



Green Software Engineering: Cloud-based Face Detection and Static Code Analysis

Ethar Abdul Wahhab Hachim¹, Yasmin Makki Mohialden^{2*}, Zeyad Farooq Lutfi³,
Nadia Mahmood Hussien⁴

^{1,2*,3,4}Department of Computer Science, College of Science, Mustansiriyah University,
Baghdad, Iraq

Email: ¹ethar201124@uomustansiriyah.edu.iq, ³zeyadfa6@uomustansiriyah.edu.iq,
⁴nadia.cs89@uomustansiriyah.edu.iq

Corresponding Email: ^{2*}ymmiraq2009@uomustansiriyah.edu.iq

Received: 10 April 2023

Accepted: 26 June 2023

Published: 16 August 2023

Abstract: *This paper presents an approach to green software engineering that integrates cloud-based face detection and static code analysis to promote sustainable software development. The proposed method uses OpenCV, a computer vision library, and a pre-trained Haar cascade classifier to detect faces in images. Faces are marked with green bounding frames that serve as visual indicators of their locations. In addition, the paper evaluates the quality of a distinct script file using Pylint library static code analysis techniques. The analysis evaluates code compliance with standards, identifies potential flaws, and identifies code odors. By integrating these practices, the proposed method seeks to reduce resource consumption, maximize energy efficiency, and enhance code maintainability, promoting environmentally friendly and sustainable software engineering practices. One outcome of our effort was creating the YasminNadiaArabicSocialMediaImages data collection, which includes faces of Arabic social media celebrities and is filled out to be accessible for public usage on the websites Kaggle and GitHub.*

Keywords: *Green Software Engineering, Static Code Analysis, Kaggle, Opencv, and Cloud Computing.*

1. INTRODUCTION

Green Software Engineering has become an essential part of software development. Its goal is to reduce the damage software systems do to the world over their entire lifecycle.

Green Software Engineering creates eco-friendly software. Software systems' environmental impact is expanding as software demand rises. Software development and operation waste



energy, resources, and carbon. Traditional software development ignores ecological impacts. Green Software Engineering integrates sustainability into the software development lifecycle to solve this problem. It stresses energy efficiency, resource optimization, and carbon footprint reduction.

Like the sustainability movement, Green Software Engineering seeks to balance technical progress and environmental responsibility. It stresses energy-efficient algorithms, renewable energy sources for data centers, optimizing resource utilization, and code quality and maintainability. Green Software Engineering includes virtualization, cloud computing, and software reworking to save energy and reduce waste. It fosters energy-efficient hardware and resource-efficient software coding.

Software development may promote sustainability and environmental conservation by using Green Software Engineering principles. It creates eco-friendly software solutions with excellent performance and functionality [1,2,3]. Integrating face detection and static code analysis promotes sustainable software development. These two strategies are unrelated, yet they support software engineering's environmental responsibility. Improving User Experience: Applications that detect and interact with faces improve the user experience. This improves facial recognition, augmented reality, and biometric authentication. Applications can provide smooth, intuitive user experiences with precise facial detection. Optimized coding and efficient facial detection methods reduce computing resource consumption. Face-related tasks use less energy and processing resources with lightweight and energy-efficient face detection methods. Static code analysis eliminates inefficiencies and resource-intensive coding methods to optimize code and reduce system resources. Face identification and static code analysis increase software performance and responsiveness. Fast and precise face identification techniques enable real-time or near-real-time facial data processing, improving application performance. Static code analysis identifies performance bottlenecks, inefficient algorithms, and memory leaks, helping developers improve code for responsiveness. Static code analysis tools like Pylint find code smells, defects, and maintainability issues. Developers can write better, more maintainable code by following coding standards. Technical debt, debugging, and teamwork improve. Face identification and static code analysis improve software development efficiency, sustainability, and the environment. Green software engineering enhances user experience, resource consumption, performance, and code quality [4,5,6].

Cloud computing is a new technological paradigm that transforms computer and technical ideas into solutions that function like utilities like water and energy systems. The cloud has many benefits, such as adaptable computing tools, cost savings, and flexible service. But it has been shown that worries about security and privacy are the key things that keep people from using the cloud more. The new ideas that clouds offer, such as resource sharing, multi-tenancy, and hiring, make things harder for the security community [7, 8]. This study suggests a way to make software development more sustainable by combining cloud-based face detection with static code analysis. By combining these techniques, we hope to reduce our resources, make the best use of energy, and make the code easier to manage, all of which will contribute to environmentally friendly software engineering.



Related work

[2010] In this paper, the experimental results show the method can ensure that the cloud neither knows the user's actual face data nor the face private matching identification result. To secure the user's face data, we develop a credible, efficient, and low-complex method to guarantee cloud computing security [9].

[2010] In this paper, Experimental results prove that the new face detector implements real-time face detection and improves detection robustness [10].

[2019] The authors evaluate their schemes based on the open-source Azure IoT Edge, and the experimental results show that the three-tier architecture "Client-Edge-Cloud" face recognition system outperforms the state-of-the-art face recognition systems in reducing the average response time [11].

[2019] the proposed face detection method achieves state-of-the-art performance on the WIDER FACE dataset, the most popular and challenging face detection benchmark [12].

[2020] Using publicly available programs to test their tool, the authors show that insecure patterns can be found in real-world code. This means that static source-code analysis is an excellent way to check for security, for example, to stop commissioning unsafe or malicious industrial task programs [13].

Proposed Method

The suggested approach provides information on both programming and visual aspects. It can be applied to numerous fields, including software quality assurance, automated code review procedures, and facial recognition systems. It provides valuable data for decision-making and enhancing image- and code-based applications. The code's suggested method is divided into two main parts.: Finding faces in images-based cloud technology and analyzing code statically using the pylint python package.

A. Finding Faces in images-based cloud technology

The determined functional requirements for the first part are very sentential. The function requirements are: -

1. The system should be able to select the required image
2. The system should find the features of an image.
3. The system should be able to compare features with the dataset
4. The design should be able to detect the object
5. The system should draw bounding boxes around detected faces

Functional requirements' output can be illustrated in a use case diagram and scenario. Figure 1 shows the use case diagram for the first part.

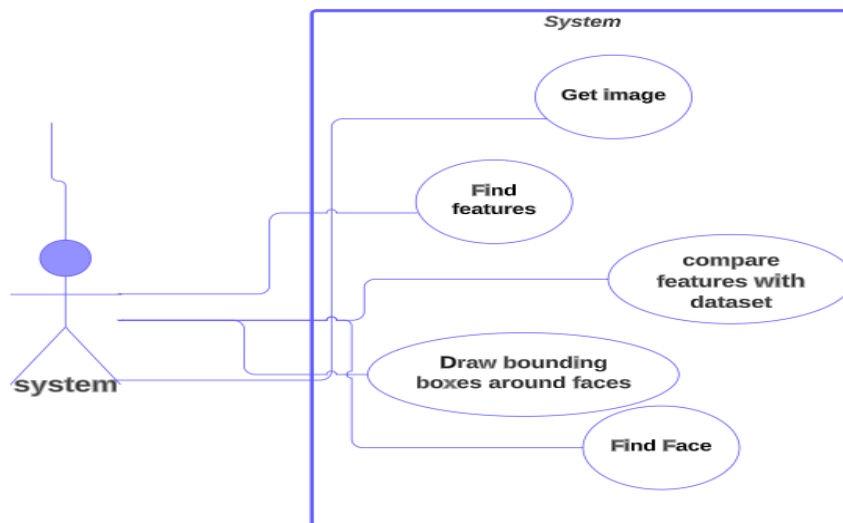


Figure 1: use case for the first part

The method uses the OpenCV library and the Haar cascade algorithm to find faces. A library for processing images and videos, OpenCV is open-source and free. It is connected to computer vision through machine learning, feature and object recognition, and other aspects. The primary OpenCV modules, components, and OpenCV based on Python are presented in this document [14].

The Haar Cascade Classifier is one of the few object detection methods to identify faces. It is based on the number of pixels inside the rectangle feature rather than the values of each pixel in the picture, and it can process information quickly. This method finds an object using a Haar-like feature, integral vision, AdaBoost learning, and a cascade algorithm. The Haar traits comprise most of the parts of the Haar Cascade Classifier that find faces. With the Haar features, you can check whether a picture has a specific quality. Add up the pixels inside the black rectangle to detect the value of each part. The Haar-like feature is a square that gives a precise signal to detect faces in a picture [15] quickly.

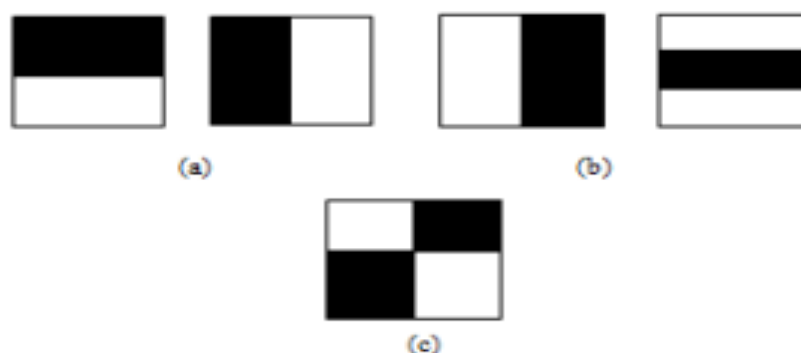


Figure 2 (a) Edge feature, (b) Line feature, and (c) Four-Triangle



B. Static code analysis

It scans source code for mistakes, flaws, vulnerabilities, and compliance concerns to assess a program's quality and security without actually running it. Static code analysis aids in finding and fixing issues early in the development cycle before they become expensive or dangerous. Performance, readability, and maintainability are all enhanced through static code analysis. Benefits of static code analysis: Testing can discover difficult-to-find concurrent bugs and memory leaks, buffer overflows, null pointer dereferences, and buffer overflows. The challenges in static code analysis [16]:

- i. It controls formatting, documentation, indentation, and name rules.
- ii. It decreases vulnerabilities and coding errors, enhancing security and quality.
- iii. It would help if you did as little debugging and hand-coding review as possible to save time and money.
- iv. It can increase confidence in code, particularly old or third-party code.

The system used static analysis to provide quick feedback on unnecessary code that is useful in practice. Static study of the code using Python's pylint.



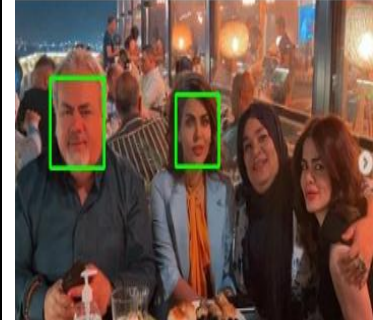

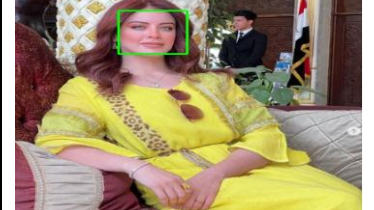
Pylint is the most well-known static analysis tool for Python to date. PyLint is open-source, accessible, and capable of finding logical flaws and warnings about specific coding standards, providing information about the complexity of the code, and making straightforward refactoring suggestions. Building an abstract syntax tree is how it functions. Additionally, Pylint enables developers to create their plugins for particular tests, making it simple to expand the tool. It is mainly used by well-known IDEs and frameworks, like PyCharm, VSCode, Django, Eclipse with PyDev, etc., to do static analysis of code in real time. Google, one of the biggest IT companies that use PyLint, shows how common and reliable it is. Google's Python source is looked chiefly at with static analysis [17].


2. RESULTS AND DISCUSSION

The proposed system uses an open-CV-based Haar cascade algorithm to detect the face in any input image. One outcome of our effort was creating the YasminNadiaArabcSocialMediaImages data collection, which includes faces of Arabic social media celebrities and is filled out to be accessible for public usage on the websites Kaggle and GitHub. Table 1 shows some test images from the dataset.

Table. 1. Test images from the dataset

Image with detected faces	The coordinate of the detected faces	Static code analysis
----------------------------------	---------------------------------------------	-----------------------------

	<p>0</p>	<p>Issue 1: Unused variable 'x' Issue 2: Missing docstring for the 'detect_faces' function Issue 3: Inconsistent indentation</p>
	<p>Face one coordinate: (22, 69, 83, 83) Face 2 coordinates: (292, 125, 77, 77) Face 3 coordinates: (268, 334, 77, 77) Face four coordinates: (88, 342, 72, 72)</p>	<p>Issue 1: Unused variable 'x' Issue 2: Missing docstring for the 'detect_faces' function Issue 3: Inconsistent indentation</p>
	<p>Face one coordinate: (40, 90, 60, 60) Face 2 coordinates: (180, 101, 48, 48)</p>	<p>Issue 1: Unused variable 'x' Issue 2: Missing docstring for the 'detect_faces' function Issue 3: Inconsistent indentation</p>
	<p>Face 1 coordinates: (147, 98, 50, 50) Face 2 coordinates: (252, 103, 47, 47) Face 3 coordinates: (334, 103, 50, 50) Face four coordinates: (351, 191, 58, 58) Face 5 coordinates: (211, 173, 58, 58)</p>	<p>Issue 1: Unused variable 'x' Issue 2: Missing docstring for the 'detect_faces' function Issue 3: Inconsistent indentation</p>
	<p>Face one coordinate: (120, 34, 79, 79)</p>	<p>Issue 1: Unused variable 'x' Issue 2: Missing docstring for the 'detect_faces' function</p>

		Issue 3: Inconsistent indentation
	Face 1 coordinates: (126, 338, 97, 97)	Issue 1: Unused variable 'x' Issue 2: Missing docstring for the 'detect_faces' function Issue 3: Inconsistent indentation

The systems develop based on google collab cloud, in which all the required tools are available online without needing installation. The system is a utility collab rather than traditional computing. Table 2 illustrate the benefit of using the cloud in developing system.

Table 2: Benefit of using cloud computing

Systems	Cost and scalability	Performance and reliability	Security and Compliance	Culture and collaboration	Trade-offs and balance	Flexibility and innovation
Traditional computing	-----	-----	-----	-----	✓	-----
Cloud computing	✓	✓	✓	✓	✓	✓

3. CONCLUSION

The comprehensive approach, which integrates the proposed method's programming and visual aspects, is presented. They can be implemented in areas as diverse as software quality assurance, automated code reviews, facial recognition systems, providing valuable data for decision-making, and optimizing image-based applications.

The first phase of the method focused on detecting faces in images using cloud technology. The functional requirements included selecting the desired image, extracting features, comparing them with a dataset, detecting objects, and drawing boxes around the detected faces. The OpenCV library and the Haar cascade algorithm, known for their high speed and accuracy, were used to detect faces.

The second stage of the method involved parsing the static code using the pylint package in the Python programming language. Static code analysis helps identify source code bugs, defects, vulnerabilities, and compliance concerns without running it. Pylint, a popular static analysis tool in Python, provides feedback on coding standards, logical flaws, code complexity, and suggested refactoring.



The results of the proposed system proved its effectiveness in detecting faces in the input images using the Haar cascade algorithm. A dataset called Yasmin Nadia Arabc Social Media Images containing the faces of Arab celebrities on social media has been created and is publicly available on Kaggle and Git Hub. The system also identified code issues such as unused variables, missing strings, and non-uniform indentation using static code analysis using pylint. System creation has been facilitated by using cloud computing, relying on Google Colab cloud technology in this case. Compared to traditional computing methods, this eligibility has benefits such as cost, scalability, speed, reliability, security, compliance, culture, and better teamwork.

We suggest that in future work to improve face detection performance, other techniques can be explored, and face detection algorithms can be improved to increase accuracy and efficiency. This may include using machine learning and deep neural networks to improve the classification of faces and better identify traits. Or Expanding the scope of application: The proposed system can be used in other fields besides detecting faces, such as detecting other objects or recognizing the pattern of specific things. The system can be extended to support different requirements in many areas. Or improve static code analysis: The capabilities of static code analysis can be extended to handle a broader range of programming languages and provide more detailed and comprehensive reports on potential defects and concerns in the code. Or expand support for cloud computing: It can Explore other options for cloud computing, such as Amazon Services, Microsoft Azure, and others, to achieve greater availability and flexibility in the system and expand its capabilities.

Acknowledgment

The Authors would like to thank Mustansiriyah University (<https://uomustansiriyah.edu.iq/>) in Baghdad –Iraq, for its support in the present work.

4. REFERENCES

1. Manotas, I., Bird, C., Zhang, R., Shepherd, D., Jaspan, C., Sadowski, C., Pollock, L., & Clause, J. (2016). An Empirical Study of Practitioners' Perspectives on Green Software Engineering. 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). <https://doi.org/10.1145/2884781.2884810>.
2. Yahaya, J., Ibrahim, S., Raisian, K., & Deraman, A. (2019). Green Software Process Based on Sustainability Dimensions: The Empirical Investigation. Proceedings of the 1st International Conference on Informatics, Engineering, Science, and Technology, INCITEST 2019, 18 July 2019, Bandung, Indonesia. <https://doi.org/10.4108/eai.18-7-2019.2287944>.
3. Nazir, S., Fatima, N., Chuprat, S., Sarkan, H., F, N., & Sjarif, N. (2020). Sustainable Software Engineering: A Perspective of Individual Sustainability. International Journal on Advanced Science, Engineering and Information Technology. <https://doi.org/10.18517/ijaseit.10.2.10190>.



4. Zhou, Y., Liu, D., & Huang, T. (2018). Survey of Face Detection on Low-Quality Images. 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018). <https://doi.org/10.1109/FG.2018.00121>.
5. Schiewe, M. (2022). Bridging the gap between source code and high-level concepts in static code analysis: student research abstract. Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing. <https://doi.org/10.1145/3477314.3508371>.
6. Schiewe, M. (2022). Bridging the gap between source code and high-level concepts in static code analysis: student research abstract. Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing. <https://doi.org/10.1145/3477314.3508371>.
7. Saba Abdulbaqi Salman, Sufyan Al-Janabi, and Ali Makki Sagheer, "Security Attacks on E-Voting System Using Blockchain," Iraqi Journal For Computer Science and Mathematics, vol. 4, no. 2, pp. 179–188, May 2023.
8. Khalil, I. M., Khreishah, A., & Azeem, M. (2014). Cloud computing security: A survey. *Computers*, 3(1), 1-35.
9. Wang, C., & Yan, H. (2010). Study of Cloud Computing Security Based on Private Face Recognition. 2010 International Conference on Computational Intelligence and Software Engineering. <https://doi.org/10.1109/CISE.2010.5676941>.
10. Tu, Y., Yi, F., Chen, G., Jiang, S., & Huang, Z. (2010). Fast rotation invariant face detection in color image using multi-classifier combination method. 2010 International Conference on E-Health Networking Digital Ecosystems and Technologies (EDT). <https://doi.org/10.1109/EDT.2010.5496601>.
11. Gil, D., Fernández-Alemán, J., Trujillo, J., García-Mateos, G., Luján-Mora, S., & Toval, A. (2018). The Effect of Green Software: A Study of Impact Factors on the Correctness of Software. *Sustainability*. <https://doi.org/10.3390/SU10103471>.
12. Li, S., Liu, F., Liang, J., Cai, Z., & Liang, Z. (2019). Optimization of Face Recognition System Based on Azure IoT Edge. *Cmc-computers Materials & Continua*. <https://doi.org/10.3970/cmc.2018.903.694>.
13. Zhang, F., Fan, X., Ai, G., Song, J., Qin, Y., & Wu, J. (2019). Accurate Face Detection for High Performance. *ArXiv*.
14. Pogliani, M., Maggi, F., Balduzzi, M., Quarta, D., & Zanero, S. (2020). Detecting Insecure Code Patterns in Industrial Robot Programs. Proceedings of the 15th ACM Asia Conference on Computer and Communications Security. <https://doi.org/10.1145/3320269.3384735>.
15. Hasan, R. T., & Sallow, A. B. (2021). Face Detection and Recognition Using OpenCV. *Journal of Soft Computing and Data Mining*, 2(2), 86-97.
16. Sharma, A., Pathak, J., Prakash, M., & Singh, J. N. (2021, December). Object detection using OpenCV and Python. In 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N) (pp. 501-505). IEEE.
17. Gulabovska, H., & Porkoláb, Z. (2019, September). Survey on Static Analysis Tools of Python Programs. In SQAMIA.