EVALUATION OF PYTHON LANGUAGE IN THE EARLY STAGES OF DIGITAL IMAGE PROCESSING

Dericson Pablo Calari Nunes [1], Jaqueline Rissa Franco [2], Marcos Monteiro Júnior [1], Jonathan de Matos [1], Rosane Falate [1]

[1] Universidade Estadual de Ponta Grossa (UEPG) 84.030-900 – Ponta Grossa – PR – Brasil
[2] Universidade Federal do Paraná (UFPR) 80.060-00 – Curitiba – PR – Brasil

dericsonpcalari@gmail.com,jaquerifr@gmail.com, mmjunior@uepg.br, jonathan@uepg.br, rfalate@uepg.br

Abstract. *Our research group has been developing computational systems to assist in solving agricultural problems. Almost all systems were developed using the OpenCV library's support, using the Java language. However, in 2018, it was announced that the most widely used programming language in different fields of application, including the scientific field, is Python. Given this scenario, this work aimed to evaluate the Python language in image processing. For this purpose, a seed analysis computational system developed in the Java was reimplemented in Python, and in order to carry out a practical evaluation, we performed execution time experiments on both codes.*

Keywords. *Seeds, digital image processing, programming languages, comparison.*

1. INTRODUCTION

Several researchers have developed digital image processing systems using different platforms and technologies, with the predominant use of the OpenCV library. This library uses open-source technology and was initially written in the C/C++ language but has now been adapted for other languages, such as Java [1] and Python [2]. Python, in particular, has become the most widely used programming language in computer programs, entertainment, scientific applications, and, in future, even in embedded devices [3].

The smallest element of an image is called a pixel, and image analysis follows a sequence of stages proposed by [4] for information extraction: acquisition, preprocessing, segmentation, feature extraction, and recognition and interpretation. All these stages directly depend on the type of problem to be solved. The fact is that many of these stages require processing since data manipulation occurs at each pixel of the image.

In light of this, to evaluate the Python language in image processing, this work proposed to migrate part of a computational system developed in the Java language for seed analysis [5] to the Python language. To evaluate the proposed study, we analysed the execution time of the proposed languages (Java and Python) in applying Digital Image Processing (DIP) techniques.

The languages defined for study in this paper, Java and Python, have been competing in the scientific area, and programming and development industry for years [3], [6]. Java can be defined as a general-purpose, object-oriented programming language designed to develop various applications. On the other hand, Python is a simple, high-level, general-purpose programming language designed for artificial intelligence, machine learning, and web development, among other things [7]. The main differences between the two languages are [7]:

Language Type: Java is strictly object oriented, implementing the concepts of inheritance, polymorphism, abstraction and encapsulation. Python supports concepts of object oriented programming, but is more flexible, also allowing structured coding.

Code: Java uses a virtual machine (Java Virtual Machine - JVM), so the code is compiled to a platform-independent byte code. Running programs in the JVM is safer, faster and more memory robust than Python, that is an interpreted language. Java is a strong type language, meaning that all variables and objects need to have their data type declared. Python is a weakly typed language, so the programmer does not need to declare types of variables. Typing variables makes programming more robust and memory efficient.

Syntax: Python requires less code compared to other languages while Java is more verbose. This characteristic of Python may hinder beginner users to migrate from Python to more complete and enterprise languages. Python does not require some code formatting like semicolons and braces, relying in identation, contributing to its low verbosity and making it easier for beginners.

Databases: JDBC (Java Database Connectivity) is much more active than Python's database connectivity, making Java the popular choice among developer programmers.

Practical Agility: Java is more famous for mobile and web applications, while Python is used for scientific data and development operations. Java has been used in more proven scenarios and has an excellent reputation, while Python is used in more experimental scenarios within a growing technological movement. Python can easily embed libraries written in C/C++, thus it has a large repository of libraries and functionalities, allowing programmers to develop some applications with less coding. As aforementioned, Java is in the market for more time, so the Java written library repositories are also vast.

Simplicity: Python is the most commonly used language by beginners due to its structure, which generally results in shorter and easier-to-handle code.

Both languages have their characteristics. In techniques involving DIP, the Java language has been the most widely used, to the best of our knowledge, while Python is currently more in demand due to its ease of use and structure.

The general objective of this project is to compare the existing algorithms in Java with those implemented in the Python language to evaluate the performance of the studied language.

Besides this Introduction Section, Section 2 describes the material used in this work and the reasons for their choices; Section 3 explains the methods used and describes the experimental setup; Section 4 shows and explains the results obtained with the experiments and finally, Section 5, there is the conclusion about this work.

2. MATERIALS AND METHODS

The work was developed using the Python programming language, version 3.7. The OpenCV library, version 3.4.9, was used to support image loading, creation, and application of necessary filters and functions.

The metric chosen for comparison and analysis of the algorithms was the execution time on the same machine.

The library or image database used was the same as the one used by the algorithm coded in the Java language when it was created. Figure 1 shows one of the images used to perform the algorithm tests.

The main functions in both algorithms were the following segmentation functions: segmentaRGB, segmentaHSV, and segmentaCieLAB.

Each function is designed for image segmentation using different color spaces such as RGB, HSV, and CieLAB. The original purpose of the Java code was to test and determine which of these functions would provide the most accurate segmentation.

For the development of the Python algorithm, we divided the project into the stages shown in the flowchart presented in Figure 2. The initial stage involved studying the techniques used in the original Java code and understanding how to implement them in Python using the OpenCV library. The final step of Python methodology involved finalizing the code implementation and conducting experiments to obtain the performance information to compare it with Java results.

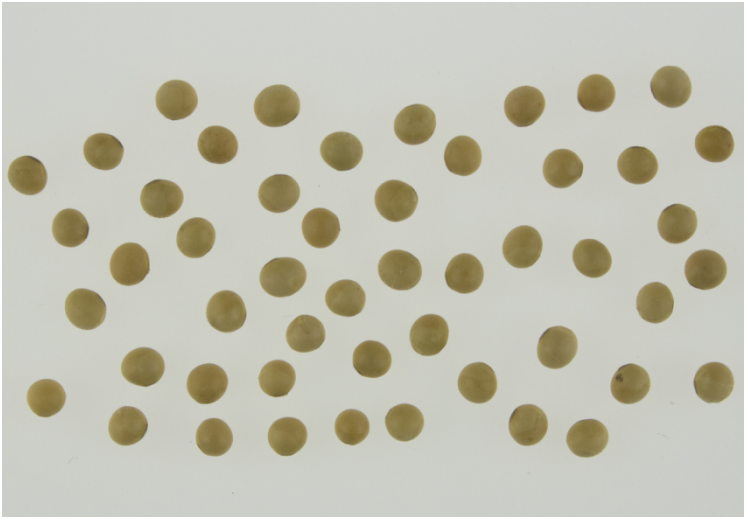

Figure 1. Image of soybean seeds from the chosen image database for algorithm testing.

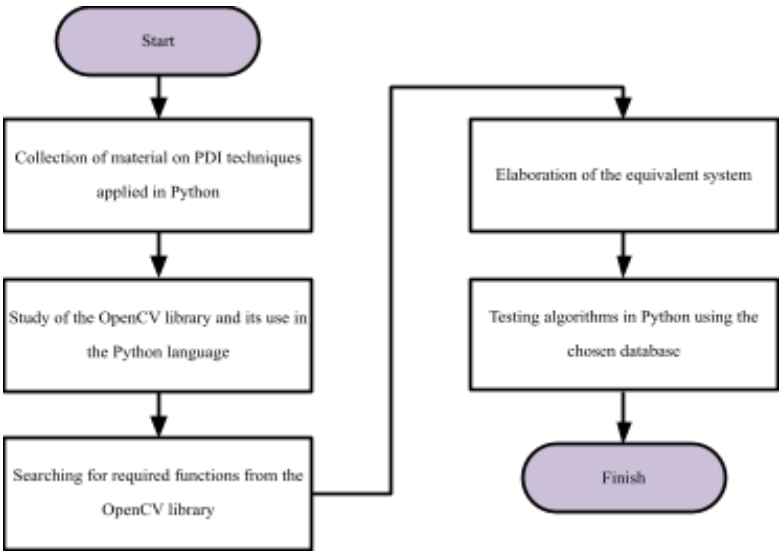

Figure 2. Flowchart of the stages in the construction of the Python algorithm.

3. RESULTS

Figure 3 presents a comparison between a segmentation function in both languages. Figure 3(a) shows that function in Python, while Figure 3(b) shows it in Java. Although the image displays the segmentaHSV function in both languages, the Java example is partially shown in the image, as it required more lines of code to complete it. In contrast, the Python function is complete in the image.

In comparing to the final versions of the algorithms, the Python language required approximately 200 lines of code, whereas the Java language required approximately 2000 lines. However, this information should not be taken as a measure of performance comparison since both languages have their ways of being read and interpreted by the computer. Therefore, the number of lines should not be used as a performance analysis metric.

PYTHON

```python
def segmenta_hsv(imagem, name):
    print("HSV")
    print("  Segmentação")
    initial_time = time.time()
    img_hsv = cv2.cvtColor(imagem, cv2.COLOR_RGB2HSV)
    img_hsv_save = cv2.cvtColor(img_hsv, cv2.COLOR_Lab2LRGB)
    if SALVA_IMAGENS:
        cv2.imwrite(f"imagens/processadas/{name}_img_hsv.png", img_hsv)
        cv2.imwrite(f"imagens/processadas/{name}_img_hsv_save.png", img_hsv_save)

    frame_h, frame_s, frame_v = cv2.split(img_hsv)

    h = otsu_threshold(frame_h, name, "HSV_h")
    s = otsu_threshold(frame_s, name, "HSV_s")
    v = otsu_threshold(frame_v, name, "HSV_v")

    image_otsu_range = cv2.bitwise_or(h, s)
    image_otsu_range = cv2.bitwise_or(v, image_otsu_range)

    if SALVA_IMAGENS:
        cv2.imwrite("imagens/processadas/image_HSV_otsu_range.jpg", image_otsu_range)
    contours = encontra_contornos(image_otsu_range, imagem, "HSV")

    image_merged_lab = cv2.merge((h, s, v))
    image_and = cv2.bitwise_and(image_merged_lab, imagem)

    print(f"Segmentação levou {time.time() - initial_time} segundos")

    if SALVA_IMAGENS:
        cv2.imwrite("imagens/processadas/merged_hsv.jpg", image_and)

    area_total, quantidade = corta_imagens(contours, imagem, image_and, name, "HSV")
    print(f"Area Total: {area_total}")
    print(f"Total: {quantidade}")
```

JAVA

```java
public void segmentaHsv(IplImage imagem, String caminho, String name) {
    System.out.println("HSV");
    int[] histogramaBin = new int[256];
    int h, s, v;
    //segumentação
    IplImage imghsv = imagem.clone();
    cvCvtColor(imagem, imghsv, CV_RGB2HSV);

    IplImage hsvsave = imghsv.clone();
    cvCvtColor(imghsv, hsvsave, CV_HSV2RGB);
//      cvSaveImage(caminho + "processadas\\" + name + "_hsv.png", hsvsave);

    System.out.println("    Segmentação");
    IplImage frame_h = cvCreateImage(imagem.cvSize(), IPL_DEPTH_8U, 1);
    IplImage frame_s = cvCreateImage(imagem.cvSize(), IPL_DEPTH_8U, 1);
    IplImage frame_v = cvCreateImage(imagem.cvSize(), IPL_DEPTH_8U, 1);

    cvSplit(imghsv, frame_h, frame_s, frame_v, null);

    h = otsuTreshold(frame_h);
//      cvSaveImage(caminho + "processadas\\" + name + "_hsv-h.png", frame_h);
    s = otsuTreshold(frame_s);
//      cvSaveImage(caminho + "processadas\\" + name + "_hsv-s.png", frame_s);
    v = otsuTreshold(frame_v);
//      cvSaveImage(caminho + "processadas\\" + name + "_hsv-v.png", frame_v);

    //binariza os canais
    IplImage imgOtsuh = cvCreateImage(imghsv.cvSize(), IPL_DEPTH_8U, 1);
    cvThreshold(frame_h, imgOtsuh, h, 255, CV_THRESH_BINARY);

    histogramaBin = calculaHistograma(imgOtsuh);
    if (histogramaBin[255] > histogramaBin[0]) {
      cvNot(imgOtsuh, imgOtsuh);
    }
//      cvSaveImage(caminho + "processadas\\" + name + "_hsv-bin-h.png", imgOtsuh);

    IplImage imgOtsus = cvCreateImage(imghsv.cvSize(), IPL_DEPTH_8U, 1);
    cvThreshold(frame_s, imgOtsus, s, 255, CV_THRESH_BINARY);

    histogramaBin = calculaHistograma(imgOtsus);
    if (histogramaBin[255] > histogramaBin[0]) {
      cvNot(imgOtsus, imgOtsus);
    }
//      cvSaveImage(caminho + "processadas\\" + name + "_hsv-bin-s.png", imgOtsus);

    IplImage imgOtsuv = cvCreateImage(imghsv.cvSize(), IPL_DEPTH_8U, 1);
    cvThreshold(frame_v, imgOtsuv, v, 255, CV_THRESH_BINARY);

    histogramaBin = calculaHistograma(imgOtsuv);
    if (histogramaBin[255] > histogramaBin[0]) {
      cvNot(imgOtsuv, imgOtsuv);
    }
//      cvSaveImage(caminho + "processadas\\" + name + "_hsv-bin-v.png", imgOtsuv);

    //faz o merge dos canais binarizados como o InRange
    IplImage imgInRangehsv = cvCreateImage(imghsv.cvSize(), 8, 1);

    cvOr(imgOtsuh, imgOtsus, imgInRangehsv);
    cvOr(imgOtsuv, imgInRangehsv, imgInRangehsv);
//      cvSaveImage(caminho + "processadas\\" + name + "_hsv-bin.png", imgInRangehsv);

    //contorno
```

Figure 3. Comparison between the segmentaHSV segmentation function in: (a) Python and (b) Java.

Tables 1 and 2 present, respectively, the measured values in seconds of the execution time for segmentation functions in different color spaces (RGB, HSV, CieLAB), and the average values of the total execution time in seconds for the algorithms.

Table 1. Measured execution time values in seconds for segmentation functions in different color spaces.

Function Name	Execution time of Java Language (s)	Execution time of Python Language (s)
segmentaRGB	1.864	1.119
segmentaCieLAB	3.008	1.856
segmentaHSV	3.157	1.603

Table 2. Average values of the measured execution time in seconds for the complete execution of the algorithms in each language.

	Java Language	Python Language
Average total execution time (s)	4.581	2.141

The Python language outperformed Java, indicating that Python has emerged as an excellent tool for image processing techniques.

However, it cannot be generalised that Python will be more efficient in all areas of image processing, as further in-depth studies in various image processing domains would be necessary to validate such claims. Nonetheless, Python has been gaining more popularity over the years due to its performance and user-friendly nature, ease of learning, and wide range of libraries that facilitate application development [3], [8].

One factor that influenced Python's performance was the simplicity of implementing most functions through the OpenCV library. The library already provided many of the required functions in the version used in this paper compared to the JavaCV plugin used in the Java algorithm development.

5. CONCLUSIONS

Python has become the most widely used language for image processing and machine learning development due to its ease of use and learning. The results of this paper demonstrate that Python outperformed Java in terms of execution performance for image processing algorithms, specifically in segmentation. Further studies should be conducted to assess its performance in other domains of image processing and even in segmentation, with more comprehensive tests involving algorithms of equal or greater complexity to validate this claim.

ACKNOWLEDGEMENTS

To Fundação Araucária for the financial support.

REFERENCES

[1] NAME, M. H.; RIBEIRO, S. S.; MARUYAMA, T. M.; VALLE, H. DE P.; FALATE R.; VAZ, M. S. M. G. Metadata Extraction for Calculating Object Perimeter in Images. *IEEE Latin America Transactions*, v. 12, p. 1566-1571, 2014.

[2] KURTULMUŞ, F.; KAVDIR, I. Detecting corn tassels using computer vision and support vector machines. *Expert Systems with Applications*, v. 41, n. 16, p. 7390-7397, 2014.

[3] CASS, S. *The Top Programming Languages 2023*. IEEE Spectrum, 2023. Disponível em: <https://spectrum.ieee.org/the-top-programming-languages-2023>. Acesso em: 24 out 2023.

[4] MARQUES FILHO, O.; VIEIRA NETO, H. *Processamento Digital de Imagens*. Rio de Janeiro: Brasport, 1999.

[5] FRANCO, J. R. *Método Computacional para Identificação do Fungo Cercospora Kikuchii em Sementes de Soja*. Dissertação (Mestrado em Computação Aplicada) – Setor de Ciências Agrárias e de Tecnologia. Universidade Estadual de Ponta Grossa, Ponta Grossa, 111 f, 2017.

[6] SHIVANGI; SAH, R. K. .; SHREEYAM; FLORANCE, G.; NIRMALA, M. CNNCalc – An Implementation of a Handwritten Mathematical Equation Solver, 2023. In: *7th International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, pp. 638-645, mai. 2023. doi: 10.1109/ICICCS56967.2023.10142278.

[7] KHOIROM, S.; SONIA, M.; LAIKHURAM, B.; LAISHRAM, J.; SINGH, T. D. Comparative analysis of Python and Java for beginners. *International Research Journal of Engineering and Technology*, *7*(8), 4384-4407. 2020.

[8] Data Science Academy. *Por que a linguagem Python é tão popular em machine learning e inteligência artificial?* 5 dez. 2020. Disponível em: https://blog.dsacademy.com.br/por-que-a-linguagem-python-e-tao-popular-em-machine-learning-e-inteligencia-artificial/. Acesso em: 24 out. 2023.