## 

## Character Recognition GUI in MATLAB

~PROJECT DOCUMENTATION~

Student: Musteata Denisa Gabriela

Faculty of Automatic Control and Computer Engineering

Contents

1. Project Overview ………………………………………………………………………………………. 3

2. Step-by-Step Description …………………………………………………………………………… 4

2.1. Creating the GUI Window ………………………………………………………………. 4

2.2. Adding Buttons and UI Elements …………………………………………………….. 4

2.3. Functionality ……………………………………………………………………………….. 5

3. Key MATLAB Functions Used ………………………………………………………………………. 6

4. Results ……………………………………………………………………………………………………. 7

5. Conclusion ……………………………………………………………………………………………… 9

6. Biblyography ……………………………………………………………………………………………. 9

1 Project Overview

This project implements a detailed graphical user interface (GUI) in MATLAB, designed for automatic text recognition from digital images using Optical Character Recognition (OCR) technology. The application provides flexibility for the user to either load an existing image from the computer or capture a new image in real-time using a connected webcam. Once an image is loaded or captured, the user can process it to extract and display the recognized text directly in the GUI.

The core of the project is built around a main script named imageGUI. This script is responsible for constructing and managing all GUI components, handling user interactions through button callbacks, and integrating MATLAB’s built-in image processing and OCR capabilities. Beyond simply reading and displaying text, the GUI also offers a user-friendly design, combining clear visual elements (buttons, image axes, and text area) with intuitive functionality, making the entire recognition process seamless and accessible to users without requiring programming knowledge.

Overall, this project demonstrates how MATLAB can be used not only for advanced numerical computation but also for building interactive applications that bridge image acquisition, processing, and automatic text extraction.

Objective: - Developing a computer vision system for recognizing characters written on a white paper.

Steps:

1. Creating a GUI for image acquisition
2. Image analysis and algorithm design
3. Character recognition
4. Visualizing the results in the GUI

2 Step-by-Step Description

2.1. Creating the GUI Window

The project starts by creating a GUI window using the uifigure function:

* fig = uifigure('Name', 'Character Recognition');

The GUI contains:

* Three buttons:
  + **Select Image** (Selecteaza imagine)
  + **Capture Image** (Captureaza imagine)
  + **Process Image** (Procesare imagine)
* An **axes** (uiaxes) for displaying the selected or captured image.
* A **text area** (uitextarea) for displaying the recognized text.

Each button is styled with specific background colors and font sizes to improve the visual appearance.

2.2 Adding Buttons and UI Elements

The buttons are created with uibutton and linked to specific callback functions through the ButtonPushedFcn property:

* btnS = uibutton(fig, 'push', 'Text', 'Selecteaza imagine', ..., 'ButtonPushedFcn', @(btn,event) selectImage());
* btnC = uibutton(fig, 'push', 'Text', 'Captureaza imagine', ..., 'ButtonPushedFcn', @(btn,event) captureImage());
* btnProcess = uibutton(fig, 'push', 'Text', 'Procesare imagine', ..., 'ButtonPushedFcn', @(btn,event) processImage());

Axes for displaying images:

* ax = uiaxes(fig, 'Position', [50 200 500 200]);

Text area for showing recognized text:

* **txt = uitextarea(fig, 'Position', [100 100 350 100]);**

2.3 Functionality

**selectImage()**

* Opens a file picker dialog using uigetfile to select an image from the computer.
* Reads the selected image with imread.
* Displays the image using imshow inside the GUI axes.
* **[filename, filepath] = uigetfile({'.jpg;.jpeg;\*.png'}, 'Selecteaza o imagine');**
* **img = imread(fullfile(filepath, filename));**
* **imshow(img, 'Parent', ax);**

**captureImage()**

* Initializes the webcam using webcam.
* Captures an image with snapshot.
* Displays the captured image in the GUI.
* Releases the webcam resource with clear cam.
* **cam = webcam(1);**
* **img = snapshot(cam);**
* **imshow(img, 'Parent', ax);**
* **clear cam;**

**processImage()**

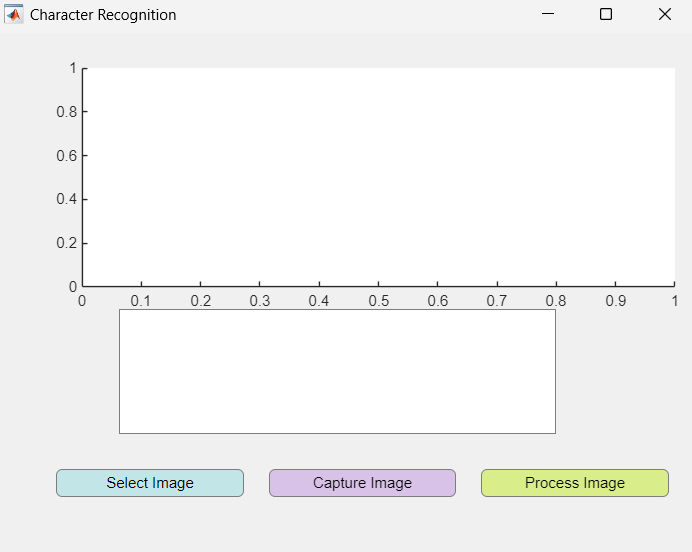
* Retrieves the currently displayed image using getimage(ax).
* Checks if the image is grayscale or RGB:
  + If RGB, converts it to grayscale with rgb2gray.
* Resizes the image using imresize to improve OCR accuracy:
* **grayImgB = imresize(grayImg, 3.5, 'nearest');**
* Performs OCR using MATLAB’s OCR function:
* **ocrResults = ocr(grayImgB);**
* **recognizedText = ocrResults.Text;**
* Displays the recognized text in the text area.

3 Key MATLAB Functions Used

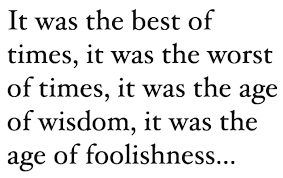
| Function | Purpose |
| --- | --- |
| uifigure | Creates the main GUI window. |
| uibutton | Creates clickable buttons. |
| uiaxes | Area to display images inside the GUI. |
| uitextarea | Displays multi-line text (recognized text). |
| uigetfile | Opens file picker dialog for selecting images. |
| imread | Reads the selected image file. |
| imshow | Displays the image in the GUI axes. |
| webcam | Connects to the system’s webcam. |
| snapshot | Captures an image from the webcam. |
| clear | Releases the webcam object. |
| rgb2gray | Converts RGB images to grayscale. |
| imresize | Changes the size of an image. |
| ocr | Performs optical character recognition. |
| getimage | Retrieves the image currently displayed in the axes. |

4 Results

* **Interface**



* **Test image**

****

* **Select image**

O imagine care conține text, captură de ecran, afișaj, Font

Conținutul generat de inteligența artificială poate fi incorect.

* **Process image**

**O imagine care conține text, captură de ecran, afișaj, software

Conținutul generat de inteligența artificială poate fi incorect.**

5 Conclusions

This project integrates user-friendly GUI design with MATLAB’s computer vision capabilities. It allows users to easily capture or select an image and extract text using OCR. The modular approach makes the code easy to understand and extend for future improvements.

In conclusion, this project not only demonstrates technical implementation but also emphasizes usability, maintainability, and potential extensions. It shows how GUI design and OCR can be combined to create practical applications. Additional diagrams, performance evaluation, and feature enhancements could further strengthen and enrich this work. Furthermore, it highlights the potential for adding more advanced features like language detection, handwriting recognition, and integration with databases. The conclusion also underlines the importance of code readability, modularity, and user experience in software design, making the project not only a technical demonstration but also a solid foundation for real-world applications and further research.

6 Bibliography

[1] Smith, R. (2007). "An overview of the Tesseract OCR engine." *Proceedings of the Ninth International Conference on Document Analysis and Recognition*.

[2] Breuel, T. M. (2008). "The OCRopus open source OCR system." *Document Recognition and Retrieval XV*.

[3] Simard, P. Y., et al. (2003). "Best practices for convolutional neural networks applied to visual document analysis." *ICDAR*.

[4] Chen, X., Yuille, A. (2004). "Detecting and Reading Text in Natural Scenes." *CVPR*.