




# COLOR-BLIND DETECTOR DOCUMENTATION



Hana Roubalová  
Summer Semester 2022/2023  
Individual Software Project NPRG045

## Obsah

User Documentation .....	2
Objective .....	2
System Requirements .....	2
Installation.....	2
Application interface .....	2
Usage .....	3
Loading image .....	3
Navigating folder .....	3
Choosing the type of color vision deficiency for analysis.....	3
Starting analysis.....	3
Downloading the analysis report .....	3
Limitations .....	3
Code Documentation .....	4
Input data .....	4
Code Overview .....	4
main.py.....	4
User interface .....	4
design.py .....	4
design.kv.....	4
Application logic .....	4
facade.py .....	4
color_grouping_algs.py .....	4
detection_algs.py .....	5
deltaE_distance_calculator.py .....	5
report_result_object.py .....	5
pdf_generator.py.....	5
report_template.j2.....	5
Experiments.....	5
dataset_generator.py .....	5
evaluation.py .....	5
System Requirements: .....	5

# User Documentation

## Objective

The Color-Blind Detector application aims to help users determine if a graph image is colorblind-friendly. It analyzes the colors in the image and checks if they are distinguishable for people with different types of color blindness.

## System Requirements

- Windows Operating System
- Python version 3.7 or higher

## Installation

Follow these steps to install and set up the application:

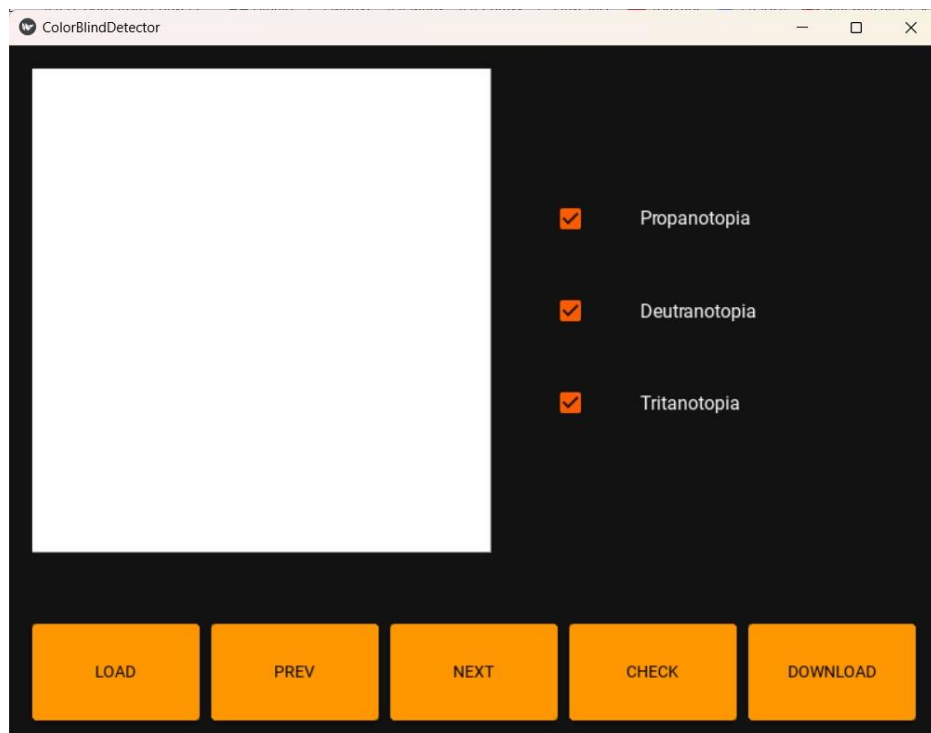
1. Download the Color-Blind Detector package from [download link] **TODO!!**.
2. Extract the package to a desired location on the computer.
3. Open a terminal or command prompt.
4. Navigate to the extracted package directory.
5. Install the required Python packages by running the following command:

```
pip install -r requirements.txt
```

6. Once the installation is complete, launch the application.

## Application interface

The Color-Blind Detector application provides a user-friendly graphical interface. After launching the application, the main screen will appear with five buttons for controlling the application and three checkboxes for selecting the color vision deficiency.



## Usage

### Loading image

Press the **Load** button and find the desired image in the file manager. Selecting a folder of images is also possible by pressing the check button while in the file manager. In this case, the first image in the selected folder will display.

### Navigating folder

To navigate the images, use the Prev and **Next** buttons. The **Prev** button will display the previous image in the folder, while the **Next** button will display the following image.

### Choosing the type of color vision deficiency for analysis

The checked boxes indicate the selected types of color vision deficiencies for analysis. Choose the desired combination by selecting appropriate checkboxes.

### Starting analysis

Pressing the **Check** button will start the analysis, which may take a while. After the analysis, if the image is unfriendly, a warning will display next to the corresponding color vision deficiency. If no warnings appear, the picture is color-blind friendly.

### Downloading the analysis report

After the analysis, download the pdf report about the current image by pressing the **Download** button. The report will show the original image and a simulated image for each triggered deficiency.

### Limitations

The application assumes that the input image is in a standard PNG or JPEG format. It is not excessively large or complex. And that the image is of a single graph.

The application has no memory. When loading the same image, the analysis must be performed again.

# Code Documentation

## Input data

The input for this application is an image of a single graph in a JPEG or PNG format, selected by the user using the file manager. Input can also be a user-selected folder with images.

To test this application, one can utilize the `TestFiles` directory of prepared images or the `GroupingDataset` directory containing generated graph images.

## Code Overview

The application consists of three main modules. The first module handles the graphical user interface. Second takes care of the underlying application logic, which is separated from the user interface module by a Facade. The third module handles the experiments performed to compare the performance of different implementations of the algorithms from the application logic module.

### `main.py`

The `main.py` serves as the entry point for the application. It initiates the execution of the Kivy application and launches the user interface.

## User interface

### `design.py`

The `design.py` defines the handling methods for events triggered in the user interface, including all the buttons pressed methods and navigation in the file manager.

### `design.kv`

Defines user interface design and layout of the Kivy application using a declarative syntax specific to the Kivy library.

## Application logic

### `facade.py`

The `facade.py` implements the Facade design pattern. It serves as a separation of the application logic from the user interface logic and as a simplified entry point to the application logic. Defines two main methods, `find_issues`, and `generate_report`, that intermediate the communication with the application logic subsystem.

It also acts as the context for the Strategy design pattern that implements the color grouping algorithms. This functionality was used during the implementation phase. The best-chosen strategy is set in the constructor as the default in the final application.

### `color_grouping_algs.py`

Defines three distinct implementations for grouping colors of an image using the Strategy design pattern. Both the abstract Strategy class and the concrete implementations are located here.

The context and client code are placed in the `evaluation.py` file.

The concrete strategies of grouping colors are implemented using the K-means algorithm from the Sklearn library, Euclidean distance, and deltaE distance calculation which is implemented in the `deltaE_distance_calculator.py` file.

#### `detection_algs.py`

The `detection_algs.py` implements the analysis part of the application logic. Given grouped colors, it computes the distance for the original and the simulated color blindness colors. Then triggers a warning if some of the original distances have shrunk considerably in the simulated versions. The results are conveyed using the `report_result_object.py` objects. For each type of colorblindness, one report object is created that carries the information on whether the corresponding issue was triggered.

#### `deltaE_distance_calculator.py`

Calculates the CIE2000  $\Delta E$  distance for every pair of given colors using the `scipy` and `colormath` libraries. It returns the distances as an upper triangular matrix flattened to a list.

#### `report_result_object.py`

Represents warning objects. Stores information about the type of colorblindness and whether it was triggered or not. It also stores the simulated image for the corresponding colorblindness.

#### `pdf_generator.py`

The `pdf_generator.py` generates the pdf report based on the `report_result_object`. It first renders the `report_template.j2` to html with the given information and then converts the html to pdf using the `xhtml2pdf` library.

#### `report_template.j2`

The `report_template.j2` contains the jinja2 template for the pdf report generation.

### Experiments

#### `dataset_generator.py`

The method `generate_graph_images` generates the random test images using the `matplotlib` library and `genData` method adapted from

[https://github.com/ddecatu/VizExtract/blob/main/create\\_graph.py](https://github.com/ddecatu/VizExtract/blob/main/create_graph.py)

Additionally, the method `generate_dataset` creates a csv dataset containing the generated images and required information about them.

#### `evaluation.py`

Runs the experiments and calculates the evaluation metrics. It is implemented as a context to the Strategy pattern that implements color grouping algorithms from the `color_grouping_algs.py` file.

### System Requirements:

- **Windows** Operating System
- Python version **3.7** or higher

The application relies on the following external libraries:

- **Kivy**: Graphical interface development framework.
- **Colorblind**: Library for simulating color blindness and evaluating color distinguishability.
- **NumPy**: Computational and array manipulation library.
- **Scikit-learn (Sklearn)**: Machine learning library. Used for K-means algorithm and metrics.
- **Matplotlib**: Plotting library used for generating datasets.
- **Pandas**: Data manipulation library used for CSV file operations.
- **PIL** (Python Imaging Library): Image manipulation library.
- **SciPy** and **colormath**: Libraries used for calculating  $\Delta E$  distance.
- **Jinja2** and **xhtml2pdf**: Libraries used for generating PDF reports.