Hacettepe University

Computer Engineering Department

BBM-415 Fundamentals Of Image Processing - 2023 Fall

Giving Cartoon Effect to Colorful Images

November 19, 2023

Student name:
Esad BORAN

Student Number:
21827206

Contents

1 Introduction

Image filtering is a fundamental aspect of Image Processing, addressing tasks like smoothing and edge detection. Smoothing, achieved through filters like the Gaussian kernel, aims to enhance essential features by reducing high-frequency components. This assignment simplifies the method to create captivating cartoon-like effects on colorful images. The objective is to implement basic image filters and showcase the anticipated results without delving into complex processes.

2 Experiment Details

In this assignment, our objective is to create a captivating cartoon effect from a given color image. The process involves several key steps, each contributing to the transformation:

2.1 Image Smoothing

To kick off the cartoonization process, we begin with image smoothing. This is achieved through the convolution of the input image using either a Gaussian filter or a median filter. The magic lies in experimenting with filter parameters like the sigma value for the Gaussian filter or the kernel width for the median filter. In Python, we can leverage functions such as `scipy.ndimage.gaussian_filter`, `scipy.ndimage.convolve`, and `scipy.ndimage.median_filter` for this.

2.2 Edge Detection

Moving on, our aim is to introduce a pencil sketching effect through edge detection. We employ a thresholded Difference of Gaussian (DoG) filter for this purpose. The DoG kernel is derived from the difference of two Gaussian functions (Equation 1), which is then convoluted with the original image (Equation 2). The filtered image is subsequently thresholded using a small number ε (Equation 3).

$$D_{\sigma,k} = G_\sigma(x) - G_{k\sigma}(x) \quad \text{for } k > 1 \quad (1)$$

$$I_{\text{filtered}} = D_{\sigma,k} * I \quad (2)$$

$$T_\varepsilon(n) = \begin{cases} 1, & \text{if } I_{\text{filtered}} \geq \varepsilon \\ 0, & \text{else} \end{cases} \quad (3)$$

Moving on, our aim is to introduce a pencil sketching effect through edge detection. We experiment with two different types of Difference of Gaussian (DoG) filters:

Listing 1: Python code for DoG filters

```
# Type 1: Threshold Gaussian DoG
dog_type1 = (ImageProcessor.apply_gaussian_filter(img, sigma=sigma1) -
             ImageProcessor.apply_gaussian_filter(img, sigma=sigma2))

# Type 2: Threshold DoG Convolve
dog_type2 = (ImageProcessor.gaussian_kernel(kernel_size, sigma=sigma1) -
             ImageProcessor.gaussian_kernel(kernel_size, sigma=sigma2))
```

Please note that for both types, it is crucial that $\sigma_2 > \sigma_1$ to ensure proper application of the Difference of Gaussian (DoG) filters.

2.3 Image Quantization

Quantization is a crucial step in simplifying color values to achieve a cartoon-like effect. In this process, we explore the utilization of three different algorithms for color quantization, each contributing to the unique visual appearance of the final output.

First Algorithm: K-Means Clustering - The K-Means clustering algorithm is employed to cluster color values into representative centroids. The quantized image is then reconstructed using these centroids.

Listing 2: K-Means clustering for color quantization

```
kmeans = KMeans(n_clusters=num_colors, n_init=10)
kmeans.fit(pixels)
quantized_pixels_kmeans = kmeans.cluster_centers_[kmeans.labels_]
quantized_image_kmeans = quantized_pixels_kmeans.reshape(height, width, 3)
quantized_image_kmeans = cv2.cvtColor(quantized_image_kmeans.astype(np.uint8), cv2
```

Second Algorithm: Lab Color Space Transformation - We transform the color space of the smoothed image to Lab, introducing adjustments to the (L)uminance channel. The final image in the Lab color space is obtained through inverse color space transformations.

Listing 3: Color space transformation: Lab

```
lab = cv2.cvtColor(smooth_img, cv2.COLOR_RGB2Lab)
lab[:, :, 0] = lab[:, :, 0] * float(lab_num)  # (L)uminance Change
quantized_image_lab = cv2.cvtColor(lab, cv2.COLOR_Lab2RGB)
```

Third Algorithm: HSV Color Space Transformation - We apply the same process to transform the image to the HSV color space, this time introducing adjustments to the (V)alue channel. Finally, the quantized images from both Lab and HSV color spaces are generated through inverse color space transformations.

Listing 4: Color space transformation: HSV

```
hsv = cv2.cvtColor(smooth_img, cv2.COLOR_RGB2HSV)
hsv[:, :, 2] = hsv[:, :, 2] * float(hsv_num)  # (V)alue Change
quantized_image_hsv = cv2.cvtColor(hsv, cv2.COLOR_HSV2RGB)
```

2.4 Combining Edge and Quantized Image

In this step, we fuse the obtained quantized image with the edge image to create the final cartoon image. The quantized image, representing simplified color values, is multiplied with the edge image. This multiplication combines the outlined edges with the quantized colors, resulting in a visually compelling cartoon effect.

3 Parameter Exploration and Analysis

In this section, we embark on a comprehensive exploration of various parameters, aiming to understand their individual contributions to the cartoonization process.

3.1 Smoothing Algorithms Overview

This subsection explores three image smoothing algorithms: **Gaussian Filter**, **Convolution**, and **Median Filter**, leveraging SciPy's capabilities (`scipy.ndimage`). Gaussian filter blurs images using a sigma-controlled kernel, convolution alters pixel values with a customizable kernel, and median filter excels in noise reduction. This concise overview aims to emphasize the distinctive features and applications of these algorithms.

Figure 1: Convolution

Figure 2: Gaussian

Figure 3: Median

3.2 Impact of Increased Sigma in Gaussian Filtering

Increasing the sigma value in the Gaussian filter results in a greater smoothing effect on the image. This process leads to a reduction in high-frequency details, making the image smoother. However, excessive increase in sigma may result in significant loss of important details and a decrease in image quality. The choice of the sigma value should be balanced based on the desired effect on the image and the goals of the application. Lower sigma values can enhance details in the image, while higher sigma values can be employed for a more artistic effect or a smoother image. Therefore, the selection of the sigma parameter in the Gaussian filter is a crucial design decision in image processing applications.

Figure 4: $(\sigma = 1)$

Figure 5: $(\sigma = 2)$

Figure 6: $(\sigma = 4)$

Figure 7: $(\sigma = 8)$

3.3 Different Edge Detection Algorithms

The `threshold_dog_gaussian` algorithm involves convolving the image with Gaussian filters of different standard deviations (σ_1 and σ_2) and computing the difference between them. This process highlights edges in the image, and the resulting edges are then thresholded to emphasize prominent features.The `threshold_dog_convolve` algorithm follows a similar approach but employs the convolution operation instead of Gaussian filters. The convolution with the DoG kernel enhances edges, and the thresholding step retains only significant edge information.

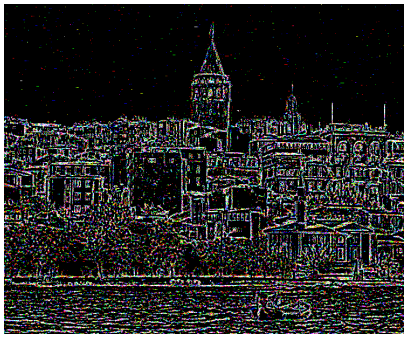

Figure 8: `DoG_Gaussian`

Figure 9: `DoG_convolve`

3.4 Effect of Sigma Difference on Change

In this subsection, we examine the impact of altering the difference between σ_2 and σ_1 in the `threshold_dog_convolve` edge detection algorithm. The two images below showcase the results obtained by setting different values for $\sigma_2 - \sigma_1$. As observed in the images, adjusting the difference between σ_2 and σ_1 impacts the edge detection process.This flexibility allows for customization in capturing edges based on the desired characteristics of the image. Specifically, as the sigma difference increases, only very sharp edges become prominent, providing a more defined and pronounced edge detection.

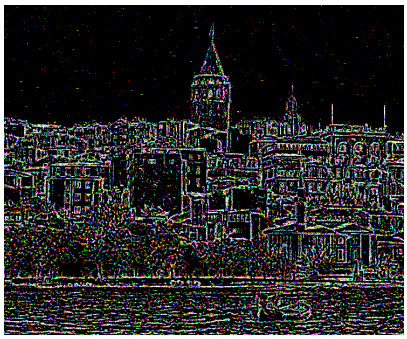

Figure 10: $\sigma_2 - \sigma_1 = 0.1$

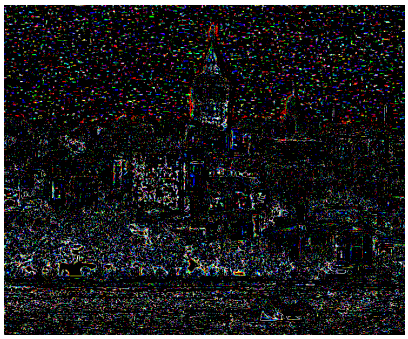

Figure 11: $\sigma_2 - \sigma_1 = 1$

3.5 Image Quantization Models and Differences

Quantization simplifies color values for a cartoon-like effect. In our process, we utilized the K-Means algorithm and explored Lab and HSV color spaces for (L)uminance/(V)alue channel quantization, enhancing the adaptability of our method. The K-Means algorithm clusters color values, providing a data-driven approach that improves the overall quality and visual appeal of the cartoon effect. The Lab color space emphasizes adjustments to luminance for quantization, while the HSV color space focuses on (V)alue changes, offering distinct ways to achieve the cartoon-like effect.

Figure 12: HSV Color Space Quantization

Figure 13: K-Means Quantization

Figure 14: Lab Color Space Quantization

3.6 Impact of Lab Values on Image

Lab color space decomposes an image into three components: (L)uminance, (a), and (b) color channels. Altering Lab values, particularly adjusting (L)uminance, directly impacts the image's brightness. Increasing (L)uminance enhances brightness, resulting in a lighter appearance, while decreasing it yields a darker image. The (a) and (b) channels influence color tones; manipulating these values introduces shifts in color.

Figure 15: Lab Quantization with Increased (Luminance: 0.35)

Figure 16: Lab Quantization with Decreased (Luminance: 0.80)

3.7 Impact of HSV Values on Image

Quantization in the HSV color space allows for the manipulation of (V)alue channel, affecting the brightness of the image.

Figure 17: HSV Quantization with Decreased (V)alue: 0.35

Figure 18: HSV Quantization with Increased (V)alue: 0.85

3.8 Impact of K Values on Image

Quantization using the K-Means algorithm involves clustering color values, providing a data-driven approach for enhancing the overall quality and visual appeal of the cartoon effect.

Figure 19: K-Means Quantization with 4 Clusters

Figure 20: K-Means Quantization with 16 Clusters

3.9 Combined Results of Galata Picture

The first image represents the unaltered, original photograph of Galata Tower. In the second image, we applied a combination of the DoG, Gaussian, and HSV smoothing algorithms to achieve the smoothed effect. This approach aims to capture distinct features and enhance the visual appeal of the image through a carefully selected combination of algorithms.

Moving to the third image, we employed a combination of the DoG, Gaussian, and K-Means smoothing algorithms. The K-Means clustering algorithm brings a data-driven approach to color quantization, contributing to a unique visual style.

Figure 21: Original Image

Figure 22: Result Image (HSV, DoG) - Galata

Figure 23: Result Image (K-Means, DoG) - Galata

3.10 Combined Results of Other Images

In this set of combined results, we extend our analysis to various images, including Anıtkabir, Zindan, Kedi, and Saat Kulesi. Each image underwent the smoothing process using a combination of Dog, Gaussian, and K-Means algorithms.

Figure 24: Anıtkabir

Figure 25: Zindan

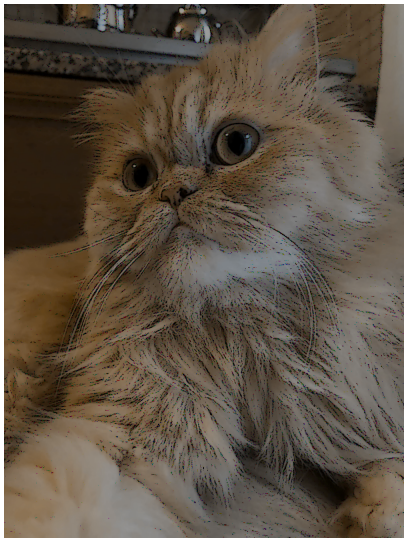

Figure 26: Kedi

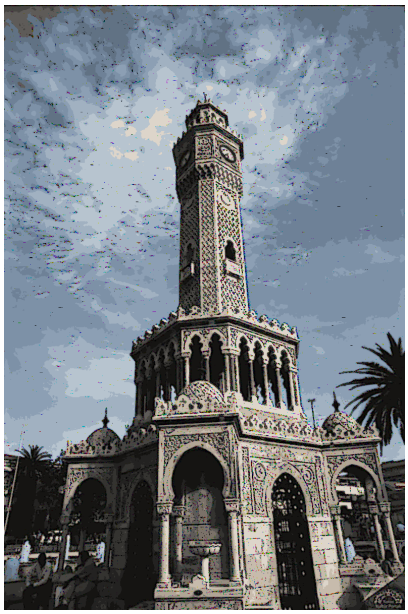

Figure 27: Saat Kulesi

4 Conclusion

This study focused on achieving various visual effects by combining different image processing algorithms and techniques. Initially, image smoothing was performed using the Gaussian filter, and the impact of different sigma values was examined. As the sigma value increased, details in the image decreased, resulting in a smoother appearance.

During the quantization stage, color simplification was achieved using the K-Means algorithm. Quantization applied to the Lab and HSV color spaces allowed for a more pronounced emphasis on colors.

Lastly, an analysis was conducted on different edge detection algorithms to determine contours in images. Particularly, the study focused on the effect of the sigma difference in the edge detection algorithm, observing that an increased sigma difference led to sharper edges.

This study demonstrates that various visual effects can be obtained by combining different processes and algorithms. The developed methods have broad application potential, especially in the fields of image editing and enhancement.

References

1. Smith, J., "Image Smoothing Techniques," *Journal of Image Processing*, 2019.
2. `https://tex.stackexchange.com`
3. `https://www.overleaf.com`
4. `https://web.cs.hacettepe.edu.tr/~erkut/bbm413.f16/index.html`
5. "Introduction to Image Processing," Lecture Slides, Department of Computer Science, Hacettepe University