ECE/CSE 576, Spring 2019 Homework 1: Filtering, Edge finding, and Clustering

Philip Pham

April 20, 2019

Task 1: Convolution

For this task, we implement a general convolution method for linear filters. My code can be found at `Convolution`. The run time complexity is $O(HWCK_wK_h)$, where H is the height of the imate, W is the width of the image, C is the number of channels, K_w is the width of the kernel, and K_h is the height of the kernel.

Task 2: Gaussian Blur

In this task, we apply the Gaussian blur filter.

$\sigma = 4.0$ was used. The darkened borders are due to zero padding.

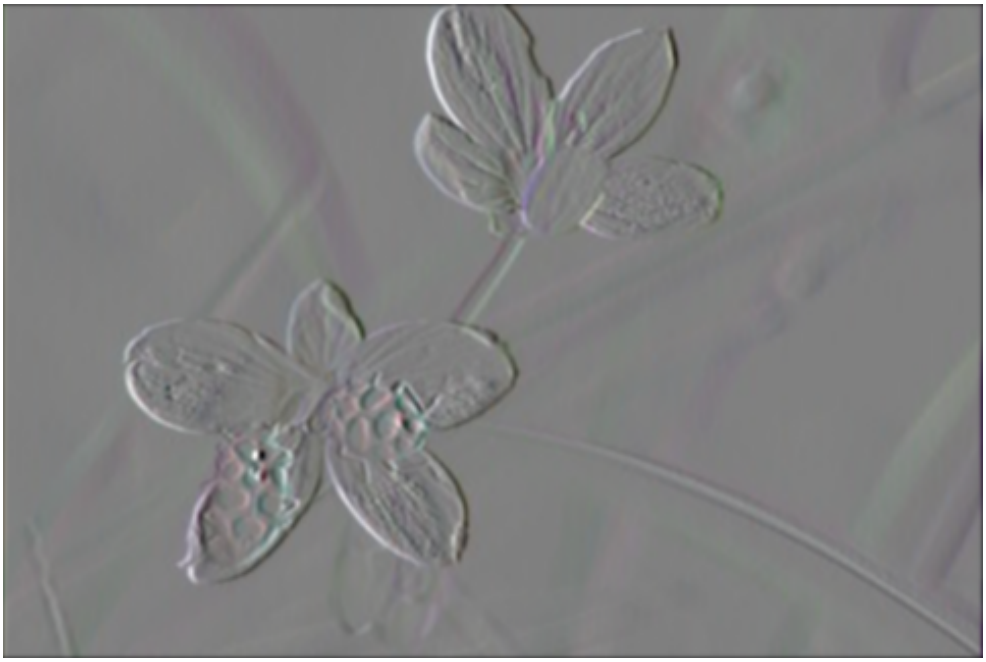
Task 3: Separable Gaussian Blur

The separable Gaussian blur applies a vertical and horizontal Gaussian blur filter separately.

The results are the same as Task 2, which demonstrates the commutativity as associativity of convolutions.

Task 4: Image derivatives

In this task we compute derivatives to help detect edges.



x-derivative of `LadyBug.jpg` with $\sigma = 1.0$.

y-derivative of `LadyBug.jpg` with $\sigma = 1.0$.

Second derivative of `LadyBug.jpg` with $\sigma = 1.0$.

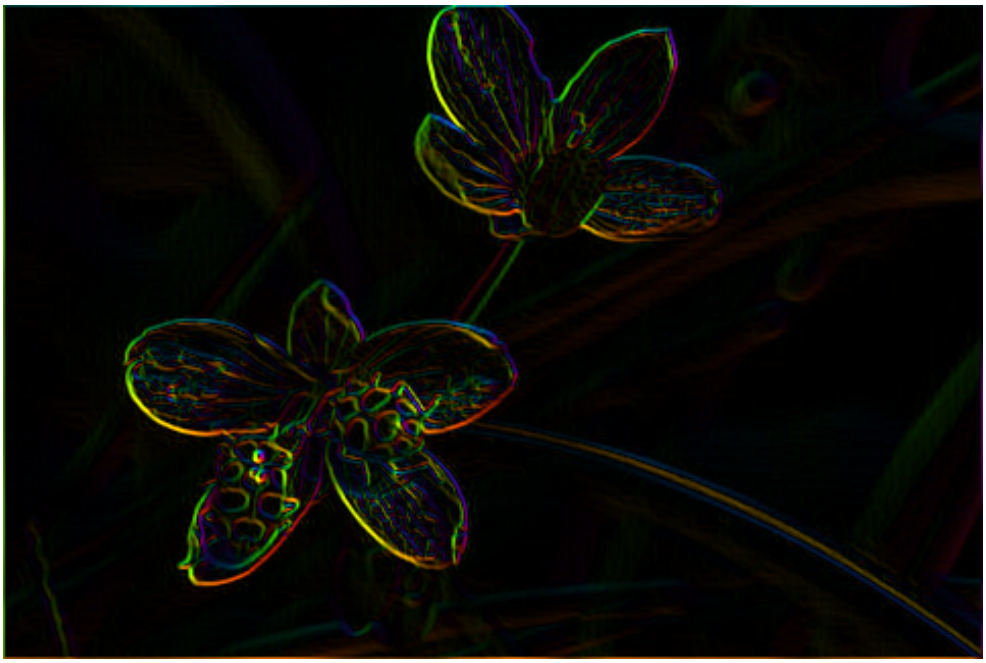
Task 5: Image Sharpening

By substracting the second derivative from the original image, we can sharpen it.

`Yosemite.png` sharpened with $\sigma = 1.0$ and $\alpha = 5.0$.

Task 6: Edge Detection

The Sobel operator can be used to detect edges.



The Sobel operator is applied to `LadyBug.jpg`.

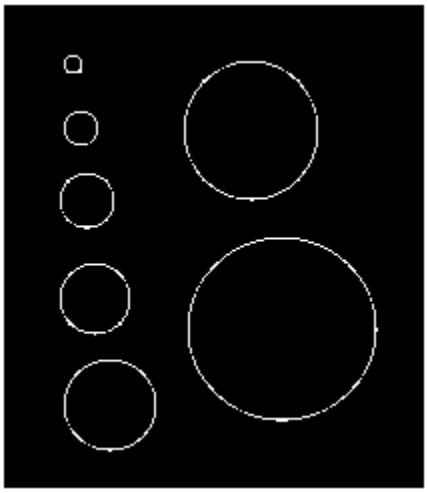
Task 7: Bilinear Interpolation

Bilinear interpolation is used to rotate an image.

`Yosemite.png` rotated by 20°.

Task 8: Finding edge peaks

Edge peaks are found with a combination of the Sobel operator and bilinear interpolation. The Sobel operator gives us a pixel's magnitude and orientation. With bilinear interpolation we can get the magnitude of the two neighboring perpendicular pixels. If the pixel is an edge peak, it should have the greatest magnitude.



Edge peaks for `Circle.png` with a threshold of 40.0.

Task 9: K-means color clustering

Each pixel of an image can be seen as belonging to a cluster based on its colors. Lloyd's algorithm with the l_1 norm is used to cluster the pixels.

Task 9a: With random seeds

One way to initialize the clusters is randomly.

K-means clustering applied to `Flower.png` with 4 clusters initialized randomly.

Task 9b: With pixel seeds

Another way to initialize the clusters is by selecting a subset of pixels to be initial seeds.

K-means clustering applied to `Flower.png` with 5 clusters initialized from pixel seeds.

Bells: Reflected padding

As seen in Tasks 2 and 3, one downside of zero padding is that the intensity of the border points change. Padding by reflecting along the image borders can fix this.

`Seattle.jpg` padded out. The original border is shown in red.

Gaussian blur with $\sigma = 4.0$ applied to a padded with reflection `Seattle.jpg`.

The pixel intensity along the border is now maintained in constrast to Tasks 2 and 3.

Bells: Histogram K-means

In Task 9b, selecting clusters from the empirical distribution of pixels can be noisy. One way to smooth out the distribution is to bucket them into a histogram and initialize clusters that way. If each color is divided into 8 buckets, there will be $8 \times 8 \times 8 = 512$ total buckets.

K-means clustering of `Flower.png` into 5 clusters initialized from a histogram.

Bells: Median Filter

Taking the median intensity of of a neighborhood can smooth and reduce noise.

Noisy black and white version of `Seattle.jpg`.

Noise reduction on greyscale version of `Seattle.jpg` with a median filter of radius 2.

Whistles: Bilateral Filter

The bilateral filter is similar to the Gaussian filter in that it smooths and reduces noise. By using a weighted average of pixel intensity, it preserves sharp edges.

Bilateral filter applied to `Yosemite.png` with $\sigma_S = 2.0$ and $\sigma_I = 20.0$.

Whistles: Hough Transform

The Hough transform takesas input edge peaks. For each edge peak pixel, the Hough transform considers all the lines that the pixel could possibly be part of by considering all θ in polar (r, θ) space.

The most likely lines are returned and merged with the edge peaks are filtered based on these lines to get line segments.

Identifying line segments with the Hough transform in `Flags.png`. The original image, edge peaks, and the result of the Hough transform are shown.

Appendix

All code used to generate these images can be found at `ppham27/cse576/hw1`. The embedded PNG files and the LaTeXcan be found in `ppham27/cse576/hw1/report`.