Image Processing Toolbox Documentation

CIS*4720 Image Processing and Vision

Tanya Nantsa
1101999

TABLE OF CONTENTS

SECTION 1 – USER'S MANUAL

1.1 What the Toolbox Does

In my image processing toolbox, I have implemented a graphical user interface where users of the toolbox are able to crop an image, flip images horizontally and vertically, scale an image using nearest-neighbour interpolation, rotate an image any number of degrees, and apply zero padding on the image.

In addition, my image processing toolbox enables users to edit and evaluate images in a variety of ways. This includes applying linear and power-law mappings to an image, calculating the histogram of an image, and performing histogram equalization, and determining the convolution of a 3 by 3 sized kernel with an image. Additionally, my toolbox allows users to carry out non-linear filtering with min, max, and median filtering.

One unique feature of my toolbox is edge detection of an image. Furthermore, my image processing toolbox handle coloured images along with greyscale image.

1.2 How to Use the Toolbox

To begin using my toolbox the user must ensure that they are in the project directory which is named "TanyaNantsa_Toolbox". The user must also ensure they have DearPyGui installed by typing 'pip install dearpygui' (this was the library used to create the graphical user interface). Then type 'python3 ImageToolBox.py' to launch the toolbox. The application will pop up in a new window.

To use the tools in the toolbox the user must first select an image from their own files to upload by pressing the "File Selector" button in the application. The image uploaded must be in one of the following formats: JPEG or PNG. When the image has successfully loaded the original image will pop up in a new window. The next step is to select which tool the user wants to use on the image by clicking the corresponding tool button in the application. Once the user has selected the tool, a new window will load where users can input the specifications for that tool.

Depending on the tool, the user will either input values or selects from a menu. After the use has clicked the button on the screen to apply the tool to their image, the new image will pop up in a new window and it will also be saved in the current project directory. To exit the specific tool's window, click the arrow located at the top right of the window. To exit the toolbox application, click the X in the top left corner.

1.3 What the Toolbox Does Not Do

My toolbox cannot calculate convolution with kernels other than of size 3 by 3. Also, the toolbox cannot do reflective indexed or circular indexing and it does not apply bicubic or bilinear interpolation when scaling an image.

SECTION 2 – TECHNICAL DISCUSSION

2.1 Crop Image Tool

To crop a selected image, the user must input the coordinates of the boundaries of the box they wish to crop to. These coordinates include the left, right, upper, and lower boundaries of the box. Once the user submits these inputs by clicking the button, a method is called which initializes a new image with the size of the cropped box and sets every pixel to black.

Next, the program iterates through every pixel within the cropped box' boundaries and applies the color of the pixel located at the same location in the original image to that location in the new image. This process ensures that the pixels in the new image correspond to the pixels of the original image within the cropped box.

After visiting and setting every pixel in the new image, the program saves and displays the newly cropped image to the user.

2.2 Flip Image Tool

To flip an image, the user selects whether they want to flip the image horizontally or vertically. A method is called which initializes a new image to be the same size as the original image and sets every pixel to black.

If the user chooses to flip the image horizontally, then the program iterates through each row of pixels in the original image and populates the corresponding row of the new image. The program iterates through the rows of the original image from left to right and populates the rows of the new image from right to left with the color of the pixel from the original image. Thus, each row of pixels appears in reverse order in the new image.

If the user chooses to flip the image vertically, then the program iterates through each column of pixels in the original image and populates the corresponding column of the new image. The program iterates through the columns of the original image from top to bottom and populates the columns of the new image from bottom to top with the color of the pixel from the original image. Thus, each column of pixels appears in reverse order in the new image.

After visiting and setting every pixel in the new image, the program saves and displays the flipped image to the user.

2.3 Scale Image Tool

To scale an image, the user inputs the scaling factor they wish to apply. Next, a method is called that initializes a new image with dimensions equal to the width of the original image multiplied by the scaling factor, and the height of the original image multiplied by the scaling factor. Then, the program iterates through every pixel in the new image and calculates the corresponding location in the original image by dividing the pixel's x and y coordinates by the scaling factor and rounding the result to the nearest integer value (nearest x and y locations). The program is performing nearest-neighbour interpolation. The program then sets the colour of the pixel in the new image to be the same as the color of the equivalent pixel in the original image.

After visiting and setting every pixel in the new image, the program saves and displays the scaled image to the user.

2.4 Rotate Image Tool

To rotate an image, the user inputs the degree of the angle they wish to rotate the image. Next, a method is called that initializes a new image to be the dimensions given from the rotation matrix formula. The program iterates through every pixel in the new image and computes the corresponding location from the original image with the inverse rotation matrix. If the corresponding location in not located out of bounds in the original image then colour at that pixel is set as the colour in the corresponding pixel in the new image.

After visiting and setting every pixel in the new image, the program saves and displays the rotated image to the user.

2.5 Zero Padding Tool

To apply zero padding to an image, the user inputs the padding size in pixels. Next, a method is called that initializes a new image to be the size of the original image plus then addition padding around all sides of the image and set every pixel colour to black. Then the program will copy the image to the middle of the new padded image. Finally, the program saves and displays the new image to the user.

2.6 Linear and Power-Law Mapping Tool

To apply linear grey-level mapping, the user can input values for the bias (a), which affects the brightness, and gain (b), which affects the contrast. Next, a method is called that initializes a new image to be the same size as the original image. If the original image is a colored image, it is converted to a grayscale image. Then, the program iterates through every pixel in the original image and applies the formula a*u + b to the grayscale value of the pixel, where u is the grayscale value. The resulting grayscale value is then set for the same pixel location in the new image. After setting the grayscale value for every pixel in the new image, the program saves and displays the new image to the user.

To apply power-law grey-level mapping to an image the user must input the gamma value the wish to use. The program then coverts the image to a greyscale image. Next, the program initializes a new image to be the same size as the original image. Then, the program iterates through every pixel in the original image and applies the mapping function to the grey level at that pixel location. The power-law mapping function used is: $255*(u/255)^{gamma}$, where is the grey level of the pixel. The grey level calculated is set as the colour at the corresponding pixel in the new image. After setting the grey-level value for every pixel in the new image, the program saves and displays the new image to the user.

2.7 Histogram Equalization Tool

To create the histogram of an image, the program iterates through every pixel and increments the correct bin in the histogram depending on the grey level of that pixel. The program then computes the cumulative distribution function for the histogram. It does this by totalling the values of each bin in the histogram. Then the program normalizes the cumulative distribution function to a range of 0-255 by taking each value and dividing it with the total amount of pixels within the image and multiplying by 255.

The program initializes a new image to be the same size as the original image. The program then applies histogram equalization through iterating over every pixel and mapping the grey level to the matching normalized cumulative distribution function value and setting the pixel's new grey level value. Finally, the program saves and displays the new image to the user.

2.8 Convolution Tool

To find the convolution of a 3 by 3 kernel with an image, the user inputs the values they wish to use in the kernel. The program initializes a new image to be the same size as the original image. Next, the program iterates through every pixel in the original image, excluding pixels located at the border. Then, the program calculates the convolution by calculating the sum of the neighbouring pixels (3 by 3 neighbours) multiplied by the corresponding kernel pixel value. The calculated sum is set for the corresponding center pixel in the new image. Once the program has iterated through all the pixels, the program saves and displays the new image to the user.

2.9 Edge Detection Tool

To find the edges of an image, the program uses Sobel kernels. First, the program converts the original image to a grayscale image and initializes a new image to be the same size as the original, with all pixels initially set to black. Then, the program iterates through all pixels in the image and, for each pixel, applies three different Sobel kernels to that pixel and its six neighboring pixels. The three different kernels used detect horizontal, vertical, and diagonal edges.

After, the magnitude of the gradient is calculated, if an edge is detected, a white pixel is added to the corresponding location in the new image. Finally, the program saves and displays the new image to the user, which highlights the edges of the original image.

SECTION 3 – DISCUSSION OF RESULTS AND FUTURE WORK

3.1 What my Toolbox Successfully Does

My toolbox has successfully implemented a graphical user interface which allows user to seamlessly interact with my application. Additionally, my toolbox successfully crops, flips, scales, and rotates an image.

Moreover, my toolbox can apply linear and power-law grey level mappings to images, accomplish histogram equalization, calculate convolution of any 3 by 3 kernel with an image. Users can also conduct non-linear filtering on an image and performs edge detection of an image.

3.2 What Doesn't Work in my Toolbox

When rotating an image, you cannot choose to rotate it in a different direction. Moreover, when implementing scaling in my toolbox, bilinear and bicubic interpolation do not work only nearest neighbour interpolation. Additionally, circular indexing and reflected indexing does not work in my toolbox.

3.3 How my Programs were Tested

My programs were tested manually with different test cases including out of bounds values. Additionally, to test my programs I used predefined library functions for the tools featured in my toolbox. I used those functions to see how those results compared to the results produced by my programs and evaluates whether my programs were correct and functioning as expected. For example, I tested my crop tool's functionality by comparing my results with the results from the crop() method in Pillow's Image library.

3.4 The Programs' Performance Strengths and Weaknesses

One of my programs' strengths is the ability it has to take user input and apply it to customize specific image processing tools to the user's desire. Additionally, my program is very user friendly so all types of users with any level of experience can use the toolbox successfully.

One of my program's weaknesses is regarding scaling. Since I used nearest neighbour to interpolate it can result in a loss of sharpness in the scaled image. Another weakness is that since many tools use single pixel operations, such as the crop, flip, scale, and rotate tools, very large image files will take longer to be processed. Moreover, some weaknesses relating to the edge detection program is that it is sensitive to noise which may result in the edges detected not actually being edges and it also has trouble detecting edges that do not go vertically, horizontally, and diagonally.

3.5 The Toolbox’s Future Functionality

Some functionalities I would implement in the future are bilinear interpolation for scaling an image to have produce sharper scaled images. Secondly, I would implement convolution with a rectangle of any size the user wishes. Moreover, I would add circular indexing and reflective indexing to my toolbox to increase the number of tools available to the user. Finally, I would change the current functionality of the zero-padding tool to allow users to customize how much padding they want to add on each side (top, bottom, left, and right) of an image.