An Interactive Tool For Gamut Masking

Ying Song Cheryl Lau Sabine Süsstrunk

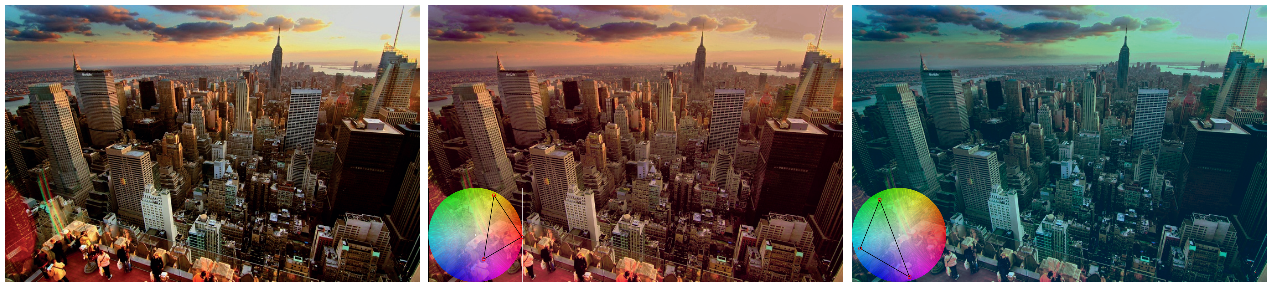
School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

ABSTRACT

Artists often want to change the colors of an image to achieve a particular aesthetic goal. For example, they might limit colors to a warm or cool color scheme to create an image with a certain mood or feeling. Gamut masking is a technique that artists use to limit the set of colors they can paint with. They draw a mask over a color wheel and only use the hues within the mask. However, creating the color palette from the mask and applying the colors to the image requires skill. We propose an interactive tool for gamut masking that allows amateur artists to create an image with a desired mood or feeling. Our system extracts a 3D color gamut from the 2D user-drawn mask and maps the image to this gamut. The user can draw a different gamut mask or locally refine the image colors. Our voxel grid gamut representation allows us to represent gamuts of any shape, and our cluster-based image representation allows the user to change colors locally.

Keywords: interactive tools for artists, aesthetic reproduction, color manipulation, gamut mapping, gamut representation, gamut masking

1. INTRODUCTION



(a) original (b) warm color scheme (c) cool color scheme

Figure 1. **Interactive tool for gamut masking.** Our tool allows users to change colors in an image in order to express a desired mood or feeling. Warm and cool color renderings of the original image were created with our interactive tool by mapping the images to gamuts computed from the user-drawn masks (bottom left). *Original image courtesy of Flickr user Jerry Ferguson.*

Artists often choose to use a particular set of colors in order to create a certain mood or feeling in a painting or image. Gamut masking is a technique developed by painter James Gurney [1] to help artists develop color palettes, or sets of paints, to use while painting in order to achieve a desired mood or feeling in their artwork. In gamut masking, a mask is placed over the color wheel as in Figure 4(a), and the artist limits the colors to those within the mask. The artist chooses the shape of a mask and where to place it. Different mask shapes placed at different locations on the wheel will lead to color schemes that evoke different feelings.

We propose an interactive tool for gamut masking for amateur digital artists. The tool allows users to design their own gamut masks and transform images in order to convey a particular mood. First, the user creates a 2D mask over the color wheel. Then, we extrapolate a 3D gamut in CIE LUV space from the 2D mask. The input

Author correspondence:
ying.song@epfl.ch, cheryl.lau@epfl.ch, sabine.susstrunk@epfl.ch

image is mapped to this 3D gamut so that it mostly contains only colors within the gamut mask. At this point, the user is free to interact with the system to locally change colors as desired. Our interactive tool enables users to digitally apply gamut masking on an input image and refine their creation to achieve their artistic goals.

Our tool supports all kinds of gamut mask shapes because it is important for artists to have the flexibility to create any gamut mask. The mask shape and placement determines the coloring and feel of the final image. For example, in Figure 1, a triangular mask is placed over different sets of colors to produce warm and cool color renderings of the same image. As with Richard Robinson's online tool,[2] the user can select from a variety of preset shapes or draw his/her own freehand mask and choose its placement over the color wheel. Robinson's tool allows users to create masks, print them, and use them as a reference when painting. Our tool creates a gamut from the mask and maps the image to the gamut. Figure 2 exhibits masks overlaid over the colors of famous paintings. These masks have extreme concavities and multiple components and so will their 3D gamut extensions. Thus, artist-drawn gamut masks would not necessarily share the same characteristics as display gamuts. This makes it hard to apply standard gamut mapping techniques. In order to handle arbitrary gamut shapes, we represent the gamut using a voxel grid. This enables us to represent gamuts of any shape, including those with concavities, multiple components, and/or holes.

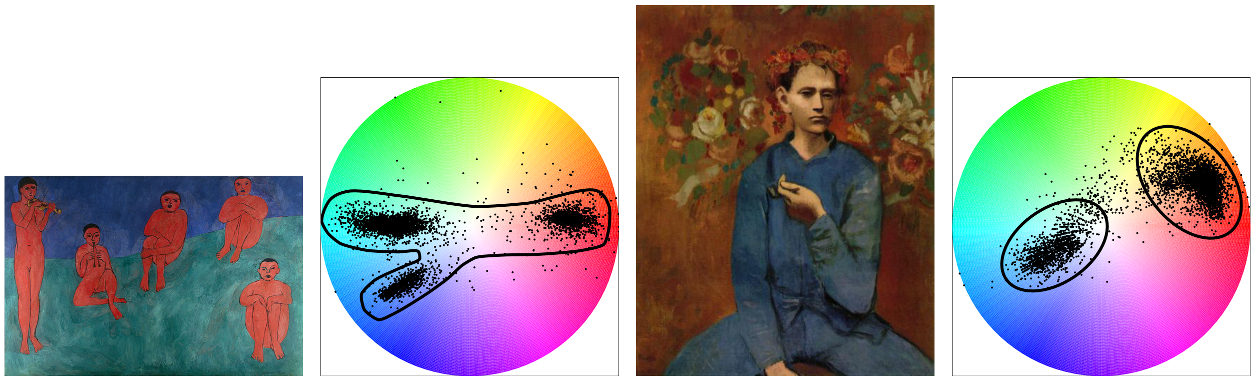

(a) Matisse (b) split complement (c) Picasso (d) multiple components

Figure 2. **Color schemes from famous paintings.** Henri Matisse's *Music* (a) exhibits a split complementary color scheme, while Pablo Picasso's *Garçon à la Pipe* (c) mostly contains colors within two main components. We plot pixel colors from each painting and overlay split complementary and multiple component gamut masks, respectively, over the color wheels in (b) and (d). *Images of paintings courtesy of Wikipedia.*

After the initial mapping to the gamut created from the mask, the user can change the colors in the image locally, similar to color grading. The goal of color grading in movies is to change the colors in an image to evoke a certain feeling, direct the viewer's attention, and/or adhere to the color script of the movie. During the process of color grading, it is common to change colors locally.[3,4] In our system, image pixels are clustered to form groups of pixels with similar colors and spatial locations, and it is these clusters that the user can manipulate in a 3D color space. By changing the colors of each cluster, the user can locally control the colors in the output image.

We propose an interactive tool for amateur artists to apply gamut masking to an image and change the colors of an image in order to achieve their artistic goals. Our voxel grid gamut representation allows us to represent gamuts of any shape, and our cluster-based image representation allows the user to change colors locally.

2. RELATED WORK

Our method relies on gamut mapping to enforce the mask constraints on the image. Many gamut mapping methods have been developed, and we refer the reader to Jan Morovic's book[5] for a summary of this topic. In particular, our tool uses the method of Lau et al.[6] to perform the cluster-based gamut mapping once the gamut has been constructed and when the user modifies the gamut or the clusters. Their method performs an optimization when mapping the clusters to the gamut, but our method skips the optimization and only maps clusters to their closest points within the gamut boundary. Instead of providing a single output, the artist is free play around with the image colors by moving clusters around to see what works best for his artistic goals. Some interactive gamut mapping tools have been developed, but they lack the flexibility to allow users to create their own gamut masks and change local image colors. Stone et al.[7] and Kalra[8] let users vary transformation parameters for translation, rotation, and scaling transformations performed on the gamut. Farup and Hardeberg[9] propose a tool in which a user can interactively change the gamut surface by moving tetrahedra on the gamut boundary. These changes are local to the gamut but not local to the image in a spatial sense. Willett et al.[10] created an applet to demonstrate gamut mapping to different gamuts under different rendering intents. The applet lets the user create gamuts by arranging 3-5 primaries in xy chromaticity space, but it is not used for changing local colors in an image. These interactive gamut mapping tools were not developed for the purpose of gamut masking, so they do not allow users to draw their own gamut masks of arbitrary shapes. Richard Robinson created an online tool for drawing gamut masks,[2] but these masks are intended to be printed and used as a reference when painting. Our tool lets users draw a gamut mask, provides an initial mapping according to the gamut mask, and lets users change the colors of the image within the context of the gamut created.

There are many solutions for color editing from commercial software to specific methods for color transfer. Commercial software for general image editing includes Adobe Photoshop and Adobe Lightroom. Commercial software specifically designed for color grading includes Adobe SpeedGrade, Blackmagic Design Da Vinci Resolve, Autodesk Lustre, Digital Vision Nucoda, and FilmLight Baselight. Bonneel et al.[11] and Pitié et al.[12] propose automatic color grading methods to color grade images and videos according to reference images and videos. Starting from the color transfer work of Reinhard et al.,[13] many methods, automatic and interactive, have been developed to transfer the colors from one image to another. With interactive color transfer methods,[14–20] the user marks corresponding regions in the source and destination images to be used for color transfer. In contrast to the commercial software and the interactive color transfer methods, our tool offers a different kind of interaction. Our tool enables the user to create gamut masks by drawing 2D masks on a color wheel and explore the limited palettes of different gamut masks.

3. METHOD

The diagram in Figure 3 illustrates the user's interaction with the tool and the underlying processes of our method. First, the user draws a 2D gamut mask from which we extrapolate a 3D gamut. Second, we cluster the input image in spatio-chromatic space and gamut map it to the gamut using the cluster-based gamut mapping method of Lau et al.[6] Then, the user is free to experiment with different gamut masks or interact with the clusters by moving them around in color space to change their colors. With this interactive stage, our tool enables users to explore the space of different masks and their effects, helping them decide which mask achieves the desired mood. The ability to change the colors of clusters lets the user make local color changes to the image. After each cluster movement, pixel colors are blended to get the output image.

3.1 User Interface

Figure 4 shows a screenshot of our tool. In the first panel 4(a), the user draws a 2D gamut mask on top of the HSV color wheel. The user can draw this mask freehand or can select a preset shape. The preset shapes include a triangle, a circle, and a mood-accent scheme consisting of a large ellipse and a smaller circle of accent colors. In the mood-accent scheme, the larger ellipse represents the dominant hue of the image, and the smaller circle represents an accent color typically used for highlights. The preset shapes can be resized and moved around the wheel. For a smoother freehand curve, we fit a piecewise polynomial cubic spline to the underlying points of the user's mouse movement. The second panel 4(b) shows the resulting 3D gamut. In the third panel 4(c), the user can view the original image, the underlying clusters, or the output image. Here, the user can select a pixel of the image to select its underlying cluster. Selecting the whole cluster lets us apply a color change to a group of pixels with a similar color in the same local spatial region. To change the cluster's color, this cluster can then be moved around in the u^*v^* plane in the second panel, and its lightness (L^*) value can also be changed through the slider. When a cluster is moved, we update the affected pixels in the output image displayed in the third panel. Our tool lets the user create different mask shapes, apply the masks to input images, and modify image colors locally through cluster movement.

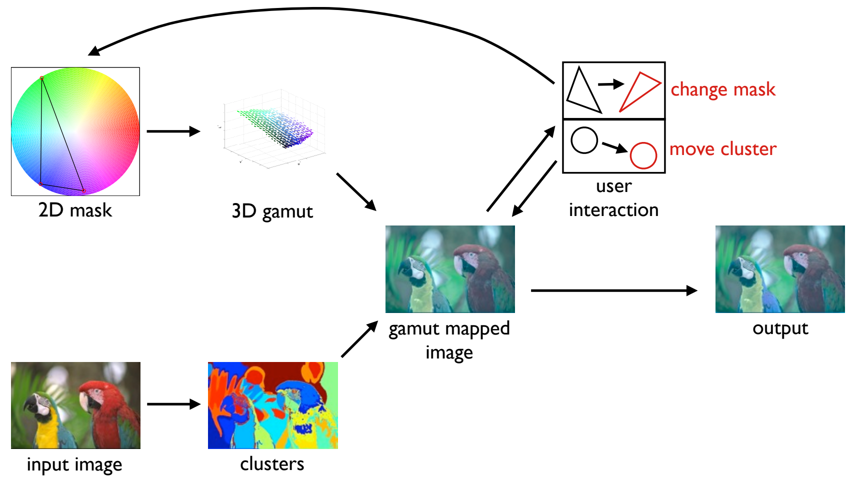
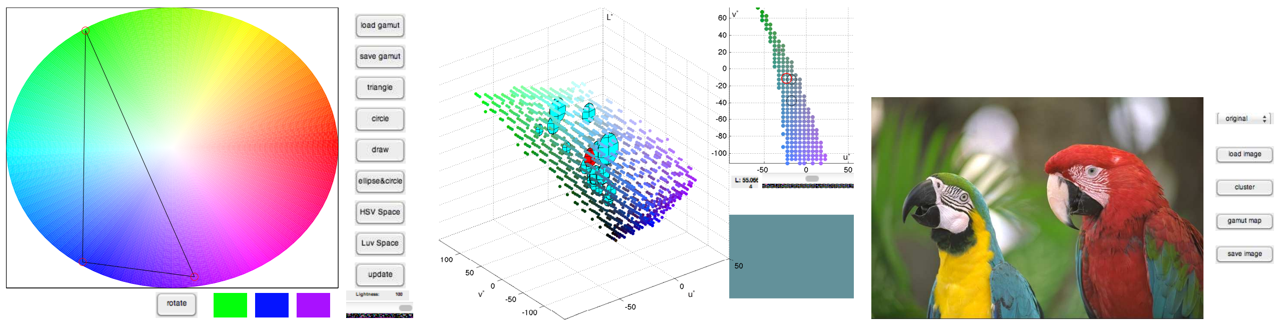


Figure 3. **Overview.** The user draws a 2D mask over the color wheel which we then extend to a 3D gamut. The input image is clustered spatio-chromatically, and these clusters are mapped to the 3D gamut. The user can then either change the mask or change a cluster's color. Our system maps clusters back to the gamut and transfers the results back to the pixels in a blending step to get the output image. *Image of birds courtesy of Kodak.*



(a) mask panel (b) gamut panel (c) image panel

Figure 4. **Interactive tool.** Our tool allows the user to create a 2D mask in (a), visualize the resulting 3D gamut in (b), and compare the input and output images in (c). The user can change the mask in (a), select clusters by clicking on an image pixel in (c), and move a selected cluster by dragging the red circle in the u^*v^* plane in (b). *Image of birds courtesy of Kodak.*

3.2 Gamut Creation

From the user's 2D gamut mask, we create a 3D gamut which we represent using a voxel grid. The user draws a 2D mask on an HSV color wheel at value $V = 1$. HSV space may be more familiar to artists than LUV space, so we let users draw their masks in HSV space while we perform gamut mapping in LUV space to achieve a more perceptual rendering when mapping based on closest distance to the gamut. To convert the HSV mask to LUV, we densely sample the HS plane, find the points inside the mask, and convert those points to u^*v^* chromaticity values. In our voxel representation, the voxel cells containing these points are marked as in-gamut. We then extrude the 2D mask in u^*v^* along the lightness dimension L^* to create a 3D volume in LUV space. Finally, we intersect the extruded volume with the device gamut to get the final gamut. We use sRGB as our device gamut. The voxel grid representation allows us to represent and map to many types of gamuts including gamuts with extreme concavities, multiple components, and/or holes.

3.3 Gamut Mapping

We choose a cluster-based representation for the input image in order to allow local modifications to the image. We follow the method of Lau et al.[6] when creating the clusters, mapping the clusters to the gamut boundary, and blending the mapped results across the pixels. The methodology is briefly summarized here, but we refer the reader to Lau et al. for more details. We cluster the pixels in the image in five dimensions (L^*, u^*, v^*, x, y) where (x, y) are spatial coordinates. To map clusters to the gamut, we map each cluster's mean to its closest point on the level set r from the gamut boundary, where r is the cluster's radius. To transfer cluster movements to the pixels, a blending step calculates the difference vector $\mathbf{d}_q$ for each pixel q as a weighted combination of cluster movements as follows.

$$\mathbf{d}_q = \sum_{i=1}^{n} \omega_{qi} \mathbf{d}_i. \tag{1}$$

The weights ω_{qi} for all pixels q over all clusters i, where $i = 1$ to n, are related to pixel q's Mahalanobis distance to cluster i. These weights do not change, so we only need to compute them once at the beginning of the interactive session. When the user moves a cluster, the cluster's difference vector $\mathbf{d}_i$ changes, and the pixels' difference vectors are recomputed according to Equation 1. Then, each pixel's color is updated by adding its difference vector $\mathbf{d}_q$ to its original color. By enabling cluster movements, we allow the user to make local color changes.

4. RESULTS

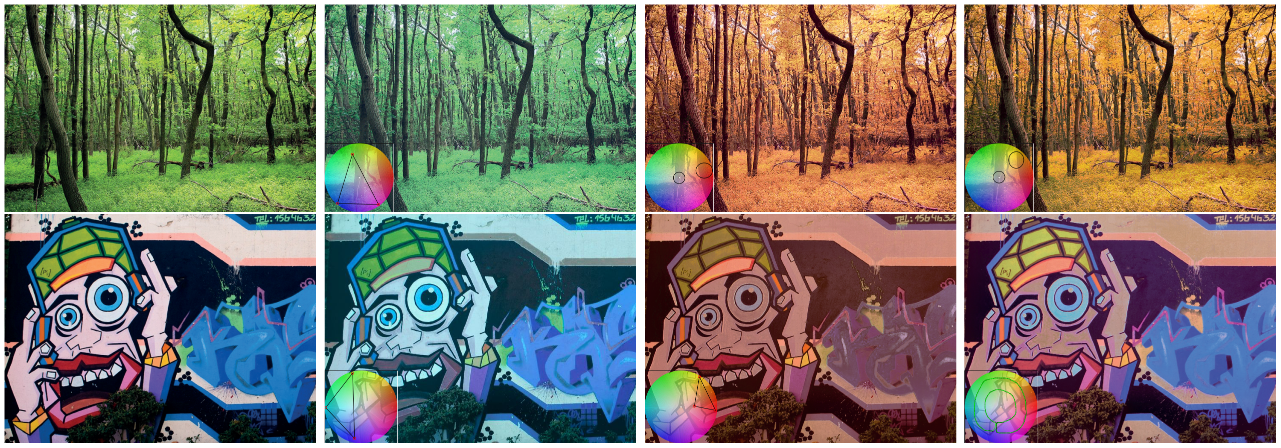


Figure 5. **Different renderings.** By applying different masks to the original images (left), the user creates different renderings of the forest scene and the graffiti artwork which can evoke certain seasonal moods or artistic renditions, respectively. *Original images courtesy of Flickr users Nicholas A. Tonelli and Pablo Romeo.*

Our tool allows users to explore different masks, create their own masks, and change local image colors to determine how to accomplish their artistic goal. Figure 5 shows results of a user creating different renderings of a forest scene and a graffiti artwork by repositioning and redrawing masks on the color wheel. The forest renderings could evoke seasonal moods corresponding to different times of the year. The user explores different artistic renditions by mapping the graffiti image to different masks. In Figure 6, we show that with our voxel grid representation our tool can handle complex gamut shapes. The image of the red shoe is mapped to a "Y"-shaped gamut mask with high curvature concavities in order to achieve a split complementary color scheme. Alternatively, a scheme with a bluish dominant hue and a red accent color is accomplished by mapping the image to a mask with two components. Most of the image is kept within the large blue component while the red shoe falls within the small red component. To make local color changes, the user changes the colors of individual clusters in Figure 7. In the first row, the user changes the color of the red door to an orange door, yellow door, and green door. In the second row, the user applies a greenish tint or bluish tint to the background by mapping the image to a greenish or bluish mask, respectively. Such coloring of the image is often performed in color

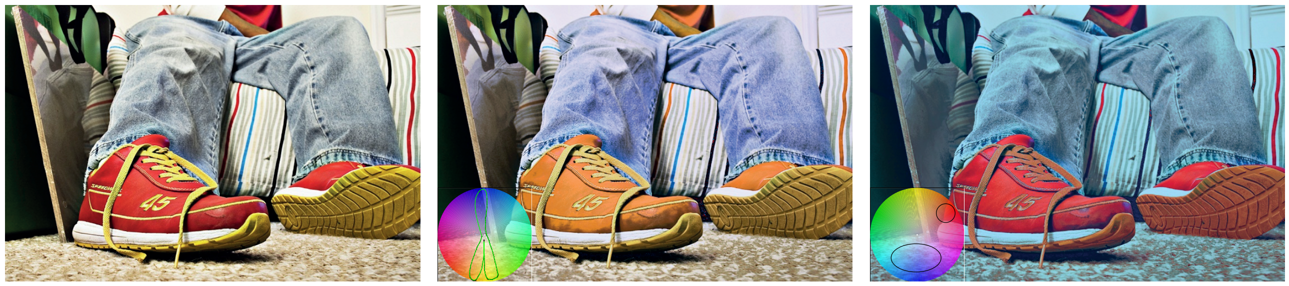

(a) original (b) split complement (c) mood accent

Figure 6. **Complex masks.** Our tool can handle arbitrary shaped masks. The original image (a) is mapped to a split complementary color scheme (b) described by a "Y"-shaped mask with a sharp concavity. The original image (a) is altered by a mood and accent scheme (c) represented by a multiple component mask containing a blue component for the dominant hue of the image and red component for the red shoe. *Original image courtesy of Flickr user malik ml williams.*

grading in order to adhere to a specified color scheme. Then, the user locally adjusts the girl's skin color to make her stand out more from the background. These results show how our tool enables the exploration of gamut masking results, handles complex gamut mask shapes, and allows local color changes.

5. CONCLUSION

We present an interactive tool for gamut masking in which the user draws a 2D gamut mask over a color wheel to specify a limited color palette for the output image. Our system determines the 3D gamut corresponding to the 2D mask and maps the image to this gamut. Our voxel grid representation for the gamut lets us handle arbitrary shaped gamuts, and our cluster representation for the image lets the user easily change the colors of local image regions while also enabling us to remap the results back to the arbitrary shaped gamut. Users can draw different masks to explore different color schemes and change local image colors in order to achieve their artistic goals.

ACKNOWLEDGMENTS

We acknowledge the following Flickr users for their images under a creative commons license: Jerry Ferguson (jerryfergusonphotography), Nicholas A. Tonelli (Nicholas_T), Pablo Romeo (Lieutenant Pol), malik ml williams, Grant MacDonald, colorblindPICASO. In using these images, we are not suggesting that the licensors endorse us or our use of the images.

REFERENCES

[1] Gurney, J., "Gamut masking method (parts 1-3)," (2011). Accessed July 2013. `http://gurneyjourney.blogspot.ch/2011/09/part-1-gamut-masking-method.html`.

[2] Robinson, R., "Gamut mask tool," (2010). Accessed July 2013. `http://www.livepaintinglessons.com/gamutmask.php`.

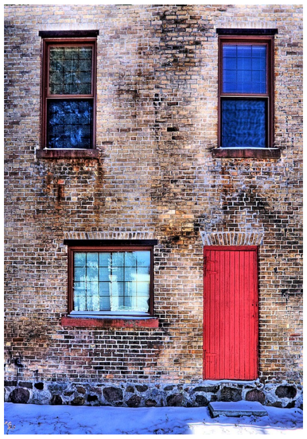
[3] "Lotr digital color grading (parts 1-2)," (2009). Uploaded by kfltd. Accessed July 2013. `http://www.youtube.com/watch?v=M4zRMLbZZxw`.

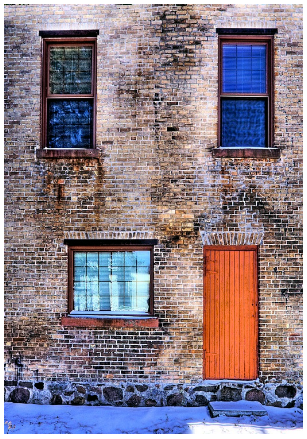
[4] "King kong color grading," (2010). Uploaded by kfltd. Accessed July 2013. `http://www.youtube.com/watch?v=Jov9e4fo7iE`.

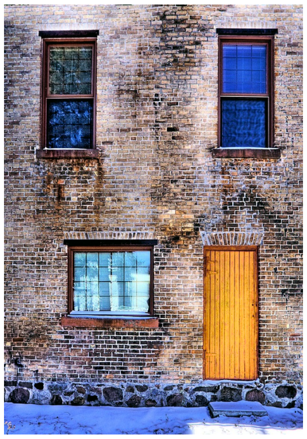
[5] Morovic, J., [*Color Gamut Mapping*], John Wiley & Sons Ltd. (2008).

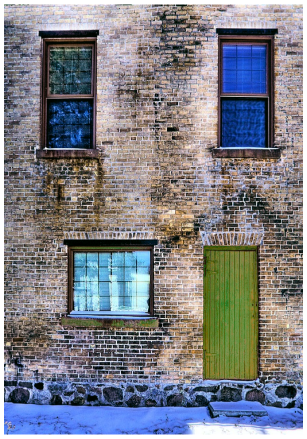
[6] Lau, C., Heidrich, W., and Mantiuk, R., "Cluster-based color space optimizations," in [*Proc. IEEE Int. Conf. Comput. Vision*], 1172–1179 (2011).

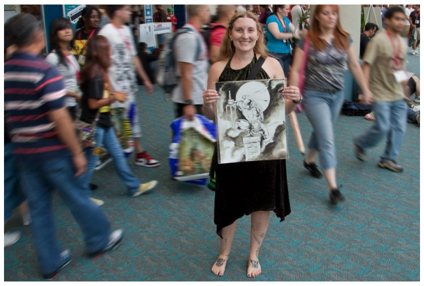
[7] Stone, M., Cowan, W., and Beatty, J., "Color gamut mapping and the printing of digital color images," *ACM Trans. Graph.* **7**(4), 249–292 (1988).

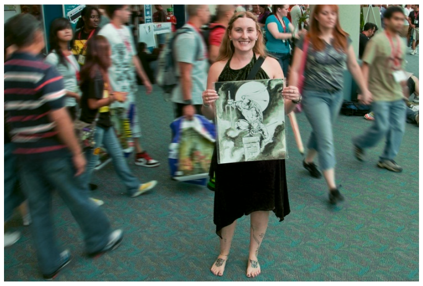

(a) original


(b) orange door


(c) yellow door


(d) green door


(e) original


(f) green tint

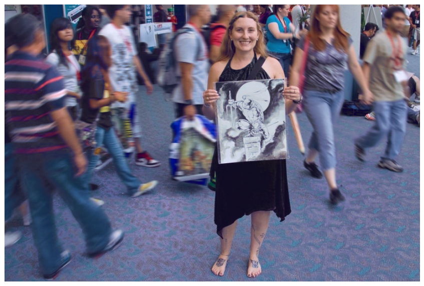

(g) blue tint

Figure 7. **Local color changes.** The user changes the red door in (a) to an orange (b), yellow (c), and green door (d) by moving the cluster representing the door. The user alters the original image (e) by applying a greenish tint to the background in (f) and a bluish tint to the background in (g). Then, the user changes the girl's skin color to make her stand out a little more from the background. *Original images courtesy of Flickr users Grant MacDonald and colorblindPICASO.*

[8] Kalra, D., "Gamopt: A tool for visualization and optimization of gamuts," in [*Proc. SPIE 2171*], 297–304 (1994).

[9] Farup, I. and Hardeberg, J., "Interactive color gamut mapping," in [*Int. Printing and Graphics Arts Conf.*], (2002).

[10] Willett, N., Levoy, M., and Adams, A., "Gamut mapping," (2010). Accessed Dec 2013. `http://graphics.stanford.edu/courses/cs178/applets/gamutmapping.html`.

[11] Bonneel, N., Sunkavalli, K., Paris, S., and Pfister, H., "Example-based video color grading," *ACM Trans. Graph.* **32**(4), 39 (2013).

[12] Pitié, F., Kokaram, A., and Dahyot, R., "Automated colour grading using colour distribution transfer," *J. Comput. Vision and Image Understanding* **107**(1-2), 123–137 (2007).

[13] Reinhard, E., Ashikhmin, M., Gooch, B., and Shirley, P., "Color transfer between images," *IEEE Comput. Graph. Appl.* **21**(5), 34–41 (2001).

[14] Abadpour, A. and Kasaei, S., "A fast and efficient fuzzy color transfer method," in [*Proc. IEEE Int. Symp. Signal Process. and Inform. Technology*], 491–494 (2004).

[15] Luan, Q., Wen, F., and Xu, Y.-Q., "Color transfer brush," in [*Proc. Pacific Conf. Comput. Graph. and Applicat.*], 465–468 (2007).

[16] Wen, C.-L., Hsieh, C.-H., Chen, B.-Y., and Ouhyoung, M., "Example-based multiple local color transfer by strokes," *Comput. Graph. Forum* **27**(7), 1765–1772 (2008).

[17] An, X. and Pellacini, F., "User-controllable color transfer," *Comput. Graph. Forum* **29**(2), 263–271 (2010).

[18] Li, Y., Ju, T., and Hu, S.-M., “Instant propagation of sparse edits on images and videos,” *Comput. Graph. Forum* **29**(7), 2049–2054 (2010).

[19] Oskam, T., Hornung, A., Sumner, R., and Gross, M., “Fast and stable color balancing for images and augmented reality,” in [*Proc. 2nd Joint 3DIM/3DPVT Conf. 3D Imaging, Modeling, Process., Visualization & Transmission*], 49–56 (2012).

[20] Xiao, Y., Wan, L., Leung, C.-S., Lai, Y.-K., and Wong, T.-T., “Example-based color transfer for gradient meshes,” *IEEE Trans. Multimedia* **15**(3), 549–560 (2013).