Implementing sGLOH in OpenCV

Ardalan Ahanchi, Michael Waite

School of STEM

University of Washington, Bothell Campus

18115 Campus Way NE, Bothell, WA 98011

{Ardalan,Mwaite2}@uwb.edu

1. Introduction

In 2010 Fabio Bellavia, Domenico Tegolo, and Emanuele Trucco presented a research paper entitled “Improving SIFT-based descriptors stability to rotations.”[1]. In this paper they put for the idea of a new descriptor that they named sGLOH, which they based on a descriptor known as GLOH. Their goal was to create a SIFT-like descriptor (like GLOH) but avoid rotating the feature patch before calculating descriptors by making discrete rotations of the descriptor possible.

2. Background and Goals

The goal of this project was to implement sGLOH in C++ using the OpenCV open source library, version 4.1.2. Implementation was intended to be done following the OpenCV project’s contribution guidelines such that this project could be added into the OpenCV library for release. Given unexpected difficulties in the development process this goal has not yet been reached. However, the initial goal of implementing the ideas in the paper have been reached. Additionally, we implemented a Key-point detector, and a brute-force matcher.

3. Implementation

Implementation was initially separated out into two tasks to maximize work between team members: implementation of the feature point extraction and image intensity gradient extraction and implementation of the sGLOH descriptor calculation, rotation, and matching.

3.1. Descriptor

The sGLOH descriptor works by partitioning the area surrounding a feature point into *n* circular rings, further separating those rings into *m* equally sized segments to define a log-polar grid of regions $R_{r,d}$ where $r = \{1, 2, \ldots, n\}$ and $d = \{0, 1, \ldots, m-1\}$. Given a descriptor *H* the central ring is then defined by a function *PSI(H)* which equates to 0 if the central ring consists of a single region and 1 if it consists of *m* regions.

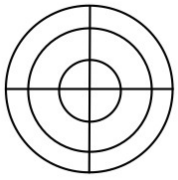

Figure 3.1.1. *sGLOH patches which use the polar coordinates,*

Each region is then divided into *m* gradient orientation histogram bins representing the distribution of image intensity gradients for *m* equally spaced directions in the region $R_{r,d}$. The sub-pixel distribution is then estimated for a bin $h^i_{r,d}$ by a weighted summation of all pixel-gradient vectors described by the Gaussian equation

$$h^i_{r,d} = \frac{1}{\sqrt{2\pi}\sigma} \sum_{p \in R_{r,d}} G_m(p)\, e^{-\frac{(M_{2\pi}(G_d(p) - m_i))^2}{2\sigma^2}}$$

Figure 3.1.2. *Gaussian Window used for estimation.*

where $G_m(p)$ and $G_d(p)$ are respectively the gradient magnitude and direction for a pixel p in the region $R_{r,d}$ with $r = \{0, 1, \ldots, n\}$ and $d = \{0, 1, \ldots, m - 1\}$, $m_i = 2*PI*i/m$ represents the center of bin I, and sigma = $2*PI*c/m$ with c being a positive real number represents the standard deviation. Finally, The periodicity of q is determined by the function $M_q(x)$ [1].

$$M_q(x) = \begin{cases} x & \text{if} \quad x < \frac{q}{2} \\ q - x & \text{otherwise} \end{cases}$$

Figure 3.1.3. *Gaussian Window used for estimation.*

Moreover, the elements of the vector are shifted by (d positions), and a histogram is defined. Finally, the output descriptor can be calculated by concatenating the histograms.

3.2. Key-point Detector

A key-point detection implementation was written which follows a SIFT like logarithm [3]. This part of the implementation was not required or used by the author of the sGLOH paper [1]. In their implementation, they used a custom Harris-Z edge detector to extract points. However, due to the unavailability of the Harris-Z detector (Not in OpenCV), and the calculation of rotations with the SIFT's [3] default key-point detector, a custom implementation was written.

Initially, the detector creates a pyramid of the images across various scales (octaves) and blur levels. Each octave will contain multiple blurring levels. Each image in the octaves is blurred by the Gaussian Blur method.

For the implemented extractor, 4 octaves, and 5 blur levels were used as suggested by SIFT's author [3], The next step is calculating the Difference of Gaussian images in the pyramid. This is achieved by negating the images from each other. This results in octaves having one fewer image in them.

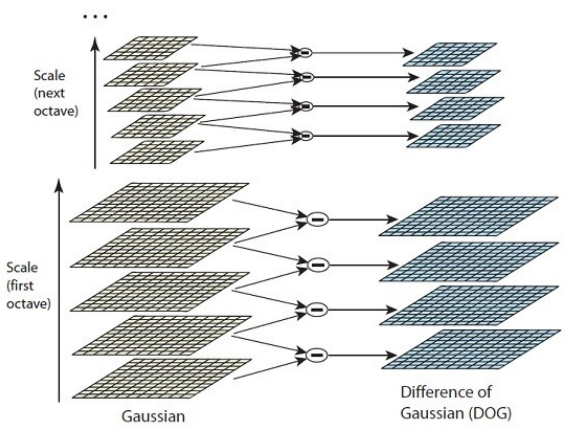

Figure 3.2. *Calculation of Difference of Gaussian in the scale-space pyramid [4].*

After the Difference of Gaussian images are produced, the maxima/minima are located for each pixel's neighbors. There are a total of 26 checks to determine if a pixel is an extreme.

For narrowing down the number of extracted key-points, a threshold of contrast is applied to the point. If the key-point meets the threshold, then a check is made to confirm that the pixel is not on an edge. Due to the strong response of the Difference of Gaussian Images along the edges, this property has to be checked to further narrow down the number of generated key-points.

4. Process and Challenges

Implementation of the descriptor calculation began with in-depth research of the mathematics equations provided by Bellavia et al.'s paper [1]. Due to lack of experience with the symbols used the initial research took multiple days and extensive perusal of mathematics articles on Wikipedia [5].

After the initial research was concluded problems began to surface regarding the quality of the explanation of the mathematics in Bellavia et al.'s paper[1], particularly in relation to variables being used in different contexts without being explicitly redefined. These problems put a halt on development for several days until in-person consultation with Professor Clark Olson could be achieved.

During the period of halted development, the example implementation of sGLOH in C by Bellavia et al.[1] was discovered and much effort was put into understanding said code. Due to the poor coding practices, and lack of comments in this implementation, the effort to understand it would continue until the week the development effort concluded. Understanding this implementation would ultimately prove unproductive, leading only to increased uncertainty in our development.

Upon resumption of development incremental progress would be achieved and demonstrated to colleagues over a period of a few weeks. During this period the algorithm for calculating the sGLOH descriptor vector was completed. Due to unexpected delays in the development of the image intensity gradient extraction algorithm, the development of the descriptor rotation and matching was delayed until the final week of development.

Sample data was chosen for testing and evaluation purposes by examining locales for the possibility of taking feature-rich pictures. Eventually a cluttered closet was chosen, and four pictures were taken at very high resolution with minor shifts in position.

The final week of development saw the completion of the descriptor rotation and matching algorithms. This effort was slowed-down by bizarre and cryptic compiler errors, overlong run-times, and repetitive stress injuries. The correctness of the rotation algorithm was ensured with careful pen-and-paper verification of code behavior due to the difficulties caused by the overlong run-times. Initially efforts were made towards using the OpenCV Brute Force Matching implementation but were met with failure. Re-implementing a brute force matching algorithm would prove successful but would also exacerbate the problem of overlong run-times during testing.

Due to an overabundance of feature point match output, both correct and incorrect, a decision was made to improve the sGLOH algorithm by adding a simple thresholding operation. This saw some success in improving the ratio of correct matches to incorrect matches.

To ease the testing of Key-point extraction and feature matching, an option was added which permits the sGLOH to switch it's key-point extractor in the run-time. Currently, sGLOH supports key-point detection using our own extractor, SIFT, and SURF algorithms.

5. Accomplishments

We implemented the sGLOH descriptor successfully following the algorithm describer in the original paper [1]. Moreover, we implemented our own SIFT like key-point detector. Which functions as expected, and it manages to extract the key-points. However, due to the lack of sub-pixel estimation, the accuracy of the points is not ideal.

Moreover, we achieved rotation in-variance by calculating the gradient magnitudes and angles for the whole image, and then rotating the extracted features by shifting them. Finally, we wrote test programs to test sGLOH, and our key-point

detector and compare it with other methods such as SIFT and SURF.

Having said the above, the most important accomplishment for this project was understanding the paper, and conceptualizing the implementation. This, by far, was the most difficult and time-consuming part of this project.

6. Results

For evaluating the project, our implementation of the sGLOH descriptor was compared with SIFT using various images, and key-point extractors.

6.1. Descriptor

To evaluate the descriptor's stability, the first and the fourth images of the bark and boat sequences in the oxford image database were checked. Additionally, the first and fourth images of the wall and graffiti sequences were used to evaluate the descriptors' accuracy when view-points are shifted/changed (Much like the test-case in the original paper) [1]. Moreover, we compared manually taken images which represented a more dramatic change in view-point.

To match the results, a custom brute force matcher was used. This custom matcher proved to be rather slow, and ended up being the bottleneck of our implementation. The matched images were compared with SIFT's output (Since it has been the benchmark in-between the descriptors).

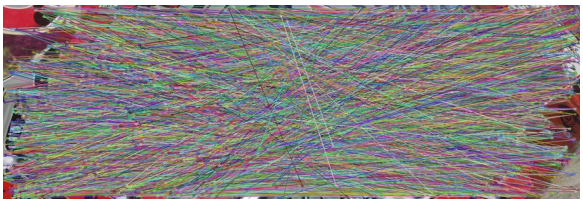

Figure 6.1.1. *Matches using the sGLOH descriptor on the Graffiti image sequence from the Oxford database [7] (Using the SIFT key-point extractor).*

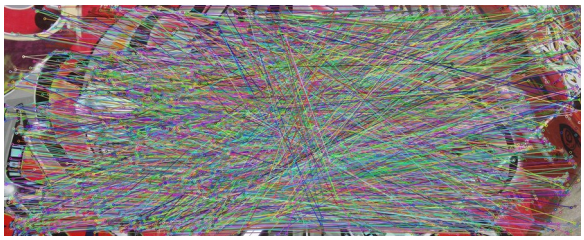

Figure 6.1.2. *Matches using the SIFT descriptor on the Graffiti image sequence from the Oxford database [7] (Also Using the SIFT key-point extractor).*

As we can observe from the figure 6.1.1 and 6.1.2, sGLOH performs slightly worse than SIFT when comparing changes in view-points. Moreover, due to the custom matching algorithm used by our implementation, the number of matches are much higher than SIFT's. These results are also mirrored when comparing the Wall sequence [7], which has drastically more matches between the images.

When comparing the results on the Bark and Boat data-sets, the results are less different. Both descriptors are finding similar matches, and the number of matches seems to be similar as well. Figures 6.1.3 and 6.1.4 show the comparison results. As it was suggested by the original author of the paper [1], the results confirm that the sGLOH descriptor is rather stable.

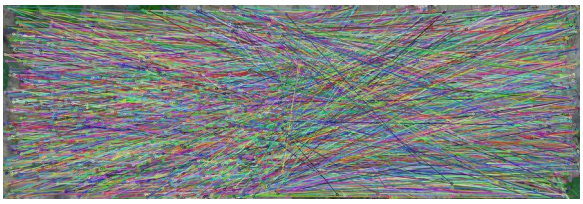

Figure 6.1.3. *Matches using the sGLOH descriptor on the Bark image sequence from the Oxford database [7] (Using the SIFT key-point extractor).*

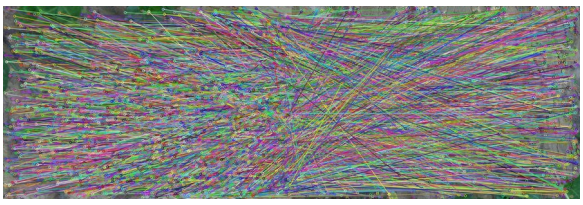

Figure 6.1.4. *Matches using the SIFT descriptor on the Bark image sequence from the Oxford database [7] (Using the SIFT key-point extractor).*

Testing showed heuristically that the optimal distance to threshold descriptor matches at is between 0.14 and 0.25. Below this range the number of overall matches decreases dramatically, and above this range the number of incorrect matches approaches equality with the number of correct matches.

6.2. Detector

To evaluate the Key-point detector, several approaches were used. Initially, we wrote a demo program which generates an image randomly with a few key-points, and then ran the detector on that image. It ended up performing as expected in those tests (Since it is not doing sub-pixel approximation, we did not expect it to be very accurate). However, it successfully detected the mentioned key-points. Figure 6.2.1 is the test program running and the blue dots are the detected key-points in the generated image.

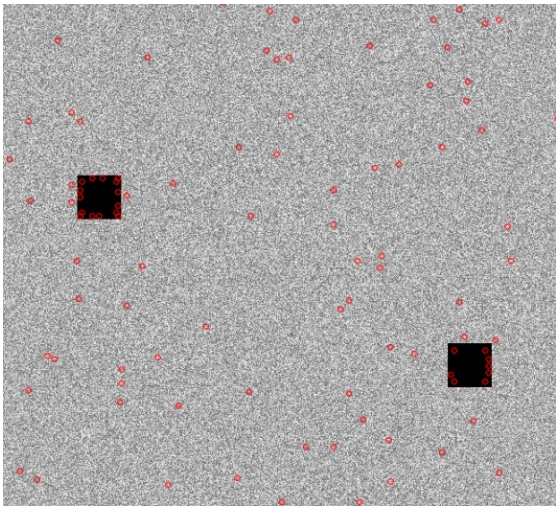

Figure 6.2.1. *A zoomed-in image of running our custom Key-point detector with a randomly generated test image.*

Next, we evaluated the Key-point detector by using it alongside our sGLOH descriptor. The same images from the oxford database were matched. Similar tests were done on SURF and SIFT's key-point extractor which allowed us to objectively compare the results.

Figure 6.2.2. *Matches using the sGLOH descriptor on the Bark image sequence from the Oxford database [7] (Using our custom key-point extractor).*

Figure 6.2.3. *Matches using the sGLOH descriptor on the Bark image sequence from the Oxford database [7] (Using the SIFT key-point extractor).*

The results are rather similar when comparing the images. However, as mentioned earlier, SIFT is more accurate for determining Key-points' locations. Additionally, we can observer that less key-points were detected in our extractor. This is mainly caused by not calculating rotation. Based on the original SIFT paper [3], for each dominant orientation, additional key-points are added at the same spot. However, this is not the case with our custom extractor.

6.3. Performance

To compare the performance, we used SIFT as the benchmark. In our implementation, none of the code written runs in parallel, so we expected our code to be significantly slower. Based on the evaluations, in average, the sGLOH descriptor is 3-5 times slower than SIFT. Same can be said about the key-point

extraction of SIFT when we compare it to our implementation.

However, due to unforeseen issues with the Brute-force matcher, we had to implement our own. As mentioned previously, the matcher ended up being the largest bottle-neck in our program. The matcher is 100-200 times slower than the sGLOH's feature detection and descriptor calculations combined. When using our custom key-point detector, the run-time is manageable. This is due to fewer key-points being generated using our algorithm (as explained in section 6.2).

7. Future Work

In the future we hope to be able to optimize and refine our code to make it acceptable for addition to the OpenCV library. We also would like to find out if there is a better method for thresholding the matching results to improve the accuracy of sGLOH.

Allowing our algorithms to run in parallel would also improve the performance significantly. A large portion of the calculations have the possibility to easily run in parallel. However, due to the time constraints we did not explore this idea.

Moreover, the key-point detector can become much more effective and comparable if we add sub-pixel approximation as it was discussed by the original SIFT paper [3].

8. Conclusions

In this project, we presented our implementation and the results of the sGLOH descriptor as it was described by it's original authors [1]. Additionally, we discussed our custom key-point extractor which was based on SIFT's which skips the rotational calculations on the key-points.

9. References

[1] Bellavia, F., Tegolo, D., & Trucco, E. (2010). Improving SIFT-based Descriptors Stability to Rotations. *2010 20th International Conference on Pattern Recognition*.doi:10.1109/icpr.2010.845

[2] K. Mikolajczyk and C. Schmid. Scale and affine in-variant interest point detectors. International Journal of Computer Vision, 60(1):63–86, 2004.

[3] D. Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2):91– 110, 2004.

[4] Sinha, U. (n.d.). SIFT: Theory and Practice: LoG approximations. Retrieved from http://aishack.in/tutorials/sift-scale-invariant-feature-transform-log-approximation/

[5] Wikipedia contributors. (2019, November 14). Taxicab geometry. In *Wikipedia, The Free Encyclopedia*. Retrieved 07:47, December 14, 2019, from https://en.wikipedia.org/w/index.php?title=Taxicab_geometry&oldid=926081770

[6] Bay, H., Ess, A., Tuytelaars, T. and Van Gool, L. (2008). Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3), pp.346-359.

[7] Affine covariant features, 2007. http://www.robots.ox.ac.uk/~vgg/researchaffine

[8] Wikipedia contributors. (2019, September 28). Circular shift. In *Wikipedia, The Free Encyclopedia*. Retrieved 07:49, December 14, 2019, from https://en.wikipedia.org/w/index.php?title=Circular_shift&oldid=918394678