SIFT-ONN: SIFT Feature Detection Algorithm Employing ONNs for Edge Detection

Madeleine Abernot, Sylvain Gauthier, Théophile Gonos, Aida Todri

SIFT-ONN: SIFT Feature Detection Algorithm Employing ONNs for Edge Detection

*

Madeleine Abernot[1,a], Sylvain Gauthier[2,a], Théophile Gonos[2], and Aida Todri-Sanial[1,3,*]

[1]LIRMM, University of Montpellier, CNRS, Montpellier, France
[2]A.I.Mergence, Paris, France
[3]Electrical Engineering Department, Eindhoven Technical University, Eindhoven, Netherlands
[a]Both authors contributed equally to this research.
[*]Corresponding author: aida.todri@lirmm.fr

Abstract

Mobile robot navigation tasks can be applied in various domains, such as in space, underwater, and transportation industries, among others. In navigation, robots analyze their environment from sensors and navigate safely up to target points by avoiding obstacles. Numerous methods exist to perform each navigation task. In this work, we focus on robot localization based on feature extraction algorithms using images as sensory data. ORB, and SURF are state-of-the-art algorithms for feature-based robot localization thanks to their fast computation time, even if ORB lacks precision. SIFT is state-of-the-art for high precision feature detection but it is slow and not compatible with real-time robotic applications. Thus, in our work, we explore how to speed up SIFT algorithm for real-time robot localization by employing an unconventional computing paradigm with oscillatory neural networks (ONNs). We present a hybrid SIFT-ONN algorithm that replaces the computation of Difference of Gaussian in SIFT with ONNs by performing image edge detection. We report on SIFT-ONN algorithm performances, which are similar to the state-of-the-art ORB algorithm.

*This work was supported by the European Union's Horizon 2020 research and innovation program, EU H2020 NEURONN project under grant n. 871501 (www.neuronn.eu).

Keywords— O scillatory Neural Networks, Image Edge Detection, Feature Detection, SIFT

1 Introduction

Navigation is a complex and pervasive problem that allows autonomous robots to navigate safely in an environment without human interaction. Robot navigation is typically divided into various tasks, such as localization to place the robot in its environment from a reference point, obstacle avoidance to avoid obstacles in real-time, and mapping to create an environment map in real-time, among others [2, 17].

In state-of-the-art robot navigation, Simultaneous Localization And Mapping (SLAM) is the most widely applied algorithm [8, 18]. SLAM uses sensor data from the robot to estimate robot's current location. Sensor data are also used to create an environment map around the robot. SLAM can use various types of sensors. For example, some SLAM algorithms use cameras as sensors and use captured images to estimate robot position and environment. Another widely applied solution is to combine feature-based object tracking from images with SLAM algorithm [7, 11, 24]. Feature-based object tracking consists of detecting and describing features from two following image frames

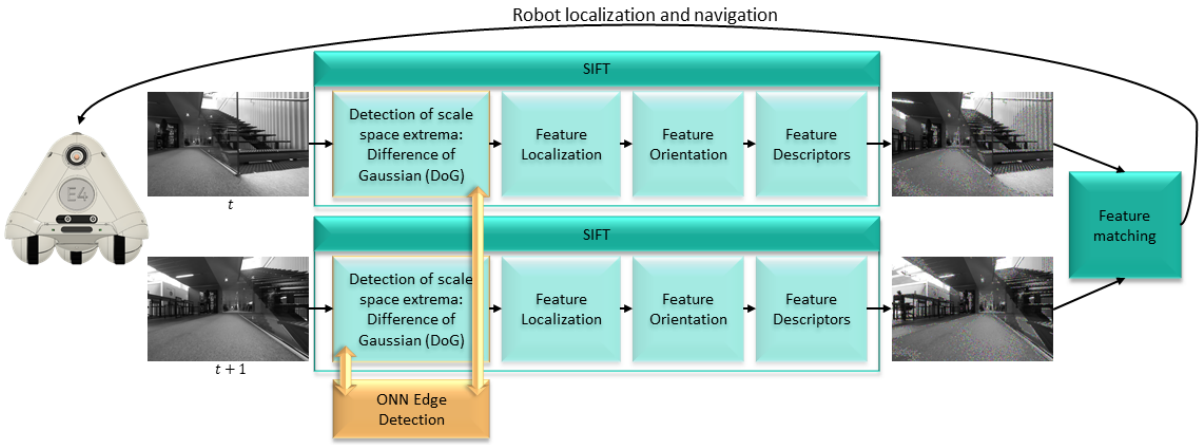


Figure 1: Robot movement computation based on SIFT feature detection and description algorithm and matching.

and matching the corresponding features to compute transformation between two frames, see Fig. 1. In state-of-the-art, SLAM algorithm is combined with ORB (Oriented FAST and Rotated BRIEF) or SURF (Speeded Up Robust Features) algorithms [11,14,24,27] because they perform fast feature detection and description algorithms.

Even if ORB gives good results, other feature detection and description algorithms can reach better precision but are too slow to compute for real-time navigation applications. For example, the Scale Invariant Feature Transform algorithm (SIFT) [22, 30] is one of the best-reported feature detection and description algorithms in terms of precision but has large computation latency. In this work, we explore a possible solution to speed up SIFT algorithm while maintaining reasonable precision compared to other algorithms. We choose to investigate SIFT as it is often used as a baseline in various feature detection algorithms. However, our proposed solution is general enough and applicable to other feature detection and description algorithms, such as SURF or others.

We propose and explore massive parallelism of oscillatory neural networks [15,19,25,28] to reduce the latency of SIFT algorithm. The first stage in SIFT algorithm is the computation of a Difference of Gaussian (DoG), in wich the output resembles an image edge detection algorithm. We propose to use ONNs to perform image edge detection [4,5] to replace the first SIFT DoG stage.

The main contributions of the paper can be summarized as i) the development of the hybrid SIFT-ONN algorithm by introducing ONNs for image edge detection at the first stage of the SIFT algorithm, and ii) benchmarking and evaluation of the hybrid SIFT-ONN solution compared to state-of-the-art feature detection and description algorithms in terms of precision and latency.

The paper is organized as follows. First, Section 2 gives an overview of SIFT feature detection and description algorithms. Then, Section 3 presents the ONN computing paradigm and its application to image edge detection. Next, in Section 4, we describe our hybrid SIFT-ONN algorithm by replacing first-stage SIFT DoG with ONN-based image edge detection. Also, in Section 4 we describe the methods to evaluate SIFT-ONN with state-of-the-art algorithms. Section 5 presents the results and benchmarking of the SIFT-ONN. And finally, Section 6 discusses the advantages, limitations, and prospective improvements and future work on SIFT-ONN.

2 Feature detection and description with SIFT

Feature detection and description algorithms are used in various applications, such as navigation with SLAM. Here, we present state-of-the-art approaches on feature detection and description algo-

rithms for navigation tasks. Also, we describe the different computation stages in the SIFT algorithm.

2.1 State-of-the-art

Over the last twenty years, various feature detection and description algorithms have been proposed. Most feature detection algorithms are based on edge or corner detection algorithms. Mainly, their differences are in the mathematical approaches to detect features or edges and corners, and in the methods to define the descriptors of the features used to match them. Important parameters are scale and rotation invariance, which are necessary to detect and describe features for matching two images with different points of view. SIFT is one of the first feature detection and description algorithms introduced with scale and rotation invariance [22], which obtains high precision while using large computing resources but also results in long computation time. Since SIFT, other solutions have been proposed to reduce computation latency depending on the target applications [6, 9, 21, 27].

For example, in mobile robotics, SIFT has been successively replaced by SURF [9] and ORB [27], which are faster, therefore, more suitable for real-time constrained applications. In this work, we investigate applying ONN image edge detection to replace SIFT DoG in order to reduce SIFT computation latency.

2.2 Scale Invariant Feature Transform (SIFT) Algorithm

SIFT stands for Scale Invariant Feature Transform [22]. SIFT is a feature detection and description algorithm, which is scale and rotation invariant. It aims to correctly detect and describe features to be able to match them between two images, even if they are rotated, with different sizes and scales.

The SIFT feature detection and description algorithm can be divided into four main stages, as shown in Fig. 1. The first stage detects extrema in various scale spaces. More precisely, it uses DoG to detect edges in images smoothed by gaussian blur filters with different smoothing scales. For each pixel at position (x, y) in the image, it computes:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \qquad (1)$$

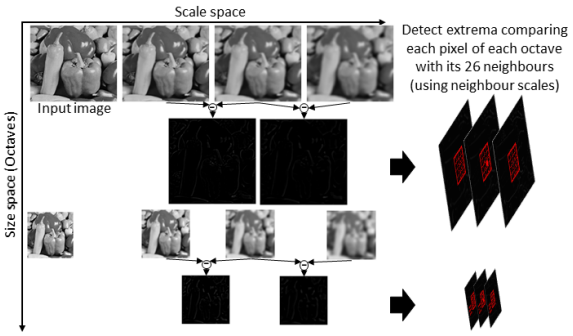


Figure 2: Detection of scale space extrema based on Difference of Gaussians.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \qquad (2)$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \qquad (3)$$

where D is the DoG output, L is the output image after the Gaussian blur is applied, G is the Gaussian blur, and σ is the smoothing scale, see Fig. 2. Then, extrema are extracted from the DoG outputs. For each octave, each pixel of a scale space is compared with its 8 neighboring pixels from the same scale space, and with its 9 neighboring pixels from lower and upper scale spaces. Extrema are pixels with the highest or smallest values compared to all 26 neighbors, see Fig. 2.

The second stage of SIFT is to select key points from the extracted extrema. Low extrema are filtered to obtain only the strong key points that are easily reproducible in different images. Extrema on edges are also filtered because edges are sensitive to noise and DoG is highly sensitive to edges. After localizing the key points, their orientation is computed. SIFT computes for each key point the magnitude and orientation on a 16x16 window. Then, it groups pixels in 4x4 windows to create orientation histograms. In this case, orientation and magnitude in each 4x4 window are combined. Then, the main orientation from each 4x4 histogram is extracted. Fig. 3 shows the feature orientation process with histograms and orientation definitions. The final stage creates the descriptors, which are vectors containing main orientations with magnitude around each key point, making descriptors scale and rotation invariant.

Once SIFT is applied to two images, descriptors from both images are matched. Matched descrip-

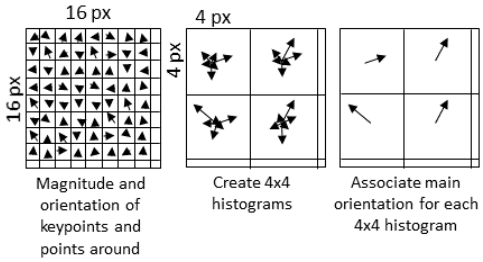


Figure 3: Feature orientation.

tors are used to determine the transformation between the two images. For example, in the case of two following images from a moving robot, the transformation can be associated with the movement of the robot. Note, it is possible to filter the number of matches used to compute the transformation to impact the obtained transformation value.

3 ONN for image edge detection

ONN is a promising neuromorphic computing paradigm for edge artificial intelligence applications due to its energy efficient and massive parallel computations [16].

3.1 ONN computing paradigm

ONN is a brain-inspired paradigm where the computations are performed on coupled oscillators [15]. In ONNs, each neuron is an oscillator coupled with synapses, which can be implemented either in analog or digital. ONNs encode information in the phase relationship among oscillators. For example, with binary information, a logic $'0'$ is represented with an oscillator phase of 0^o, and a logic $'1'$ is represented with an oscillator phase of 180^o. Phase-based computing allows to lower voltage amplitude, thus reducing power consumption. ONNs use the dynamics of coupled oscillators for inference. At first, oscillator phases are initialized from input information. Then, phases evolve in a parallel fashion depending on the coupling strength among oscillators. Finally, oscillators stabilize and their phases are read to construct the output information. Overall, both low power and parallel computations make

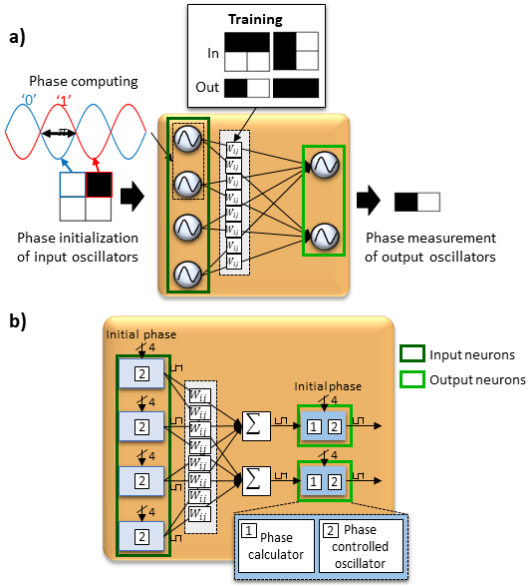


Figure 4: a) Feedforward ONN architecture for classification. b) Digital feedforward ONN implementation on FPGA.

ONN attractive for edge computing.

In literature, ONNs are typically designed as a fully-connected recurrent architecture, and trained with unsupervised learning rules, like Hebbian, to solve pattern recognition or auto-associative memory tasks, as in Hopfield neural networks [20, 23]. Recently, authors in [4, 5], reported on a two-layer ONN architecture with feedforward connectivity that can efficiently solve image edge detection tasks. Fig. 4.a) illustrates the ONN computing paradigm with a two-layer feedforward architecture configured for classification.

In this work, we utilize the feedforward digital ONN implemented on FPGA as in [3–5], see Fig. 4.b). Each oscillator is a 16-stage phase-controlled oscillator allowing phases between 0^o and 360^o with phase precision of 22.5^o. Oscillators from the output layer also include a phase calculator, which integrates post-synaptic phases from the input layer. Synapses are 5-bits signed registers allowing integer weights between -15 and 15.

3.2 ONN edge detection

Here, we investigate in incorporating the feedforward ONN architecture configured for image edge

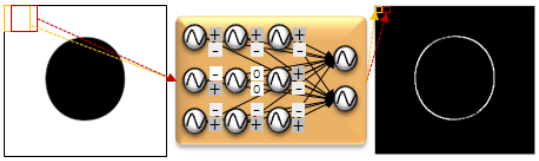


Figure 5: Feedforward ONN configured for image edge detection applied to a black and white image.

detection to be part of the SIFT algorithm. In literature, image edge detection uses convolutional filters, typically 3x3, 5x5, or 7x7, to scan an image and detect edges. Sobel [29] and Canny [12] are the two main convolutional-based edge detection algorithms.

ONNs for image edge detection are introduced in [4] using a two-layer ONN connected with bidirectional connections that perform hetero-associative memory tasks. In [4], authors use Sobel convolutional filters as training patterns to compute synaptic couplings between the two ONN layers. However, such architecture had some missing edges, thus in [5], authors proposed a customization of the coupling weights to improve the precision of the two-layer bidirectional architecture. Nevertheless, the two-layer bidirectional ONN architecture has important latency limitations, and uses a large amount of digital resources, limiting the number of possible ONNs that can be implemented in parallel in order to reduce latency. Thus, in [5], authors introduced instead an architecture with a two-layer feedforward ONN. Authors reported on feedforward ONN for image edge detection with various sizes (3x3, 5x5, and 7x7) to scan the image, see Fig. 5. Using a larger size ONN can increase the stride and help reduce the image scanning latency. In [5], it was shown that feedforward ONN for image edge detection has similar precision as the state-of-the-art Sobel algorithm, and it can achieve better latency by using multiple ONNs in parallel. Table 1 highlights the main characteristics of the feedforward digital ONN. Note, the ONN configured for image edge detection from [5] only outputs binary phase information $0^o, 180^o$.

In the SIFT algorithm, the DoG computation can be approximated to an edge detection computation. Thus, in this work, we aim to replace the SIFT DoG stage with 3x3, 5x5, or 7x7 feedforward ONN for image edge detection– creating a hybrid SIFT-ONN algorithm.

Table 1: Characteristics of edge detection feedforward ONN from [5].

ONN	3x3	5x5	7x7
Latency	1.15 μs	1.15 μs	1.15 μs
LUTs	211	302	457
Flip-Flops	277	437	597

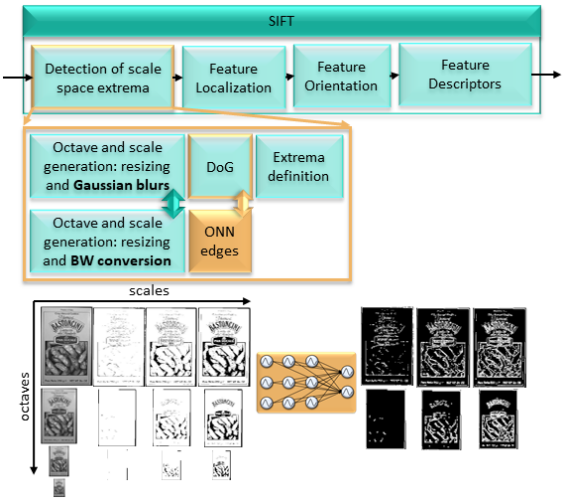
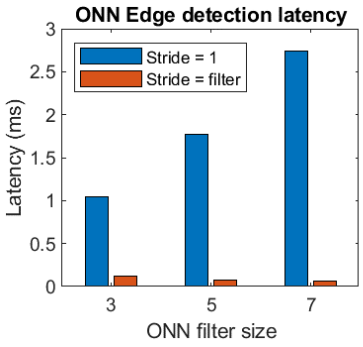


Figure 6: SIFT-ONN algorithm process.

4 SIFT-ONN adaptation

Here, we replace the DoG with the feedforward ONN for image edge detection.

In the first stage of SIFT, a first pyramid is generated from the input image by resizing it to create various octaves. The first octave is composed of the input image upsampled by two, and the following octaves are generated by subsampling by two of their previous octave until the size is too small to make a Euclidean division by two. Then, a second pyramid is created by applying various gaussian blurs on the image from each octave to create different scales of the same image, see Fig. 2. Next, the DoG outputs are computed for each octave between neighboured gaussian blurs to obtain images with highlighted edges. In this work, the SIFT-ONN keeps the generation of the first pyramid, creating octaves by subsampling the input image. Then, the gaussian blurs are replaced with image binarization



ONN filter size	#Parallel ONNs
3x3	290
5x5	170
7x7	110

Figure 7: ONN edge detection latency estimation for SIFT-ONN 512x512 image process.

with various black-and-white thresholds. Next, the ONN for edge detection is applied to each generated black-and-white image to obtain images with highlighted edges. Finally, the ONN edge detection output images replace the SIFT DoG stage output for the rest of the SIFT algorithm, see Fig. 6.

Fig. 7 shows time estimations to scan all generated images from a 512x512 input image with parallel feedforward ONNs. The number of parallel ONNs is defined from the maximum resource utilization of the Zybo Z7 board, which is equipped with a Xilinx Series-7 FPGA. Fig. 7 highlights that using a stride of the size of the filter really improves latency. In [5], authors assigned the ONN output to the full selected window to avoid discontinuities. However, in SIFT-ONN, tests showed the precision is higher if only the central pixel is assigned, even though it induces discontinuities. Thus, for the rest of the paper, the ONN output is assigned to the central pixel of the scanning window. For example, for a 3x3 ONN filter, the stride is equal to 3, and the ONN output is assigned to the central pixel of the 3x3 window.

4.1 Validation and evaluation methods

The feedforward ONN is validated and evaluated for image edge detection in [5] using the digital design of Fig.4.b) implemented on FPGA. In this work, the feedforward ONN configured for image edge detection is first emulated in python with a Hopfield-based feedforward network. It allows using the built-in functions of SIFT algorithm from Python libraries [1] combined with the ONN image edge detection. Also, it allows comparing precision and latency with other state-of-the-art feature detection and description algorithms available in Python libraries. However, in order to have a better latency assessment of the SIFT-ONN, we estimate the latency of the digital feedforward ONN on various image sizes.

4.1.1 Dataset

This work uses a custom dataset generated from 36 standard 256x256 and 512x512 gray scale images. Each image is resized and normalized to a 512x512 size and becomes a base image. Then, 5 sub-images are generated by applying rotation, perspective and size transformations from each 512x512 base image. Thus, the full dataset contains 36 base images, and 180 transformed images.

4.1.2 Hopfield-based emulator in Python for precision evaluation

SIFT-ONN is validated in python using a Hopfield-based emulator of the digital feedforward ONN configured for image edge detection combined with built-in SIFT Python functions [1]. Precision is obtained using images from the dataset applied to image relocation application. In feature detection and description algorithms, precision is evaluated by checking repeatability. An algorithm is repeatable if features can be correctly detected in images before and after transformation. In this work, to assess SIFT-ONN repeatability, the goal is to retrieve the position of parts of an image, the sub-images from dataset, in the original image, the corresponding base image from dataset. To do so, feature detectors and descriptors are first generated from all base and sub-images using the SIFT-ONN feature detection and description algorithm and matched between the sub-images and the corresponding base image. Matching is performed using brute force approach. From matching between one base image and one sub-image, a transformation is computed, and the corresponding position of the sub-image in the base image is defined. The obtained position is then compared with the real position of the sub-image, obtained during its generation, using the Jaccard Similarity Coefficient (JSC) [13]. JSC computes the ratio of intersection over union

pixels between the real sub-image position, and the computed sub-image position.

$$JSC = \frac{Intersection}{Union} \quad (4)$$

The JSC final score is computed performing a mean of the JSC scores of all sub-images. Moreover, we use the same method with other state-of-the-art feature detection and description algorithms to have a consistent benchmark of the SIFT-ONN solution.

4.1.3 Digital ONN for latency assessment

After assessing the precision and repeatability of the SIFT-ONN, it is necessary to evaluate its computation time. Using parallelism for the ONN edge detection in Python is difficult, so for the Python tests with a Hopfield-based emulator, we consider sequential treatment, which makes the computation time really high, around hundreds of seconds. However, considering the digital feedforward ONN implemented on FPGA, it is possible to implement multiple ONNs in parallel to improve latency. Thus, to evaluate and compare the latency of SIFT-ONN with state-of-the-art algorithms, we use estimations of digital ONN latency to scan images using parallel ONNs combined with a computation time of later steps of SIFT-ONN from SIFT Python libraries. The latency of SIFT Python libraries is estimated based on the GeForce GTX 1050 Mobile GPU hardware.

5 Results

Here, we present results obtained with the SIFT-ONN solution in terms of precision, latency, and compared them with well-known feature detection and description algorithms. SIFT-ONN is compared with SURF, ORB, BRISK, SIFT, and KAZE as they are state-of-the-art feature detection and description algorithms.

5.1 Precision

Fig. 8 highlights the JSC mean score obtained for each tested algorithm depending on the percentage of matches used for transformation. Note, SIFT-ONN is tested with the three ONN filter sizes: 3x3, 5x5, and 7x7, and in each case, the stride of image scanning is equal to the filter size. For example, a stride of 3 for the 3x3 filter. Also note that when scanning with a stride of the size of the filter, the ONN output is applied only to the central pixel of the 3x3 window scanned. Table 2 presents the real computation times obtained with the sequential process of a 3x3 Hopfield emulator, the estimated computation time obtained with digital 3x3 ONN implemented on FPGA and precision for scanning strides equal to 1 and 3 with no filtered matches. It highlights that scanning with a stride of the size of the filter does not diminish the precision score, it even increases it. However, it really decreases the real latency to test on Python, as well as the estimated latency. Thus, we do not consider small strides due to the long computation time.

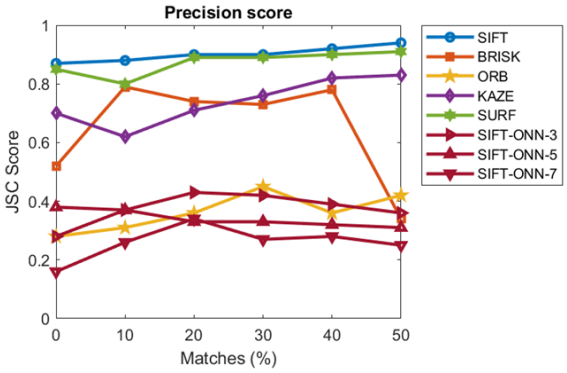


Figure 8: Precision scores of feature extraction algorithms compared with SIFT-ONN solution.

Table 2: ONN-SIFT real computation time based on python processing, estimated computation time based on digital ONN estimations, and score for a 3x3 ONN filter size for two scanning strides and no filtered matches.

Stride	1	3
Real comp. time (s)	3712	563
Est. comp. time (s)	0.0878	0.0632
JSC score	0.16	0.28

Fig. 8 shows that SIFT-ONN reduces SIFT precision score by a factor larger than two. SIFT-ONN precision score is also lower than BRISK, SURF, and KAZE results, but similar to ORB for most of the cases. SIFT-ONN with a 3x3 filter gives the

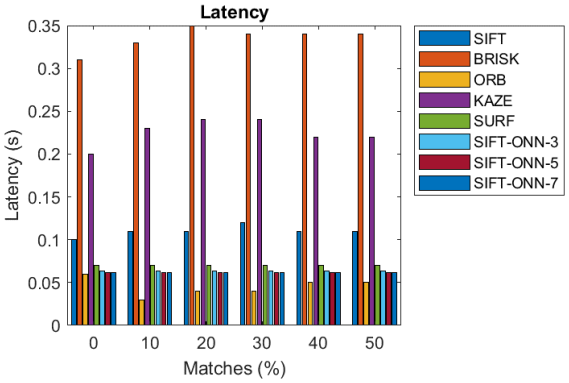


Figure 9: Latency of feature extraction algorithms compared with SIFT-ONN solution.

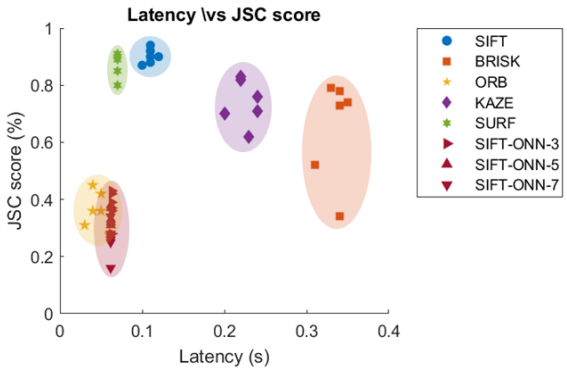


Figure 10: Combination of precision score and latency of SIFT-ONN compared with state-of-the-art feature detection and description algorithms.

best precision score compared with larger ONN filter sizes.

5.2 Latency

Fig. 9 shows latency estimations of ONN-SIFT for various ONN filter sizes, considering the digital ONN design, compared with the latency of state-of-the-art algorithms computed in Python based on the GeForce GTX 1050 Mobile GPU hardware. It shows the ONN filter size does not affect much the global SIFT-ONN latency. SIFT-ONN improves latency from the original SIFT algorithm and reduces it approximately by a factor of two. Fig. 10 combines both precision score and latency results to highlight SIFT-ONN moves the SIFT algorithm in the same precision, and latency ranges as ORB, with a lower precision but a faster computation time.

6 Discussion

This work presents an adaptation of SIFT feature detection and description algorithm by using a hybrid SIFT-ONN solution. In the SIFT-ONN, a feedforward ONN for image edge detection [5] replaces the original SIFT DoG stage. The aim of SIFT-ONN is to reduce SIFT computation time, while keeping high precision to allow feature detection and description on edge devices.

First, it is important to highlight that SIFT-ONN adaptation does not require important changes. Only the SIFT scale generation and DoG stages are replaced with the ONN image edge detection. However, the following SIFT stages do not require any change. Yet, the results show the SIFT-ONN solution induces a drastic reduction of precision in comparison with SIFT, by a factor of 2. This decrease is mainly due to the binary ONN edge detection output. The DoG from SIFT algorithm outputs gray-scale images with various edge strengths such as strong or weak edges, while the ONN edge detection outputs black-and-white images with binary edge information. Thus, this binary output can have a consequence in the extrema definition. Each edge can become a maximum, and each background can become a minimum, depending on the position. However, in principle, ONN can stabilize to non-binary phases among $0^o - 360^o$ range. Thus, a solution to improve ONN-SIFT precision is to investigate how to perform ONN image edge detection with non-binary ONN outputs. Additionally, this work does not compare SIFT-ONN with SIFT adapted with other image edge detection algorithms, such as state-of-the-art Sobel [29] or Canny [12]. Both algorithms generate gray-scale images with various edge significance, thus it can help assess if a gray-scale ONN edge detection could solve feature detection and description with higher precision.

Furthermore, during tests, some divergences in the precision results were observed between two python simulations with equal parameters. Thus, we believe the generated dataset may not be large enough to clearly assess the precision of the SIFT-

ONN algorithm and further tests are necessary. However, due to the large computation time, it was not possible to enlarge the dataset. Additionally, precision is tested on the image relocation application. However, it is also important to test feature detection and description algorithms on other applications, like image occlusion. Thus, additional tests on a larger dataset and on additional applications are also necessary to have a better assessment of the SIFT-ONN precision.

Still, SIFT-ONN is close to ORB performances both in terms of precision and estimated latency. ORB is currently a standard to perform feature detection and description on embedded devices due to its fast computation time, even though precision is lower than other state-of-the-art algorithms. Thus, the SIFT-ONN can become an alternative to ORB for feature detection and description, for example, in robotic applications. Furthermore, ONN shows promising low-power computation properties [16], which can be advantageous for edge computing in robotic applications.

Finally, this work aims to present, validate, and evaluate the SIFT-ONN solution, but up to now, there is no hardware demonstrator implemented. In literature, it is reported various FPGA implementations of SIFT [10, 26], so future work is to develop a real-time full FPGA implementation of the SIFT-ONN algorithm.

7 Conclusion

This work explores the SIFT-ONN solution to accelerate the SIFT feature detection and description algorithm to make it suitable for embedded applications with important latency constraints, such as in robotic navigation. The SIFT-ONN solution replaces the SIFT Difference of Gaussian stage with an image edge detection algorithm based on the promising Oscillatory Neural Network (ONN) neuromorphic computing paradigm. ONNs are currently being explored for their fast and low-power computation, which can be suitable for embedded feature detection and description applications. SIFT-ONN is an adaptation of SIFT which does not require major changes. The SIFT-ONN induces a diminution of the SIFT precision by a factor of two, however, latency estimations based on a digital implementation of the ONN image edge detection algorithm are promising, reducing latency also by a factor of two. Thus, SIFT-ONN performances are comparable with the ORB algorithm, which is the state-of-the-art feature detection and description algorithm for robotic navigation applications and other applications with fast timing requirements. Finally, this work is the first investigation to use the ONN computing paradigm as a hardware accelerator for image processing algorithms, and results are encouraging to explore further image processing tasks.

Acknowledgement

We thank T. Gil (LIRMM - CNRS), M. Jimenez, J. Nunez, and M. J. Avedillo (IMSE - CSIC) for fruitful exchanges and constructive suggestions on this research work.

References

[1] A clean and concise python implementation of sift (scale-invariant feature transform).

[2] Mobile robot navigation and obstacle avoidance techniques: A review. *International Robotics & Automation Journal*, Volume 2(Issue 3), May 2017.

[3] M. Abernot, T. Gil, M. Jiménez, J. Núñez, M. J. Avellido, B. Linares-Barranco, T. Gonos, T. Hardelin, and A. Todri-Sanial. Digital implementation of oscillatory neural network for image recognition applications. *Frontiers in Neuroscience*, 15, 2021.

[4] M. Abernot, T. Gil, and A. Todri-Sanial. Oscillatory Neural Network as Hetero-Associative Memory for Image Edge Detection. In *Neuro-Inspired Computational Elements Conference*, pages 13–21. Association for Computing Machinery, Mar. 2022.

[5] M. Abernot and A. Todri-Sanial. Two-Layer Oscillatory Neural Networks for Image Edge Detection: Study of Bidirectional and Feedforward Architectures, Oct. 2022.

[6] P. F. Alcantarilla, A. Bartoli, and A. J. Davison. Kaze features. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer Vision – ECCV 2012*, pages 214–227. Springer Berlin Heidelberg, 2012.

[7] A. M. Ali and M. J. Nordin. Sift based monocular slam with multi-clouds features for indoor navigation. In *TENCON 2010 - 2010 IEEE Region 10 Conference*, pages 2326–2331, 2010.

[8] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (slam): part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

[9] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision – ECCV 2006*, volume 3951, pages 404–417. Springer Berlin Heidelberg, 2006.

[10] V. Bonato, E. Marques, and G. A. Constantinides. A parallel hardware architecture for scale and rotation invariant feature detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(12):1703–1712, 2008.

[11] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, Dec. 2021.

[12] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov. 1986.

[13] N. C. Chung, B. Miasojedow, M. Startek, and A. Gambin. Jaccard/Tanimoto similarity test and estimation methods for biological presence-absence data. *BMC Bioinformatics*, 20(15):644, Dec. 2019.

[14] C. Cocaud and T. Kubota. SURF-Based SLAM Scheme using Octree Occupancy Grid for Autonomous Landing on Asteroids. page 8.

[15] G. Csaba and W. Porod. Coupled oscillators for computing: A review and perspective. *Applied Physics Reviews*, 7(1):011302, Mar. 2020.

[16] C. Delacour, S. Carapezzi, M. Abernot, and A. Todri-Sanial. Energy-Performance Assessment of Oscillatory Neural Networks based on VO2 Devices for Future Edge AI Computing. Mar. 2022.

[17] G. Desouza and A. Kak. Vision for mobile robot navigation: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267, 2002.

[18] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006.

[19] T. Endo and K. Takeyama. Neural network using oscillators. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 75(5):51–59, 1992.

[20] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, Apr. 1982.

[21] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 International Conference on Computer Vision*, pages 2548–2555, 2011.

[22] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004.

[23] R. Morris. D.O. Hebb: The Organization of Behavior, Wiley: New York; 1949. *Brain Research Bulletin*, 50(5-6):437, Nov. 1999.

[24] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct. 2015.

[25] D. E. Nikonov, G. Csaba, W. Porod, T. Shibata, D. Voils, D. Hammerstrom, I. A. Young, and G. I. Bourianoff. Coupled-Oscillator Associative Memory Array Operation for Pattern Recognition. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 1:85–93, Dec. 2015.

[26] J. Peng, Y. Liu, C. Lyu, Y. Li, W. Zhou, and K. Fan. Fpga-based parallel hardware architecture for sift algorithm. In *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 277–282, 2016.

[27] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[28] N. Shukla, W.-Y. Tsai, M. Jerry, M. Barth, V. Narayanan, and S. Datta. Ultra low power coupled oscillator arrays for computer vision applications. In *2016 IEEE Symposium on VLSI Technology*, pages 1–2, June 2016. ISSN: 2158-9682.

[29] I. Sobel and G. Feldman. A 3×3 isotropic gradient operator for image processing. *Pattern Classification and Scene Analysis*, pages 271–272, Jan. 1973.

[30] H. Zhou, Y. Yuan, and C. Shi. Object tracking using sift features and mean shift. *Computer Vision and Image Understanding*, 113(3):345–352, 2009.