601.220 Intermediate Programming

Summer 2023, Midterm project overview (June 23th)

Midterm project

Image processing: like Photoshop(tm), but a command line program which

- reads an input image
- applies a transformation
- writes an output image

PPM files

All widely-used image formats (PNG, JPEG, etc.) are compressed, so the data representation is very complicated.

We will use the PPM format, which is uncompressed, and relatively easy to read and write.

The starter code provides `ppm_io.h`/`ppm_io.c` for reading and writing PPM files.

Images and pixels

An image is a rectangular grid of *pixels*. Each pixel is a colored dot. Pixel color is represented as *red*, *green*, and *blue* intensity values in the range 0–255. An RGB triple can represent (essentially) any color that can be perceived by the human eye.

Example colors:

- $(0, 0, 0)$: is black
- $(255, 255, 255)$: is white
- $(17, 59, 94)$: the darker shade of blue on this slide
- $(26, 89, 142)$: the lighter shade of blue on this slide

Pixel data type

```
typedef struct _pixel {
  unsigned char r;
  unsigned char g;
  unsigned char b;
} Pixel;
```

Image data type

```
typedef struct _image {
  Pixel *data;  // pointer to array of Pixels
  int rows;     // number of rows of Pixels
  int cols;     // number of columns of Pixels
} Image;
```

Note that:

- The `data` field points to a dynamically-allocated array of `Pixel` elements
- The pixels are stored in *row major* order: the top row of pixels is first, then the second row of pixels, etc.

Pixel coordinates

X coordinates: 0 is the left most column.

Y coordinates: 0 is the *top* row. (Not the bottom row!)

Original image

Grayscale

Convert each pixel in the input image so that

1. The "intensity" of the pixel is preserved, but
2. The RGB component values are equal to each other

Project description has a formula for doing the converstion.

Grayscale example output

Binarize

Convert each pixel to either black or white based on comparing its computed grayscale value to a threshold value.

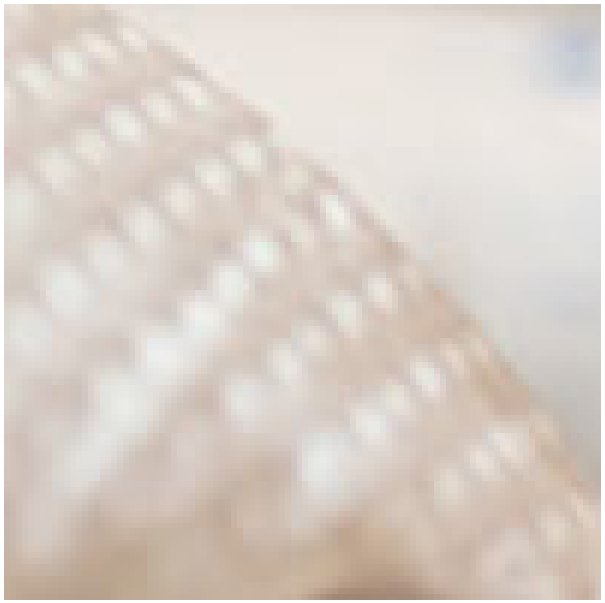
Binarize example output

(With threshold value set to 127.)

Crop

Select a rectangular region of the input image. The region is specified as coordinates of top left and bottom right corners of region.

Crop example output



(With region specified as $(200, 200)$ and $(300, 300)$.)

Transpose

Output image has the same pixels as input image, but the x/y coordinates are swapped.

Transpose example output

Gradient

Convert image to grayscale.

Compute "gradient" grayscale intensity for each pixel in both horizontal and vertical directions.

Result image grayscale intensity for each pixel is the sum of the absolute values of the horizontal and vertical gradient values.

Gradient example output

Seam carving

Rescale an image while avoiding changing the aspect ratio of the "interesting" parts of the image.

Uses gradiant image to determine where "seams" should be identified (to guide which parts of the original image should be removed.)

Details in project description. This will be fairly tricky to implement!

Seam carving example output

Viewing images

The image files, particularly the *result* image files, will be in your ugrad account. How to view them on your local computer?

Option 1: Preview them in VS Code. If you have VS Code set up to access your ugrad account, this should work transparently.

Option 2: Use `scp` to copy them to your local machine.

Option 3: Use X forwarding (`ssh -X`) and the `feh` image viewer program.

Team dynamics

This is a team project.

Team projects work best when team members work together synchronously. This is known as *pair programming*.

Recommendation: avoid having overly-specific division of responsibilities between team members. The best scenario is when

1. All team members know what everyone is working on, and
2. Any team member can contribute to *any* part of the project

Avoid "critical path" dependencies. I.e., team member 1 can't make progress because they are waiting for team member 2 to complete something.

Using Git for a team project

Everyone on the team should clone the team's midterm project repository.

Each team member will use `git push` and `git pull` to synchronize their work with the project repository.

Merges and merge conflicts

Sometimes, before you `git push` your own work, you will need to `git pull` to bring your local repo up to date with your teammates' work.

Most of the time, `git` automatically creates a *merge commit* to reconcile your work with your teammates' work. This is possible when your changes affect different parts of the code than the code your teammates modified.

However, this could also result in a *merge conflict* if your changes conflict with your teammates' changes.

Resolving a merge conflict

If `git` tells you a merge conflict occurs, it will add *conflict markers* to the affected code to indicate where the conflicts are. (They are very obvious: look for places where there are <<<< and >>>>.)

You will need to edit the affected files to

1. Remove the conflict markers, and
2. Manually reconcile your changes with your teammates' changes

When you're done, and you've tested the code, just do `git add` on the files were conflicts where found, then `git commit`.

Planning your time

As always:

1. Start early (i.e., *now*)
2. Make steady, incremental progress
3. Work on the simpler image transformations first (the order in the project description is the recommended order)

Let us know if you have questions!