CS 3258/5258 Spring 2021
Assignment 3
Image Resizer

March 4, 2021

1 Purpose

In this assignment you will construct a module to the CLI of the last project to resize an image with anti-aliasing.

2 Due Date

The assignment is due by midnight on Thursday, March 18, 2021.

3 Resizer

3.1 Command Format

Implement two additional commands in your CLI, resize and zoom. The format is:

Command	Parameters	Description
Resize	x_{scale}, y_{scale}	Resize by x_{scale} in x and y_{scale} in y
Zoom	s	Scales the image in both directions by the amount s

Both these commands take non-negative floating point arguments. The way they should work is to take the image that was just displayed using tiffread, resize it appropriately, and redisplay the resized image. The command should also print out the dimensions of the new image (so you can tiffwrite it easily).

3.2 Implementation Details

If the original image was n pixels in x (horizontal) and m pixels in y (vertical), the resulting image should be $\texttt{floor}(x_{scale}n)$ by $\texttt{floor}(y_{scale}m)$.

Remember that there are no negative intensity values. In this assignment you must do all your calculations in floating point, and then convert back to the legal intensity range of $[0 \ldots 255]$. Anything outside this range should be *clamped*.

When implementing discrete convolution, a decision about what the color values for pixels outside the range of the image must be made. There are typically three things done, although I mandate that you use one in the Requirements section below. The three methods are:

1. Values outside the image are 0.
2. Values outside the image are the same color as the border pixels.

3. You can reflect the image with respect to the border.

Also, do NOT assume that the image is bigger than the filter as this may not be true.

To resize in 2D, first do the resize on the horizontal image and then do it on the vertical image (or vice versa, at your choice). This type of transformation is *separable*, which means you can do it one dimension at a time.

You may find it easier to implement the magnification part of the assignment separately from the minification part. If you approach the assignment this way, I suggest you check to see if the scale factor is a magnify ($scale > 1$) and then call a routine `magnify_x` or `minify_x`.

Apply each operation to each color component separately.

3.3 Requirements

For this assignment, pixels outside the borders of the image should have a value of 0 in all three components. Also, you should use the Lanczos kernel as your base convolution kernel ($sinc(x)sinc(x/2)$, $sinc(x)sinc(x/3)$), etc., as described in class).

Be sure your program does something reasonable when magnified or minified by by a non-integral amount. It should also do the right thing when magnified by an integer, of course! The program should be able to tiffwrite out the resulting image. Make sure that your program checks the arguments to prevent me from magnifying an image to bigger than 1024x1024 if you use a static array to display values.

3.4 Extra Credit

For extra credit, you can implement one or all of the additional features.

- Implement a command `border` that takes a string argument and changes the way borders are dealt with in the image. For example, "border zero" would set the convolver up so that pixels outside the borders of the image are 0. "border freeze" would use the border pixel's value for pixels outside the range of the image, and "border circular" would reflect the image around its border for values outside the image range (7 pts.)

- Implement a command `select` that takes a string argument and changes the base convolution kernel for the resize command. For example, "select gaussian" would use a gaussian instead of a Lanczos filter, or "select mitchell" would use the Mitchell filter described in class, in the textbook (Section 9.3), or handout (Chapter 19 of *Computer Graphics* by Hughes et al. (2014)). Possible filters are gaussian, mitchell, hamming, b-spline cubic, catmull-rom, triangle (tent), and box. (2 pts./filter). Use a default width that makes sense, e.g., 2 pixels wide, and tell us what it is in accompanying documentation. Formulas for the filters can be found in Section 9.3 of the book, and in the handouts. Make sure that the Lanczos kernel is selectable. Add optional arguments to "select gaussian", "select lanczos", "select triangle", and "select box" that govern their width, i.e., makes them sharper or wider (3 pts.). For a Lanczos kernel, for example, a wider filter would be given by $sinc(x)sinc(x/3)$.[1]

- Extend the resize command to handle negative scale factors. A negative scale should reflect as well as resize (3 pts.).[2]

- Explain and justify the efficiency of your implementation (3 pts.).

To receive extra credit, you must submit a document, e.g., README, README.doc, describing what extra features you implemented, compare them with the method required by this assignment, e.g., which filter gives more visually pleasing results, and submit some sample images resized by your filter that are the results of your tests. You can add all this to the README, or point me to the document in a README. The document doesn't have to be long, but it needs to show that you thought about the comparisons.

[1] This "Extra Credit" is mandatory for anyone taking CS5258, i.e., taking the course for graduate credit.

[2] This "Extra Credit" is also mandatory for anyone taking CS5258, i.e., taking the course for graduate credit.