Assignment 4

**Prepared by**Brendan Burke

**Computer Vision  
CAP 4410.01**

**4/15/2018**

**Table of Contents**

[**List of Figures**](#_bl6c85yx81ay)**…………………………………………………….................................................................. 2**

[**List of Acronyms**](#_6955l5hos812)**………………………………………………………………………………………………. 3**

[**Assignment 4**](#_aj5i4n7gsb6c)**………………………………………………………………………………………………….. 5**

[Assignment Details](#_1fob9te)………………………………………………………………………………………………………………………………... 5

[SIFT](#_2et92p0)……………………………………………………………………………………………………………………………………………………….. 5

[Implementation](#_vvldakxgr3li)…………………………………………………………………………………………………………………………………….. 6

[Figures](#_q3fqrv7nxyq4)…………………………………………………………………………………………………………………………………………………... 7

**References…………………………………………………………………………………………8**

# **List of Figures**

**Figure 1** Octaves……….……………………………………………………………............................... 1

**Figure 2** DoG Results…………................................................................................................................. 2

**Figure 2** MinMaxLoc Results………………………………………………………………………….... 3

# **List of Acronyms**

SIFT………………………………………………………………..Scale Invariant Feature Transform

DoG…………………………………………………………………………....Difference of Gaussian

LoG…………………………………………………………………………….Laplacian of Gaussian

# Assignment 4

## Assignment Details

This assignment required in depth study and research of the Scale Invariant Feature Transform (SIFT). Although certain versions of OpenCV include a built in SIFT function, it does not show the output of each step. This was the goal of this assignment: to show the output of each step of the SIFT function.

## SIFT

As I learned, the first step of the SIFT function is to find the scale space extrema. This step is performed by many substeps. The first of which is to create several octaves of the original image. An octave consists of several layers. Each layer of the octave has a sigma value. Typically, the first layer in the octave has the lowest sigma value, and the last layer has the largest sigma value. These sigma values are used by the Gaussian function to progressively blur the image. Then, every progressive octave is actually half the size of the preceding octave. So, this substep saves several sizes of the image, with different levels of blur. This is good, because it helps the program view the image with different sizes and clarity. Thus, making it scale invariant.

Once the octaves have been calculated, the next step is to perform Different of Gaussian (DoG). The reason why DoG is calculated instead of Laplacian of Gaussian (LoG), is because LoG is computationally expensive. DoG on the other hand is not computationally expensive, and is a very close approximation of LoG. Therefore, DoG is calculated instead of LoG. Now, not only is DoG computationally inexpensive, it is also incredibly easy to implement. During my research, I discovered that finding the DoG of two images is as easy as subtracting one from the other. After calculating the DoG for each octave, the results were as I expected. Each DoG image was mostly a black image, with small white dots spread throughout the image. These white dots are key feature points.

The next substep for finding the scale space extrema is to use the DoGs to find possible key feature points. A sliding window is used to compare the current pixel with the surrounding eight pixels. Separate sliding windows are used in the DoG images that are before and after the current one being analyzed. So, instead of eight pixels being compared to the current pixel, twenty six pixels are being compared. The purpose of this is to find the best key feature point with respect to the image’s sigma value.

The second step in the SIFT function it to localize the key point. In this step, the Taylor series is used to refine the accuracy of the keypoints. This is because an image might have numerous key points to use, but the majority of the points might not have very accurate features. So, the Taylor series is used to compare the extrema intensity value to a certain threshold, the Contrast Threshold. If the intensity value is less than the Contrast Threshold, it is rejected. If the intensity is greater than the Contrast Threshold, it’s value is kept. Furthermore, techniques similar to the Harris Corner Detection algorithm are used to remove edges. This, again, is to help obtain the best key feature points.

The next step is Orientation Assignment. This step is very important because it is used for image rotation invariance. This is obtained by taking the neighboring pixels around the key feature points found in the previous step. The neighborhood is determined based on the size of the image scale. The magnitude and direction of the key feature point is determined based on the info found in the key point neighborhood. Finally, an orientation histogram is created to hold thirty six bins over . What this step does, is allows the program to recognize the key feature points at different rotations. So even if the key feature point is rotated differently than it is in the “training” image, the program will still be able to recognize it, because it has performed the Orientation Assignment step.

Next, Keypoint Descriptor is performed. I found this step particularly interesting because of how much detail going into exploring the key feature points. First, a 16 by 16 area is selected around a key feature point. This neighborhood is then divided into sixteen blocks of 4 by 4 sub-neighborhoods. Then, from each sub-neighborhood, an eight bin histogram is created. This heavy analysis of each key feature point is fantastic for achieving invariance to illumination, rotation, and other changes.

The last step of the SIFT function is Keypoint Matching. In this step, the key feature points between two images are compared. This is done by matching to the nearest neighbors. Yet in some matching cases, the first closest match might be very similar to the second closest match. To resolve this problem, a ratio is taken. If the ratio is greater than 0.8, then they are rejected as matches.

## Implementation

This was a rather difficult and large assignment. It required the most amount of time and energy implementing. Yet, there were two very useful websites I found that helped me tremendously, by helping me understand the steps of the SIFT function. Both websites provided in depth understanding of the inner steps of each major step of the SIFT function.

The first step I took with implementing the SIFT function from scratch was creating all of the octaves. I performed this step through nested for loops. The first for loop created the octaves, while the second for loop created the layers inside each octave. I learned through my research that the author of the SIFT function recommended four octaves with five layers in each octave. Each image in each layer also has a Gaussian filter applied. The sigma value changes at a constant rate of *sigma+2* per each image. Then, each image is saved, respective of its octave and layer, in a folder called Octaves.

The next step in my implementation, was calculating the Difference of Gaussians. I used nested for loops to access the images that were saved in the previous step. First I load the “current” image, then I load the succeeding image. Then, after validating the images loaded correctly, I performed DoG by subtracting the current image from the succeeding one. Then I saved the results in a folder called DoG. The images were saved with respect to the octave and layer.

The next step I performed was finding the key feature point extrema. Again, I used nested for loops to access the images in the previous step. I had two different ideas when first attempting this step. The first idea was to use two vector variables to store the minimum and maximum values and locations. This way, it would be easy to access the data. The second ideas was to create a Mat image of zero values to store the location of the minimum and maximum values. This would actually be easier to access the data, because I would be able to use the OpenCV built in function, findnonzeros(), which finds the location of nonzero values in an image. So, whenever the algorithm would find a minimum or maximum value, it would change the same location in the zero Mat image with a one. Then, the algorithm could easily find the “one” locations in the zero Mat image via the findnonzero() function, then search for those same locations in the DoG image. Yet, after further research, I discovered a more optimised and much simpler option by using the minMaxLoc() built in function. This function find the minimum and maximum values, and find their locations as well. I used respective variables to hold these values. Then, after loading in the image, I performed the minMaxLoc() function. I did run into some trouble with this function at first though. I discovered that this image does not accept multi-channel images; RGB images. So, after converting the image into grayscale, minMaxLoc() performed wonderfully.

After finding the minimum and maximum values and locations, I hit a roadblock. I understood that the next step was to perform Keypoint Localization using the Taylor series. But I was unable to make the connection between the different variables in the Taylor series equation, and the information I had obtained from the previous steps. I searched through numerous websites and sources, yet was unable to understand the connection. What I did understand, was that the Taylor series was used to calculate the first order derivatives, and get more accurate key feature points by comparing it to the Contrast Threshold. Then, one would use techniques similar to the Harris Corner Detection to eliminate edges.

From this point on I was unable to continue to the remaining steps of implementing the SIFT function. Reading from various sources online I completely understand the process that undergoes during the SIFT function. The SIFT section in this report proves I have a deep understanding of how the SIFT function works. Therefore, given more time, I would strive to complete the remaining SIFT function steps. Specifically, I would work towards understanding how to translate the data I have gathered so far, into the Taylor series to obtain more accurate key feature points.

## Figures

Please see figures by following this link: <https://docs.google.com/document/d/1iVihsg8Z55omMr0MKqEzpatlgvA2M39NaV9j4aAAGzE/edit?usp=sharing>

**References**

[1] Sinha, U. (n.d.). The scale space. Retrieved April 15, 2018, from <http://aishack.in/tutorials/sift-scale-invariant-feature-transform-scale-space/>

[2] <https://www.researchgate.net/post/How_can_I_check_keypoint_localization_step_in_SIFT_works>

[3] Find local maxima in grayscale image using OpenCV. (n.d.). Retrieved April 15, 2018, from <https://stackoverflow.com/questions/5550290/find-local-maxima-in-grayscale-image-using-opencv>

[4] (n.d.). Retrieved April 15, 2018, from <https://docs.opencv.org/3.4.1/d2/de8/group__core__array.html#ga8873b86a29c5af51cafdcee82f8150a7>

[5] O. (n.d.). Opencv/opencv\_contrib. Retrieved April 15, 2018, from <https://github.com/opencv/opencv_contrib/blob/master/modules/xfeatures2d/src/sift.cpp#L332>

[6] (n.d.). Retrieved April 15, 2018, from <https://docs.opencv.org/3.1.0/de/d25/imgproc_color_conversions.html>

[7] Operations with images¶. (n.d.). Retrieved April 15, 2018, from <https://docs.opencv.org/2.4/doc/user_guide/ug_mat.html>

[8] Taylor Series Function - C Forum. (n.d.). Retrieved April 15, 2018, from <http://www.cplusplus.com/forum/beginner/15537/>