POINT-AND-CLICK REGION BASED METHOD FOR COLOR EDITING AND CONTROL FOR DIGITAL COLOR PRINTERS

Siyu Zhu[1], *Sohail A. Dianat*[2], *Lalit K. Mestha*[3]

[1,2], Department of Electrical Engineering, Rochester Institute of Technology, Rochester, NY 14623, USA; [3], Xerox Corporation, Webster, NY 14580, USA

ABSTRACT

To meet ever changing customer demand, the commercial printing industry requires the capability of producing colors accurately and consistently. In this presentation, we propose to use a region growing process with a spatial (i.e., 2-D) color gradient matrix in RGB or L*a*b* or CMYK space to pick a region/object of interest. A grid with Laplacian edge map is constructed for the whole image a priori using the spatial color gradient matrix. The Laplacian edge map (a template), if needed, is shown on the GUI to guide the user to point at his or her object of interest. After clicking the mouse, the algorithm will grow the region until it coincides with the Laplacian edge map. A color constrained cost function is introduced to prevent leakage through mild transitions between different objects. Once the object is selected, a customized rendering LUT is applied to improve the color quality of the object. We also present control-based techniques to create inverse transformation that produce more accurate CMYK values. Control based techniques minimize the rendering errors normally occur during inversion and gamut mapping process.

Index Terms— *Color control, Object image segmentation, Image rendering.*

1. INTRODUCTION

Object segmentation is a popular topic for image processing. To extract a meaningful object from color images, the prerequisite is to perform good edge detection. On the other hand, present edge detection method cannot automatically make ideal edge map for a random color image. So a fuzzy edge map is proposed in reference [1], to restrict the object, and a region growing process is performed on top of the fuzzy edge map to produce a mask covering a meaningful region. The algorithm proposed in this paper is simple and fast, and will select an object instantly after a point and click operation. Then the segmentation result will be sent to the rendering algorithm. The rendering Look-up tables are customized to enhance the color representation for each of the selected objects. A control based algorithm as introduced in reference [3] which is used here to achieve an optimized rendering result. This technique is proved via simulations using MathWorks MATLAB®. It has greater potential to improve the image quality when compared to traditional methods.

2. FUZZY EDGE MAP

Edge detection is done by computing the gradient of pixel values of a target image. A common way to compute a picture's gradient is to calculate its first derivative of its gray scale. But gray scale might fail in some cases, because different colors may result in same gray scale intensity. Hence, a Laplacian edge detection algorithm is introduced that will eliminate the possibility of information loss during edge detection. We compute the eigenvalues of Laplacian matrix corresponding to each pixel which is then used to detect the edges. The Laplacian matrix is defined for each color pixel at position (x, y) as:

$$\mathrm{L} = \mathrm{D}^{\mathrm{T}}\mathrm{D}, \text{where } \mathrm{D} = \begin{bmatrix} \frac{\partial \mathrm{R}}{\partial \mathrm{x}} & \frac{\partial \mathrm{R}}{\partial \mathrm{y}} \\ \frac{\partial \mathrm{G}}{\partial \mathrm{x}} & \frac{\partial \mathrm{G}}{\partial \mathrm{y}} \\ \frac{\partial \mathrm{B}}{\partial \mathrm{x}} & \frac{\partial \mathrm{B}}{\partial \mathrm{y}} \end{bmatrix} \tag{1}$$

Here, R, G and B represent the color values of the current pixel in each channel. $\frac{\partial \mathrm{R}}{\partial \mathrm{x}}$, is the gradient of R channel in x direction. The color gradient at each pixel is defined as:

$$\lambda = \frac{\mathrm{a}+\mathrm{b}+\sqrt{(\mathrm{a}-\mathrm{b})^2+4\mathrm{c}^2}}{2} \tag{2}$$

where,

$$\mathrm{a} = \left(\frac{\partial \mathrm{R}}{\partial \mathrm{x}}\right)^2 + \left(\frac{\partial \mathrm{G}}{\partial \mathrm{x}}\right)^2 + \left(\frac{\partial \mathrm{B}}{\partial \mathrm{x}}\right)^2,$$
$$\mathrm{b} = \left(\frac{\partial \mathrm{R}}{\partial \mathrm{y}}\right)^2 + \left(\frac{\partial \mathrm{G}}{\partial \mathrm{y}}\right)^2 + \left(\frac{\partial \mathrm{B}}{\partial \mathrm{y}}\right)^2,$$
$$\mathrm{c} = \left(\frac{\partial \mathrm{R}}{\partial \mathrm{x}}\right)\left(\frac{\partial \mathrm{R}}{\partial \mathrm{y}}\right) + \left(\frac{\partial \mathrm{G}}{\partial \mathrm{x}}\right)\left(\frac{\partial \mathrm{G}}{\partial \mathrm{y}}\right) + \left(\frac{\partial \mathrm{B}}{\partial \mathrm{x}}\right)\left(\frac{\partial \mathrm{B}}{\partial \mathrm{y}}\right).$$

Here gradient λ equals to the largest eigenvalues of the Laplacian matrix. For digital images, whose pixel values are discrete, the first derivative is replaced by difference equation. For instance, $\frac{\partial \mathrm{R}}{\partial \mathrm{x}}$ is replaced by $R(x+1)$-$R(x-1)$. The edge map is the Laplacian gradient itself without the threshold. As a result, the edge detection process will produce a map with real numbers which is fuzzy instead of binary indices. The values of the edge map indicate to what extent a pixel belongs to an edge.

3. REGION GROWING

A seed is selected manually using point and click method. Once the pointing device has picked a seed pixel, it is marked with 1 while all other pixels are marked with 0. The binary map indicating the seed pixel is called Object Mask (OM). A window scanning process is then applied. Window can be of size 3×3 or greater (figure 1). The larger the window, the faster is the growing process, but less accurate the results are. Window will scan in a raster sequence centering at every pixel. For each pixel, two terms are computed inside the current window. Term A is called the growing term, which is the normalized mean value of OM. Term B is called the convergence term, which is the normalized gradient value of current pixel. A cost function is then computed as:

$$f(x,y) = \omega_a(x,y)A(x,y) - \omega_b(x,y)B \qquad (3)$$

Where $A(x,y)$ is the growing term, $B(x,y)$ is the convergence term. While the cost function is smaller than zero, the OM at current pixel is set to 0, which means stop growing; otherwise, the OM at current pixel is set to 1, which means grow the target object region to the current pixel. $\omega_a(x,y)$ and $\omega_b(x,y)$ are adjustable weight scalars used to control the final growing result. After the window scanned the entire image, the OM is refreshed to let the object grow to new regions. Then the algorithm loop for object growth will continue until convergence. Convergence is defined as the end of the growing process which occurs when the difference of Object Mask at current loop and last loop are less than a threshold, say 10 (which means 10 (or less than 10) pixel indices are changed in the last loop). The flow chart of region growing loops is shown in figure 2.

While mild transitions of colors occur between objects, the fuzzy edge will not be strong enough to stop leakage between objects. Color constraints will add another boundary to prevent leakage, and hold the growing process inside a meaningful object. For every pixel, the color distance between current pixel color and a reference color are computed. The reference color can be the color of the seed or the average color of the current object. The distance is presented by the Euclidian distance between two colors (or ΔE^*2000 [2] between the two for a more accurate presentation of visual color difference). The color constraint is computed for every pixel, and recorded as a constant for each loop in the growing process. This constraint will be added to the convergence term B(x, y) in equation (3).

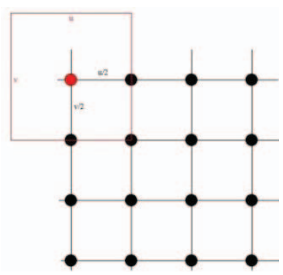

Figure 1 The window scanning sequence for growing algorithm.
[The window size is 3×3 and centered at the first pixel in the image. While scanning, the window will center at all pixels in a raster order and mean values of Object Mask are computed in each window.]

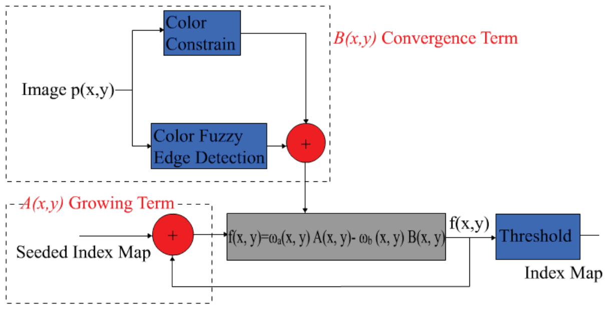

Figure 2 The flowchart of region growing algorithm.
[The color fuzzy edge detection and seed is first computed and restored. Then the growing procedure start as window is scanned in raster sequence. The growing algorithm repeats until the object comes to a convergence area.]

4. CONTROL BASED INVERSION

Control based inversion method which is introduced in reference [3] is implemented here to render an image based on the segmentation result. Here the purpose of extracting a meaningful object is to render it properly for printing device to enhance the quality. A printing device commonly uses CMYK tones to recreate colors from the Profile Connection Space. So a printer model could usually be presented as a CMYK to CIE L*a*b* Look-Up Table. The structure of the table is shown in figure 3. So the inversion transformation is from CIE L*a*b* to CMYK. The printer model is assumed available for a particular device, and the inverse transformation is to be generated. We use control based approach to complete a typical rendering structure as in Figure 4.

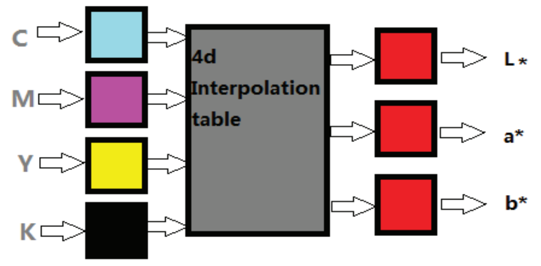

Figure 3 Printing device to L*a*b* transformation or say printer model

For those colors outside the CMYK gamut, we used control based ray tracing method with centroid gamut mapping to find the appropriate mapping positions. Here we describe the algorithm briefly.

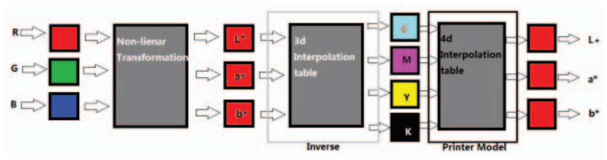

Figure 4 Color space rendering introduced in this paper.
[The RGB image is first converted to CIE L*a*b* space and then transformed into CMYK using inverse LUT. The forward transformation can be performed directly on the printer model.]

Any display or printer has its own color gamut which generally does not cover the whole CIE L*a*b* color space. Larger the gamut is, richer the color the device can generate. For those colors outside the gamut of a printer, we have to map the color into its gamut to be able to produce it. For example, a saturated blue color on display will probably fail while printing because normal CMYK ink printer can't generate such a saturated color in blue area.

The forward LUT which is the characterized version of the printer is used to find the corresponding inverse transformation with gamut mapping. In this approach, a feedback control algorithm is used at each node to accurately convert the in-gamut L*a*b* nodes to CMYK color space. A block diagram shown in figure 5 illustrates how inversion operations are executed at a given L*a*b* node [3].

For an individual node color, the system with the integrator in figure 5 can be expressed in state space form as

$$x(k+1) = Ax(k) + Bu(k) \tag{4}$$

where x(k) represents the L*a*b* values from the printer model obtained at iteration k. A is an identity matrix; B is the Jacobian matrix computed around the initial CMYK value. u(k) is the control law applied to the input of the printer. The Jacobian matrix B is different for each node color and is computed as follows:

$$B = \begin{bmatrix} \frac{\partial L*}{\partial C} & \frac{\partial L*}{\partial M} & \frac{\partial L*}{\partial Y} & \frac{\partial L*}{\partial K} \\ \frac{\partial a*}{\partial C} & \frac{\partial a*}{\partial M} & \frac{\partial a*}{\partial Y} & \frac{\partial a*}{\partial K} \\ \frac{\partial b*}{\partial C} & \frac{\partial b*}{\partial M} & \frac{\partial b*}{\partial Y} & \frac{\partial b*}{\partial K} \end{bmatrix} \tag{5}$$

The control law is designed using MIMO state-feedback controllers. Thus, $u(k) = -Ke(k)$, where $e(k)$ is the error between the target L*a*b* and the model L*a*b* at iteration step k. In pole placement design, the gain matrix K is derived based on the pole values specified such that the closed-loop system model shown in Figure 5 is stable. This is achieved by assigning pole values within the range [0,1) in the z-domain.

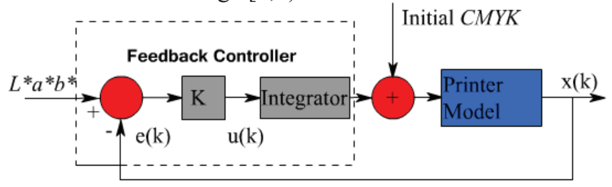

Figure 5: Closed-Loop control algorithm with a gain matrix and the integrator as controller for a 4-to-3 control-based inversion.

All gamut-mapping methods need to determine accurately whether the node color is inside or outside the destination gamut. A color that is wrongly considered as outside the gamut may be handled incorrectly by the gamut-mapping algorithm, which can lead to unacceptable artifacts and color errors when images are rendered through those multidimensional LUTs.

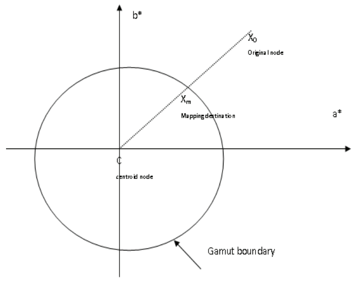

Figure 6 The section between start and end points.
[The control based inversion is run on each node between X_0 and C, once the control loop achieved convergence, or say reached the point of X_m, then we use color values X_m to find the corresponding CMYK color.]

For a node color that is outside the gamut, we create a line (i.e., ray) which connects the current node with the end point (figure 6). For a centroid gamut mapping method, the end point is the gamut centroid which is a node at the center of the gamut. We divide the line into hundreds of nodes. The control loop as described above is applied to each node on the line (from outside node towards the centroid along the line for a centroid gamut mapping algorithm). It is to be noted that for other ray-based gamut mapping algorithms, the end point can be away from the centroid. Once the control loop achieved convergence with error near the threshold first time, the out-of-gamut node will result in mapping along the ray. The process also results in CMYK values for mapped color.

The control-based inversion gives much better result than other known inversion methods. It improves inversion accuracy, because at the in-gamut nodes it results in zero inversion error unlike other known methods (e.g., unstructured interpolation). Also, the mapping of out-of-gamut colors to the gamut boundary can be accurate especially when the gamut boundaries are curved. For mapping out-of-gamut colors, this algorithm can take large computing time. To optimize the computational aspects, a binary ray tracing method is applied [4]. Instead of dividing the ray into hundreds of nodes, the binary ray tracing algorithm will divide the ray into two equal sections, and the midpoint is examined with control based inversion to see whether the point is inside or outside the gamut. If the point is outside the gamut, then the inner section will be divided into two smaller sections and the new midpoint will be examined again. Otherwise, the outer section will be divided and the corresponding midpoint. The procedure will continue until we determine the mapping point.

5. SIMULATION RESULTS

Face color usually preferred to be rendered with CMY tones only except K tone. It is because the elimination of tone will produce a visually better color for human flesh tones, and make it smoother. Implementing the proposed object segmentation could easily extract human face from back ground and control the rendering selectively using different LUTs for each object. The original images shown here include human faces, see figure 7.

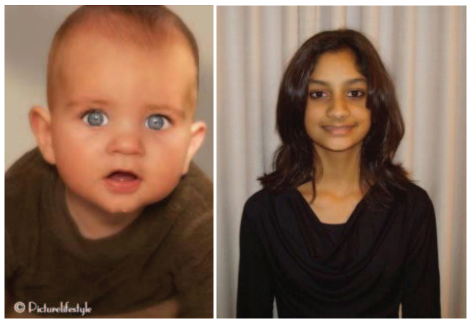

Figure 7 Original image with human face flesh color.

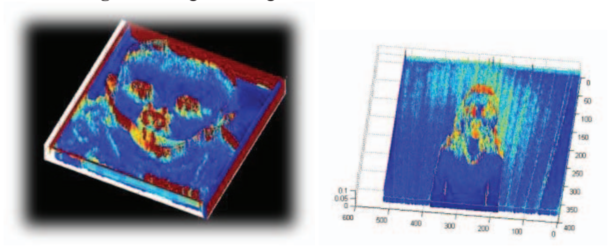

Figure 8 Mesh plot of fuzzy edge map produced using Laplacian fuzzy edge detection method.
[The heights of the mesh columns indicate the value of the gradient or say the largest eigenvalue of each pixel's Laplacian matrix.]

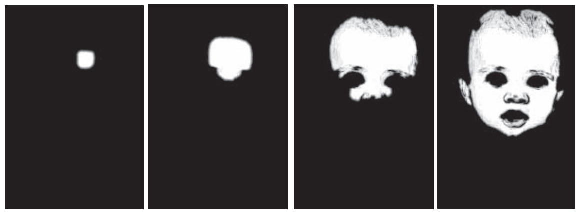

Figure 9 The Object Mask grows during the region growing procedure. [The luminance of the pixel indicates the value of the cost function as in equation 3. The value of the cost function describe the trend to continue growing for pixels. The very left image indicates the object while the growing procedure is terminated because very pixel comes to convergence point.]

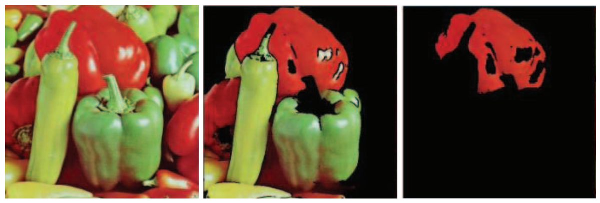

Figure 10 The original image (left); segmentation result without color constraint (middle); segmentation result with color constrain (right).

Fuzzy edge map is created first for each input image (figure 8). The height of the mesh plot indicates the value of the color gradient. We can see the peaks of fuzzy edges surrounding the face area in each image. The region growing method based on the fuzzy edge map is like climbing the hill of edges gradually.

Using the point-and-click segmentation method, the seed is selected, region growing method will run and the Object Mask grows continuously, as shown in figure 9. After each loop the object climbs to the area with higher fuzzy edge values, but the distance between the pixel and the seed also influences the growing result. Steeper the edge is, slower the object growth at the pixel.

In some cases the mild transition between objects are likely to cause leakage while growing the region. Hence, color constraints are introduced to eliminate the possibility of such errors. For example, as in figure 10, the color constraint restricts the segmentation to one meaningful object instead of leaking into other regions.

After the object is detected, customized Look-up tables are used to interpret the colors for printing device. An example of the printer output is given in Figure 11 with comparison to original image.

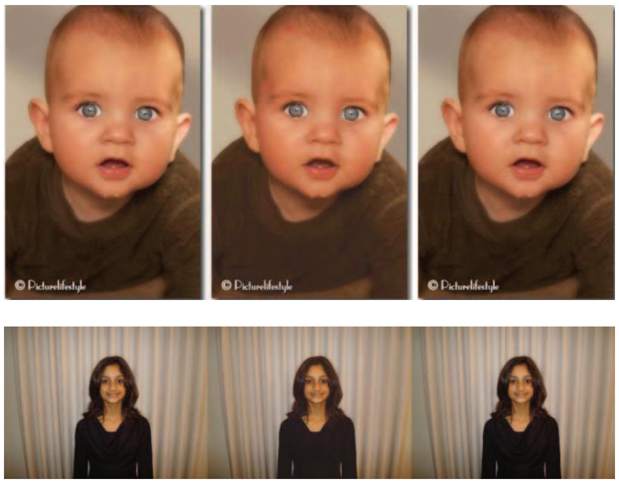

Figure 11 The original image (left), the rendered image with global LUT for every pixel (center), the rendered image with customized LUT (right).

6. CONCLUSION

The proposed object segmentation is simple and quick. After a point and click operation, it is used to detect the target object automatically. Color constraints eliminate the leakage through mild transitions between objects. Tuning weight factors of cost function (3) can prevent leakage to neighboring undesirable regions. After the object is selected, it is then rendered using inverse Look-Up Table with customized rendering. The inverse is performed using a control based inversion method to achieve the minimum error. A binary ray tracing algorithm is used to optimize the computational time while mapping out-of-gamut colors. The rendered image with customized LUTs for target objects gives improved quality without contours than when compared to a global rendering LUT.

7. REFERENCE

[1] Chung-Chia Kang, Wen-June Wang, "Fuzzy Based Region Growing for Image Segmentation", The 28th North American Fuzzy Information Processing Society Annual Conference (NAFIPS2009) Cincinnati, Ohio, USA - June 14 - 17, 2009.

[2] Sharma, Gaurav, Wencheng Wu, Edul N. Dalal (2005). "The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations". Color Research & Applications (Wiley Interscience) 30 (1): 21–30. doi:10.1002/col.20070.

[3] L.K. Mestha and S.A. Dianat, "Control of Color Imaging Systems", CRC Press, Taylor and Francis Group, ISBN 978-0-8493-3746-8, p303-429, May 2009.

[4] Carole Bakhos, "Gamut mapping with ray-based control model", Graduate thesis paper, Department of Electrical Engineering, Rochester Institute of Technology, 2009.