GENERATING NON-PHOTOREALISTIC ILLUSTRATIONS FROM IMAGES

Hana Ali Rashid (05940)

Habib University
hr05940@st.habib.edu.pk

ABSTRACT

There has been work done in Non-Photorealistic Rendering (NPR) using image processing techniques (such as mosaicking, glass painting, caricatures and more) but given the variance in types of images, the methods used for each application vary. This paper discusses a novel application of existing image processing techniques: generating illustrated versions of objects from images to be used as graphic elements in digital design. Since MATLAB has a rich variety of algorithms available in its Image Processing Toolbox in the form of functions, the scope of this project is to utilize these algorithms to achieve an illustrated effect in methods that are similar to those used for other NPR purposes that were explored in the literature review. This would give us an insight into the effectiveness of existing algorithms for this application and set the stage for further work.

Index Terms— Non-Photorealistic Rendering (NPR), image abstraction and stylization, filtering, graphic design

1. INTRODUCTION

Graphic design is prevalent in all fields for various purposes, whether it be to create a poster for recruiting employees or compiling a book for children. However, most design work relies on the assistance of graphic elements - including but not limited to photographs and drawings - to increase the effectiveness of the design via visual aids and/or make them more visually appealing (Figure 1). Of these, digital illustrations are used widely for their ability to be manipulated (in terms of color, size etc.) as well as for the flexibility of being able to pick and use particular objects from the illustration, in order to make more complex works by combining them. However, such illustrations require equipment (drawing tablet, scanners) and skill to make, no matter how simple they appear. An alternative to this is also to buy illustrations made by other artists in order to use them for commercial use. However, for individuals looking for another option, it may be possible to ”generate” illustrated objects from images using non-photorealistic rendering (NPR). An effective implementation of this technique will aim to make it simpler for designers to acquire the illustrated elements they require.

The characteristic feature of this project is that it aims to generate illustrations of objects to be used as graphic elements. Therefore, instead of applying NPR rendering to the entire image as is commonly seen, the object of interest will first be extracted from the image before it is processed further for an illustrated effect. Moreover, in this project an ”illustrated” or ”cartoon” effect is one where an object is made up of fewer fine details, fewer colors - and shades of colors - than the original image, and has its borders enhanced. This project focuses primarily on RGB images.

2. LITERATURE REVIEW: IMAGE PROCESSING FOR NPR ILLUSTRATED EFFECT

There has been extensive work done on segmenting the foreground of an image, using thresholding methods, graphs, distance-based methods and more. Since segmentation does not have much to do with NPR and is more of a pre-processing step, the literature review was focused more on processing the image further for an illustrated effect.

According to [1], NPR techniques can be classified into one of four categories: stroke-based rendering, image analogy, region-based techniques, and image processing and filtering techniques. Given the nature of this project, this review will be focusing on image processing and filtering techniques. The primary steps that need to be taken to give an image an illustrated effect are to detect its edges and enhance them, as well as to simplify the colors and features within it.

2.1. Edge Detection and Construction

In order for an image to appear illustrated, it’s edges need to be enhanced. However, only the principal edges are required by an illustration, and popular edge detection algorithms like Canny edge detection detect more edges than required due to noise and fine texture, and such ”over-connectedness” may reduce recognizability of the object [2]. One technique used by [3] used a bilateral filter on the image in order to smoothen it and remove fine detail while preserving edge information. They also improved the contrast of the image and then used

978-1-4799-1180-6/13/$31.00 ©2013 IEEE

this "abstracted" form of the image to extract edges using an FDoG filter. The process is summarized in Figure 2.

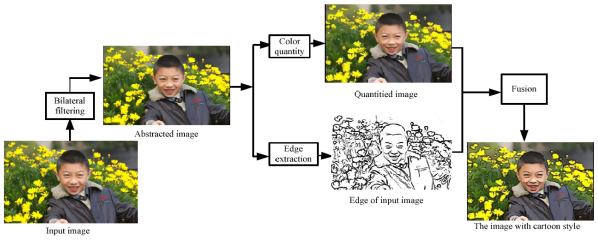


Fig. 2. Algorithm to give an image a cartoon effect [3]

A Flow-based Difference-of-Gaussian (FDoG) method has been widely considered a better alternative to Canny edge detection for line drawing in NPR rendering. This is a modification of the Difference-of-Gaussian (DoG) method used by [2] which was based on the Marr-Hildreth edge detector. A coherent line drawing algorithm was developed based on FDoG for NPR rendering [4]. This used the concept of *Edge Tangent Flow* (ETS) which describes the edge tangent direction of principal edges in an image while disregarding weaker edges. Thus, pairing ETS with DoG allows for line construction that overcomes the limitation associated with DoG such as isolated points around weak edges or regions that look more like points than lines due to lack of clear direction [4].

Coherent line drawing has been widely, for instance to generate cartoon-style bas-releifs from photographs [5], due to the well-connected edges it produces without including weak edges or fine texture, as well as the variance in edge thickness which gives a hand-drawn look - ideal for an illustrated effect.

2.2. Color

The second part of giving an image an illustrated effect is to simplify the colors used in it by forming "blocks" of uniform color to define regions within the image. [6] and [3] both use bilateral filtering in order homogenize color regions while also preserving edge information in an image, as opposed to simply using a Gaussian filter to smoothen the image as a whole. Additionally, in order to simplify the color palette of the image, color quantization was done [3, 6]. For an RGB image, [6] used the following formula to requantize the image where p is the pixel value and a is the factor by which the number of colors is to be reduced:

$$p_{new} = \lfloor \frac{p}{a} \rfloor * a$$

Determining the scale factor that produces the most reliable results depends on the application and must be tested.

2.3. Recombination

Once edges are effectively extracted and constructed, and color quantization is complete, the final step taken to generate an illustrated effect was to overlay the edges on the requantized image [3, 6].

This review looked at various techniques that may be combined to extract the object of interest from an image and give it a non-photorealistic illustrated effect. As is seen from the number of papers on the topic, much work is done in order to perform NPR rendering of images for various applications, and the method used by each - such as edge detection algorithm or image segmentation - differs depending on the application. This review, however, only highlighted techniques that appeared to cater best to the aim of this project.

3. METHODOLOGY

The objective of this project was to explore how currently available MATLAB algorithms can be used to give images illustrated effects. The process of illustrating an object from an image can be broken down into three primary steps:

1. Segmentation for object (foreground) extraction
2. Color quantization
3. Edge detection for enhancing object boundaries

As informed by the literature review, the work done in edge detection and reconstruction for NPR is extensive and it is unlikely to do it justice within the scope of this project. Therefore, this project focuses on existing algorithms for implementing the first two steps i.e. segmentation and quantization, for an illustrated effect. Of the available algorithms, the ones that performed best for the desired application are mentioned below.

3.1. Pre-Processing

The conditions of image acquisition vary with location, ambient light, illumination and more. Thus, in order to ensure no unnecessary noise or textural detail interferes with further processing, a bilateral filter was used to remove noise and smoothen images using Gaussian kernels [7] while preserving edge information. The MATLAB function `imbilatfilt()` was used for this purpose [8].

3.2. Object Extraction

In order to segment RGB images, there are two main methods that can be followed. The images can either be converted to grayscale and segmented (using algorithms that only take grayscale images as input), and their results then used as binary masks to segment the original images. Or, algorithms that cater directly to RGB images can be used to segment

them. The latter approach was favored in this project as it reduced the need for additional processing in order to effectively segment the images using masks.

Segmenting RGB images is not a simple task and is still being worked on. A popular technique to segment such images is based on **K-Means Clustering** which iteratively determines the minimum average squared distance for points in the same cluster combined with a randomized seeding technique [9]. This algorithm is known for its speed and accuracy and was therefore chosen as one option for extracting the object(s) of interest from images. The `imsegkmeans()` [10] MATLAB function was used to implement this technique. The advantage of this technique is that it does not require any user interaction and performs the segmentation automatically. Therefore, it also gives reproducible results despite the number of times it is run [10].

Another segmentation technique explored for this application was the **Lazy Snapping** technique developed by [11] which is an interactive technique based on K-Means clustering. The MATLAB implementation of this, `lazysnapping()` [12], provides an interactive method to allow foreground and background pixels be identified which are then used by the algorithm to segment images. This method was also chosen for this project for the flexibility it gives users as well as better segmentation of images with varying backgrounds that may be mistaken for the foreground (due to illumination or noise).

The third major technique that was tested for this application was that of MATLAB's `imseggeodesic()` [13] which is segmentation based on **adaptive weighted distances** as put forth by [14]. This is also an interactive technique that requires the user to identify regions of interest and these regions are used as starting points from where other points in the image are added to segments based on their distances from the selected regions and thus the probability that they are part of a particular segment [14, 13]. This method also gives users more flexibility in the variety of images it can effectively segment without requiring heavy computation.

3.3. Color Quantization

Once the desired object has been effectively extracted from the image, it can be given an illustrated look by simplifying its color palette i.e. reducing the number of colors represented within the object. This can be achieved using the `rgb2ind(img,Q)` function of MATLAB which reduces the number of colors within an rgb image to Q number of quantized colors [15]. This number can be set based on the complexity of the image and the desired effect.

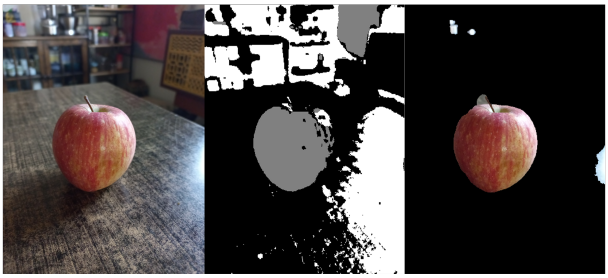
`rgb2ind()` quantizes the image as a whole but another method of quantizing colors is to use thresholds to quantize each plane in the RGB image. The `multithresh(img,N)` function helps achieve this by using Otsu's global thresholding method to generate N threshold values for the given image. The resultant threshold values are then passed as input to the `imquantize()` function which quantizes the image for each of its $N+1$ planes. Both of these quantization methods perform well (or not) depending on the input image [16, 17].

4. RESULTS AND DISCUSSION

In order to test the effectiveness of these algorithms, a sample of 24 images were taken using a smartphone. These images are all of common objects that may be used as graphic elements in graphic design (paint tubes, book(s), paintbrush(es)) and were taken using a smartphone as that is the most accessible means of acquiring images for most people.

4.1. Object Extraction Results

Each of the segmentation algorithms mentioned above performed differently depending on the complexity of the image. The K-Means segmentation method proved effective for images that did not have other objects in the background, but included background objects for images with busier backgrounds (Figure 3). Moreover, in order to get the desired object, the user has to identify desired clusters and add them manually, which removes the advantage of automation in this method. However, for images where the object is segmented as desired, the result is more precise than most other methods (Figure 4) which is a major advantage.

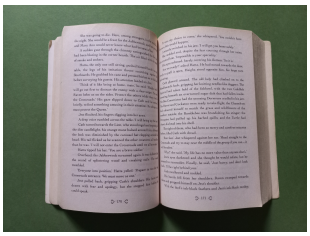


(a) Input Image (b) K-Means (c) Lazy Snapping

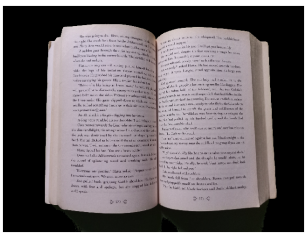
Fig. 3. Comparison of segmentation results for an image with a busy background.

The segmentation technique based on Lazy Snapping allowed users to determine regions required and therefore was able to identify the object(s) of interest. It worked better than K-Means for images with busy backgrounds as seen in Figure 3, although the result was not always precise.

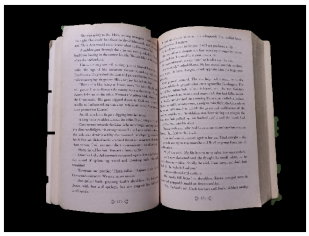
However, the geodesic segmentation technique performed least effectively. As it followed a distance-based algorithm from the regions of interest (ROIs), it also gave imprecise boundaires and the results were not useful for this aplication. The results of the three algorithms can be easily seen in Figure 4.



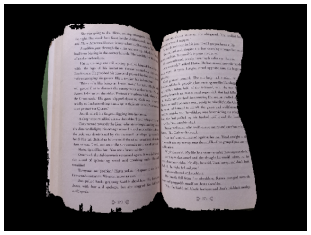
(a) Input Image



(b) K-Means



(c) Lazy Snapping



(d) Geodesic

Fig. 4. Comparison of segmentation results using the three techniques.

4.2. Color Quantization Results

Both quantization methods worked effectively and gave similar results in simplifying colors in the objects and giving them an illustrated effect. The book in Figure 4 can be seen to have an illustrated effect after quantization in Figure 5. Moreover, Figure 6 shows more examples of input images and their illustrated versions.

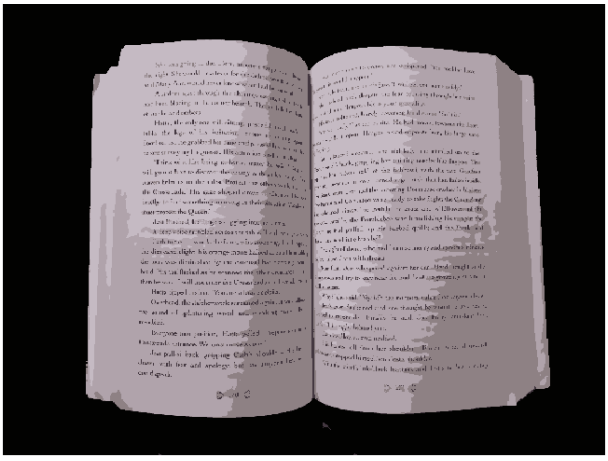


Fig. 5. Book image after quantization.

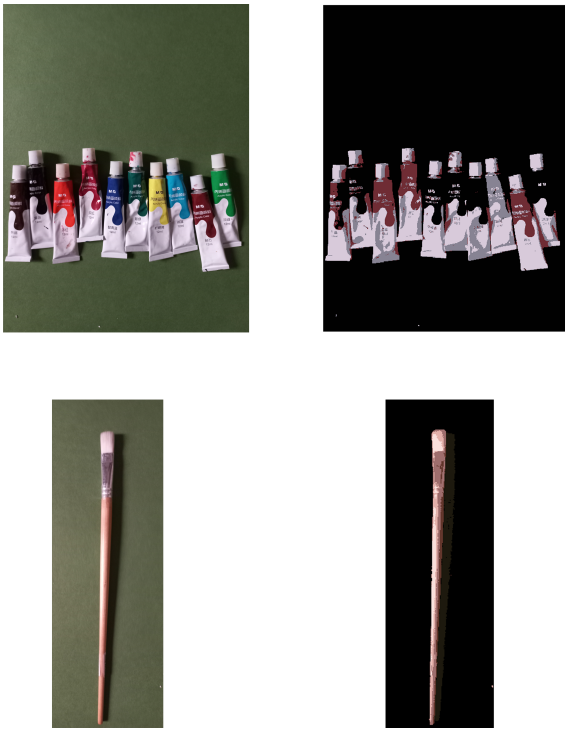


Fig. 6. Left column: input images. Right column: segmented and color quantized.

4.3. Limitations and Future Directions

A limitation of this implementation is that images must have a solid colored background with no other objects or colorful backgrounds in order for accurate segmentation to take place. Moreover, quantization of colors does give an illustrated effect but also eliminates detail which may in some cases include major parts of the object in question.

Further work on this application can attempt to determine if a pattern exists between the nature of an image (illumination, background) and the most effective segmentation technique for it. That may allow us to group together input images and use the segmentation technique best suited to it. Moreover, the ”illustrated” images produced in this implementation have sharp edges and real-world forms. In order to give a more illustrated look, perhaps methods of softening sharp edges and distorting proportions can be implemented as well. In addition, an effective coherent line drawing algorithm can also be implemented in order to enhance only the principle boundaries of illustrated objects for a more effective illustrated effect.

5. CONCLUSION

This paper aimed to explore existing MATLAB algorithms/functions that may be used to extract objects from images and give them an illustrated effect for use in graphic design by smoothing

(bilateral filter), extracting the object (segmentation), and color quantization. As can be seen in Figure 7, existing algorithms are capable of producing such illustrations. Further work on this application can look into various other illustrated effects and art styles that can be implemented, as well as implementing an edge detection and linking method to enhance the edges of the illustrations in a visually appealling manner.

Fig. 7. Posters made using illustrations generated in this project from smartphone-captured images.

6. REFERENCES

[1] M. P. Pavan Kumar, B. Poornima, H. S. Nagendraswamy, and C. Manjunath, "A comprehensive survey on non-photorealistic rendering and benchmark developments for image abstraction and stylization," *Iran Journal of Computer Science*, vol. 2, no. 3, pp. 131–165, Sept. 2019.

[2] Bruce Gooch, Erik Reinhard, and Amy Gooch, "Human facial illustrations: Creation and psychophysical evaluation," *ACM Transactions on Graphics*, vol. 23, no. 1, pp. 27–44, Jan. 2004.

[3] Li-wen Lu, Yuan-yuan Pu, Heng Zhang, and Dan Xu, "A Non-photorealistic Rendering Algorithm For Cartoons," p. 6, 2013.

[4] Henry Kang, Seungyong Lee, and Charles K. Chui, "Coherent line drawing," in *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering - NPAR '07*, San Diego, California, 2007, p. 43, ACM Press.

[5] Seungchan Lee and Bong-Soo Sohn, "Generation of cartoon-style bas-reliefs from photographs," *Multimedia Tools and Applications*, vol. 78, no. 20, pp. 28391–28407, Oct. 2019.

[6] Kevin Dade, "Toonify: Cartoon Photo Effect Application," p. 3.

[7] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, Bombay, India, 1998, pp. 839–846, Narosa Publishing House.

[8] "Bilateral filtering of images with Gaussian kernels - MATLAB imbilatfilt," .

[9] David Arthur and Sergei Vassilvitskii, "k-means++: The Advantages of Careful Seeding," p. 9.

[10] "K-means clustering based image segmentation - MATLAB imsegkmeans," .

[11] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum, "Lazy Snapping," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 303–308, Aug. 2004, Place: New York, NY, USA Publisher: Association for Computing Machinery.

[12] "Segment image into foreground and background using graph-based segmentation - MATLAB lazysnapping," .

[13] "Segment image into two or three regions using geodesic distance-based color segmentation - MATLAB imseggeodesic," .

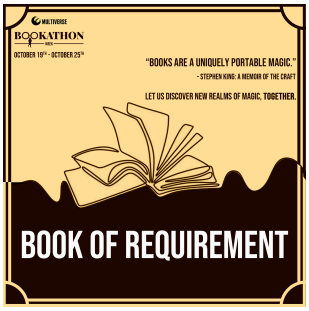
[14] A. Protiere and G. Sapiro, "Interactive Image Segmentation via Adaptive Weighted Distances," *Trans. Img. Proc.*, vol. 16, no. 4, pp. 1046–1057, Apr. 2007, Publisher: IEEE Press.

[15] "Convert RGB image to indexed image - MATLAB rgb2ind," .

[16] "Multilevel image thresholds using Otsu's method - MATLAB multithresh," .

[17] "Quantize image using specified quantization levels and output values - MATLAB imquantize," .

(a) Poster - summer program



(b) Poster - literary event

(c) Badge - blood drive

Fig. 1. Examples of illustrated graphic elements as used in graphic design. (a) uses plants as graphic elements for decorative purposes. (b) uses an illustrated book as a graphic element for a book-related event. (c) uses a drop as a graphic element in a badge for blood drive donors.