Machine Vision Project Documentation

Mitroi Bianca, Grupa 30235

May, 2023

1 Description

The machine vision project aims to develop a computer vision system that recognizes the colors of a Rubik's cube face.
The input is a picture captured from the device camera.
It is recommended that he room where the picture is taken has be illuminated with white light. The output is a matrix of colors detected and shown in console.

2 Related work and research

Link metoda Link webcam photos

3 Code Overview

The provided code is written in C++ and consists of multiple functions that perform different operations related to image processing and computer vision. Here is an overview of the main functions and their functionalities:

- `lab3_calchist`: Calculates the histogram of a given grayscale image.
- `lab2_binarization`: Converts a grayscale image to a binary image using a threshold value.
- `lab2_isInside`: Checks if a pixel coordinate is inside the image boundaries.
- `lab2_toGray`: Converts a color image to grayscale.
- `lab8_pragAutomat`: Calculates an optimal threshold value for image binarization using the Otsu method.
- `createTemplate`: Creates a binary template image with a specific structure.
- `countWhitePixels`: Counts the number of white pixels in a region of interest within an image.

- `color_detect`: Performs color detection on a pixel and determines its color category.
- `mask`: Applies a mask to a specific region in an image and performs color detection on the masked area.
- `continueing`: Continues the image processing workflow after the initial binarization step.
- `developing`: Implements the image processing pipeline, including denoising, edge extraction, binarization, and subsequent steps.
- `proiect`: Initiates the project by capturing video frames from a camera, saving an image, and starting the image processing pipeline.

4 Workflow and Functionality

The project follows the following workflow:

1. The `proiect` function captures video frames from a camera and displays them in a window.
2. Pressing the spacebar initiates the image processing pipeline by saving the current frame as "frame_0.png" and calling the `developing` function.
3. The `developing` function performs the following steps:
   (a) Converts the captured image to grayscale.
   (b) Applies denoising to reduce noise in the image.
   (c) Extracts edges using the Laplacian operator.
   (d) Calculates the histogram and determines an optimal threshold value for binarization.
   (e) Binarizes the image using the obtained threshold value.
   (f) Calls the `continueing` function to proceed with further processing.
4. The `continueing` function performs the following steps:
   (a) Creates a template image with a specific structure.
   (b) Iterates over different positions in the binarized image, comparing the template with the image region.
   (c) Counts the number of white pixels in each region and identifies the region with the maximum white pixel count.
   (d) Updates the template size and position based on the identified region.
   (e) Applies a mask to the region of interest and performs color detection on the masked area transforming the initial image from rgb to hsv taking the color from the center of the template squares.
5. The program continues capturing video frames and repeating the image processing pipeline until terminated.

5 Usage

To use the machine vision project, follow these steps:

1. Ensure that you have the OpenCV library installed on your system.
2. Compile the provided code using a C++ compiler, linking against the OpenCV library.
3. Run the compiled executable file.
4. A window will appear, displaying the video feed from the camera.
5. Press the spacebar to start the image processing pipeline.
6. The processed frames will be displayed in a separate window, showing the detected objects and their colors.
7. Continue capturing frames and processing them by pressing the spacebar.
8. To terminate the program, close the windows or press the "Esc" key.

6 Results

- In Figure 1, we have the image taken with device camera.
- In Figure 2, we have the binarized image.
- In Figure 3, we have the template put over the binarized image.
- In Figure 4, we have the template applied on the initial image transformed into hsv.
- In Figure 5, we have what console displays after processing. It shows the colors detected from each square at its position on cube, then the hsv codes for those colors

7 Conclusion

The machine vision project provides a comprehensive computer vision solution for image processing, object detection, and color recognition tasks. By utilizing the OpenCV library and implementing various algorithms, the project demonstrates the capabilities of computer vision in real-time applications.

Figure 1: Captured image
(Go back)

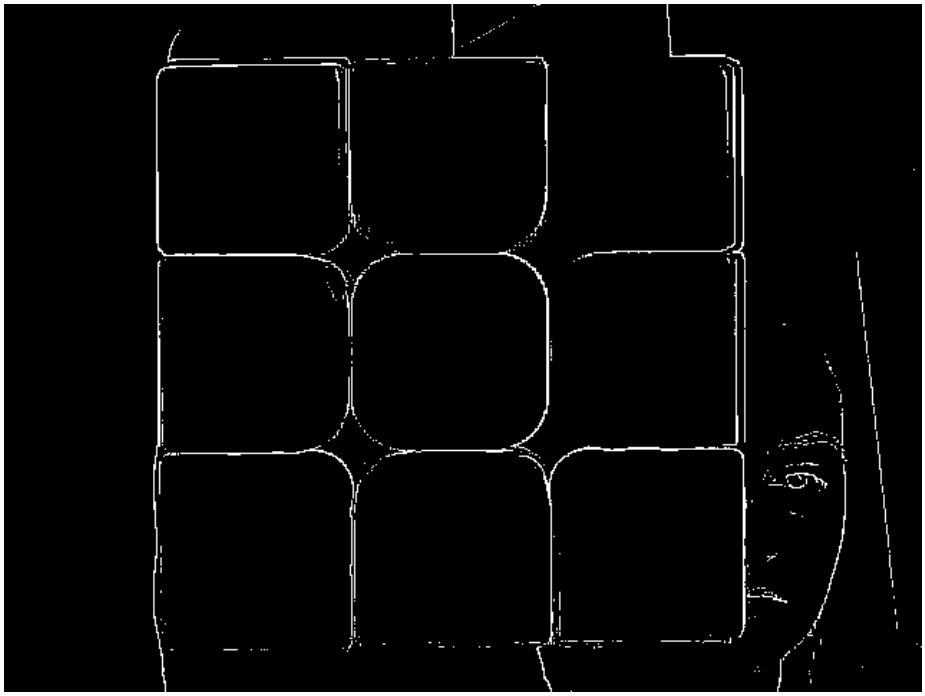

Figure 2: Binarized edge detection frame
(Go back)

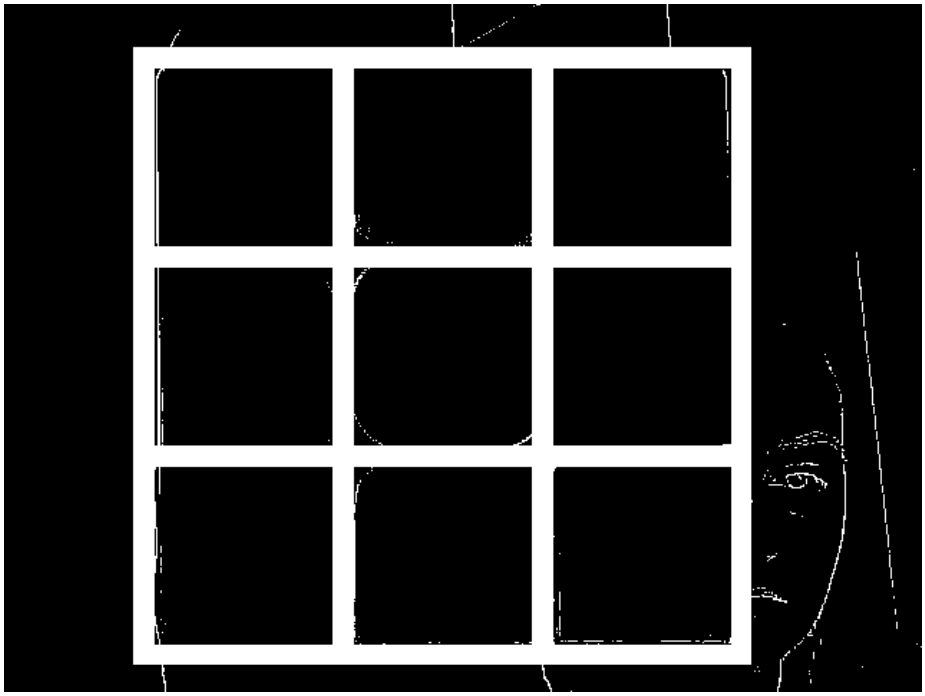

Figure 3: Template aplied on binarized image
(Go back)

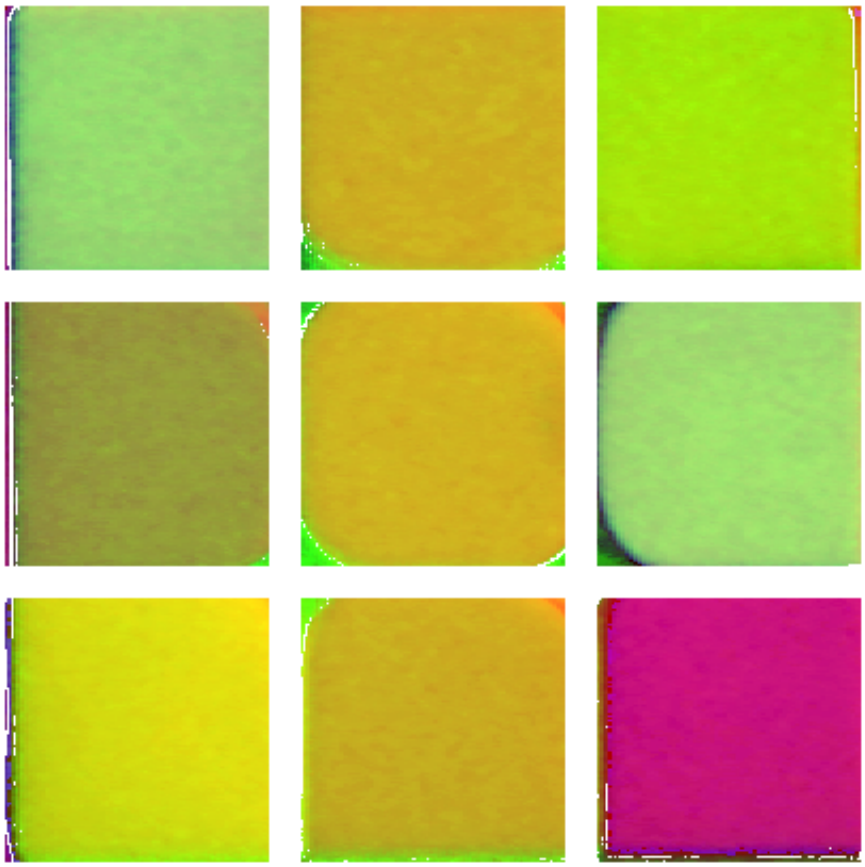

Figure 4: Colors passed to HSV representation
(Go back)

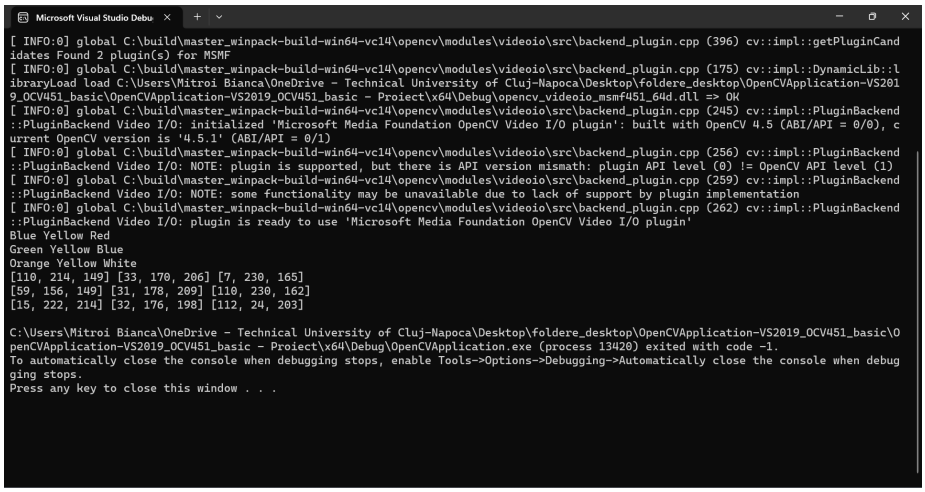

Figure 5: What console displays after detection
(Go back)