Automatic Rotoscope

Vicki Chen, Ben Liu
ECE 160
UCSB
vchen@umail.ucsb.edu , bliu00@umail.ucsb.edu

***Abstract*—In order to produce a function which can produce a cartoon effect, we must explore different methods of image processing and determine which filter can provide accurate results while preserving key components of the original image file. The functionalities of a bilateral filter proved to be a popular method due to its ability to smooth images while preserving edges much better than other filters.**

I. INTRODUCTION

Nowadays you can hardly browse the internet without running into a GIF. They populate the internet on blogs and social media and play a huge part in providing entertainment as they are often used as statements or commentary.

A GIF(Graphics Interchange Format) is essentially an image file format that is animated by combining several other images or frames into a single file. Unlike other image formats, GIFs typically use a compression algorithm for easy storage of multiple image frames.

In this challenge, we are told to create a rotoscope animation from a short video file. This technique is often used to convert live -action movies into animations and was developed by Max Fleischer who saw a need for methods producing more realistic animations. After researching different methods of image processing, we decided to use a combination of Bilateral filtering with edge preservation and color adjust to produce a moving animation with a cartoon effect.

II. METHOD

A. Extracting Frames from a Video

The first step in this function is to extract multiple frames from a video file. The function reads a video file and saves the image in a local folder. The number of resulting images depends on the length of the video and the how many seconds per frame the user choses.

When the images are extracted to the local directory, a filter will be applied to all of them. Our next step reads all image files located in the local directory and applies a bilateral filter to create a cartoon effect.

B. Applying Filter to Images

A bilateral filter is a spatially varying filter that better preserves edges than the Gaussian Filter. Bilateral filtering smooths images while preserving edges through a nonlinear combination of nearby image values. It also combines gray levels or colors based on both their geometric closeness and their photometric similarity, and prefers near values to distant values in both domain and range. In contrast with filters that operate on the three bands of a color image separately, a bilateral filter can smooth colors and preserve edges in a way that is tuned to human perception.

Given a special width σx and a value width σy, the filter can be defined as

$$\beta_{\sigma x \sigma y}(fx) = \frac{1}{Zx}\sum_{y} G_{\sigma x}(x-y)G_{\sigma y}(f(x)-f(y))f(y)$$

When the normalizing constant is

$$Z_x = \sum_{y} G_{\sigma x}(x-y)G_{\sigma y}(f(x)-f(y))f(y)$$

at a given pixel x, it corresponds to an an averaging with the data dependant kernel:

$$G_{\sigma x}(x-y)G_{\sigma y}(f(x)-f(y))f(y)$$

As the range parameter σ_y increases, the bilateral filter gradually approaches Gaussian convolution more closely because the range Gaussian widens and flattens. As the spatial parameter σ_x increases, the larger features get smoothened.

Since the files we are working with are color video files, we must define the edges of the images as part of our filter process to make sure our final output performs smoothing accurately.

By measuring the pixel values of a photo, it is able to detect edges if certain areas of the picture have colors that differ too greatly from its neighbors. Bilateral filters determine which colors are different or the same. After which only the similar colors are grouped together and its color values, averaged together. Once an average of the similar colors is determined, this new color replaces the old pixel values. This difference in color results to more outlined edges since the difference in colors with neighboring shapes are greater and more clear.

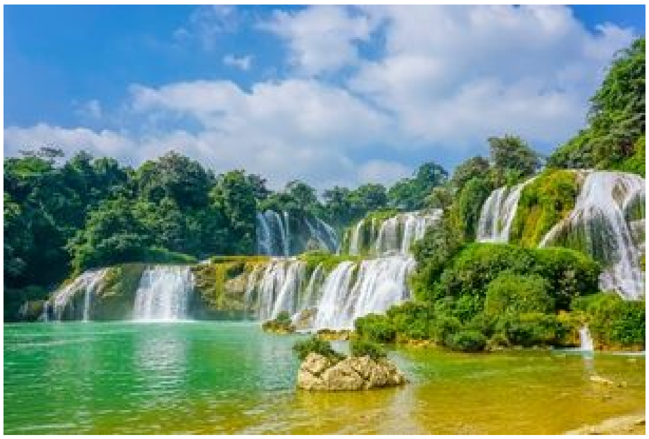

Fig 1. Original Image of a Scenery

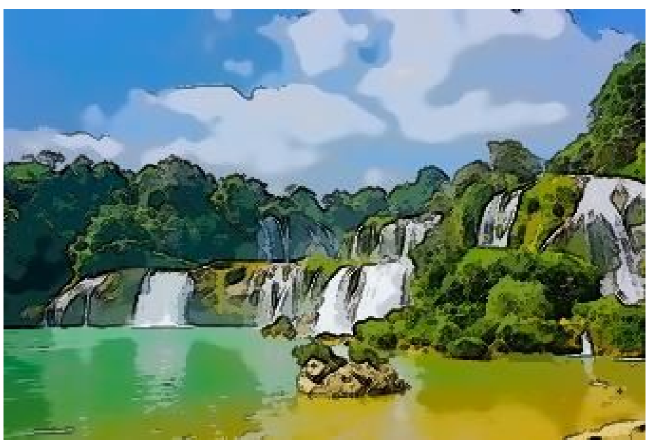

Fig 2. After applying bilateral filter

Bilateral filtering is named due to its process of combining domain and range filtering .It replaces the pixel value at x with an average of similar and nearby pixel values. In smooth regions, pixel values in a small neighborhood are similar to each other, and the normalized similarity function is close to one. As a consequence, the bilateral filter acts essentially as a standard domain filter, and averages away the small, weakly correlated differences between pixel values caused by noise

C. Changing Color

After deciding that we needed one more filter to produce cartoon like results, we decided to adjust the RGB scale of the image, creating a pop-art like output image.

By setting :

RGB = [255 133 114];

We can easily choose which color we want by their RGB values. After applying the bilateral filter along with edge detection, we converted the image to grayscale and filled in the white areas with the chosen color. The resulting image accurately mimics a pop art style filter

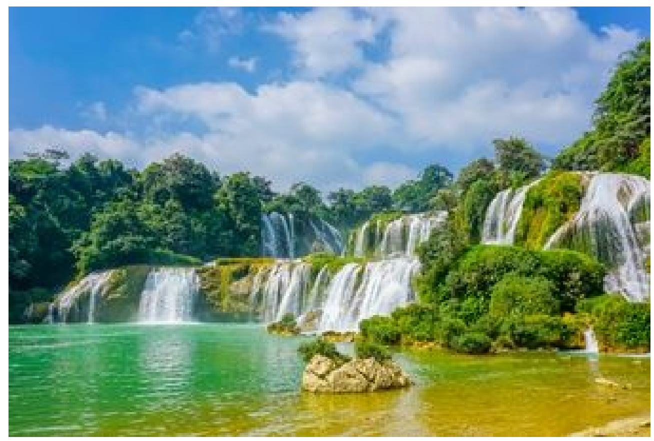

Fig 3. Original Image of a Scenery

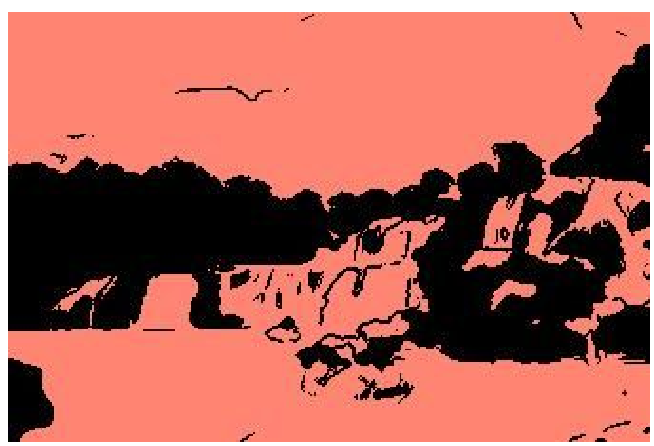

Fig 4. After applying bilateral filter with color adjustment

D. Turning the images into a GIF

The last step of our function takes all the filtered images and produces a gif. This can be easily done by combining all the output image files and creating a map which stores and compresses the selected image files.

III. RESULTS

The resulting product is a gif of the input video that has been cartoonfied

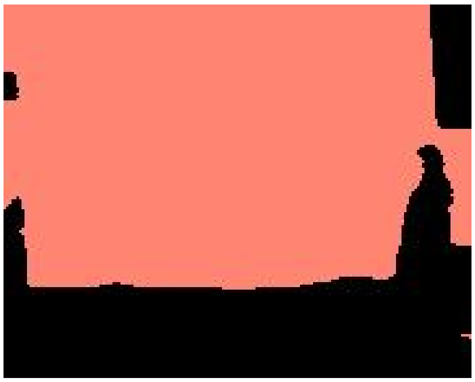

Fig 5. Image of final GIF cartoon effect

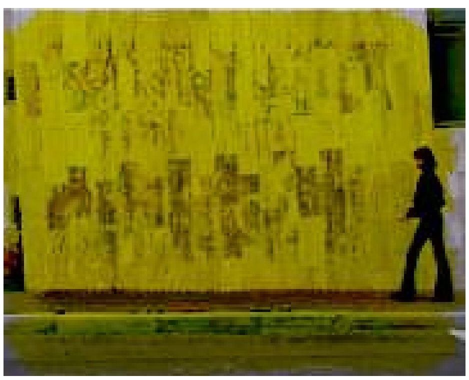

Fig 6. Image of final GIF with Yellow effect

IV. CONCLUSION

For this project, we first individually researched the topic until we had a solid understanding of the different options we had. After that, we decided on one possible solution and worked together on constructing the method. We built off each others code/ideas as we went and used the knowledge we learned from different sites to build our image processing method.

The challenging part of this project was determining how to correctly produce a rotoscope like animation simply by using image processing methods and practices. Another challenge we faced was that our animation for some reason would jump back to a beginning frame every couple of seconds. This stumped us for a while as we couldn't figure out what in our code caused this. However, after looking into the file array, we noticed that the program read our files in a strange order. Instead of reading from 1, 3, 5, 7, 9, the program would jump straight from 1,11,13,15 and only go back to 3 once it got to the 30's. From this we learned that the program can't read digits properly, so we decided to just start the count at 11 so all index values would be 2 digits.

Fig 7. Image of GIF with teleporting problem

We were able to explore areas that we were not exposed to before which made this challenge problem a fun experience. With a more visual project, we had a much easier time exploring different concepts as it was more clear what exactly what is happening to our photo.

REFERENCES

[1] Homepages.inf.ed.ac.uk. (2018). *Bilateral Filtering*. [online] Available at: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MANDUCHI1/Bilateral_Filtering.html [Accessed 7 Jun. 2018].

[2] Mathworks.com. (2018). *Generate Cartoon Images Using Bilateral Filtering- MATLAB & Simulink*. [online] Available at: https://www.mathworks.com/help/visionhdl/examples/generate-cartoon-images-using-bilateral-filtering.html [Accessed 7 Jun. 2018].

[3] "Bilateral filter," *Wikipedia*, 12-May-2018. [Online]. Available: https://en.wikipedia.org/wiki/Bilateral_filter. [Accessed: 07-Jun-2018].

[4] "cartoonizing real images," *Stack Overflow*. [Online]. Available: https://stackoverflow.com/questions/3954484/cartoonizing-real-images. [Accessed: 07-Jun-2018].