**Interactive Computer Graphics 2023 Spring, Term Project Report**

**Color Harmonization**

Chin-Chieh Hsu 許晉捷 F09922184

# Abstract

An image can be considered harmonious if we feel comfortable based on our individual perspective. Conventionally, artists and visual designers have been relying on their own instincts and artistic feelings to create impressive harmonic artworks. Color harmonization, on the other hand, is a computerized technique that applies a statistical algorithm to alter a digital image into a harmonized one. In this report, I will provide a detailed explanation of the methodology behind color harmonization, along with my own comprehensive implementation with a Graphical User Interface (GUI) using Python. Additionally, to elevate the performance in terms of time cost, some additional features not included in the original paper have been implemented.

# Contributions (貢獻)

The main contributions in my implementation can be categorized into two areas: GUI-related and algorithm-related, and are summarized as follows:

* + Algorithm-related contributions
    1. The first full implementation of the original paper (Cohen-Or et al., 2006) thus far, including all details of the algorithm, with additional features as undermentioned, among resource that can be found on public Internet, to the best of my knowledge.
    2. The harmonic scheme of the harmonization process can be either based on the target image itself, or on an optional reference image.
    3. In conjunction with a Super Resolution (SR) technique (Wang et al., 2021), an enhancement has been made to improve the efficiency of the algorithm. This enhancement enables the use of smaller-sized inputs for the main process without sacrificing quality. Further details are described in Section 4.
    4. In conjunction with GrabCut (Rother et al., 2004), a foreground-background image segmentation technique, a high-level application has been developed to let users selectively apply the color harmonization process on background (and the color harmony is based on the foreground), or vice-versa. This was also mentioned in the original paper. Further details are described in Section 4.
    5. Users have the option to choose between selecting a specific template type or allowing the program to automatically search for the best template type. The concept of template types will be discussed in greater detail in Section 3.
  + GUI-related contributions
    1. The first implementation along with a stand-alone GUI, including several extensive and user-friendly features, listed undermentionedly.
    2. Four ways to load image: 1) from local 2) drag-and-drop 3) from URL 4) from clipboard. This enables the ample flexibility for individual users.
    3. A specific GUI panel for algorithm settings for each single loaded image individually. The settings incorporate the template type of the harmonic schemes, the resize-ratio, and an optional reference image. These terms will be further explained in Section 3.
    4. All algorithms run on other threads. The user-interface is not frozen; thus, users can continue operating (e.g., load another image) while the harmonization process is concurrently running.
    5. An immediate and clear comparison will be displayed right after a process is done. This further enhances the user experience in terms of convenience and practicality.

# Introduction

The research of the color harmony has been elapsed a long time, intertwined with the some modern color model and color theory with respect to the human perception. Among those previous works, Matsuda introduced a comprehensive set of harmonic templates for color design (Matsuda, Y. 1995) (Tokumaru et al., 2002) that we leverage for harmonious evaluation and a further calculation, shown in *Figure 1*.

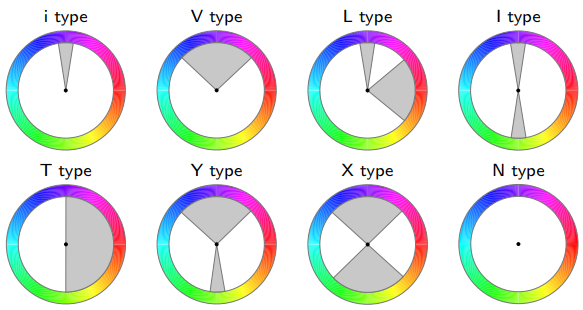


Figure 1. The eight harmonic templates as hue wheels, represented in HSV color space, developed by Matsuda (The figure originates from the paper (Cohen-Or et al., 2006)).

As Figure 1 shown, there exist eight types of harmonic templates: i-type, V-type, L-type, I-type, T-type, Y-type, X-type, and N-type. Each template is represented as a ring with hues in the HSV (hue, saturation, and value) color space and one or two sectors, except the N-type. The sectors in each template signify each type of color harmony, i.e., if all pixels’ hues of an image are located inside that one or two sectors in any one of the templates, the image is considered harmonious. Note that here we do not incorporate the N-type template into consideration, since it was proposed for grayscale images.

The goal of this computerized color harmonization algorithm is to alter a digital image by “shifting” the hues of all pixels along with a ring (a template) in order to make them be inside the sectors. This is depicted in Figure 2, which shows the goal of the algorithm. The upper row is the raw hues in a ring-shaped histogram and the raw image; the second row is the eventual result after the algorithm. As the figure shows, we “pushed” the raw hues on the ring and enforcedly “shifted” them into sectors of the chosen I-type template in a statistical way.



Figure 2. The goal of the color harmonization algorithm: the upper row is the raw hues in a ring-shaped histogram and the raw image; the second row is the eventual result after the algorithm. As the figure shows, we “pushed” the raw hues on the ring and enforcedly “shifted” them into sectors of the I-type template (This figure with ring-shaped histograms is drawn and generated by my own program; the raw image originates from the paper; the harmonized image is the result of my program).

In details, algorithmically speaking, in this project, I have implemented the full content of the original paper, including 1) harmonic scheme selection for the best one, 2) Graph-cut optimization (Boykov & Jolly, 2001) based on the selected best harmonic scheme, and 3) the color-shifting in conjunction with a normalized Gaussian function, as well as 4) some high-level application such like foreground-only or background-only harmonization with the help of image segmentation. Furthermore, 5) the option for users to apply the harmonization process based on either the target image itself or an optional reference image. These all result in a satisfied completeness and accomplishment implementation, which is the first one that implemented the full process and features of the original paper among the resource that can be found on public Internet, to the best of my knowledge. Further details will be illustrated in the following sections.

# Method

In this section, the entire color harmonization algorithm (Cohen-Or et al., 2006) is elaborated in details. The algorithm can be subdivided into three stages: 1) Harmonic scheme selection. 2) Graph-cut optimization. 3) Color shifting.

1. Harmonic scheme selection

A harmonic scheme is defined as , where is the “index” of a template () and is defined as an orientation (offset) of a template. In this stage, we aim to select a best harmonic scheme along with the , the template indexed by , that minimizes the following objective function



, where *X* is the input image, is the harmonic scheme, *p* belonging to *X* represents all pixels in *X*, *H* and *S* refers respectively to the hue and saturation, and *E* denotes the closest sector’s border from *p*. In particular,



represents that given a specific harmonic scheme , the closest border of the sector in the , which is “rotated-by-“, from a pixel *p*. Finally, denotes the arc-distance on the ring.

This objective function *F* measures the color harmony of an image *X*, given an arbitrary harmonic scheme . The author also considered the saturation channel to mitigate the overestimated effect caused by the hue channel. We aim to minimize *F* to search for the best harmonic scheme, using Brent’s method (Press et al., 2007), , such that



In practical, in order to save the time cost to do optimization, instead of applying Brent’s method seven times to find out seven different -values for each template type, we can let the user to choose a designated template type and apply merely once Brent’s method. This may lead to a suboptimal result, since the chosen-by-user template type may not be the best one that can actually minimize *F*; however, it will be much more efficient by doing this.

The visualized process finding the best harmonic scheme can be depicted in Figure 3, as the red arrow represents. The “rotated” I-type template at the lower-right corner in Figure 3 can now be construed as the result of the first stage, i.e., the best harmonic scheme that we have found out.

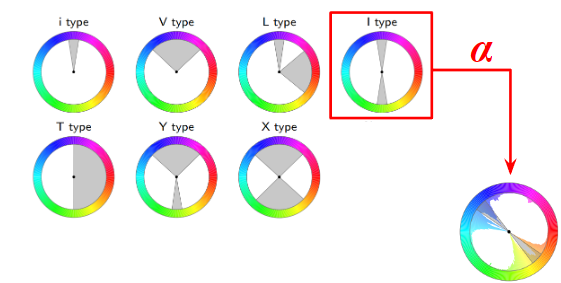


Figure 3. The visualization of the goal of searching for the best harmonic scheme . In this particular case, is I (thus is the I-type template). The “rotated” I-type template at the lower-right corner can now be construed as the result of the first stage, i.e., the best harmonic scheme that we have found out.

1. Graph-cut optimization

Now we have the best harmonic scheme, as shown as the “rotated” template at the lower-right corner in Figure 3. The problem now we have is that: can we shift the hues on the ring-shaped histogram along with the template directly and coerce them be located inside the sectors, e.g., the nearest sector? Unfortunately, the answer is NO, because we have not taken the spatial coherence into consideration yet.

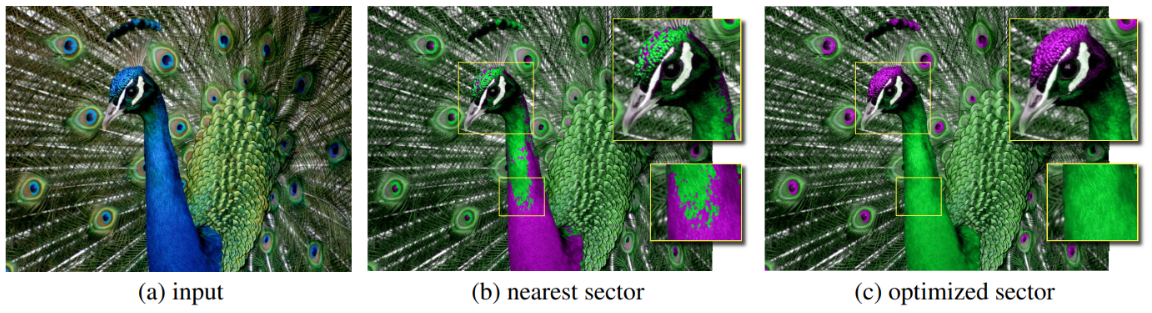
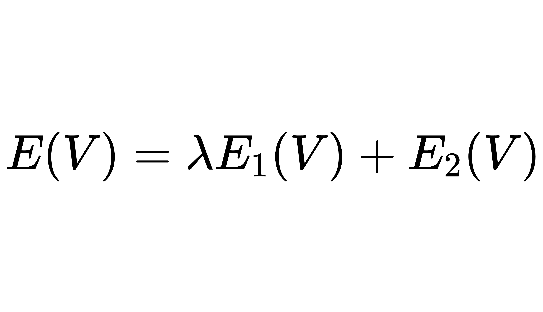
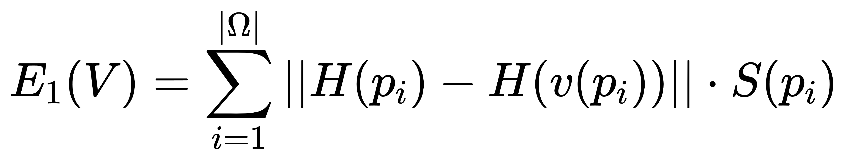


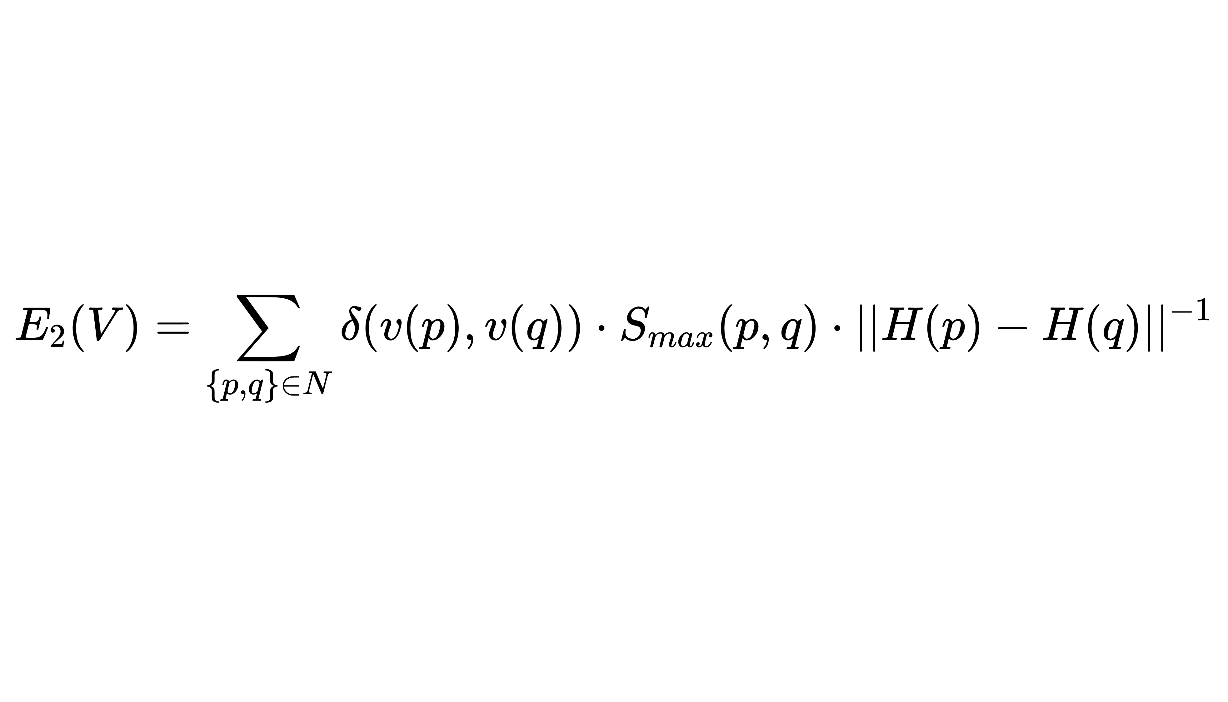
Figure 4. (a) the raw image (b) the result if we directly move each hue which are originally outside any sectors to the border of its nearest sector (c) the result after applying the Graph-cut optimization (The figure originates from the paper (Cohen-Or et al., 2006)).

As shown in Figure 4, If we simply coerce and shift each hue toward the border of its nearest sector, the unpleasant visual artifacts will be generated clearly. To produce the appealing results like (c) in Figure 4, in contrast to employing a harsh shifting method, the renowned Graph-cut optimization technique is adopted here, which offers a more refined approach to attain color harmonization (Boykov & Jolly, 2001). By utilizing this technique, the process becomes more precise and nuanced, allowing for better control over the harmonization process.

Initially developed for foreground-background image segmentation, Graph-cut optimization is a technique originally used for binary classification tasks on images. In the context of the color harmonization process, the objective is to assign a label to each pixel indicating its movement direction: clockwise or counterclockwise? This labeling process can also be viewed as a binary classification task on images. Consider the following energy function *E*:







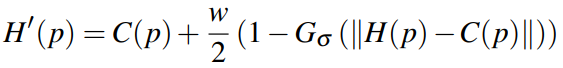
, where is the set of input pixels, *N* is the set of all adjacent pixels in , and is a balancing parameter between equations and . Our objective is to search for a set of labels , where each is a binary label with respect to a pixel denoting the destination sector to which whether the hue of the pixel is moved clockwisely or counterclockwisely .

Similar to the harmonic scheme selection in described in the previous subsection, the objective of the equation is to evaluate the distances between the raw hue and the “shifted-to-the-destination” hue. Here, the saturation channel is also considered to weigh the contribution from those pixels with low or high saturations. Secondly, , as a regularization term, accounts for enhancing the spatial color coherence among adjacent pixels which are assigned to the same label by penalizing the different assignments among the adjacent labels with same or similar hues. The term , where *p* and *q* are a pair of arbitrary adjacent pixels, in takes responsibility for this penalization: if and are different, i.e., one moves clockwisely; another one moves counterclockwisely, this delta function results in 1 and the penalization ensues; otherwise, 0.

To do this minimization on the energy function *E*, Graph-cut represents the image as a graph, where nodes are construed as pixels and edges represent relationships between adjacent pixels. By assigning weights to these edges based on color similarities in terms of spatial coherence, the optimization algorithm can determine the most harmonious color transitions within the image. This method ensures that color modifications are carried out smoothly and seamlessly, resulting in visually pleasing and natural-looking harmonized images.

1. Color shifting

Up to this point, we have already acquired the best harmonic scheme , along with the binary label set that depicts the prescribed movements for each pixel. Now, it is time to do the actual color-shifting:



, where *C(p)* is the center of the destination sector with respect to which the pixel *p* is moving to, *w* denotes the arc-width associated with that sector, and is a normalized Gaussian function with mean equals to 0. The adoptation of is to exert the ease of color changes and to avoid terse color alterations, e.g. shifting colors all to the sectors’ borders, which can probably lead to visual artifacts.

# Implementation

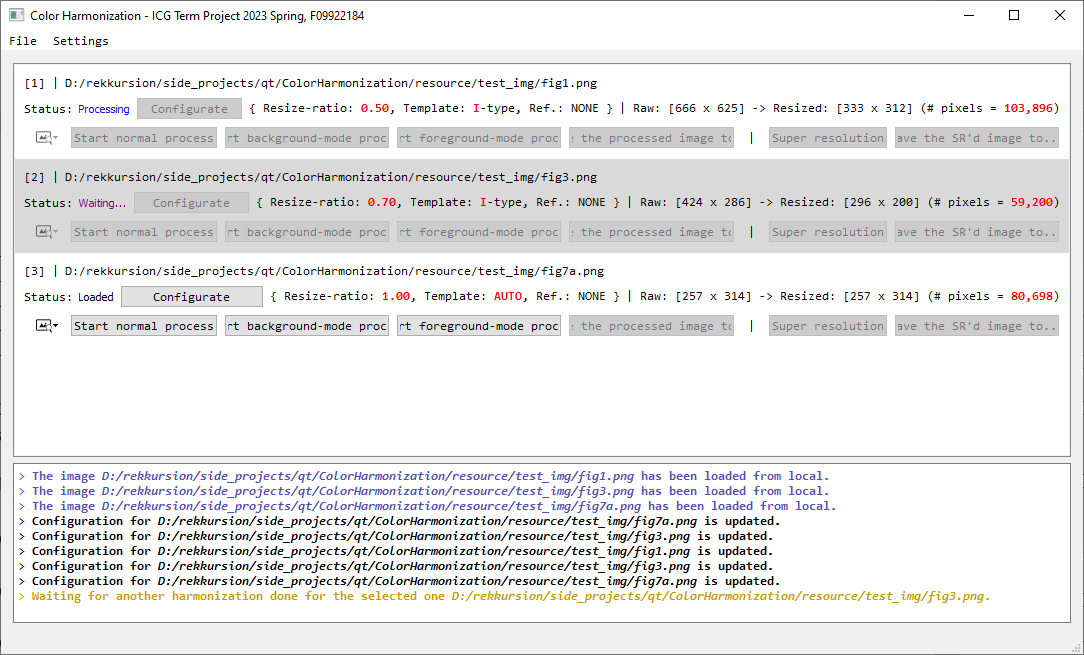
I have implemented this project in Python 3.8.0 with a user-friendly GUI system. In this section, I will introduce all details and features regarding my own implementation.

1. Implementation Environment

|  |  |
| --- | --- |
| Platform | Windows 10 |
| Development tools | Visual Studio Code with Cygwin |
| Programming language | Python 3.8.0 |
| Third-party packages | See the attached *requirements.txt* |
| Other dependencies | Real-ESRGAN (Wang et al., 2021) |
| How to run | See the attached *README.md* |

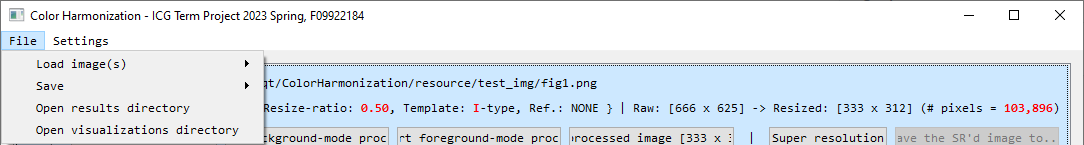
1. Features with GUI
2. Overview

(A)



(C)

(B)



(A)

Figure 5. The main window of my program. (A) the title and the toolbar; (B) the list of items, i.e., all loaded images; (C) the log area.

Figure 5 shows the primary window of my program, which consists of three major components: (A) the toolbar, taking responsibility for global operations, such like file input/output, directory operations, and global settings; (B) the list of items, presenting all loaded images with comprehensive information, for each loaded image, including the path of the image, current status (*loading*, *loaded*, *waiting*, *processing*, *done*, or *error*), algorithmic configuration (i.e., resize-ratio, template type, and the optional reference image), and several operational buttons (alter configuration, start process, start Super Resolution process, save the result to the designated path, and show the image and visualization); and (C) the log area.

1. Image loading/writing

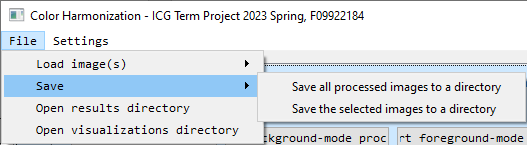
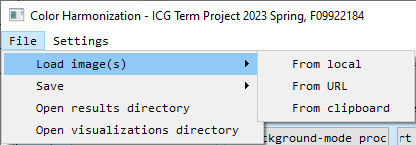


Figure 6. The program provides up to four ways to load images. The left figure displays the first three ways: (A) from local storage; (B) from URL (for Internet resources); (C) from clipboard; and (D) by drag-and-drop. The right figure shows that the program can let users conveniently save results, either all, or all selected, to a designated directory.

Figure 6 regards the file input, the program provides up to four different ways for users to load images: (A) from local storage via OS-native file dialogue (B) by dragging an image and dropping within the interface of the program (C) from a Uniform Resource Locator (URL) to let users conveniently load the image through Internet without explicitly downloading it (D) from the OS-native clipboard: the user can copy an image and it will be stored in the clipboard provided by OS. Once the user clicks the button that activates the image-loading “from clipboard”, the program will automatically read the stored image from the clipboard. This is the most convenient feature for image loading, in my personal opinion. As for the file output, the process results and visualizations will automatically be saved to a pair of pre-determined paths, respectively. These paths could be altered by users via a dedicated global preference panel. Moreover, the results could also be saved to arbitrary local storage explicitly by user, if intended.

1. Global settings

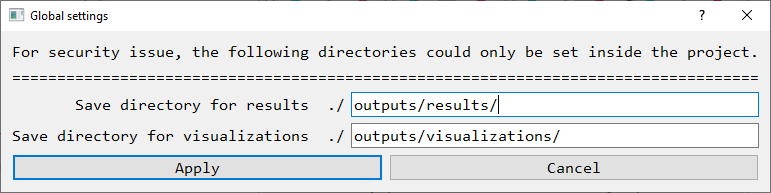
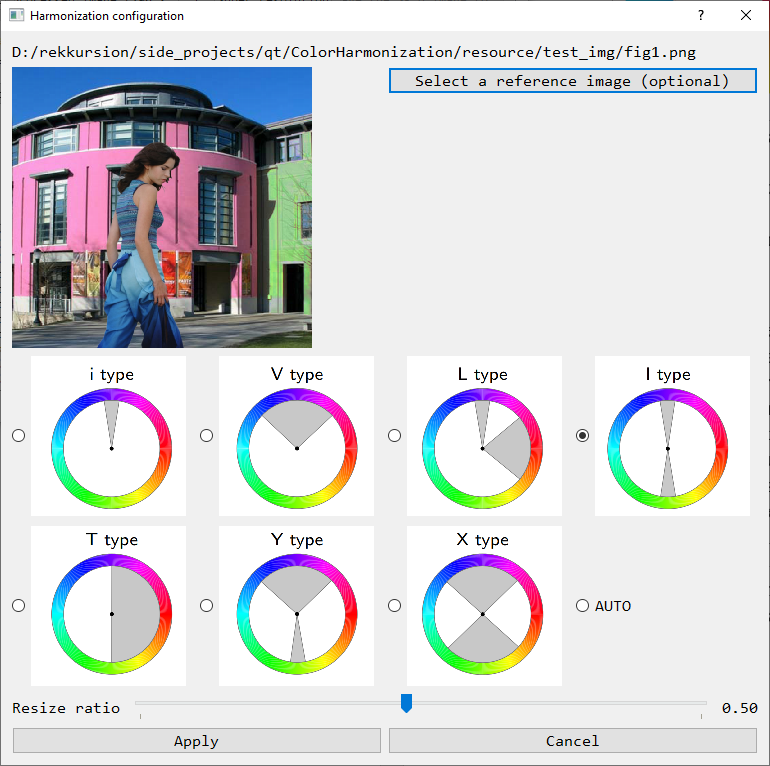


Figure 7. The global settings panel can let users set their prefered saving directories for both results and visualizations.

The global settings panel, which can be invoked be clicking the second item *Settings* at the toolbar, undertakes the arragement of saving directories, both for results and visualizations, as Figure 7 reveals. The default directories are respectively *./outputs/results/* and *./outputs/visualizations/*. Note that for security issue, both directories can only be inside the executable, i.e., inside the current working directory.

1. Individual settings for each loaded image

Following is Figure 8, which exposes the individual settings for each loaded image. These specific settings are comprised of: (A) which template type is prefered, besides, there is also a radio-button *AUTO* for automatically search for the best template type, by the algorithm elaborated in Section 3; (B) the resize-ratio, for shrinking the image; thus the computational cost can be reduced drastically since the number of pixels, as well as the number of vertices in the graph in the Graph-cut stage explained in Section 3, are reduced significantly; and (C) the selection of the optional reference image, which enables the ability to do color harmonization based on the harmony from another reference image, instead of the input image itself.



(C)

(B)

(A)

Figure 8. The case-specific algorithmic configuration panel.

1. Process and visualization

SD



1. Harmonization based on another reference Image
2. Adoptation of Super Resolution as a post-process after the core process
3. Background (or foreground) harmonization
4. s

# Results and Discussion

# Conclusion

# Implementation-Related Acknowledgements

# Reference

Boykov, Y. Y., & Jolly, M.-P. (2001). Interactive graph cuts for optimal boundary & region segmentation of objects in ND images. Proceedings eighth IEEE international conference on computer vision. ICCV 2001,

Cohen-Or, D., Sorkine, O., Gal, R., Leyvand, T., & Xu, Y.-Q. (2006). Color harmonization. In *ACM SIGGRAPH 2006 Papers* (pp. 624-630).

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.

Rother, C., Kolmogorov, V., & Blake, A. (2004). " GrabCut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, *23*(3), 309-314.

Tokumaru, M., Muranaka, N., & Imanishi, S. (2002). Color design support system considering color harmony. 2002 IEEE world congress on computational intelligence. 2002 IEEE international conference on fuzzy systems. FUZZ-IEEE'02. Proceedings (Cat. No. 02CH37291),

Wang, X., Xie, L., Dong, C., & Shan, Y. (2021). Real-esrgan: Training real-world blind super-resolution with pure synthetic data. Proceedings of the IEEE/CVF International Conference on Computer Vision,

# Matsuda, Y. 1995. Color design. Asakura Shoten.