Application of Computer Vision to image review process in the software development

Wendell Marques
Sidia Institute of Technology
Manaus, Brazil
wendell.marques@sidia.com

Andrezza De Melo Bonfim
Sidia Institute of Technology
Manaus, Brazil
andrezza.bonfim@sidia.com

Eduardo Filho
Sidia Institute of Technology
Manaus, Brazil
eduardo.oliveira@sidia.com

Edluce Leitao Veras
Sidia Institute of Technology
Manaus, Brazil
edluce.leitao@sidia.com

Daniel Souza
Sidia Institute of Technology
Manaus, Brazil
daniel.souza@sidia.com

Vinicius Gabriel da Silva
Sidia Institute of Technology
Manaus, Brazil
vinicius.silva@sidia.com

ABSTRACT

The main goal of reviews in software development is to grant the quality and correctness of the products. One of those tasks that we can highlight is image review. This process could require more time to process and tends to be less accurate due to human error. Considering that developers who perform this type of task also have other responsibilities, the longer the review takes, the more work time is required, and this can affect the time of deliverables in the industry. Therefore, the proposal of this research is to use Optical Character Recognition (OCR) and Computer Vision (CV), to automate the process of image review, for the interpretation and validation of the content, both textual and objects. To carry out this research a set of 100 images was used in manual and automated review process and was verified that the automated approach reduced the execution time by 96.45%, and reduced the number of errors by 4%, proving that it is feasible in this development context. The application of this tool helped to increase efficiency by reducing the time spent on manual reviews, increased the consistency of image analysis, ensuring better accuracy compared to manual reviews. It also promoted improvements in development and the team, through a more productive work environment.

KEYWORDS

Computer Vision, reviews in software development, software engineering, systems for automation, Optical character recognition

1 Introduction

Computer Vision, which stands out as a branch of artificial intelligence [2], enables machines to process and analyze digital images to understand their contents. This technology has revolutionized various industries worldwide, including healthcare and public security [1, 5, 6]. By automating image analysis, computer vision provides valuable insights, transforming multiple sectors.

In software development, this capability is crucial for review processes, ensuring high-quality products or services. It serves as a vital element in the creation of digital solutions, particularly for large institutions [7]. However, traditional reviews are labor-intensive and require significant time and attention from reviewers, who often face shortages due to high workloads [3]. Computer vision can be automated by these processes to improve efficiency.

Research focuses on automating image reviews with precision and speed, minimizing human error [4]. A tool known as ITR (Image Test Review) was developed using computational vision technologies. This tool primarily functions to interpret images and validate textual information within them, facilitating more efficient reviews. The proposed ITR tool has enabled teams to develop independently, compared to traditional manual reviews, thereby enhancing productivity and reducing dependency on labor-intensive methods.

2 Methodology

In this section, the methods and scenarios used in the context of automating image reviews through mentioned computational vision techniques are described, along with a brief contextualization of the current scenario and the proposed scenario.

2.1 Research Application Context

The research team comprises members involved in operational development, with responsibilities divided based on differences in expertise levels. The study was guided by a group that validates the work of others, including senior members who check the contributions of several developers.

2.2 General Overview of the Current Process

The current process of submitting an image to the software involves both less experienced developers and reviewers, each with their own roles. Below, the stages of creating and reviewing the image, as well as the consulted image specification documents, are briefly described.

2.2.1 Code Specification Documents. The information is gathered from a set of documents and systems that are not standardized and decentralized. These documents contain specifications for creating the image, including a unique code for a specific equipment.

2.2.2 Creation of the Image and Submission for Review. The image is generated by querying a database, from which 8 objects are subsequently inserted into specific locations on the image. These objects consist of 5 static objects and 2 dynamic objects derived from the specifications provided in the reference documents (2.2.1).

To create this image, the developer retrieves information from the specifications mentioned in section 2.2.1, fills out a form with the data to be included in the image, and then generates the image with the added information. After completion, the developer submits the

generated image for collaborative review and change management in source code and digital files via Helix Swarm.

2.2.3 Manual Review. Upon receiving a revision request, an assigned reviewer checks if the image data matches the information from consulted sources. If discrepancies are found, the developer is notified to correct the image using the established generation process. If the image is accurate, the review concludes, enabling the author to proceed with their submission.

However, this process presents some challenges that hinder its efficiency and agility. The primary issue is the ongoing dependence on a reviewer's availability after the image creation. This setup can lead to delays in the review process and may overload the reviewers, potentially causing unnecessary idle time for developers while they wait for feedback.

2.3 Preprocessing for automated review

For the presented proposal, some pre-processing activities were carried out for important parts in the development of the solution, and these parts are explained below.

2.3.1 Database creation. A standardized database in XLSX format was created. This database contains information previously collected, unified and registered from the original documents mentioned in 2.2.1.

2.3.2 Model definition. In the context of computational vision, the model is an image composed of predefined elements. These elements consist of both textual and graphical components. As observed in the subsection 2.2.2, these static elements do not change independently of the dynamic code being applied. In this research, the model used contains all the fixed elements that are repeated in the test images.

2.4 ITR - *Image Test Review*

Given this scenario, a tool called ITR was proposed. The functioning of this tool consists in automating the process of reviewing images. The basic operation of the tool is to receive a request for review sent by the developer, collect the image, and submit it to accuracy validations for graphical and textual elements. The validation of the image is performed using OpenCV combined with PyTesseract . For the first validation (of non-textual objects), OpenCV is used to verify the content of non-textual objects by comparing it with the model described in Section 2.3.2, as well as ensuring that the positioning of all components of the image is correct. In this stage, each image to be validated undergoes a process of extraction (cutting) of non-textual objects using predefined coordinates (based on the correct positioning according to the defined model), and then compares the collected content to the content that should be present at those coordinates as per the defined model (section 2.3.2).

In the second validation process, conducted parallel to the first, PyTesseract , which utilizes the Tesseract OCR engine , was selected for the OCR function. This function extracts textual information from images. To apply OCR, the image must be converted entirely into text. After this conversion, the extracted texts are compared against the database found in section 2.3.1 to verify matches. After, the tool verifies both the accuracy of positioning and content. If everything is correct, it indicates that the process has passed; otherwise, it highlights what is incorrect.

It is important to note that all elements must be correct (both position and content), otherwise, a file is generated indicating what is out of compliance in the image. An example of this can be seen in Figure 1, where the expected value should be a combination of the upper element with the value "AM-Z896P" and the lower element with the value "TPADSPS-417362."

Figure 1: Example of flagging errors in the image

3 Results and Discussions

The tests of the implemented tool were conducted in two phases. It is necessary to emphasize that the number of images used in both phases is the same, being 100 images in each phase. Additionally, all tests were performed using PyCharm Community Edition 2021.1.1 on a Windows 11 computer with an i7 processor and 16GB RAM. These phases and their variables will be displayed and detailed in the following subsections.

3.1 First Testing Phase

For the first phase, a random database was generated, which can be found in the file database_final_ultimate.xlsx. From this database, 96 images with correct textual and object values were generated, as well as 4 images with incorrect values for validation of automation. These images can be found in the folder Imagens_geradas.zip.

In this first phase of testing, the tool successfully completed all necessary verifications, achieving 100% accuracy and completing the process in 8.3 minutes.

3.2 Second Test Phase

For the second phase of the tool's testing, 100 images previously manually reviewed by evaluators were selected from the "swarm" based on their links (from the *swarm*). From these 100 reviews , the total time spent by each reviewer was collected, summing up to a total of 2,755 minutes. This results in a median of approximately 28.533 minutes per image (Figure 2).

Additionally, quantitative data from incorrectly conducted reviews were collected, which summed up to 6 wrongly approved images.

The images mentioned above were collected and stored as a test base. After executing the *ITR* using this data set, the total

execution time was 8,01 minutes which generated a median of 4,65 seconds/image.

In this scenario, the tool presented 2 false negatives, where discrepancies were incorrectly identified during the validation of the textual content.

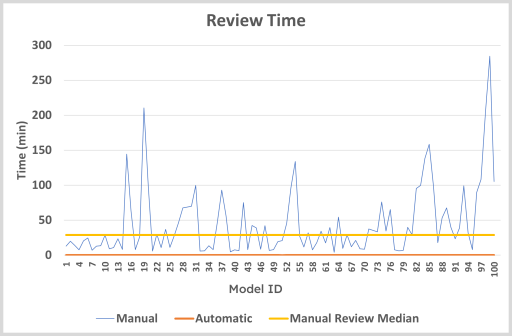

Figure 2: Manual and Automatic Time Review

3.3 Discussion

A comparative analysis of equivalent scenarios was conducted regarding execution times and the most notable observation from this analysis was that the review process using the proposed tool achieved a time difference of 2746.99 minutes less than the current manual process (manual review).

As mentioned earlier, several factors can contribute to this difference, including the availability of the reviewer's time.

It is important to remember that reviewers also have other activities to perform during their workday and, therefore, often spend a significant amount of time on a single review. Conversely, the proposed tool does not require manual review work for its execution. This suggests that incorporating new tools to assist with the reviewing process may help reduce idle time and improve workflow efficiency.

Given the dataset where the tests were conducted using 100 images, the manual review accuracy was found to be 94%, while the automatic review accuracy was 98%. Comparing these results specifically for this dataset, both accuracies can be considered acceptable. However, if the dataset were expanded to 10,000 images, with proportional distribution, there would be 600 errors in manual reviews and 200 errors in automatic reviews.

However, the issue with the review tool (ITR) may stem from a potential deficiency in its ability to identify certain special characters and differentiate between uppercase and lowercase letters, such as distinguishing a capital 'I' or a lowercase 'l.' To address these limitations, adjustments and improvements are necessary to reduce these deficiencies and bring accuracy as close to 100% as possible.

4 Conclusions and Future Work

Given the results obtained, it is suggested that the proposed method contributes to reducing execution time. In the scenario with 100 validations, the tool's execution time was 96.45% less compared to the same number of manual validations. Additionally, the use of ITR showed a 4% increase in accuracy. Based on these results, it is possible to project that the use of the tool as part of the process for validating images is viable, as it automates the process and takes less time to complete.

The application of this automated review tool significantly contributes to some points within the development team. First, it aids in increasing efficiency by reducing the time spent on manual reviews, allowing reviewers to dedicate more time to more complex and challenging tasks. Furthermore, as previously observed, automation increased the consistency of image analysis, ensuring greater accuracy than manual reviews. In this way, the use of this tool has improved both development and teamwork through a more productive work environment.

The main limitation of this tool is that for each image embarked on software by the team, a specific template must be created and implemented based on individual characteristics. Additionally, automation in this case depends on predefined rules, meaning the tool needs to be constantly updated to follow changes in standards or specifics.

Another limitation arises from the hardware used. It's possible that the proposed tool would have a shorter execution time if better equipment were utilized.

Future work aims to automate image creation upon database updates and implement verification using deep learning tools.

ARTIFACT AVAILABILITY

Repository cannot be shared due to industrial restrictions and confidentiality constraints.

ACKNOWLEDGMENTS

This paper is a result of the Research, Development & Innovation Project (ASTRO) performed at Sidia Institute of Science and Technology sponsored by Samsung Eletrônica da Amazônia Ltda., using resources under terms of Federal Law No.8.387/1991, by having its disclosure and publicity in accordance with art.39 of Decree No.10.521/2020. The author, Andrezza de Melo Bonfim, is a former employee and can currently be reached at the email andrezza.bonfim@hotmail.com.

REFERENCES

[1] Cheryl Angelica, Hendrik Purnama, Fredy Purnomo, et al. 2021. Impact of computer vision with deep learning approach in medical imaging diagnosis. In *2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI)*, Vol. 1. IEEE, ACM Press, New York, NY, 37–41.

[2] T Arrighi, JE Rojas, JC Soto, CA Madrigal, and JA Londono. 2012. Recognition and classification of numerical labels using digital image processing techniques. In *2012 XVII Symposium of Image, Signal Processing, and Artificial Vision (STSIVA)*. IEEE, 252–260.

[3] Erika Ap Garefa dos Santos, Gilmar Barros Ferreira, and Mariela Ferreira. 2023. AGRICULTURA 4.0: estudo de caso sobre a eficiência da indústria 4.0 aplicada ao agronegócio. *Ciência & Tecnologia* 15, 1 (2023), e1517–e1517.

[4] Kenish Rajesh Halani, Kavita, and Rahul Saxena. 2021. Critical Analysis of Manual Versus Automation Testing. In *2021 International Conference on Computational Performance Evaluation (ComPE)*. 132–135. doi:10.1109/ComPE53109.2021.9752388

[5] S. Rama Krishna. 2025. Text Detection and Extraction Using OpenCV and OCR. *International Scientific Journal of Engineering and Management* (March 2025).

[6] CH Nanda Kumar, E Nithin Computer, Ch Sai Krishna, and Ch Bindhu Madhavi. 2023. Real-Time Face Mask Detection using Computer Vision and Machine Learning. In *2023 Second International Conference on Electronics and Renewable Systems (ICEARS)*. IEEE, 1532–1537.

[7] Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko, and Alberto Bacchelli. 2018. Modern Code Review: A Case Study at Google. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*. 181–190.