Computational Photography Final Project

Full Name1: James Johnson
GTID1: 903047181
Full Name2: Joshua Terry
GTID2: 903067329

Spring 2019

B&W to Color

Inspired by Sergey Prokudin-Gorsky's three-color principle

The Goal of Your Project

Original project scope:

The original scope of the project was automatic white-balancing to computationally enhancing images through color adjustment in order to enhance images to match the natural world or experiment with what the photographer desires to demonstrate.

What motivated you to do this project?

Often a camera has automatic white-balancing that senses the temperature of the scene and then adjusts the capture image according, but this implementation can be limited and/or wrong. Using Photoshop you can change the white balance of an image, and we wanted to attempt our own implementation of that tool.

Scope Changes

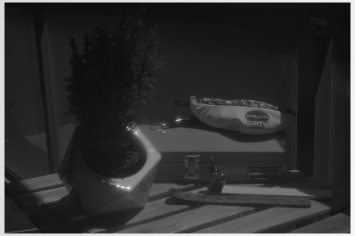
- Did you run into issues that required you to change project scope from your proposal?

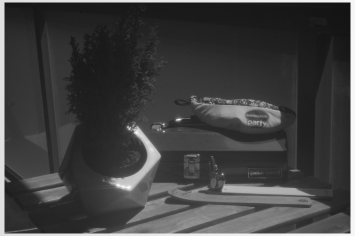
Yes, due to our original idea falling more into the scope of computer vision rather than computational photography, we pivoted to a process that also encapsulated the method with which the images were taken.

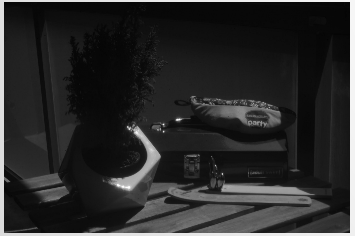
- Give a detailed explanation of what changed

Inspired partly by the mantis shrimp’s 12 color receptors and Sergey Prokudin-Gorsky’s photography technique that capture black and white (B&W) images of the individual color channels (RGB additively) and then combining them into a fully spectrumed image, we bought filters for our camera of differing colors and attempted to capture B&W images and inserting those pixel values into the individual channels of a composite image to attempt to create novel images.

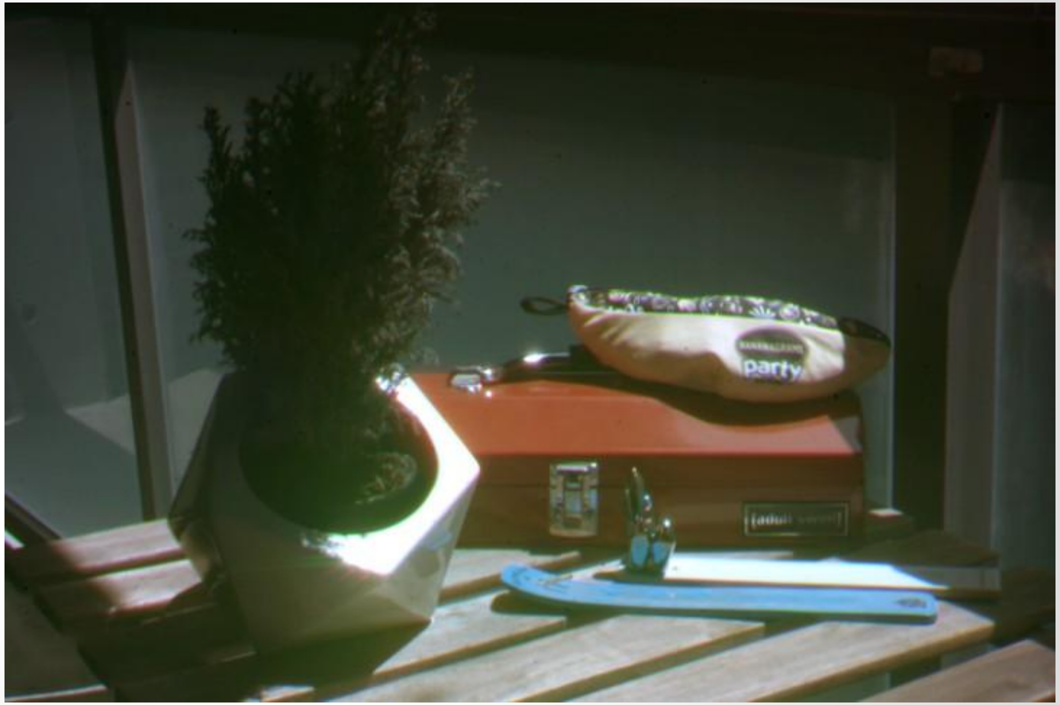
Showcase


Red


Green

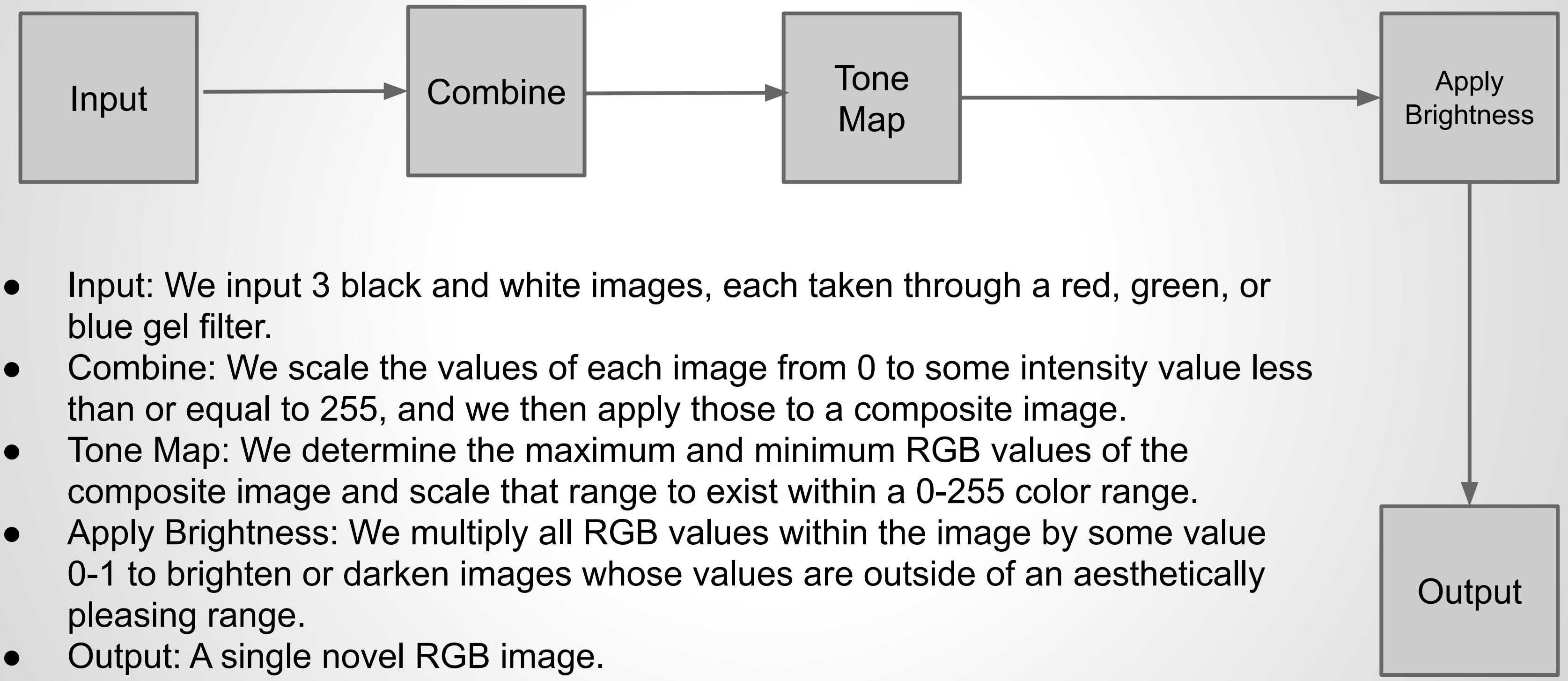

Blue

Input



Output

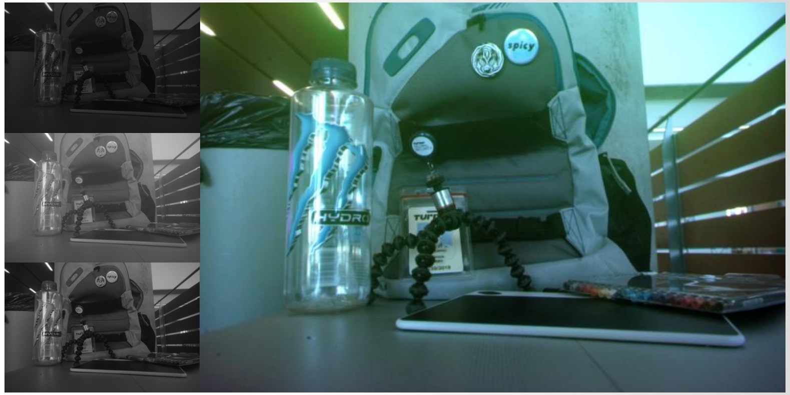
Project Pipeline

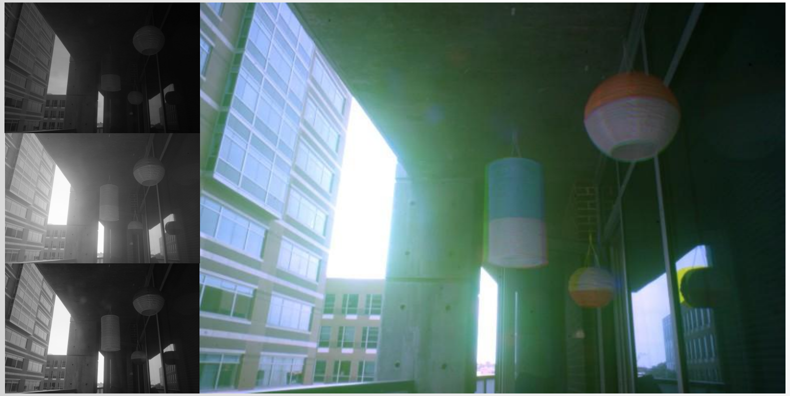


- Input: We input 3 black and white images, each taken through a red, green, or blue gel filter.
- Combine: We scale the values of each image from 0 to some intensity value less than or equal to 255, and we then apply those to a composite image.
- Tone Map: We determine the maximum and minimum RGB values of the composite image and scale that range to exist within a 0-255 color range.
- Apply Brightness: We multiply all RGB values within the image by some value 0-1 to brighten or darken images whose values are outside of an aesthetically pleasing range.
- Output: A single novel RGB image.

Demonstration: Result Sets

The three black and white images to the right were taken through red, green, and blue filters, top to bottom.
The color image to the right is a modified version of those three images in which the left images' values were multiplied by some value less than 255 and mapped onto the right image's corresponding color channel.

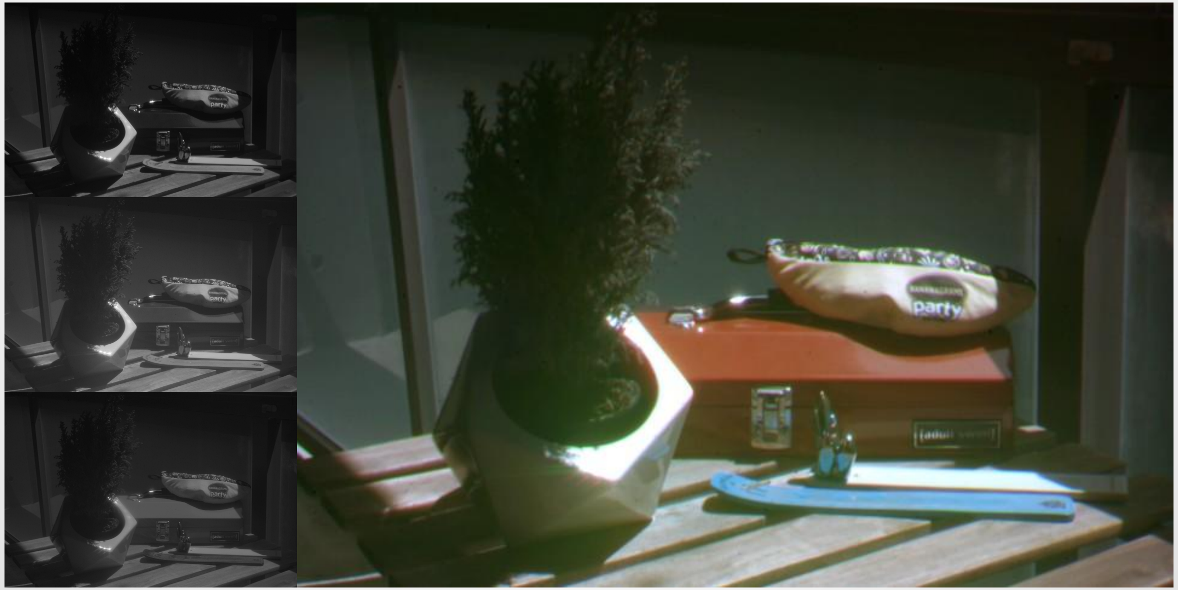




The same process that was applied to the above image is applied to every set of 3 input images, but the relative intensities of each image and the brightness of the final image all scale based on manual user input.
The green flares in the image to the right are a result of diffusion from the green filter used, and the colored artifacts are due to the lanterns' movements in the wind.

Demonstration: Result Sets

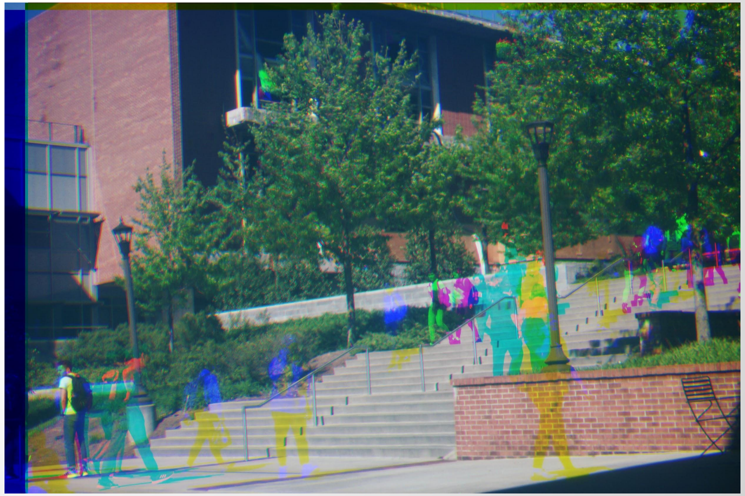
- The below resulting image has very few discolored artifacts and fairly accurate colors due to optimal lighting conditions. Very little light shone directly into the camera, and the scene was brightly illuminated by sunlight. Notice the vignette effect present in the image as a result of a high shutter speed and small aperture paired with multiple color filters and low incoming light.



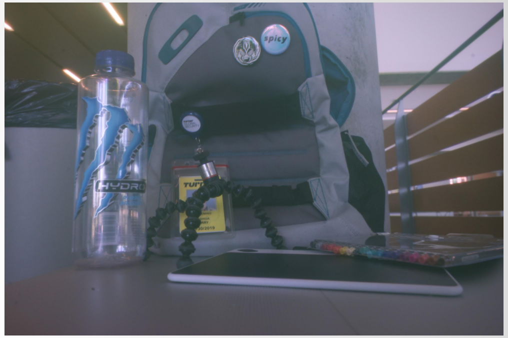
Project Development

Over the lifetime of our project, we encountered several issues, but ended up with a successful digital implementation of Produkin-Gorsky's three-color image synthesis experiment. While our initial idea of auto white balancing images fell more in the camp of computer vision than computational photography, our reworked project idea of mapping black and white photographs onto the red, green, and blue channels of a digital photograph is definitely more in line with the core principles of the class. Taking photos for the project was no simple feat: we had to plan according to wind, lighting, and timing, due to the impacts made on gel filters by wind and lighting as well as the impact made on the composite images due to timing of the images.

For instance, the image to the right was captured in windy conditions, in bright daylight, during class change, with a wobbly tripod. This resulted in a washed out appearance of the image, many multicolored artifacts within the image, and misaligned edges and features within the image. To address this, we aimed to photograph static subjects outside of windy environments or environments with large amounts of direct sunlight.



Project Development



To the right is an example of one of the status subjects we photographed, a backpack. Once we had several consistent image sets, we began programming the project using the java-based platform Processing. We implemented a simple, intuitive user interface using the ControlP5 toolkit, and we ensured that the user would be able to save the composite image at any point while running our program.

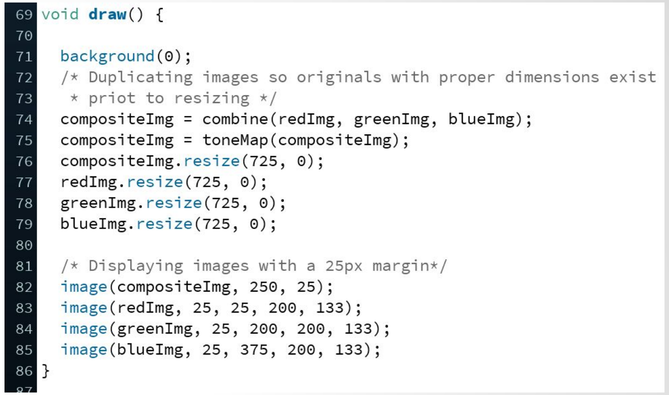
While coding, we encountered difficulties when one of the three input images was too dark or bright relative to the others, so we gave users the option to control the intensity of each color channel to account for this. Since this makes images darker or brighter, we also implemented a brightness slider to allow users to brighten or darken their images based on their modified intensities. Additionally, we ran into issues with the final image being either too pixelated or taking too long to load. To address this, we downscale all images to only fill the largest window, and display them as slightly smaller versions to more intuitively display to the user what images are being combined and how they are being combined.

We finished all goals set out for except for that of a way to dynamically control which image is mapped to which color channel, but this was deemed unnecessary as all input images are meant to represent red, green, or blue color channels.

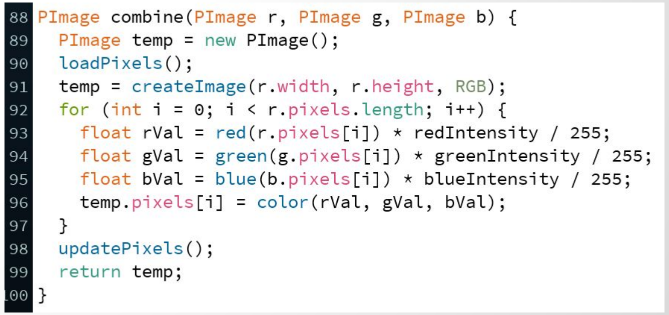
Given another opportunity to do this project, we may have gathered more image sets.

Computation: Code Functional Description

- Our draw function combines the 3 input image, scales their colors to be more rich, resizes the input and output images, and draws those images at several locations on screen.
- To combine our source images, we create a new temporary image and iterate through the pixels of that image, adding relevant values from the source color channel images after multiplying those values [0,255] by some other value [0,1] based on user input.
- This combined image is then passed back into the draw function and passed into the toneMap method.



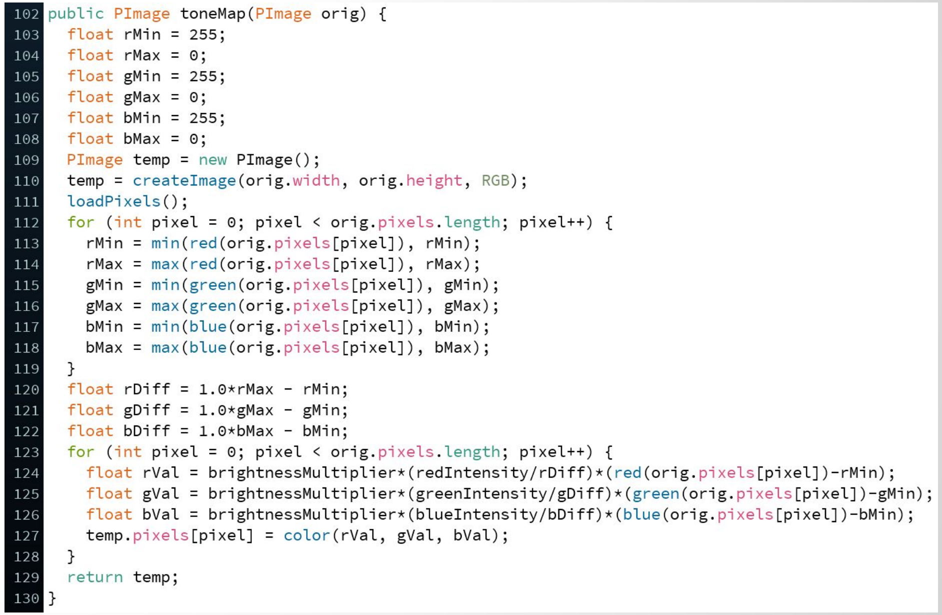
```
void draw() {

  background(0);
  /* Duplicating images so originals with proper dimensions exist
   * priot to resizing */
  compositeImg = combine(redImg, greenImg, blueImg);
  compositeImg = toneMap(compositeImg);
  compositeImg.resize(725, 0);
  redImg.resize(725, 0);
  greenImg.resize(725, 0);
  blueImg.resize(725, 0);

  /* Displaying images with a 25px margin*/
  image(compositeImg, 250, 25);
  image(redImg, 25, 25, 200, 133);
  image(greenImg, 25, 200, 200, 133);
  image(blueImg, 25, 375, 200, 133);
}
```



```
PImage combine(PImage r, PImage g, PImage b) {
  PImage temp = new PImage();
  loadPixels();
  temp = createImage(r.width, r.height, RGB);
  for (int i = 0; i < r.pixels.length; i++) {
    float rVal = red(r.pixels[i]) * redIntensity / 255;
    float gVal = green(g.pixels[i]) * greenIntensity / 255;
    float bVal = blue(b.pixels[i]) * blueIntensity / 255;
    temp.pixels[i] = color(rVal, gVal, bVal);
  }
  updatePixels();
  return temp;
}
```

Computation: Code Functional Description

- In the toneMap function, we determine the minimum and maximum values of each channel within the image in the first for loop. We then determine the difference between those.
- Then, for each pixel in the original image, we subtract that image's minimum value and multiply its value to fully inhabit a dynamic range from 0 to that image's max intensity value as passed in by the user.
- This value is then multiplied by the brightness modifier slider, and the final image is returned, passed back into the draw function, and is displayed to the user.



```
public PImage toneMap(PImage orig) {
  float rMin = 255;
  float rMax = 0;
  float gMin = 255;
  float gMax = 0;
  float bMin = 255;
  float bMax = 0;
  PImage temp = new PImage();
  temp = createImage(orig.width, orig.height, RGB);
  loadPixels();
  for (int pixel = 0; pixel < orig.pixels.length; pixel++) {
    rMin = min(red(orig.pixels[pixel]), rMin);
    rMax = max(red(orig.pixels[pixel]), rMax);
    gMin = min(green(orig.pixels[pixel]), gMin);
    gMax = max(green(orig.pixels[pixel]), gMax);
    bMin = min(blue(orig.pixels[pixel]), bMin);
    bMax = max(blue(orig.pixels[pixel]), bMax);
  }
  float rDiff = 1.0*rMax - rMin;
  float gDiff = 1.0*gMax - gMin;
  float bDiff = 1.0*bMax - bMin;
  for (int pixel = 0; pixel < orig.pixels.length; pixel++) {
    float rVal = brightnessMultiplier*(redIntensity/rDiff)*(red(orig.pixels[pixel])-rMin);
    float gVal = brightnessMultiplier*(greenIntensity/gDiff)*(green(orig.pixels[pixel])-gMin);
    float bVal = brightnessMultiplier*(blueIntensity/bDiff)*(blue(orig.pixels[pixel])-bMin);
    temp.pixels[pixel] = color(rVal, gVal, bVal);
  }
  return temp;
}
```

Any additional details?

-

Teamwork

- Generally, Josh took on more photography-oriented tasks while James handled most of the coding. After we had each completed our respective work, James provided valuable insight on the photographs and Josh fleshed out the program’s user interface and functionality slightly more with his former experience with Processing.
- Both team members contributed significantly to project presentations, brainstorming, and general progress on the project.

Resources

- By SharkD - Own work. Download source code., CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=9803283
- Brown, Evan N. “Images of Old Russia, From a Photography Pioneer.” 28 March 2019. https://www.atlasobscura.com/articles/early-color-photography-russia
- Caldwell, Roy L. Department of Integrative Biology, University of California, Berkeley - National Science Foundation [1], Public Domain, https://commons.wikimedia.org/w/index.php?curid=10483908

Appendix: Your Code

Code Language: Processing (Java)

List of code files:

- bnw_to_rgb.pde
- red.JPG
- green.JPG
- blue.JPG

Credits or Thanks

- Thanks Frank and TAs for helping us figure out a doable project that falls under the umbrella of Computational Photography!