Image Processing Project

CIS*4720

Due on Friday April 9, 2023 at 11:59pm

Professor Pascal Matsakis

Michael Dombrovsky
1040227

Contents

User Manual

This is meant to be an image processing toolbox, it is meant to do a wide array of image operations. The full capabilities of the toolkit can be split up into 5 categories as can be seen on the left side of the diagram; Basic image operations, Mapping image operations, Convolution and Indexing image operations, Histogram image operations, and Filtering image operations.

Each category of operation can be selected by clicking on it, which will reveal available operations.

Image Setup

Upload

To upload an image, click on the browse button and select the desired image.

Reset

To reset the state of the image click on the reset button.

Hide Secondary Display

To hide the secondary display click on the hide secondary display button. Please note the button is only able to be clicked when a Histogram is shown in the form of a secondary display below the image.

Basic image operations

Crop

To crop an image, fill in the amount of pixels you wish to crop from each side into the Top, Left, Bottom, Right boxes then click the crop button.

Scale

To scale an image, fill in the scale box with you desired scale factor, select your desired method of interpolation, and click the scale button.

Rotate

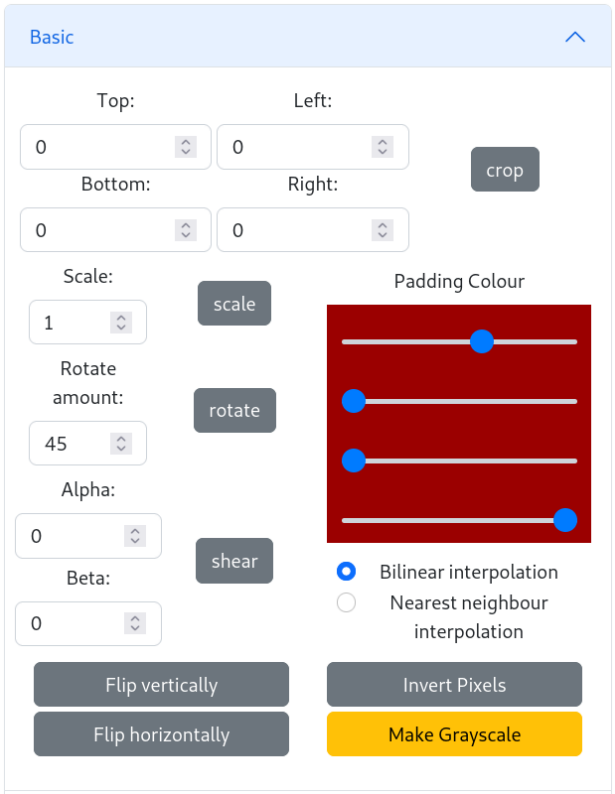
To rotate an image, fill in the rotate amount box with your desired rotation degrees, select your desired method of interpolation, select your desired padding colour, and click the rotate button.

Shear

To shear an image, select alpha and beta shear values, select your desired method of interpolation, select your desired padding colour, and click the shear button. Note that alpha is related to the horizontal shear, and beta is related to the vertical shear.

Padding Colour

To select your padding colour, there are 4 sliders representing the Red, Blue, Green, and Alpha colour values in order. Drag the 4 sliders, left or right to get your desired colour, the background colour of the box will change in relation with your selected colour.



Interpolation

To select your interpolation type, click either of the Bilinear interpolation or Nearest neighbour interpolation options.

GrayScale Converson

To convert a colour image into a grayscale image, click the make grayscale button.

Flip Image

To flip the image click either the flip vertically or flip horizontally buttons.

Inversion

To invert the image, click the invert pixels button.

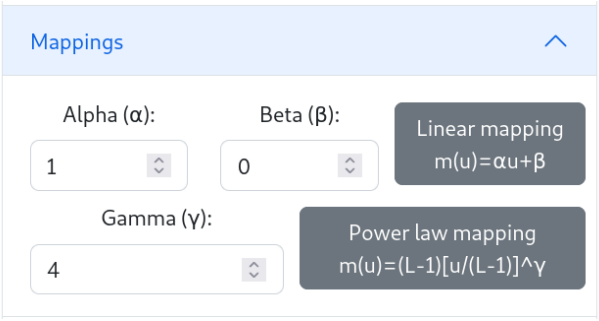
Mapping image operations

Linear Mapping

To do a linear mapping operation, select your desired alpha and beta values, then click on the linear mapping button

Power Law Mapping

To do a power law mapping operation, select your desired gamma value and click the power law mapping button



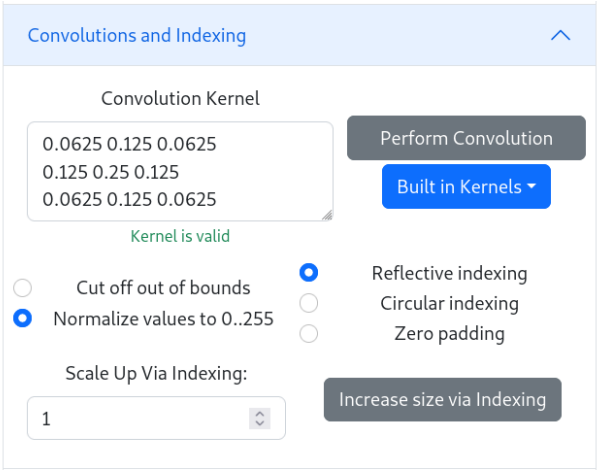
Convolutions and Indexing operations

Convolution

To perform a convolution, select your desired convolution kernel, select your desired method of handing out of bound values, select your desired indexing type for out of bounds values, and click of the perform convolution button.

Convolution Kernel Selection

To select a convolution kernel you can either type in your own, or click the built in kernels button and select from a number of built in kernel values. Please note that if you enter an invalid kernel the perform convolution button will become greyed out.



Convolution Bounds Handling

To select your method of handling out of bounds values of pixels, meaning they are greater than 255 or less than 0, which may happen after a convolution kernel is applied,

- Select the cut off bounds options to map all the values less than 0 to 0 and all the values greater than 255 to 255.
- Select the normalize values option to map the resulting range of numbers down to 0 to 255.

Image Extension Indexing

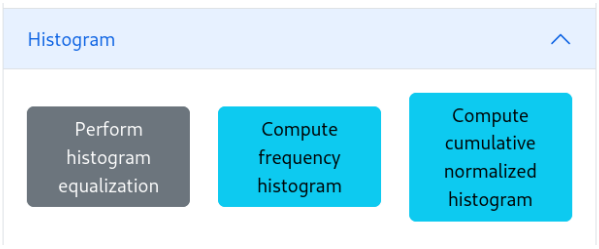
To select your method of indexing to get out of bounds values, choose from the reflective indexing, circular indexing, or zero padding indexing option.

Image Indexing Upscaling

To perform an upscale via indexing, enter your desired scale factor in the scale up via indexing box, and click the increase size via indexing button.

Histogram image operations

Histogram Equalization



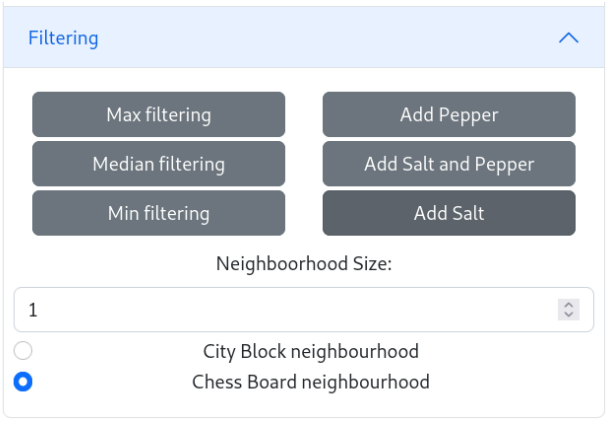
To perform a histogram equalization, click the perform histogram equalization button.

Show Histogram

To compute the frequency and cumulative normalized histograms, click those respective buttons. Note: To close the histograms click the hide secondary display button at the top right of the screen.

Filtering image operations

Max Filtering



To perform max filtering, select the desired neighborhood, and click the min filtering button.

Median Filtering

To perform median filtering, select the desired neighborhood, and click the min filtering button.

Min Filtering

To perform min filtering, select the desired neighborhood, and click the min filtering button.

Salt and Peper

This comes with the option to add salt, pepper, or salt and pepper noise to an image to test out the effects of max, median, and min filtering respectively. To select one of the noise enhancing operations, click the respective button.

Technical Discussion

This is meant to discuss the implentation of how different operations in the image toolkit work. Please note that trivial operations such as crop are omitted.

Basic image operations

Scale

Scaling works by multiplying the height and width by a desired scale factor and iterating over the new height and new width and interpolating what the corresponding pixel from the original image is, by dividing each new pixel location by the desired scale factor.

$$\texttt{newWidth} = \texttt{oldWidth} \times \texttt{scale}$$
$$\texttt{newHeight} = \texttt{oldHeight} \times \texttt{scale}$$

Rotate

Rotation works by calculating the new height and width as:

$$\texttt{newHeight} = \texttt{round}\left(|\texttt{oldHeight} \times \cos\theta| + |\texttt{oldWidth} \times \sin\theta|\right)$$
$$\texttt{newWidth} = \texttt{round}\left(|\texttt{oldHeight} \times \sin\theta| + |\texttt{oldWidth} \times \cos\theta|\right)$$

Then iterating over every pixel in the new height and width, and multiplying it by the following inverse matrix to get corresponding pixel in the origial image:

$$\begin{bmatrix} \cos\theta & \sin\theta & xTranslation \\ -\sin\theta & \cos\theta & yTranslation \\ 0 & 0 & 1 \end{bmatrix}$$

The x and y translation are needed because originally the image has the top left corner as (0,0) and we need to translate the image so that the center is at (0,0) before we can apply the transformation properly.
Then we have to translate the image back, we do this by iterating over elements of the new image and adding them to a new 2d array so that they forget their absolute locations.

Shear

Shear works by calculating the new height and width as:

$$\texttt{newHeight} = \texttt{round}\left(\texttt{oldHeight} + |\texttt{oldWidth} \times \beta|\right)$$
$$\texttt{newWidth} = \texttt{round}\left(\texttt{oldWidth} + |\texttt{newHeight} \times \alpha|\right)$$

The new width depends on the new height because of the matrix that is being used, combines both the alpha and beta shear into one operation. The matrix is a result of combining the

horizontal and vertical shear matrices by multiplying them:

$$\begin{bmatrix} 1+\beta\alpha & \beta & xTranslation \\ \alpha & 1 & yTranslation \\ 0 & 0 & 1 \end{bmatrix}$$

The x and y translation are needed because originally the image has the top left corner as (0,0) and we need to translate the image so that the center is at (0,0) before we can apply the transformation properly. Then we have to translate the image back, we do this by iterating over elements of the new image and adding them to a new 2d array so that they forget their absolute locations.

Nearest Neighbour Interpolation

Nearest neighbour interpolation works by rounding every input pixel location to the closest pixel location that is inside the image bounds:

$$\begin{aligned} \texttt{x} &= \texttt{round}(\texttt{inputX}) \\ \texttt{y} &= \texttt{round}(\texttt{inputY}) \end{aligned}$$

Bilinear Interpolation

Bilinear interpolation works by first getting 4 coordinates, based off the floor and ceiling of the input coordinate.

$$\begin{aligned} \texttt{topLeft} &= [\texttt{floorX}, \texttt{floorY}] \\ \texttt{topRight} &= [\texttt{ceilX}, \texttt{floorY}] \\ \texttt{bottomLeft} &= [\texttt{floorX}, \texttt{ceilY}] \\ \texttt{bottomRight} &= [\texttt{ceilX}, \texttt{ceilY}] \end{aligned}$$

Then we calculate the bias, which is how far each pixel is from the the input coordinates. This determines the weighting of each pixel value.

$$\begin{aligned} \texttt{biasX} &= \texttt{ceilX} - \texttt{originalX} \\ \texttt{biasY} &= \texttt{ceilY} - \texttt{originalY} \end{aligned}$$

Then we average out the top two pixels, with the bias:

$$\begin{aligned} \texttt{top} &= \frac{\texttt{topLeft} \cdot (1 - \texttt{biasX}) + \texttt{topRight} \cdot \texttt{biasX}}{2} \\ \texttt{bottom} &= \frac{\texttt{bottomLeft} \cdot (1 - \texttt{biasX}) + \texttt{bottomRight} \cdot \texttt{biasX}}{2} \\ \texttt{averaged} &= \frac{\texttt{top} \cdot (1 - \texttt{biasY}) + \texttt{bottom} \cdot \texttt{biasY}}{2} \end{aligned}$$

Then we the averaged pixel we create is the result.

Mapping image operations

Linear Mapping

Linear mapping is carried out in the following way:

$$\texttt{pixelValue} = \alpha \cdot \texttt{pixelValue} + \beta$$

Power Law Mapping

Power law mapping is carried out in the following way:

$$\texttt{newPixelValue} = (L-1) \cdot \left(\frac{\texttt{pixelValue}}{L-1} \right)^{\gamma}$$

Convolutions and Indexing operations

Convolution

Convolutions work according to the defintion that Pascal Matsakis provided in class:

$$\text{newPixelValue} = \sum_{-\frac{m-1}{2}}^{\frac{m-1}{2}} \sum_{-\frac{n-1}{2}}^{\frac{n-1}{2}} \text{kernelValue}_{i,j} \cdot \text{pixelValue}_{x-i,y-j}$$

Where we create a new image with the result of the summation mentioned above.

Convolution Bounds Handling

The value of each pixel value after a convolution is applied can be handled in one of two ways:

$$\texttt{cutOffPixel} = \begin{cases} 0 & \text{if pixelValue} < 0 \\ \text{pixelValue} & \text{if } 0 \leq \text{pixelValue} \leq 255 \\ 255 & \text{if pixelValue} > 255 \end{cases}$$

$$\texttt{normalizedPixel} = \frac{\text{pixelValue} - \min}{\max - \min}$$

Reflective Indexing

The reflective indexing operations requires two steps, as first we want to bound every index to an area from 0 to twice the max size.

$$\texttt{intermediateIndex} = ((\text{index} \,\%\, (\text{size} \times 2)) + (\text{size} \times 2)) \,\%\, (\text{size} \times 2)$$

And then after its bound into an area from 0 to twice the size, we check if it falls in the first half and can be itself, or if it falls in the second half and needs to be reflected.

$$\texttt{reflectiveIndex} = \begin{cases} (2 \times \text{size} - 1) - \text{intermediateIndex}, & \text{if intermediateIndex} \geq \text{size} \\ \text{intermediateIndex}, & \text{otherwise} \end{cases}$$

Circular Indexing

Circular indexing essentially takes the modulo of the height if the index is greater than or equal to 0, and otherwise subtracts the modulo of the height from the height if the index is less than 0.

$$\texttt{circularIndex} = \begin{cases} \text{size} - 1 - ((-\text{index} - 1) \,\%\, \text{size}) & \text{if index} < 0 \\ \text{index} \,\%\, \text{size} & \text{otherwise} \end{cases}$$

Zero Padding Indexing

Zero padding indexing is fairly simple, it just returns zero if the index is outside of the bounds of the image, and pixel at the index otherwise

$$\texttt{zeroPaddingIndex} = \begin{cases} \text{pixelValue}, & \text{if index is in bounds} \\ 0, & \text{otherwise} \end{cases}$$

Image Indexing Upscaling

This works by increasing the amount of pixels on either side of the image according to the value that the chosen indexing type, such as reflective indexing, returns.

Histogram image operations

Histogram Equalization

Histogram equalization consists of making a normalized cumulative histogram by the following steps:

1. We have to calculate the frequencies of each colour.
2. We have to make the values normalized, by dividing every value by the total amount of pixels.
3. We have to make the normalized values cumulative in order. This gives us a cumulative normalized histogram at this stage
4. Then we multiply each row value in the histogram by the maximual luminosity value, $L - 1$

Then we can use each column in the histogram as a mapping to its row value. We then apply that mapping to every pixel in the image.

Filtering image operations

Max Filtering

Max filtering is done by creating a new image and setting each pixel value to the maximal value of the neighbourhood of the corresponding pixel in the old image.

Median Filtering

Median filtering is done by creating a new image and setting each pixel value to the media value of the neighbourhood of the corresponding pixel in the old image.

Min Filtering

Min filtering is done by creating a new image and setting each pixel value to the minimal value of the neighbourhood of the corresponding pixel in the old image.

Chessboard Neighbourhood

The chessboard neighbourd for a pixel for a given distance d is defined as:

$$N_{\text{chessboard}}(x_0, y_0, d) = \{(x, y) \mid \max(|x - x_0|, |y - y_0|) \leq d\}$$

City-block Neighbourhood

The city-block neighbourd for a pixel for a given distance d is defined as:

$$N_{\text{city-block}}(x_0, y_0, d) = \{(x, y) \mid |x - x_0| + |y - y_0| \leq d\}$$

Salt and Peper

Salt and Pepper noise enhancement is done by replacing 1% of the values in the image with either salt or pepper as selected.

Results and Future Work

Strengths

- A particularly interesting capability that this features is the ability to apply multiple image transformations in series, for example scale an image and then perform histogram equalization on it.
- A big strength is the ability to display a frequency histogram of pixel values that lets you see the effect of image transformations more acutely
- A big effort was undertaken to make the user interface simple to use, and to avoid clutter where possible to minimize mental strain to the user. As the user interface

is designed to be accessed from the web, a big achievement is that the user interface works on both mobile and desktop browser.

Weaknesses

- This program is built to do all the image transformations on all 3 RGB channels, this makes some transformations such as applying an edge detection kernel look a bit awkward. Although this is partially remedied by the ability to transform an image into a grayscale image by setting the 3 RGB channels equal to each other (by averaging them).

Future Steps

- Add ability to undo and redo image operations.
- Add ability to crop image by dragging crop box on the image, instead of specifying crop in pixels.
- Add capability to do Fast Fourier Transform convolutions that would speed up the convolution speed significantly.
- Add ability to use Bicubic interpolation, this offers some benefits over the Bilinear interpolation that is currently supported in terms of image fidelity.
- Add multithreading support, as many image operations are parallelizable.

Testing

This was tested by running the program in a web browser. Multiple test images were used such as the TV test card image (this is the multicolour default image shown in the User Manual section), as well as a wide variety of images provided by Professor Pascal Matsakis in the lecture slides such as the image of nuts directly to the right of this.