Available online to ejournal.pnc.ac.id

Journal of Innovation Information Technology and Application

Journal Page is available to https://ejournal.pnc.ac.id/index.php/jinita

Comparative Analysis of Keypoint Detection Performance in SIFT Implementations on Small-Scale Image Datasets

Arif Rahman [1], Suprihatin [2], Imam Riadi [3], Tawar [4], Furizal [5]

[1,2,3,4] *Department of Information System, Universitas Ahmad Dahlan, Yogyakarta, Indonesia*

[5] *Master Program of Informatics, Universitas Ahmad Dahlan, Yogyakarta, Indonesia*

email: [1] arif.rahman@is.uad.ac.id, [2] suprihatin@uad.ac.id, [3] imam.riadi@is.uad.ac.id, [4] tawar@is.uad.ac.id, [5] 2108048014@webmail.uad.ac.id

ARTICLE INFO

Article history:
Received 18 July 2024
Revised 27 November 2024
Accepted 09 December 2024
Available online 30 December 2024

Keywords:
Image
Keypoint
OpenSIFT
SIFT
VLFeat.

IEEE style in citing this article:
A. Rahman , Suprihatin , I. Riadi , Tawar , and Furizal , "Comparative Analysis of Keypoint Detection Performance in SIFT Implementations on Small-Scale Image Datasets," Journal of Innovation Information Technology and Application (JINITA), vol. 6, no. 2, pp. 92–100, Dec. 2024.

ABSTRACT

Scale-invariant feature transform (SIFT) is widely used as an image local feature extraction method because of its invariance to rotation, scale, and illumination change. SIFT has been implemented in different program libraries. However, studies that analyze the performance of SIFT implementations have not been conducted. This study examines the keypoint extraction of three well-known SIFT libraries, i.e., David Lowe's implementation, OpenSIFT, and vlSIFT in vlfeat. Performance analysis was conducted on multiclass small-scale image datasets to capture the sensitivity of keypoint detection. Although libraries are based on the same algorithm, their performance differs slightly. Regarding execution time and the average number of keypoints detected in each image, vlSIFT outperforms David Lowe's library and OpenSIFT.

1. INTRODUCTION

Feature extraction is critical in image processing and computer vision, enabling machines to accurately recognize and categorize visual data. Scale-Invariant Feature Transform (SIFT), introduced by David Lowe, is one of the most widely used in image processing and computer vision, among other algorithms. This is due to its robustness and effectiveness in extracting features invariant to scale, rotation, and illumination changes [1], [2]. SIFT descriptors enable consistent feature matching across various image transformations, particularly useful in fields like object recognition, image stitching, and image retrieval [3]. SIFT's capability to identify and match distinct image features has made it a fundamental tool in applications like object recognition, medical imaging, augmented reality, and autonomous navigation.

Various studies have recently utilized the SIFT algorithm in diverse applications. For example, researchers evaluated SIFT's robustness in detecting image deformation in woven motifs, achieving 100% matching accuracy under specific settings [4]. In another study, the SIFT algorithm demonstrated its effectiveness in copy-paste forgery detection, proving capable of identifying altered regions in tampered images from challenging datasets [5]. Moreover, SIFT has been successfully applied in palm pattern recognition, providing faster and more accurate results than traditional methods [6]. These examples

**) Corresponding Author : arif.rahman@is.uad.ac.id*

highlight SIFT's versatility and underscore the importance of selecting an optimal implementation for different applications.

Performance analysis of the SIFT algorithm has been studied in previous research. In [7], the focus is on comparing the open-source SIFT library with David Lowe's original SIFT executable regarding runtime and accuracy in matching keypoints and computing image transforms. The analysis reveals that the library is competitive with the original implementation while providing additional flexibility for developers. The study in [8]explores the SIFT anatomy, including algorithmic components, theoretical underpinnings, and practical implementations for image feature detection and matching. SIFT's ability to handle lighting, viewpoint, scale, and rotation variations ensures consistent performance across diverse conditions. However, SIFT has limitations that impact its practical use. The algorithm is computationally intensive, making it less suitable for real-time or large-scale applications where speed is crucial. Devices with limited processing power, such as mobile or embedded systems, may struggle to implement SIFT effectively. In [9] theoretical analysis of the SIFT algorithm with practical implementation insights was explored to understand its functionality and applications comprehensively. Experiments with feature extraction and matching under varying conditions validate SIFT's effectiveness, bridging the gap between theory and application while highlighting its reliability and adaptability for real-world use.

Despite SIFT's robustness and effectiveness, different algorithm implementations may exhibit variations in performance due to differences in optimization and library design. These variations can impact critical metrics, such as execution time and the number of keypoints detected, which are essential for applications requiring real-time processing or sensitivity to fine-grained image features. Therefore, a direct comparison of these libraries is necessary to guide professionals in selecting the implementation best suited to their performance requirements. However, a comprehensive comparison of SIFT implementations' computational efficiency and keypoint detection sensitivity still needs to be explored, especially on small-scale image datasets.

This study focuses on the comparative performance of three major SIFT libraries: Lowe's original SIFT implementation, OpenSIFT, and vlSIFT (a part of the VLFeat library). Although each library is based on the same foundational algorithm, differences in performance metrics, such as keypoint detection speed and sensitivity, could impact their suitability for various applications. This research aims to identify the most efficient library for image keypoint extraction by examining these metrics using small-scale datasets. The purpose of choosing datasets with small image sizes is to capture the sensitivity of each library's keypoint detection. Our study contributes to the field by offering a systematic performance analysis of these libraries. This analysis will aid practitioners and researchers in selecting the most appropriate SIFT implementation for their needs.

2. SCALE-INVARIANT FEATURE TRANSFORM (SIFT)

2.1 SIFT Keypoint Descriptor

SIFT extracts robust local descriptors from an image against transformations like viewpoint changes, noise, and contrast variations. The algorithm has two primary operations: detecting keypoints and extracting descriptors. It proceeds in four stages to identify and describe keypoints [8]. The first step is identifying keypoints as local maxima and minima in a Difference-of-Gaussian (DoG) pyramid, which approximates the scale-space's second-order derivatives [10], [11]. Each octave in the DoG pyramid is built recursively, starting with the original image for the first octave.

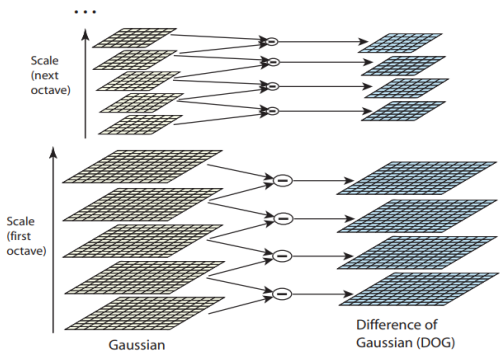

Figure 1. DoG scale-space

DOI: doi.org/10.35970/jinita.v6i2.2399

Next, a set of DoG images D_k is calculated as the difference of two consecutive Gaussian-blurred images, formally $D_k = G_{k+1} - G_k$. Within an octave $D(x, y, \sigma)$, where (x, y) denotes spatial position and σ is the scale, local extrema are detected by comparing a pixel value with its 26 neighbor pixels in $D(x, y, \sigma)$. These are considered as keypoint candidates. Figure 2 illustrates this process.

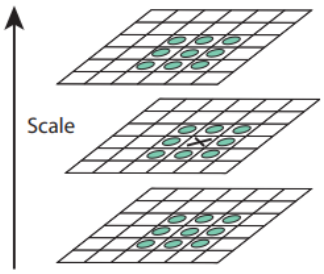

Figure. 2. Scale-space extrema detect

After keypoint candidates had been identified in the previous step, their locations were refined to sub-pixel accuracy by interpolating in scale space. Keypoints with low contrast or high edge response are discarded due to instability. First, keypoint candidates' positions (x_0 , y_0, σ_0) are refined to sub-pixel precision (x', y', σ'). A quadratic function $D(x, y, \sigma)$ interpolates nearby data points, and its maximum is found by setting its derivative to zero. This produces a 3×3 linear equation system, solvable by singular value decomposition. If (x', y', σ') is closer to a different integer position than (x_0, y_0, σ_0), the process repeats from the new position. Next, unstable keypoint candidates are discarded. Low-contrast candidates are identified by an absolute function value $|D(x', y', \sigma')|$ below a threshold. Edge candidates are detected using H, an approximation of the Hessian 2×2 matrix of $D(x, y, \sigma)$ at the refined position. Candidates with low cornerness, calculated from H, are discarded.

Keypoints from the filtering are assigned canonical orientations based on dominant local scale-space gradients. After this, each keypoint's descriptor is computed relative to its location, scale, and orientation to ensure invariance to these transformations. Each keypoint candidate's orientation φ is determined by first identifying the nearest Gaussian-blurred image L in scale. From L, the gradient magnitude and orientation are calculated pixel-wise. A weighted orientation histogram with 36 bins is constructed from the gradient orientations of sample points around the keypoint, weighted by gradient magnitude. The keypoint orientation φ is identified from the peak in the histogram and refined by parabolic interpolation. If other significant peaks (within 80% of the highest peak) exist, additional keypoints with the same position and scale but different orientations are created.

Next, the keypoint descriptor is calculated. Image gradient magnitudes and orientations are sampled in a 16×16 region around the keypoint. To ensure orientation invariance, the coordinates of the descriptor and the gradient orientations are rotated relative to φ. A Gaussian weighting function is applied, and weighted orientation histograms over 4×4 sub-regions are created. Trilinear interpolation distributes gradient values into adjacent histogram bins, resulting in a 128-bin descriptor aggregated from the 8-bin histograms of all 16 sub-regions. The descriptor is then normalized to unit length for robustness against illumination changes.

2.2 SIFT Implementations

SIFT keypoint detector implementation from David Lowe's paper is in the form of compiled binaries that can run under Linux or Windows. The computer program uses images in PGM format as the input. Output of the program are keypoints and all information needed for keypoints matching to a file in an ASCII format. A Matlab program and C code are provided that can read the keypoints and match them between images. The program and examples can be found at [12].

The open-source SIFT library [13] is written in C and utilizes the OpenCV library. The SIFT library's API uses OpenCV data types for images and matrices, making integrating SIFT functions into existing OpenCV-based vision code easy. All internal operations in the SIFT library are performed using OpenCV functions. The library has four main components, each represented by a different header file. The primary component includes functions for detecting SIFT keypoint that include two keypoint detection functions: one using default parameter settings from Lowe's paper and another allowing custom parameters.

The library implements a method for keypoint matching using a kd-tree and an approximate nearest-neighbor search. The library also includes functions using the RANSAC algorithm to compute

DOI: doi.org/10.35970/jinita.v6i2.2399

image transforms from feature matches. These functions are flexible, allowing developers to implement custom transform functions.

VLFeat open-source library implements popular computer vision algorithms specializing in image understanding and local feature extraction and matching[14]. VLFeat is written in C with interfaces in MATLAB. vlSIFT includes a feature detector and a feature descriptor. The detector extracts a collection of frames or keypoints from an image. These are oriented disks attached to blob-like structures of the image. The frames are covariant in tracking image translations, rotations, and scalings. The effect of such transformations can then be undone by canonization, i.e., warping the frames (with their appearance) to a canonical disk. The descriptor is a coarse statistic of the gradients of the frame appearance. Due to canonization, the descriptors are invariant to translations, rotations, and scalings of the image. Due to their statistical nature, they are also robust to other and not modeled noise sources.

3. METHOD

The performance analysis of image keypoint extraction using various SIFT implementations was carried out through a structured and replicable process. First, program setup and parameter configuration were conducted to ensure fair conditions for all tested SIFT implementations. Each implementation was integrated into the experimental environment, i.e., C programming language and parameters critical to SIFT performance were standardized. These included the number of octaves, scales per octave, and thresholds for keypoint detection. Additionally, input images were preprocessed to a consistent resolution to eliminate variability unrelated to the algorithms. This setup ensures that the performance differences observed stem solely from the algorithm implementations.

Then, the keypoint extraction was conducted on two small-scale image datasets, CIFAR-10 [15] and CINIC-10 [16]. These datasets were selected to challenge the SIFT implementations under varying complexity, scale, and content diversity conditions. Each image was processed through the SIFT implementations to extract keypoints, with two primary performance metrics logged during the process: runtime execution and the number of keypoints detected. This step ensures the analysis accounts for computational efficiency and feature extraction robustness.

The logged data formed the basis for analysis and comparison among the different SIFT implementations. The average runtime and the average number of keypoints detected were computed for each implementation across the datasets. Comparative statistical analysis was performed to evaluate the trade-offs between runtime efficiency and detected keypoint density. These metrics ensure a balanced assessment of algorithmic performance, capturing both computational overhead and detection capabilities. Visual representations in graphs and tables highlighted the distinctions between the implementations. As illustrated in Figure 3, the entire process captures the key stages: setup, keypoint extraction, data logging, and comparative analysis.

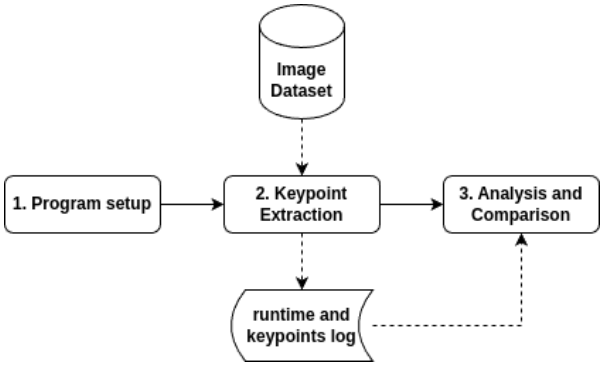

Figure 3. Performance analysis process

3.1. Program Setup

Each SIFT implementation was configured with identical experimental settings to ensure a fair and unbiased comparison. The input images were uniformly processed as 8-bit grayscale images, simplifying computational requirements by focusing exclusively on pixel intensity and discarding color information. Keypoint descriptors were set to a fixed size of 128 dimensions, consistent with the original

SIFT design, ensuring a robust representation of image features. Keypoint locations were defined using four specific parameters: row, column, scale, and orientation. The orientation parameter, ranging from $[-\pi,\pi]$ radians, is crucial in achieving rotation invariance, enabling the algorithm to match keypoints regardless of image rotation reliably.

The experiments were conducted on a system equipped with an Intel Core i3 processor and 4GB of memory, reflecting typical computational resources available in standard environments. This hardware choice ensured consistency and generalizability, allowing the results to be applied to real-world scenarios where resources may be limited. A uniform computational setup was used in this study to minimize hardware-induced variability and ensure the observed differences in performance could be attributed merely to the SIFT implementations being tested.

Keypoint extraction was performed on 1,000 randomly selected images for each dataset. This approach ensured the dataset was representative and avoided biases associated with specific image content. To enhance the reliability and robustness of the results, the process was repeated 10 times for each dataset, effectively reducing the influence of random variations or outliers.

3.2 Image Datasets

The SIFT performance analysis experiments were conducted on small-scale image datasets CIFAR-10 and CINIC-10. The purpose of choosing a dataset with a relatively small image size is to capture the sensitivity of each library's keypoint detection. The CIFAR-10 dataset (Canadian Institute for Advanced Research) is a collection of images commonly used for machine learning and computer vision algorithm research. The CIFAR-10 dataset contains 60,000 color images with 32x32 pixel sizes organized into ten classes. Each class has 6,000 images representing airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The classes are completely mutually exclusive. There is no overlap between automobiles and truck classes. The automobile class includes sedans, SUVs, and things of that sort, and the truck class includes only big trucks. Figure 4 shows 10 random examples of images from each class in the CIFAR-10 dataset.

The CINIC-10 benchmarking dataset was presented to address the inadequacies of previous benchmarking datasets. It is a CIFAR-10 extension that includes downsampled ImageNet images. CINIC-10 has 270,000 images, which is 4.5 times larger than CIFAR. Since the images are the same size as in CIFAR, CINIC-10 can be used as a drop-in replacement for CIFAR-10. It has train, validation, and test splits of the same size. In some experimental situations, more than one training dataset may be necessary. Nonetheless, equal dataset split sizes allow for a fair assessment of generalization performance. Each subset has ten identical classes to CIFAR-10 classes (90,000 images). Each class and subset has 9,000 images. CINIC-10 contains 1.8 times more training samples than CIFAR-10 when using the specified data split (an equal three-way split). CINIC-10 is intended to be interchangeable with CIFAR-10. Table 1 shows the number of images in CIFAR-10 and CINIC-10 datasets for each class.

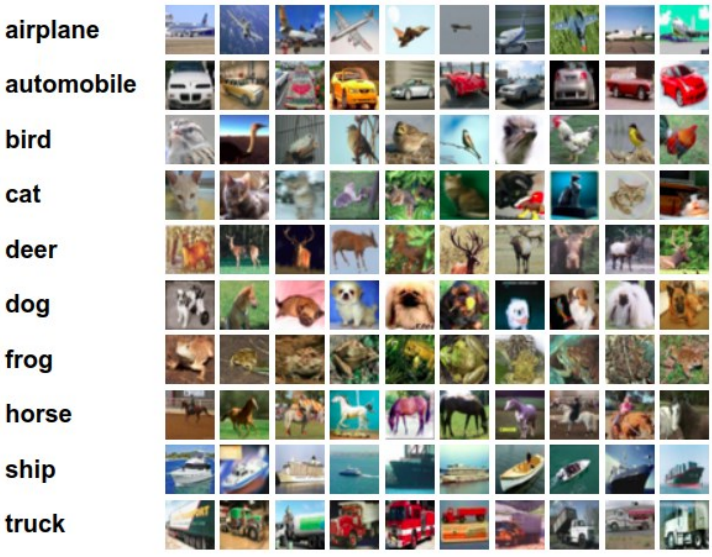

Figure 4. CIFAR-10 dataset classes with 10 random images from each class.

Table 1. Number of Images in CIFAR-10 and CINIC-10 Datasets

No	Class	Number of images	
		CIFAR-10	CINIC-10
1	airplane	6,000	27,000
2	automobile	6,000	27,000
3	bird	6,000	27,000
4	cat	6,000	27,000
5	deer	6,000	27,000
6	dog	6,000	27,000
7	frog	6,000	27,000
8	horse	6,000	27,000
9	ship	6,000	27,000
10	truck	6,000	27,000
	Total	60,000	270,000

3.3 Keypoint Extraction

The keypoint extraction process in this study was performed using three SIFT implementations: David Lowe's original implementation, OpenSIFT, and vlSIFT from the VLFeat library. Though each library is based on the same underlying SIFT algorithm, its design and optimization vary, offering distinct performance characteristics. The experiment utilized binary implementations of these libraries, called directly from a C program. This ensured a consistent execution environment and standardized the process, enabling an accurate comparison of their keypoint extraction capabilities. David Lowe's original SIFT implementation, available as a compiled binary, accepts images in PGM format and outputs keypoints and descriptors in an ASCII file. This implementation served as the benchmark for comparison. OpenSIFT, an open-source library written in C and integrated with OpenCV, was utilized for its flexibility and ease of integration into computer vision pipelines. vlSIFT, part of the VLFeat library, provides an implementation of SIFT, with support for keypoint detection and description invariant to transformations such as scale and rotation. Like OpenSIFT, vlSIFT is implemented in C and supports MATLAB interfaces. During the experiments, the keypoint extraction process involved processing grayscale 8-bit images, ensuring consistency in input data across all libraries. Parameters, such as the number of octaves, scales per octave, and detection thresholds, were uniformly configured to provide a fair comparison. For each image, the program invoked the binary implementation of the respective SIFT library to perform keypoint extraction. The execution time for each extraction and the total number of keypoints detected were recorded in a CSV file for analysis. This structured approach facilitated a detailed evaluation of the three SIFT implementations' computational efficiency and detection sensitivity, providing insights into their suitability for various applications.

4. RESULTS AND DISCUSSION

This section analyzes keypoint detection experiments using various SIFT implementation libraries, focusing on their computational efficiency based on average execution time. The comparison highlights each implementation's practicality for time-sensitive applications. Table 2 shows the average execution time of keypoint detection for each image using the default settings provided by each SIFT implementation library. The bold font denotes the lowest execution time.

Table 2. Average execution time (in seconds) of SIFT implementations

Library	Dataset	
	CIFAR-10	CINIC-10
Lowe's	0.095	0.214
OpenSIFT	0.120	0.192
vlSIFT	0.090	0.146

For the CIFAR-10 dataset, vlSIFT exhibits the fastest execution time at 0.090 seconds, closely followed by Lowe's at 0.095 seconds, while OpenSIFT is slightly slower at 0.120 seconds. When applied to the CINIC-10 dataset, the execution times increase for all implementations. Lowe's implementation shows the highest execution time at 0.214 seconds, while OpenSIFT improves relative performance with an execution time of 0.192 seconds. vlSIFT maintains its lead with the lowest execution time of 0.146 seconds. This consistent performance of vlSIFT across both datasets indicates its higher efficiency in handling larger and potentially more complex image data compared to Lowe's and OpenSIFT implementations.

Regarding keypoint detection performance, experiments are conducted by calculating the average number of keypoints detected per image, which are then grouped for each class. This method helps in assessing the sensitivity of each SIFT implementation. The underlying assumption is that more detected keypoints indicate better library sensitivity to image features, suggesting a more thorough extraction of distinctive points from the images. In other words, the more keypoints a library detects, the better its performance in identifying and representing unique aspects of the images.

However, there are instances where no keypoints are detected in specific images, which can significantly impact the effectiveness of image retrieval processes. This issue was notably observed in the OpenSIFT experiments, where two images failed to yield any keypoints. The absence of keypoints poses a significant problem because no data is available to compare with other images in a database, undermining the retrieval and matching process. When an image has no detected keypoints, it cannot be effectively matched or compared, leading to gaps in the overall image retrieval system and potentially reducing the accuracy and reliability of the implementation.

The average number of detected keypoints for each class by each library is expressed as a graph in Figure 5 for the CIFAR-10 dataset and Figure 6 for the CINIC-10 dataset. Figure 5 shows that vlSIFT consistently detects the highest number of keypoints in every image class, with averages generally staying above 15 and peaking above 20 for deer and ship. The Lowe method displays moderate keypoint detection, typically hovering around 15, with notable consistency across all image classes. OpenSIFT shows the lowest average keypoints detected, often below 15, with significant dips in classes like automobile and cat. While all three methods exhibit similar trends for specific classes, such as airplanes and trucks, where the detection rates are closely aligned, vlSIFT consistently outperforms Lowe and OpeSIFT. This trend is particularly pronounced in classes like deer and ship, where vlSIFT detection rates surpass 20 keypoints on average. The data suggests that vlSIFT is more effective in identifying keypoints across diverse image classes, whereas Lowe and OpenSIFT have relatively stable but lower performance.

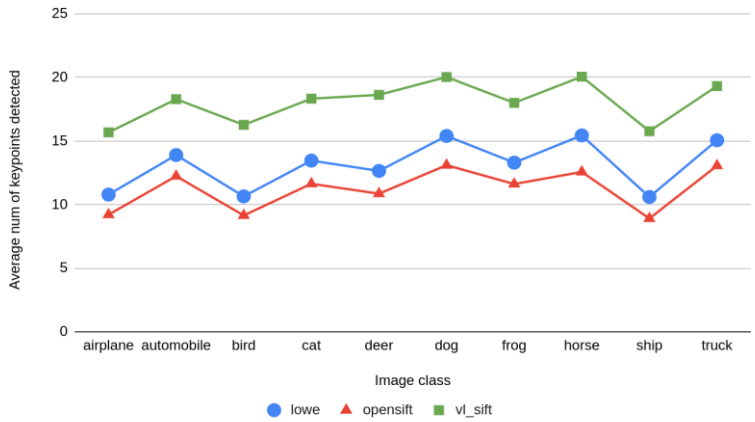

Figure 5. Average number of detected image keypoints for each class in the CIFAR-10 dataset.

DOI: doi.org/10.35970/jinita.v6i2.2399

Results on the CINIC-10 dataset, as described in Figure 6, show that the vlSIFT method consistently detects the highest number of keypoints, with averages mostly between 14 and 16 keypoints across all image classes. However, the Lowe method typically detects around 10 keypoints on average, showing relatively stable performance but generally lower than vlSIFT. OpenSIFT has the lowest average keypoints detected, fluctuating around 8 to 10, with noticeable dips in certain classes like automobile and cat.

Although all three methods vary in performance across different image classes, vlSIFT consistently outperforms the other two methods. The data indicates that while Lowe and Opensift have some overlap in their detection rates, vlSIFT provides the highest keypoint detection in all classes. For example, in the bird and horse classes, vlSIFT maintains higher detection rates than Lowe and OpenSIFT. In summary, the average number of detected image keypoints for all classes in both datasets is presented in Table 3, and the bold font denotes the highest number.

Table 3. Average image keypoints detected in the CIFAR-10 and CINIC-10 dataset

Library	Dataset	
	CIFAR-10	CINIC-10
Lowe's	13.13	11.36
OpenSIFT	11.24	9.71
vlSIFT	18.04	14.81

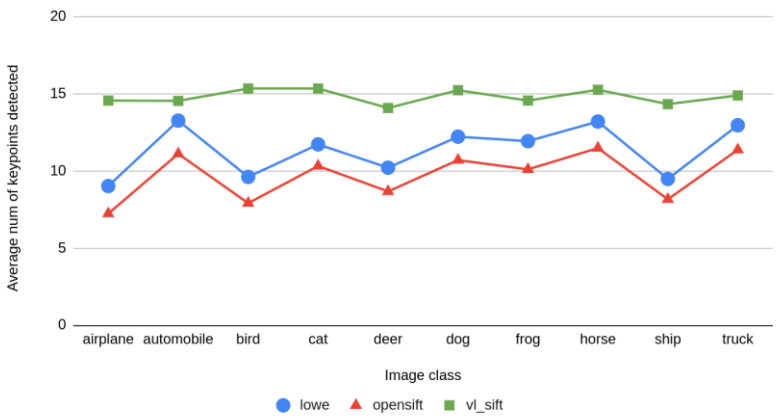

Figure 6. Average number of detected image keypoints for each class in the CINIC-10 dataset.

5. CONCLUSION

In this paper, three widely used implementations of the Scale-Invariant Feature Transform (SIFT) algorithm were evaluated using selected small-scale image datasets. While all three libraries (David Lowe's original implementation, OpenSIFT, and vlSIFT from the VLFeat library) are based on the same theoretical framework, they exhibited varying performance. Image keypoint extraction experiments revealed that vlSIFT consistently outperformed the other implementations regarding execution time and the average number of keypoints detected per image. Its ability to process images faster and detect more keypoints on average makes vlSIFT a more efficient and practical choice for applications requiring robust feature detection, particularly in scenarios with constrained computational resources.

The implications of these findings extend beyond the direct comparison of SIFT implementations. For the broader research field, our results highlight the importance of implementation choices when deploying feature detection algorithms in practical applications. The superior performance of vlSIFT suggests its potential for use in computationally constrained environments, such as embedded systems, or applications requiring rapid processing, such as real-time object detection, autonomous navigation, and augmented reality systems. Furthermore, its ability to detect more keypoints on average enhances its utility in tasks requiring high feature sensitivity, such as fine-grained image matching and medical image analysis.

However, the study has limitations that require further discussion. The experiments were conducted on a single hardware configuration, ensuring a controlled and consistent environment for comparison. However, this setup may not comprehensively represent the performance of these libraries when executed on more advanced or specialized hardware, such as GPUs or high-performance computing servers. Future studies could explore the scalability and efficiency of these implementations across a range of hardware configurations to better understand their suitability for diverse application environments.

ACKNOWLEDGEMENTS

The authors thank the Information System Department, Universitas Ahmad Dahlan Yogyakarta for the support and facilities for this research.

REFERENCES

[1] D. G. Lowe, "Object recognition from local scale-invariant features," *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pp. 1150–1157 vol.2, 1999, doi: 10.1109/ICCV.1999.790410.
[2] D. G. Lowe, "Distinctive image features from scale invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–11020042, 2004, doi: 10.1023/B:VISI.0000029664.99615.94.
[3] Z. Wang, Y. Liu, J. Zhang, C. Fan, and H. Zhang, "Interference image registration combined by enhanced scale-invariant feature transform characteristics and correlation coefficient," *J Appl Remote Sens*, vol. 16, no. 2, p. 26508, 2022.
[4] S. Joseph, I. Hipiny, H. Ujir, S. F. S. Juan, and J. L. Minoi, "Performance evaluation of SIFT against common image deformations on iban plaited mat motif images," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, no. 3, pp. 1470–1477, Sep. 2021, doi: 10.11591/ijeecs.v23.i3.pp1470-1477.
[5] P. Selvaraj and M. Karuppiah, "Enhanced copy-paste forgery detection in digital images using scale-invariant feature transform," *IET Image Process*, vol. 14, no. 3, pp. 462–471, Feb. 2020, doi: 10.1049/iet-ipr.2019.0842.
[6] M. Kasiselvanathan, V. Sangeetha, and A. Kalaiselvi, "Palm pattern recognition using scale invariant feature transform," *International Journal of Intelligence and Sustainable Computing*, vol. 1, no. 1, pp. 44–52, 2020.
[7] R. Hess, "An open-source siftlibrary," in *Proceedings of the 18th ACM international conference on Multimedia*, 2010, pp. 1493–1496.
[8] I. Rey Otero and M. Delbracio, "Anatomy of the SIFT Method," *Image Processing On Line*, vol. 4, pp. 370–396, Dec. 2014, doi: 10.5201/ipol.2014.82.
[9] M. Y. Yin, F. Guan, P. Ding, and Z. F. Liu, "Implementation of image matching algorithm based on SIFT features," in *Applied Mechanics and Materials*, Trans Tech Publications Ltd, 2014, pp. 3181–3184. doi: 10.4028/www.scientific.net/AMM.602-605.3181.
[10] E. Yang, F. Chen, M. Wang, H. Cheng, and R. Liu, "Local Property of Depth Information in 3D Images and Its Application in Feature Matching," *Mathematics*, vol. 11, no. 5, Mar. 2023, doi: 10.3390/math11051154.
[11] Y. Gu, H. Wang, Y. Bie, R. Yang, and Y. Li, "Research on Image Registration of Different Wavebands Based on SIFT Algorithm," in *2022 3rd China International SAR Symposium (CISS)*, 2022, pp. 1–5.
[12] D. G. Lowe, "The SIFT Keypoint Detector," 2005 [Online]. Available: https://www.cs.ubc.ca/~lowe/keypoints/.
[13] R. Hess, "OpenSIFT: An Open-Source SIFT Library," 2012 [Online]. Available: https://robwhess.github.io/opensift/.
[14] A. Vedaldi and B. Fulkerson, "VLFeat - An open and portable library of computer vision algorithms," *Proceedings of the international conference on Multimedia - MM '10*, p. 1469, 2010, doi: 10.1145/1873951.1874249.
[15] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009.
[16] L. N. Darlow, E. J. Crowley, A. Antoniou, and A. J. Storkey, "CINIC-10 Is Not ImageNet or CIFAR-10." 2018 [Online]. Available: https://datashare.is.ed.ac.uk/