+1})}{m \times \Delta_{max}} \tag{2}$$

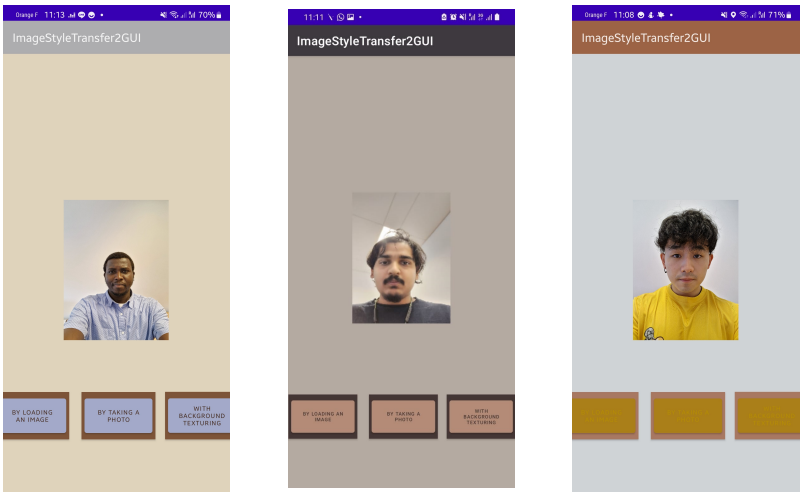
where m represents the number of edges in between components; i.e $m = n - 1$. $\Delta(C_j, C_{j+1})$ corresponds to the Euclidean distance in between dominant color values which are respectively assigned to the components C_j and C_{j+1}. Δ_{max} corresponds to the highest contrasting value being which can be obtained

e.g. when adjacent components are black and white, respectively. In this case, the contrast ratio reaches its largest value $r = 1$.

The number of identified dominant colors is n since it corresponds to the number of adjacency components (i.e, number of components of $\mathbf{v}$). These retained dominant colors are stored in a vector $\mathbf{w} = (col_1, ..., col_n)$. Therefore, the largest contrast $\mathbf{r}^\star$ can be reached by determining the optimal color distribution $\mathbf{w}^\star$. Specifically, we seek a permutation σ such that $\mathbf{w}^\star = (col_{\sigma(1)}, ..., col_{\sigma(n)})$ maximizes the contrast ratio r. The maximum is reached after at most n! steps ($n!$ is the number of permutations of $\{1, ...n\}$). The computation of $\mathbf{w}^\star$ is not time consuming when n is not large as it is the case in our application. Otherwise, the search space can be reduced by randomly selecting a fixed number of permutation. From these retained permutations, we keep the one which provides the best contrast ratio.

$$\mathbf{r}^\star = arg \max_{\mathbf{w}} c \tag{3}$$

3 Experimental results and evaluation



(a) User test 1. (b) User test 2. (c) User test 3.

Fig. 3: Application dynamically styled by taking successive selfie photos in the context of try-on of clothes.

Figure 3 displays a dynamic restyling of a default application by successively taking selfie photos in the context of try-on of clothes. This study case can be seen as a simulation of a virtual clothes try-on system for which the wearing of

a virtual garment accordingly involves the instantaneous harmonization of the GUI.

We can observe that dominants colors from image objects have been transferred to the graphical components of the interface. In particular, we remark that the photo backgrounds have colored the backgrounds of the application in Figures (3a) to (3c). In Figure 3a, buttons seems colored from the shirt color, button borders seems colored from the skin color. In Figure 3b, buttons seems colored from the skin color, the header seems colored from the T-shirt color. In Figure 3c, buttons seems colored from the T-shirt color, button borders seems colored from the skin color. Visually, outputs of user tests 1 and 2 can be satisfying. However, the output of the user test 3 is not satisfying in reason of the color of the buttons and their inner texts which is relatively close. This makes the corresponding texts unreadable and this application unusable.

Figure 4 shows the application styled by loading images of different natures. Quality of the applications styled from the loaded images is displayed as a score ranging in [0,1] which has been obtained by computing the contrast metric c. Figures 4a to 4f are ordered by their score from lowest to highest style transfer quality.

Figure 4a displays the loading of a kind of chessboard image for which a single dominant color has voluntarily been retained. The resulting design of the application is homogeneous making the background, button borders, buttons and writings appearing as a whole. The obtained null score value corresponds to an application which is neither aesthetic nor usable.

Figures 4b and 4c show outdoor photos. In Figure 4b, we can observe that the color of the vegetation seems reported to the background, button borders and writings. The color of the inner buttons seems to come from the background. The color of cars, although visually consistent, is not used for coloring components since not dominant. The quality of the image transfer is low (score value 0.21) in reason of the low contrast in between components related to the buttons which also makes the interface complex to use. In Figure 4c, the sky color is reported to the header, the remaining components got close colors from the bricks of the building facades. Quality of the styled application is similar to the one of the Figure 4b. No color optimization processes have been operated to Figures 4a to 4c.

Figures 4d depicts an indoor photo. The colors, although relatively limited in term of variety, are distributed to components with a satisfactory way. We observe a high contrast, distinguishable components, readable writings and usable button functionality. We also observe the same results by using artistic images which have a variety of colors; e.g. (see Figures 4e and 4f). It seems that the quality of the styled application is satisfying in term of aesthetic (graphical coherency) and usage when the contrast score is superior or equal to 0.44.

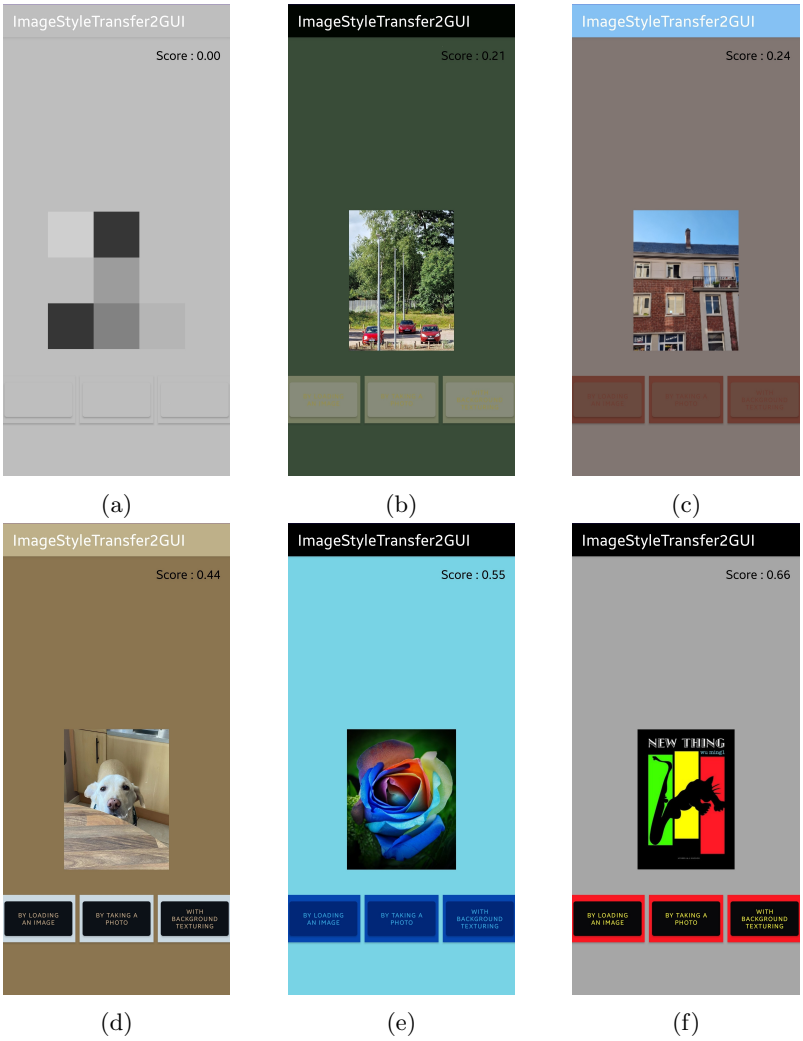


Fig. 4: Application styled by loading images of different natures. Quality of the applications styled from the loaded images is displayed as a score ranging in [0,1].

4 Conclusion

An approach has been proposed for restyling the graphical user interface of applications from an image. Such approach can be integrated as a complementary tool for application development cycles in order to facilitate the creation of designs. Such approach can also be included in a large spectrum of applications towards permitting to end-users the self-changing of the design through an image loading or taking functionality. The transfer of colors is performed without

invoking neural analysis techniques which can be time-consuming in training stages. The employed graph design and optimization process provide a direct and promising solution for restyling applications from query images.

Acknowledgments

The authors would like to thank the students Goutam Manjunath Shanbhag, Wei-Chien Chao, Mariam Helala, Serge Niyigena Gihozo, Olakunle Saheed Ogunsolu, Prabhu Kumar Reddy Marry, Shailesh Kumar of the Master training in Software Engineering & Digital Transformation, ESIGELEC School of Engineering, Rouen, France for their assistance in development and experiment stages.

References

1. E. Grinstein, N. Q. K. Duong, A. Ozerov, and P. Pz, "Audio style transfer," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 586–590, 2018.
2. K. Yin, J. Gao, M. Shugrina, S. Khamis, and S. Fidler, "3dstylenet: Creating 3d shapes with geometric and texture style variations," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2021.
3. S. Cygert and A. Czyzewski, "Style transfer for detecting vehicles with thermal camera," in *2019 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, pp. 218–222, 2019.
4. O. Texler, J. Fišer, M. Lukáč, J. Lu, E. Shechtman, and D. Sýkora, "Enhancing neural style transfer using patch-based synthesis," in *Proceedings of the 8th ACM/Eurographics Expressive Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, Expressive '19, (Goslar, DEU), p. 4350, Eurographics Association, 2019.
5. D. Kotovenko, M. Wright, A. Heimbrecht, and B. Ommer, "Rethinking style transfer: From pixels to parameterized brushstrokes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12196–12205, June 2021.
6. X. Liu, W. Wu, H. Wu, and Z. Wen, "Deep style transfer for line drawings," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 353–361, May 2021.
7. E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Computer Graphics and Applications*, vol. 21, no. 5, pp. 34–41, 2001.
8. Y. Shih, S. Paris, F. Durand, and W. Freeman, "Data-driven hallucination of different times of day from a single outdoor photo," *ACM Transactions on Graphics (TOG)*, vol. 32, 11 2013.
9. J. Yoo, Y. Uh, S. Chun, B. Kang, and J.-W. Ha, "Photorealistic style transfer via wavelet transforms," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9035–9044, 2019.
10. F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep painterly harmonization," *arXiv preprint arXiv:1804.03189*, 2018.

11. S. Yang, L. Jiang, Z. Liu, and C. C. Loy, "Pastiche master: Exemplar-based high-resolution portrait style transfer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7693–7702, June 2022.
12. G. Kwon and J. C. Ye, "Clipstyler: Image style transfer with a single text condition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18062–18071, June 2022.
13. Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, "Neural style transfer: A review," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, pp. 3365–3385, nov 2020.
14. M. H. Fischer, R. R. Yang, and M. S. Lam, "Imaginenet: Restyling apps using neural style transfer," 2020.
15. Q. Zheng, M. Lu, S. Wu, R. Hu, J. Lanir, and H. Huang, "Image-guided color mapping for categorical data visualization," *Computational Visual Media*, 2022.
16. A. Tremeau and P. Colantoni, "Regions adjacency graph applied to color image segmentation," *IEEE Transactions on Image Processing*, vol. 9, no. 4, pp. 735–744, 2000.