IMAGE PROCESSING DSL

Elena NIDELCU[*], Daniela VORNIC, Felicia NOVAC, Nikita TABANSCHI

Department of Software Engineering and Automatics, FAF-222, Faculty of Computers, Informatics, and Microelectronics, Technical University of Moldova, Chișinău, Republic of Moldova

*Corresponding author: Elena Nidelcu, elena.nidelcu@isa.utm.md

Coordinator: **Dumitru CREȚU**, university assistant, Technical University of Moldova

Abstract. *This paper introduces a domain-specific language (DSL) for image processing, that will overcome the limitations of existing tools in batch processing and automation, crucial for data science and machine learning applications. Unlike traditional image manipulation software that requires extensive programming knowledge or fails to efficiently handle multiple files, this DSL simplifies complex operations, enabling seamless batch processing of images. Its intuitive syntax makes it a useful tool for data preprocessing, a vital step in machine learning model development. This DSL stands out by offering a terminal-based interface, which significantly reduces resource consumption, making it accessible on lower-end hardware. This approach not only democratizes advanced image processing tasks but also aligns with the needs of data science professionals, facilitating their workflows without the steep learning curve typically associated with image processing libraries.*

Keywords: *image editing, batch processing, data science, ANTLR.*

Introduction

Image processing is pivotal across diverse sectors, notably in media and machine learning, where efficient data preprocessing is essential. Current tools for image manipulation often fall short in batch processing capabilities and automation, posing challenges to streamlined data handling. Addressing these gaps, this paper presents a novel domain-specific language designed expressly for image processing. This DSL stands apart from conventional software by offering simplified operation commands and robust batch processing features, significantly improving data preprocessing efficiency vital for developing machine learning models.

Role of a DSL for image processing

In the realm of image processing, a DSL plays an important role in addressing the complex challenges and requirements inherent to this domain. A DSL is a programming language specifically designed to tackle a narrow set of problems within a particular domain, offering specialized syntax, semantics, and features tailored to the specific needs of that domain [1]. HIPA^{cc} [2] and magick [3] are other DSLs which showcase the diversity of tools available for image processing tasks. The suggested DSL differs from other libraries in Unix terminals given the folder processing features and the domain-specific focus for data science tasks, that often involve a series of intricate steps, such as filtering, segmentation, feature extraction, and analysis [4].

With a DSL, developers can abstract away the low-level details of these operations, enabling them to focus on the logic and algorithms. In addition, a DSL makes sophisticated image processing capabilities more accessible to a wider audience, including data scientists, researchers, and enthusiasts with varying levels of expertise. Furthermore, the proposed DSL is intuitive because of its clear and concise commands. By providing a focused and accessible programming environment, it provides help in handling complex image data with ease.

Language Overview

The computational model of this domain-specific language is designed with an imperative and command-driven framework to simplify image processing tasks, making them accessible via command-line interfaces. Furthermore, the model supports pipeline processing, enabling the chaining of commands for complex transformations on the same images, thereby enhancing the tool's versatility for various image processing workflows.

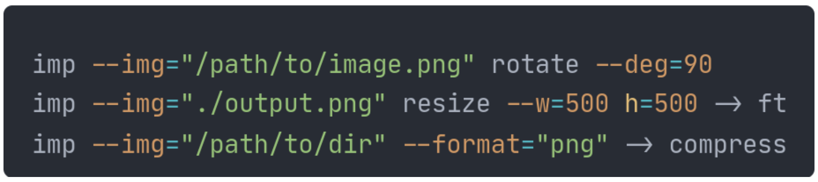
In terms of data handling, the language employs strings for specifying image names and command parameters, alongside numerical values for defining dimensions and adjustment levels. Input consists of the system paths to digital images, while the output consists of the modified pictures. Additionally, effective error detection is a crucial component, ensuring resource efficiency, validating user inputs, and providing feedback for troubleshooting.

Commands

In the suggested DSL, functions, which are alternatively referred to as actions or commands, encompass all the possible image modifications that can be applied to the declared image or folder of images. Below are the actions to be implemented:

- imp – initiates a new image processing command sequence with the specified image;
- crop – crops the image to a specified rectangle;
- convert – converts the image to a different format;
- rotate – rotates the image by a specified number of degrees;
- resize – changes the size of the image to the specified width and height;
- flipX – flips the image horizontally;
- flipY – flips the image vertically;
- bw – converts the image to black and white;
- colorize – applies a color filter over the image;
- contrast – adjusts the image contrast;
- brightness – adjusts the image brightness;
- negative – inverts all colors of the image;
- blur – applies a blur effect to the image;
- sharp – sharpens the image;
- compress – compresses the image file to reduce size;
- ft – performs a Fourier transform on the image;
- threshold – applies a threshold filter to the image;
- reduceNoise – reduces image noise;
- remBg – removes background;
- help – displays help information for commands.

To exemplify, Fig. 1 highlights examples of syntactically correct commands.



```
imp --img="/path/to/image.png" rotate --deg=90
imp --img="./output.png" resize --w=500 h=500 → ft
imp --img="/path/to/dir" --format="png" → compress
```

Figure 1. Examples of correct DSL commands

Reference Grammar

The reference grammar for the proposed DSL is depicted in Fig. 2, along with the notations used. Each command is described using a combination of terminal and non-terminal symbols, allowing a sequence of image manipulation actions to be concisely expressed. The core of the grammar is defined by the <img_command_sequence>, which outlines a sequence of image processing operations. Each operation, represented by the non-terminal <img_command>, can perform a variety of actions.

Symbol	Meaning
<foo>	non-terminal.
foo	terminal.
[x]	zero or one occurrence of x.
x*	zero or more occurrences of x.
x+	one or more occurrences of x.
|	separates alternatives.

```
                    S → imp <command>
            <command> → --img=<image_arg> <img_command_sequence> | help
<img_command_sequence> → <img_command> | <img_command> -> <img_command_sequence>
        <img_command> → crop --x=<int> --y=<int> --w=<int> --h=<int>
                      | convert --format=<image_type>
                      | rotate --deg=<int>
                      | resize --w=<int> --h=<int>
                      | flipX
                      | flipY
                      | bw
                      | colorize
                      | contrast --lvl=<int>
                      | brightness --lvl=<int>
                      | negative
                      | blur --lvl=<int>
                      | sharpen --lvl=<int>
                      | compress
                      | ft
                      | threshold--lvl=<int>
                      | reduceNoise
                      | remBg
          <image_arg> → <file_path> | <folder_path>
          <file_path> → "alpha_num*.image_type"
        <folder_path> → "alpha*"
         <image_type> → png | jpg | jpeg | bmp | gif | tiff | webp
                <int> → <digit>+
          <alpha_num> → <alpha> | <digit>
              <alpha> → a | b | . . . | z | A | . . . | Z
              <digit> → 0 | 1 | ... | 9
```

Figure 2. Reference Grammar of the DSL

Language Recognition

Following the establishment of the grammar, language recognition becomes an essential phase, ensuring that the syntactical structures outlined are adhered to during command execution. This process is underpinned by the integration of ANTLR, which transcribes the specified grammar into a parser that interprets the input commands [5]. The ANTLR v4 extension for Python, particularly its "Preview" component, plays a crucial role in this phase by enabling the generation and visualization of parse trees.

The language recognition stage consists of the Lexer, which serves as the initial filter and transforms the input into a series of tokens, and the Parser which organizes these tokens into a parse tree. This tree represents the hierarchical syntactic structure of the command, enforcing the grammar's rules. To demonstrate this, the following command designed for the DSL – imp --img="image.png" crop --x=100 --y=100 --w=200 --h=200 -> convert --format=html -> rotate --deg=90, aims to enact a sequence of image processing actions which include cropping an image, attempting a format conversion, and applying a rotation.

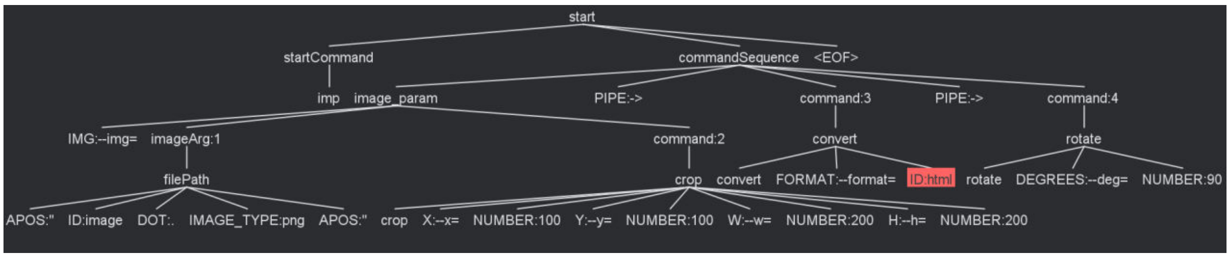


Figure 3. Example of a Parse Tree

Fig. 3 depicts the parse tree generated from the command. The structure elucidates the parsing process, showing each operation as a branch in the tree. It also reveals a critical error at the conversion operation where --format=html is flagged. This error is a direct result of 'html' not being a defined image format within the DSL grammar, illustrating the system's ability to validate inputs against the grammar and provide feedback on syntactic correctness.

The development of this DSL is currently ongoing, with Python chosen as the primary language for implementation. This decision leverages Python's extensive array of libraries, particularly those dedicated to the image manipulation field.

Conclusions

In conclusion, the development of a domain-specific language for image processing represents an advancement in the field, particularly for data science and machine learning applications. By addressing the limitations of existing tools in batch processing and automation, this DSL streamlines complex operations and enables efficient data preprocessing. Its integration with Python and terminal-based interface makes it accessible and user-friendly, democratizing advanced image processing tasks and facilitating workflows. Moving forward, further research and development in DSLs for image processing holds promise for enhancing the efficiency and accessibility of image manipulation tasks.

References

[1] *What are Domain Specific Languages (DSLs)?* [Online] [Accessed: 29.03.2024]. Available: https://www.jetbrains.com/mps/concepts/domain-specific-languages
[2] *Hipacc/Hipacc.* [Online] [Accessed: 01.04.2024]. Available: https://github.com/hipacc/hipacc
[3] *ImageMagick – Command-Line Tools: Magick.* [Online] [Accessed: 01.04.2024]. Available: https://www.jetbrains.com/mps/concepts/domain-specific-languages
[4] Gonzalez, Rafael C, and Richard E Woods. 2018. *Digital Image Processing*.
[5] *About the ANTLR Parser Generator.* [Online] [Accessed: 29.04.2024]. Available: https://www.antlr.org/about.html