Shinkai Makoto's Imagination: An attempt to mimic the Japanese Anime Style

Jing Qian, Xiaofei Liu, Ruihao Zhu
http://dianaqian1995.wixsite.com/cs534-project

Abstract:

The project is motivated by the artwork of Japanese anime director, Shinkai Makoto. This report demonstrates the process of converting a real life landscape image into the typical hand drawing style of Shinkai Makoto. The process is divided into three steps, including detecting the sky region of an input image, animefying the input image and replacing the original sky region with the selected sky source image. While we successfully transform some input images to the anime style, there are also some limitations in this program. One is that sky detection algorithm is not robust for all kinds of input images and the other is that color adjustment works better for input images with higher brightness and saturation.

Introduction:

As photo apps such as Instagram become popular, people are having fun with different filters and enjoy various images effects by applying these filters. Motivated by the popularity of artistic image filters, we aim to develop a project that allows users to transform a real-life landscape image to a anime-style image. Although we did not find specific papers related to achieve the anime-style effect, we did find some guidance online that gave us insights about how to mimic the anime effect. In order to achieve such effect, we divided the process into three steps: 1) segmenting out the sky region 2) manipulating the input image to make it more "anime-like", and 3) replacing the sky region of the original image with the source sky image that comes from anime. Accordingly, this paper gives an overview of our project idea and project motivation, and highlights the methodologies we have used and the results we have achieved for this project. This paper also gives details about the algorithms we have used for each step, the problems and challenges we have encountered and how we have solved these problems.

Motivation:

Makoto Shinkai is a Japanese director of anime and former graphic designer, and our idea was inspired by his most recent anime movie called "Your Name". This film has been a big success in Japan and China, and we were fascinated by Director Shinkai's anime style. A lot of scenes in his movie actually come from real world. We then wanted to build an application that allows users to animefy a real-life image. Another reason that we are interested in this project idea is that this project could bring some fun and romantics to people's routine lives.

Problem statement:

The main problem of our project is to convert a real-life landscape photo into a typical hand-drawing style of Japanese anime. We approached to this problem by dividing it into three main steps that each was implemented in a separate module and then merged them together in the end. The first step is to detect and return a boundary between the sky region and ground region of an input image. The second part is to apply several filters to the input image to make it look more a anime style image. The last step is to replace the sky region segmented by step one with a candidate sky background image from the existing image set.

Methodology:

Sky segmentation:

We completed this section by following the method described in a paper that is specifically about detecting a sky region in an image. This paper makes two assumptions about any input images. One is that "The luminance of the sky region changes smoothly", and the other is that "The application is in autonomous ground robot navigation, and so we assume that the sky region is above the ground region" (Yehu Shen & Qicong Wang, p4).

The entire algorithm contains three main parts or functions. The first part is the "rough selection" of the border. The algorithm first asks for the gradient changes of the input image, and for each x (1<=x<=width), the algorithm, going from top to down, compares the gradient change on a pixel at (y,x) (1<=y<=height) with a threshold t, and assigns y to be the value of the border function of x (= b(x) =y) if the gradient is greater than t at this pixel.

The second part is about border optimization which is build upon the first function described above. The paper defines its own energy function which is used as a criteria to determine the best border. It provides the threshold min, threshold max and step size it uses and also defines a formula based on these three variables to calculate the threshold t as a parameter for the function to find the border. For each threshold t it tries, the algorithm calculates the energy function and chooses the border, b(x), that has the largest energy function value.

The last part of the algorithm is to deal with some special cases. In the paper, there is actually a part to detect if a sky region exists in an image, and since in our case, we will purposely select our input images with a sky region in it, we decided to skip the part that checks whether a sky region exists or not. One special case the paper discussed is that when the sky region doesn't occupy the whole but partial upper region of an image. For instance, there may be a building that takes over the entire left space of an image. The reason that such case is particularly handled is that the algorithm for detecting the border is based on sudden gradient change from top to down and left to right. As a result, if a large object such as a building occupies the upper left or upper right region of an image, the previous functions described above might not be able to detect the correct y-value to assign to the border function b(x). Accordingly, in order to detect such cases when some columns that do not contain a sky region, the paper defines an equation:

$$t = thresh_{min} + \frac{thresh_{max} - thresh_{max}}{n-1} * (k-1)$$

where n is the step size and k is any integer from 1 to n.
It calculates the absolute differences of the border's row value between two adjacent border pixels. If there is at least one absolute difference is greater than certain threshold, then the method concludes that the special case is detected. The paper develops a function for sky region refinement to handle such special case if it is being detected. It applies k-means to the rough sky region detected in the earlier steps to get two clusters based on the color space of the region, and we call them sky-true and sky-false. It then compares the Euclidean distance between the two centroids generated by k-means and the centroid for ground region. The method determines the cluster that has the centroid closer to the ground centroid is not really belongs to the true sky region. Then the algorithm recalculates the border based on the updated sky region.

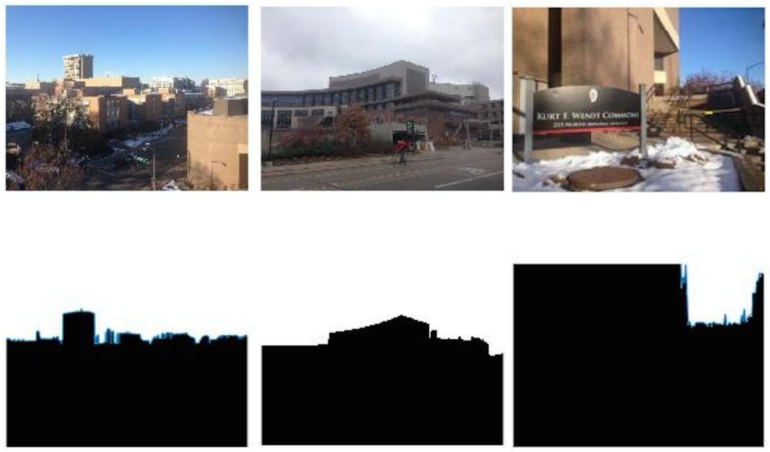

Figure 1. sky detection examples

Image filtering and color adjustment:

After sky border is detected, several image filters are applied to the input image to achieve the anime style. Bilateral filter is used to cartoonify the image because of its ability to smooth an image and preserve the edges. Then, an unsharp masking is used to make the edge more obvious. Color adjustment including increasing brightness and saturation is applied on RGB channels separately.

Specifically, a bilateral filter is a combination of a domain and range filter. It is defined as:

$$h(x) = k^{-1} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\varepsilon) c(\varepsilon, x) s(f(\varepsilon), f(x)) d\varepsilon$$

where measures spatial closeness between the neighbor pixel and center pixel and measures the photometric similarity between a neighbor pixel and the neighborhood center, k is a normalization constant. (Tomasi, C., Manduchi, R)

Traditional filtering computes the weighted average of nearby pixels in the neighborhood in which the weights of nearby pixels decrease with distance from the neighborhood center. Similarly, a range filter averages pixel values with weights that decreases with photometric similarities, such as intensity values. The range filter by itself does not make much sense because a pixel far away from the center pixel should not affect the value of the center pixel even though they have similar photometric values. Therefore, the intuition behind the bilateral filter is to combine the domain and range filter to preserve the spatial and photometric locality. In this way, the bilateral filter can successfully smooth an image and also preserve its edges.

The bilateral filter implemented in our project is using Gaussian function to measure the closeness and similarity. The closeness function is defined as a Gaussian function of Euclidian distance between a nearby pixel and the center pixel, while the similarity function is then defined as a Gaussian function of the intensity difference between two pixels.

According to the equation shown above, the bilateral filter replaces the pixel value with an average of similar and nearby pixel values. In smooth region, the similarity function is close to one so that the bilateral filter is basically a Gaussian blur filter. Near the edge, the similarity function is close to one for pixels on the same side of the edge and is close to 0 on the other side of the edge. As a result, areas on the same side are smoothened and no smoothing happens on the other side of the edge. In addition, the sigma_d is the parameter associated with domain filter and determines the amount of low-pass filtering. With a large sigma_d, the larger features get smoothened. The sigma_r is associated with the range filter. As sigma_r increases, the bilateral filter approaches to a low-pass filter because the range filer flattens and becomes nearly constant over the intensity value.

When we analyzed the artwork of the artist, we realized his work only has small smoothing effect. Thus, we apply the bilateral filter with sigma_d = 10, sigma_r = 6 to smooth small features.

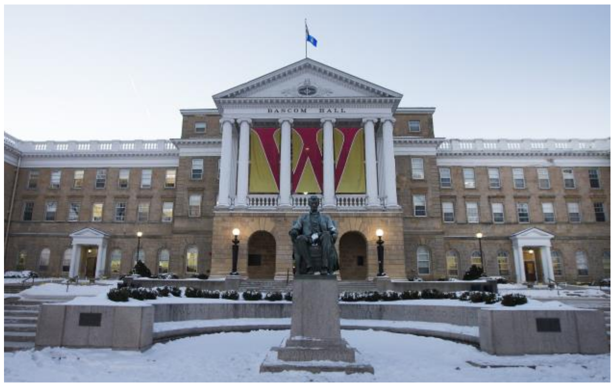
Figure 2a: input image

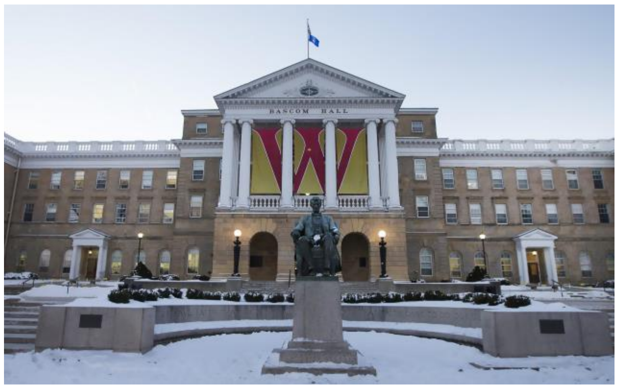
Figure 2b: image after bilateral filtering

After bilateral filter is applied, an unsharp masking is applied to make the edges more obvious. The unsharp masking is defined as:

$$h(x) = f(x) - k(f(x) * g)$$

where k is a positive constant and is a Laplacian mask.

The last step is to adjust the brightness and saturation of the input image. We experimented multiple ways to increase the saturation, for example, increasing the saturation channel in HSV color space. It turns out that the best result is to apply color adjustment on RGB channels separately so that each red, green and blue tones are enhanced naturally. Specifically,

$$extra_{red} = \max(red - \max(blue, green), 0)$$

$$new_{red} = \min(red + k * extra_{red}, 1)$$

where k is constant that defines how much red tone to add, $extra_{red}$ denotes which part of the red channel has higher red tone, and new_{red} is the red channel after red tone is enhanced (Liberman D).

After brightness and saturation increase, the right image is the final result after applying all image filters.

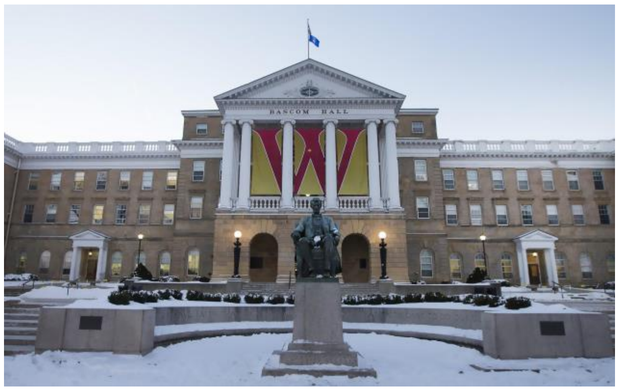
Figure 2c. after bilateral filter

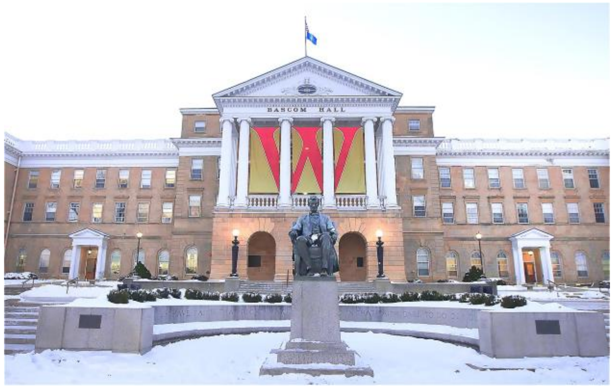
Figure 2d. after color adjustment

Sky replacement:

The sky replacement module takes in two parameters. The first parameter is the input image, which needs to be converted and returned as an output file. The second parameter is the horizontal seam that is generated by sky segmentation.

The horizontal seam is a row vector that detects the boundary between sky region and ground region of an image. The index of the seam array represents the width of the sky region while the values of the array are row indices. For example, if the horizontal seam is [1, 2, 3], this means the sky boundary of the input image contains three pixels with its coordinate (1, 1), (2, 2), and (3, 3). The first step is to identify whether the input image is taken during the day or night based on the grayscale pixel values of sky region.

First of all, the image is converted from RGB to grayscale values. A nested for loop is used to go through each pixel above the seam. For each pixel above the seam, we check whether the value of that pixel is closer to 0 or 1 by comparing the absolute difference between the pixel value and

0 or 1. If the pixel value is closer to 0, then it is a dark pixel. Otherwise, it is identified as a light pixel. After we count the number of light pixels and the number of dark pixels, if the number of light pixels is more than the number of dark pixels, we assume the input image is taken in day time. Otherwise, we assume the input image is taken in night time. Then, we added some background sky images as replacement candidates into our image set. Half of the data set is day-time sky images, while the other half of data set is night-time sky images. Once the image has been identified, the module would provide corresponding sky source images from the data set.

Once we selected the sky image, we could replace the sky region of input image with the selected sky background images. However, before the replacement, we checked the size of input image and the size of sky background image. We also resized the sky background image to make sure that the width of sky background image matches the width of the input image and the height of sky background image is equal to the maximum entry value of the horizontal seam.

Finally, we replaced the sky by copying pixels from sky background image to the input image. Specifically, we used a nested for loop to go through each pixel in the sky region of the original image, we assigned its value with the pixel value with the same coordinates from the sky background image.

Figure3 shows a pair of images that before sky replacement and after sky replacement.

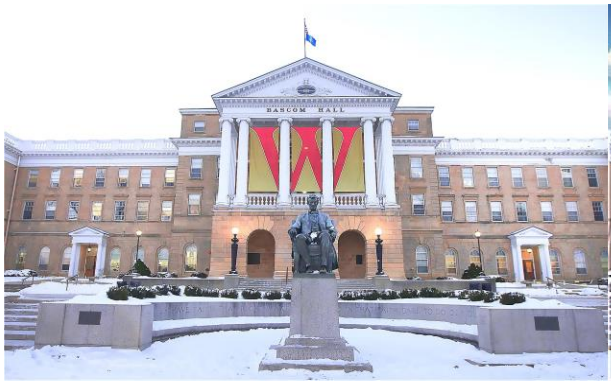

Fig 3a. input image without sky replacement

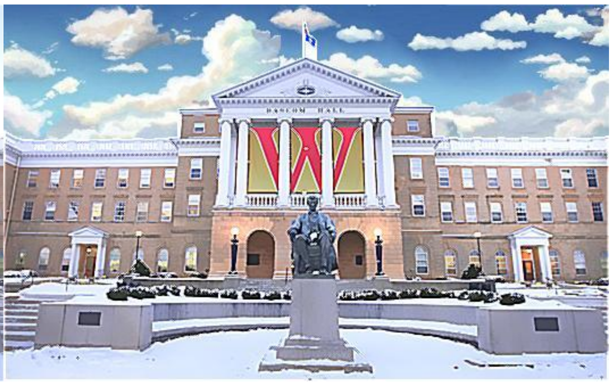

Fig 3b. output image after sky replacement

Experimental Results and limitations:

The final results of our project are shown below, with the first three images as our output images and the last one as the original input image.

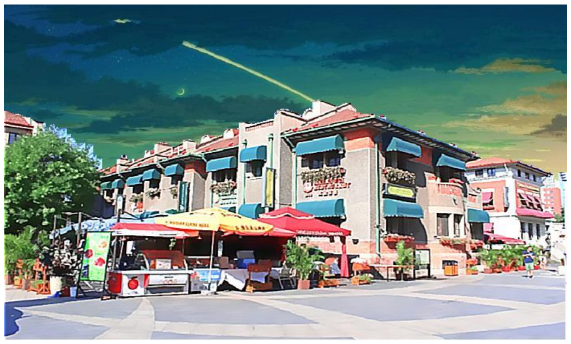
Figure 6a. output image using background 1

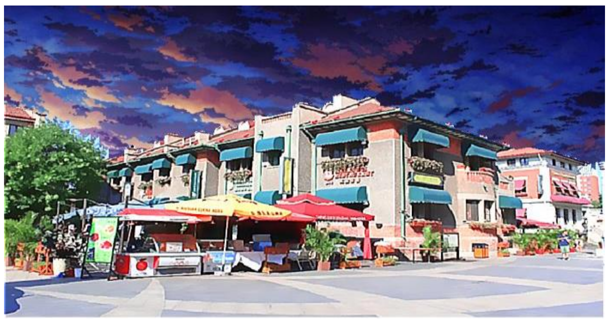
Figure 6b. output image using background 2

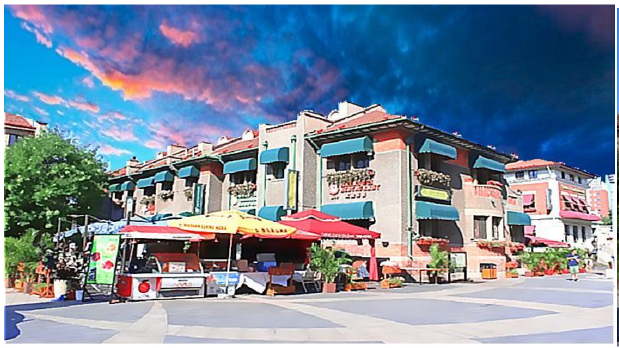
Figure 6c. output image using background 3

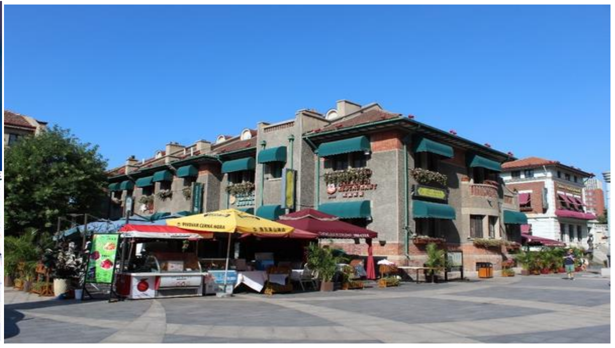
Figure 6d. original input image

However, our method has some limitations and due to these limitations, some results are not very satisfactory. One of the limitations are that the paper we used for implementing sky detection is by no means suitable for all the input images with a sky region in it. For example, it cannot detect the sky region that is partially covered by trees (see upper right of Figure 4). Additionally, the algorithm is highly dependent on the threshold t that is calculated based on thresh_min, thresh_max, and step size, meaning different t values are highly influential to the final selection of border. As a result, sometimes during the border selection, there are spikes which are not supposed to be generated.

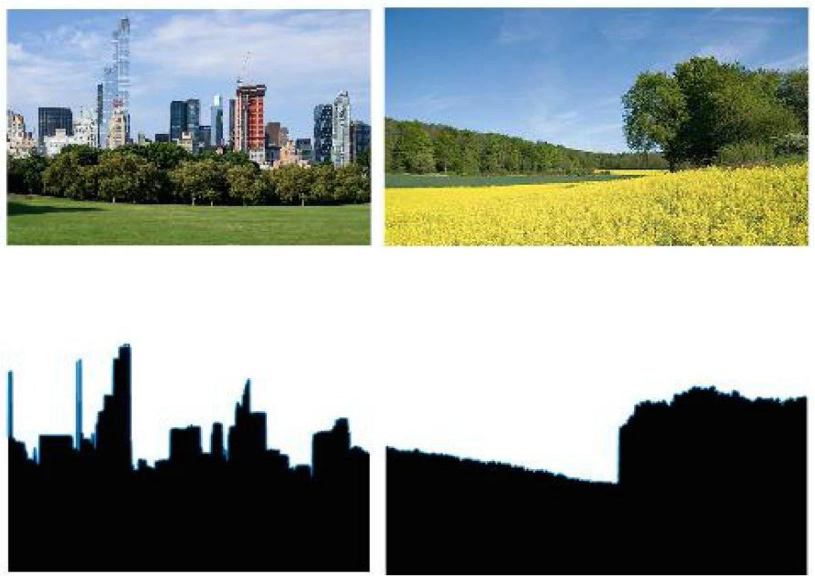
Figure 4. inaccurate sky detections

Additionally, in the image filtering step, we experimented photos with different intensity values and exposures. It turned out that the filter achieves the best result when the source image is taken in a sunny day. Under this circumstance, the source image has higher contrast and the RGB color is more obvious. Thus, it is easier to increase the RGB tones in this image and gain a better result.

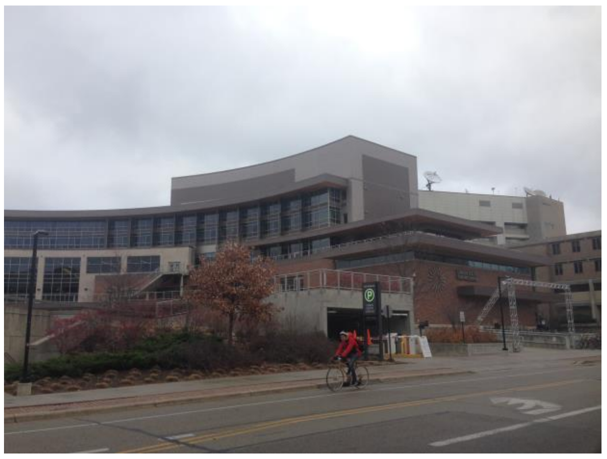
Figure 5a. image with low contrast

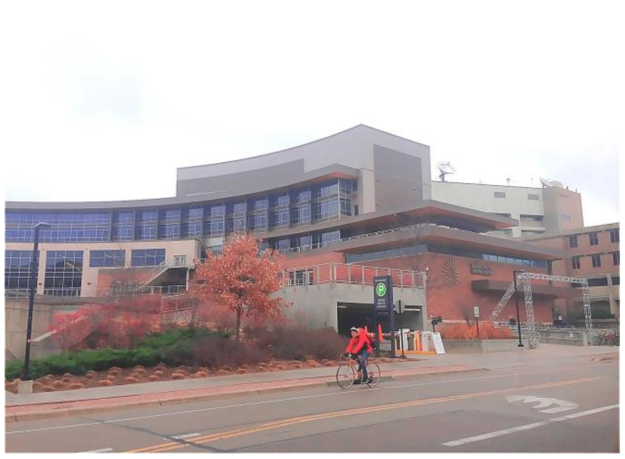
Figure 5b. bad filtering result

Conclusion:

In conclusion, we successfully transformed a real-world image to the anime style. Our program can be used in many photo apps to create some fun and artistic effects on the photos.

In the future, we hope to improve the sky segmentation step so that it can perfectly segment out the sky region. Image filtering can also be improved to achieve various anime effects. Moreover, we would like to replace not only the sky region but also other types of objects such as replacing a building or adding an anime character into the input image.

Notes (workload):

1. Jing Qian worked on sky segmentation. This section contains seven functions with a total of roughly 170 lines of codes in Matlab.
2. Xiaofei Liu worked on image filtering. The filtering function is about 70 lines of code. Part of the bilateral filter and color adjustment are based on answers found in MathWorks.com.
3. Ruihao Zhu worked on sky replacement module. The sky replacement function is about 75 lines of code. He also used the solution from homework 3 to help resize images used in his function.
4. The total lines of code written by our group, including the main module but not the use of extra source and solution from previous homework, is approximately 350 lines.

References

Liberman D. MathWorks. Retrieved from https://www.mathworks.com/matlabcentral/answers/48053-how-to-enhance-blue-color-alone-of-an-image

Tomasi, C., Manduchi, R. (1998). Bilateral filtering for gray and color images. Retrieved from https://users.cs.duke.edu/~tomasi/papers/tomasi/tomasiIccv98.pdf

Shen Y., & Wang Q. (2013). Sky region detection in a single image for autonomous ground robot navigation. *International Journal of Advanced Robotic Systems, 10*(362). doi: 0.5772/56884