Keionne Derousselle

Seam Carver Write-Up

# Height

## Description

This method returns the height of the current image.

## Analysis

O(1)

# Width

## Description

This method returns the width of the current image.

## Analysis

O(1)

# Energy

## Description

This method calculates the energy of a pixels by observing its surrounding pixels. I first check to see if a pixel within the bounds of the picture. If this is so, I find the pixels located at the top, bottom, left, and right of this pixel. I calculate the change in the red, green, and blue energy along the pixel’s x axis by subtracting the amount of red, green, and blue pixels from left to right respectively. I then square these values to find the total change in energy along the x axis. I find the change in red, blue, and green along the y axis by subtracting the red, green, and blue found in the top pixel from the bottom pixel. I square these three values and add them to get the change in energy along the y axis. I then add the change in x energy with the change in y energy. This value is known as the energy of a pixel.

Example:

redXChange = red(left) – red(right)

greenXChange = green(left) – green(right)

blueXChange = blue(left) – blue(right)

xChange = readXChange2 + greenXChange2 + blueXChange2

redYChange = red(top) – red(bottom)

greenYChange = green(top) – green(bottom)

blueYChange = blue(top) – blue(bottom)

yChange = readYChange2 + greenYChange2 + blueYChange2

energy = xChange + yChange

## Analysis

O(1)

# Find Horizontal Seam

## Description

I keep track of paths taken and energies by using an array of objects called Vertex. Each Vertex has an integer representing its energy, the cost of the path taken to get to that Vertex, and the index of the parent by which the Vertex was traveled to. This allows me to loop through every pixel in the picture, observing each pixels children/neighbors. If the neighboring pixel has already been visited by one of its parents, I check to see if the total cost to get to that neighbor is greater than the total cost to get from the current pixel to that neighbor. If that is the case, I update the neighboring pixel’s total cost to the current pixel’s total cost plus its energy. I also change the neighboring pixel’s parent index to be that of the index of the current pixel. During this process, I check to see if a neighboring index is located in the last column of pixels in the picture. If this is the case, I check to see if it currently has the lowest total cost in that column. By the time the process is complete, I have the index of the vertex with that is last in the horizontal seam. I place this index at the end of the horizontal seam array. I then loop through the rest of the horizontal seam array, placing the next vertex’s parent in the array until there are no more parents.

## Analysis

O(W\*H)

# Find Vertical Seam

## Description

I keep track of paths taken and energies by using an array of objects called Vertex. Each Vertex has an integer representing its energy, the cost of the path taken to get to that Vertex, and the index of the parent by which the Vertex was traveled to. This allows me to loop through every pixel in the picture, observing each pixels children/neighbors. If the neighboring pixel has already been visited by one of its parents, I check to see if the total cost to get to that neighbor is greater than the total cost to get from the current pixel to that neighbor. If that is the case, I update the neighboring pixel’s total cost to the current pixel’s total cost plus its energy. I also change the neighboring pixel’s parent index to be that of the index of the current pixel. During this process, I check to see if a neighboring index is located in the last row of pixels in the picture. If this is the case, I check to see if it currently has the lowest total cost in that row. By the time the process is complete, I have the index of the vertex with that is last in the vertical seam. I place this index at the end of the vertical seam array. I then loop through the rest of the vertical seam array, placing the next vertex’s parent in the array until there are no more parents.

## Analysis

O(W\*H)

# Remove Horizontal Seam

## Description

I first check to see that the seam’s length is not greater or less than the width of the current picture. Then I check to see that the width of the picture is greater than 1. If both of these cases are true, I proceed to check to see if every pixel in the array is within the bounds of the picture and that the each succeeding vertex is located in the column after the proceeding vertex and the distance between the two vertices is 1. If this holds true, I redraw the desired pixels into a new image, skipping over the pixels to remove. (e.g. for every pixel that is to be removed, I move all pixels beneath it in the same column up).

## Analysis

O(W\*H)

# Remove Vertical Seam

## Description

I first check to see that the seam’s length is not greater or less than the height of the current picture. Then I check to see that the height of the picture is greater than 1. If both of these cases are true, I proceed to check to see if every pixel in the array is within the bounds of the picture and that the each succeeding vertex is located in the row after the proceeding vertex and the distance between the two vertices is 1. If this holds true, I redraw the desired pixels into a new image, skipping over the pixels to remove. (e.g. for every pixel that is to be removed, I move all pixels after it in the same row to the left).

## Analysis

O(W\*H)