Zachary Banducci

May 2, 2018

CSE 3353

Program 3

Image Manipulation

Grayscale

In order for a pixel to be greyscale, all color channels of that pixel must be of the same value, ranging from all channels equaling 0 (black) to equaling 255 (white). There are two distinct ways to transform an image to greyscale. The first method is to take the average of all RGB channel values for a given pixel, meaning that each color has a equal weight. Although this method is effective for transforming an image into a grayscale image, the resulting image lacks contrast. In order to combat this, by utilizing a weighted average of the channel values for a given pixel, a compressed RGB value becomes gamma expanded to decompress the image. The difference between the two methods are represented below:

Original Image

Average of channel values

Weighted average of channel values

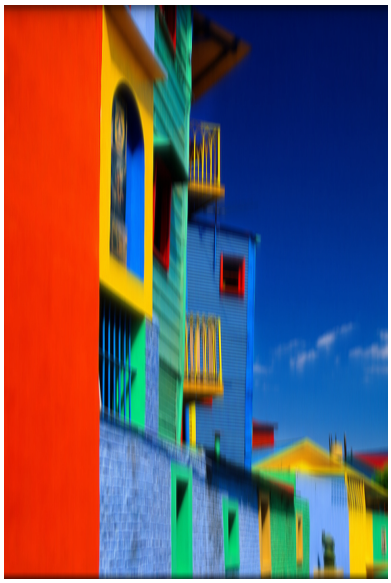
Convolution

Convolution is used to create a new version of the original function or matrix by using a pointwise multiplication of two matrices. Essentially, the convolution acts a mask over the original image in order to manipulate the intensity of the neighbors of a pixel. Different convolution matrices yield different resulting images when mapped to a source image. Convolution is essential in order to both blur and sharpen an image. A specific kernel is used for each matrix in order to achieve a certain filtering effect.

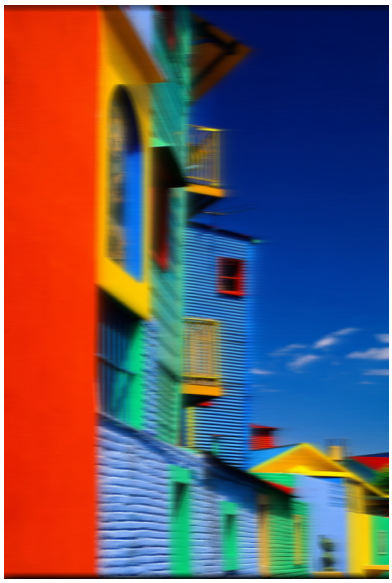
Blurring

Blurring an image, in the case of the box blur, is the process of applying a linear convolution filter over an image where each neighboring pixel is of equal weight and taking the average of the pixel values. The calculated average for each RGB channel of the pixel is then mapped to the destination pixel. Based on the dimensions of the

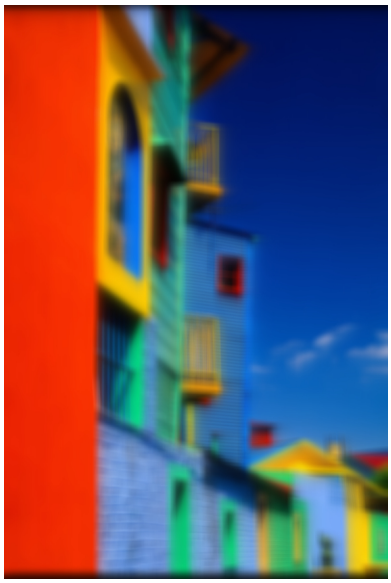
convolution matrix, as the number of pixels being evaluated in the average increases, the “blurring” effect also increases accordingly. In order to increase efficiency, the average for a pixel of interest can be calculated in both the x and y direction independently.



X – Axis Blur (10px radius)



Y – Axis Blur (10px radius)



Combined Blur (Box Blur)

Sharpening

Much like with the blur method, the sharpening method also involves an image mask produced by a convolution matrix. In this case, however, the kernel for the matrix increases the weight of the target pixel and decreases the weight of the surrounding pixels in order to sharpen the pixel of interest.

Connected Components

In order to determine connected components within an image, the image must be converted into a grayscale image. After it has been converted, the image can then be transformed into a binary image where the pixel is either on (white) or off (black). To accomplish this, one can utilize a histogram in order to calculate the most common pixel intensity within the image, signifying the background. For my implementation, I utilized a two-pass algorithm. I first obtained the average intensity value of all grayscale pixels within the image. This average was then used as a threshold for a pixel to determine if it would be black or white. If the pixel intensity was below that of the threshold, then it was labeled as a background pixel and was labeled as a foreground pixel if it was above that threshold. By changing the threshold value, one can manipulate how many pixels appear on.

The next step in determining the components was to iterate through the image by row and column. In order to determine the connected components of the image, labelling the pixels was an important part of the process. On each pixel of interest, there were three possible situations. If the pixel was a background pixel, move on to the next column in the row. If the pixel was a foreground pixel without any neighbors, the pixel was given a unique identifier. This identifier was then added in a disjoint set where it could later be merged with other labels in order to determine an equivalence relation. If the pixel contained any neighbors, the label of the pixel of interest was set to be the minimum of its two neighbors. At this point, if the remaining neighbors are not a background pixel, their labels are merged with the disjoint set with that of the pixel of

interest in order to establish the equivalence relation. This process repeats until all pixels have been assigned labels.

The final step in the process is to set the pixel color based upon its determined equivalence relation within the disjoint set. By iterating over each pixel and determining its equivalence relation, every pixel within a given region will obtain the same color since their labels are all equivalent.

The complexity of the connected components algorithm can be determined to be O(height x width) since one must iterate over every pixel two times in order to set it's labels and determine color based on the pixel label's equivalence relation.

Process Scheduling

In order to store a graph in memory, one could use an adjacency list. An adjacency list is a collection of vertices where each vertex contains a linked list to all edges that can be traversed from that vertex. One solution for determining the topological order of a directed acyclic graph is a depth first search. In a depth first search, a stack is utilized in order to determine what the last vertex that can be visited. Once all edges have been traversed for a given vertex, the vertex is pushed onto the stack. Once a vertex has been visited, it is marked as visited so that it is not revisited. This process repeats until all vertices have been visited, resulting in a stack that is the traversal of the tree. The big-O of this algorithm is O(|V| + |E|), since for every node in the adjacency list contains a set of edges, you can discover all neighbors relative to a

given vertex in linear time. Therefore, the time complexity of the algorithm is the sum of vertices plus the edges.