Photo Booth Utilizing Non-Photorealistic Rendering

Chao Li, Bryan Suzan, Alayna Truttmann

http://photobooth.cs.wisc.edu/

http://pages.cs.wisc.edu/~cli/Webpage/

1. Abstract

This project aims at filtering input images with various effects, via means of color quantization, bilateral filtering, tone mapping, color adjustments, and more. Specifically, the filters we implemented were sketch, X-ray, sepia, black and white, cartoon, and pop art. Our program allows users to select images quickly and easily and customize filter effects on them through our user-friendly GUI. Users can also compare different effects through a photo booth style setup. Users can then share these images if they choose through social media.

2. Introduction & Inspiration

Image filtering is popular and frequently used by people sharing photos on social media. Snapchat, Instagram, Facebook, and countless other apps and websites can be used to achieve the desired effects. When choosing a project to work on, we were motivated by the prevalence of image filtering and chose to create a program that would allow users to upload a photo and add a custom filter to it. We drew our inspiration from the Photo Booth for macOS program, originally released on October 2005. This program was chosen due to its variety of filters for users to put on their images, including both color and style transformations.

3. Goals

1. Create a GUI that makes it easy for people to filter the photos they upload.
2. Implement Sketch, X-ray, sepia, black and white, cartoon, and pop art filters.
3. Allow users some level of customization with the filters.

4. Related Work

As mentioned in the introduction, our photo booth filter was inspired by the Photo Booth program for macOS. As seen in Figure 1, the program allows users to filter their images in a variety of ways. The biggest difference between our program and Photo Booth is that the images in Photo Booth are filtered in real time, while in our program they are static uploaded images. Our program also offers levels of customization not seen in the Photo Booth program.

Figure 1

In creating each of the filters, we took influence from several sources that detailed the process needed to be taken to create said filters. For the sketch filter, we used the paper titled *Combining Sketch and Tone for Pencil Drawing Production* (Lu et al., 2012). The process taken by them produced a beautiful pencil sketch image that was transformed from a photograph. We also used the *Real-Time Video Abstraction* paper (Winnemöller et al., 2006) to create the cartoon effect.

5. Method

When using Photo Booth, the user inputs an image of their choice to be altered. Options for the filters are sketch, sepia, X-ray, black and white, cartoon, and pop art.

5.1 Sketch Filter

The sketch filter has a process of several stages that can be seen in Figure 2. The original image is used to make a gradient and a tone map. From the gradient, the stroke map is created. This map is used to determine how thick the outline of contents of the image will be. It allows for the final image to contain a well detailed depiction of the original image. The tone map is combined with an image of pencil texture to create the sketch-like result. This image is called the pencil map, as it used as the background of the final image. Finally, the stroke map and the pencil map are combined to create the final image. The details from the original picture are maintained throughout the process to not lose the objective of creating a well-drawn image of the contents.

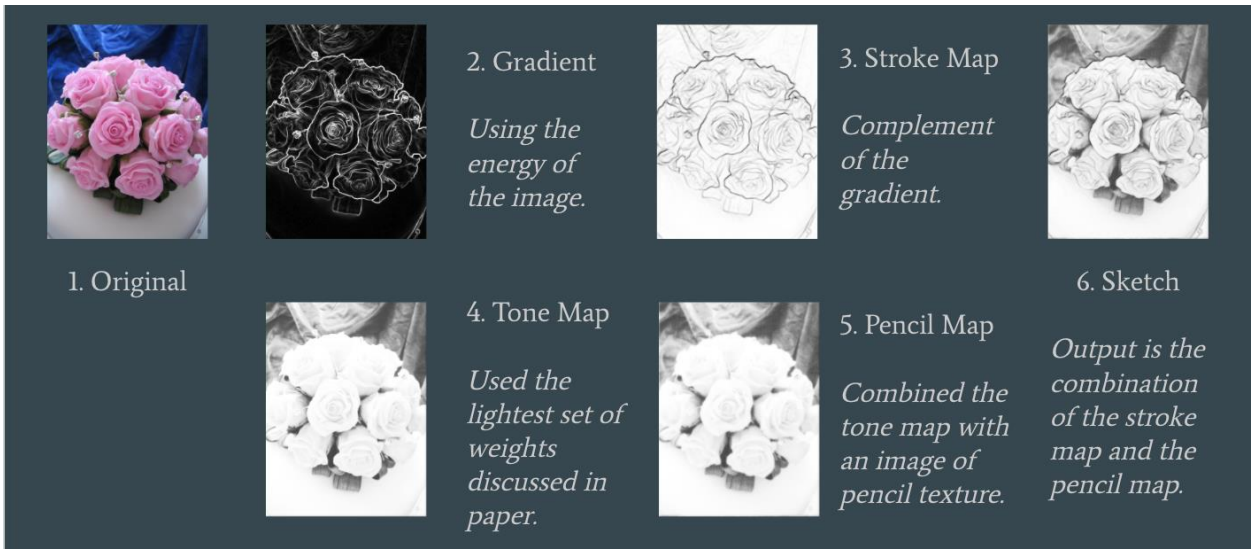


Figure 2

5.2 Sepia Filter and X-Ray

These filters were simple to make. The process begins with extracting the RGB channels from the original image. The numbers in these channels are then used with the following equation to produce the output image's RGB channels:

outputRed = (inputRed * .393) + (inputGreen *.769) + (inputBlue * .189)
outputGreen = (inputRed * .349) + (inputGreen *.686) + (inputBlue * .168)
outputBlue = (inputRed * .272) + (inputGreen *.534) + (inputBlue * .131)

These enhancements were used to create the Sepia image. When analyzing the channels, the Red and Green channels are significantly higher than the Blue channel to obtain the copper and golden color that a Sepia image contains. For the X-Ray filter, we used the same equation except that the Red channel of the output image was enhanced more than the other channels. This gave the output image a light red to pink color with white. Then we calculated the complement of the image with a built-in Matlab function (imcomplement). This gave the X-Ray effect to the output image, creating the filter.

5.3 Cartoon

The cartooning filter was created using the method described in Holger's *Real-Time Video Extraction* paper (2006). To achieve the effect, first a bilateral filter was applied to the image for a user-specified number of times. The bilateral filter serves to blur local contrast of pixels while preserving the contrast between strong edges. We left the intensity of the effect up to the user so they could determine the amount of overall blurriness. The next step is color quantization. This too was left up to user specification to determine how many colors were allowed in the image, which varies the level of detail. Then, an edge map is calculated and added to the image to give a black outline to the edges of the image. Figure 3 illustrates these steps.

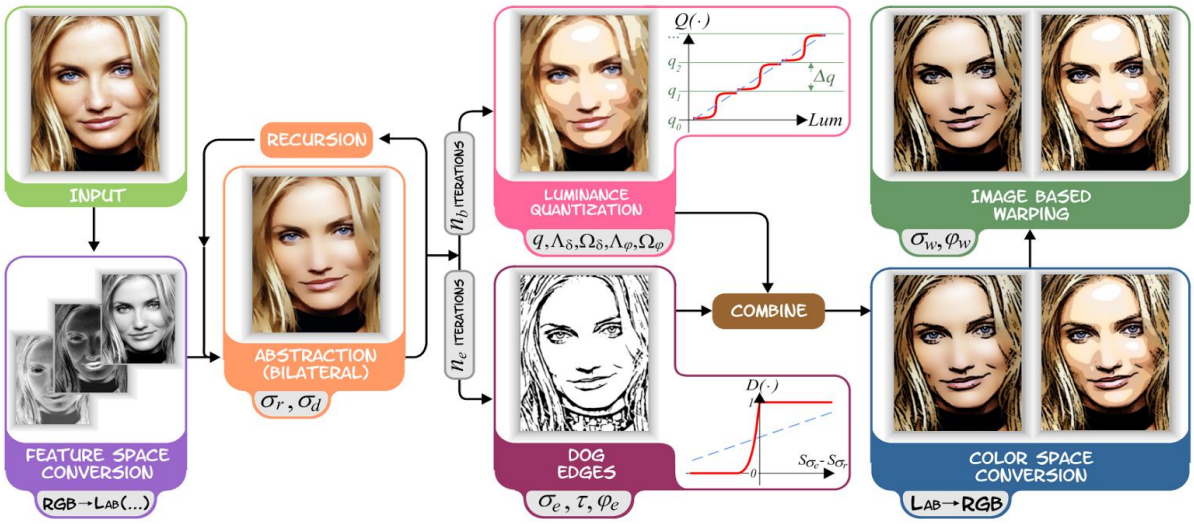


Figure 3

5.4 Pop Art

The pop art filter was created based off of results from the cartooning filter inspired by Andy Warhol's *Marilyn Diptych* work. First, the user selects 4 RGB color values. Then the image is cartoonized and converted to a binary black and white image. Then, the image is duplicated so that there are four images, and the white areas of each image are filled with the chosen RGB colors. This method has been illustrated below in Figure 4.

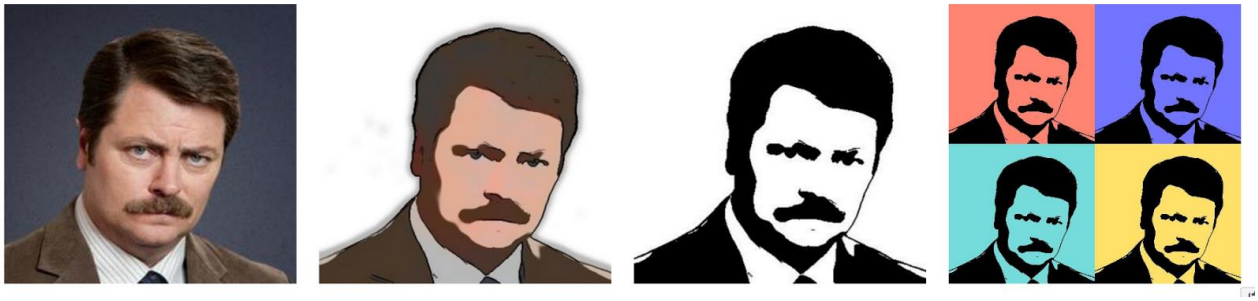
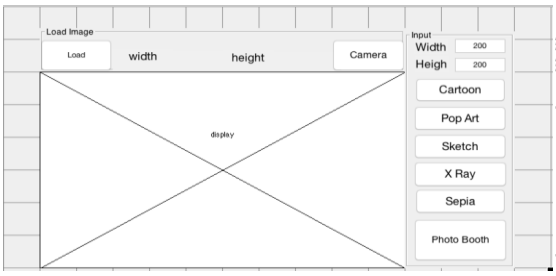


Figure 4

5.5 GUI

After all the filter functions are implemented, we created a Matlab GUI that incorporates all these functions. Users can directly interact with different filtering functions by loading input images. The GUI is implemented in three steps. First, we created a GUI skeleton that contains display panels and control buttons. By clicking different control buttons, user can apply different filtering effects to input images. The input image is uploaded to Matlab through load button that allows user to get files from their local computer. Next, we connect every functional button the filter function created.




```
% --- Executes on button press in Cartoon.
function Cartoon_Callback(hObject, eventdata, handles)
% hObject    handle to Cartoon (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
img = handles.image;
img = bilateralFilter(img, 3);
img = edgeDetect(img, 9);
handles.image = img;%create filtering effect
axes(handles.display); %display the text
imshow(handles.image)
guidata(hObject, handles);
```

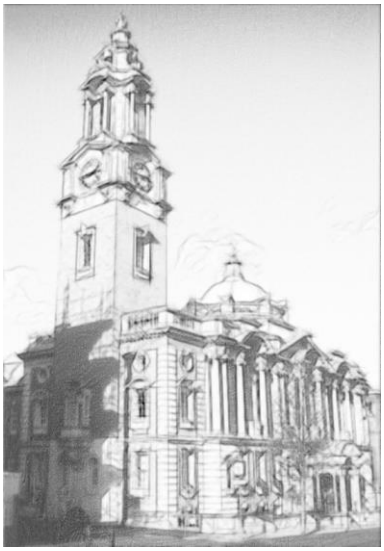
5.6 HTML Webpage

Initially, the host for the web-page we used was our personal HTML page from our CS account. With professor's help, we were able to use a website photobooth.cs.wisc.edu. The web page is written in HTML, CSS, and JavaScript. It contains basic structure of a website, and utilizes various CSS format styles and basic JavaScript to add some interactive functions.

6. Experimental Results

The following images are the results for each filter. The first image is the original and the next is the output with the filter.

6.1 Sketch



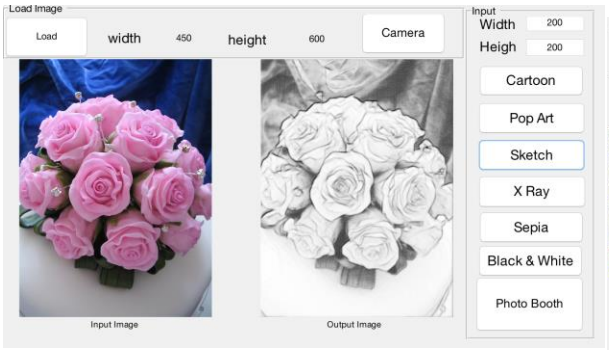
6.2 X-Ray

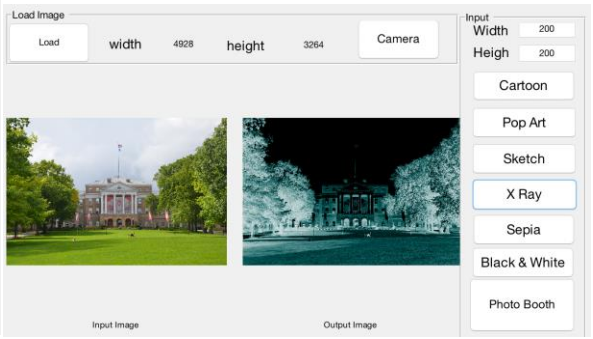
6.3 Sepia

6.4 Cartoon

6.5 Pop Art

6.6 GUI





7. Concluding Remarks & Ideas for Improvement

Our Photo Booth works well as a basic image filtering program. There are no “failure” cases of filtering gone wrong, but there are some areas where we hope to refine the filtering process so that it is more successful for all images. In the future, we could add more filters and improve the filters already created.

There are a few aspects of the pop art effect that could be improved. First, the effect is rather oversimplified in that it only has two colors. It could be modified to incorporate more colors and detail, as seen with the three-color pop art effect in Figure 5. Ideally, the user could give some sort of input to choose where each color would go. Second, the pop art filter works best on images that have simple backgrounds, so implementing a background removal operation would significantly help to improve images like Figure 6.

Figure 5

Figure 6

The sketch filter implementation we retrieved was not perfect. Its results, when compared to the results obtained by the paper's process, were blurrier and did not have as much detail. Upon analyzing the code and how the programmer who developed it went about the implementation, we noticed that the calculations for the tone map were a bit off. In the future, we plan on further analyzing this code and finding a new way to implement the tone map as described by the paper. This will allow the final sketch image to have a closer resemblance to the paper's results, allowing the image to seem like an artist drew it.

Despite the areas that need improvement, our Photo Booth works well on most images and serves the purpose to transform images in a delightful way. By developing these filters, we learned to notice visual clues on how a filter was created from our inspirational images. It took time and experimentation to assess if we were making the correct color and style changes to output a consistently visually pleasing image. We are satisfied with the results and hope that future users will enjoy our Photo Booth.

8. References

Lu, C., Xu, L., & Jia, J. (2012). Combining Sketch and Tone for Pencil Drawing Production. Proceedings of the Symposium on Non-Photorealistic Animation and Rendering. Eurographics Association, 65-73.

Winnemöller, H., Olsen, S. C., & Gooch, B. (2006). Real-Time Video Abstraction. ACM SIGGRAPH 2006 Papers on - SIGGRAPH '06, 1221-1226. Retrieved from http://holgerweb.net/PhD/Research/papers/videoabstraction.pdf

9. Team Member Contributions

Chao Li: Wrote abstract, created GUI and project web-page.
Bryan Suzan: Created sepia, X-ray, and sketch filters.
Alayna Truttmann: Created cartoon and pop art filters.
All: Wrote progress & final report.

10. Code

Approximate lines of code written: 1200

Code we created:

- Sepia filter
- X-ray filter
- Black and white filter
- Cartoon filter
- Pop art filter
- Matlab GUI Interface

Code retrieved elsewhere:

- Sketch filter
    - Written by Feng Lele, edited by Bryan Suzan
    - Source: https://github.com/candycat1992/PencilDrawing