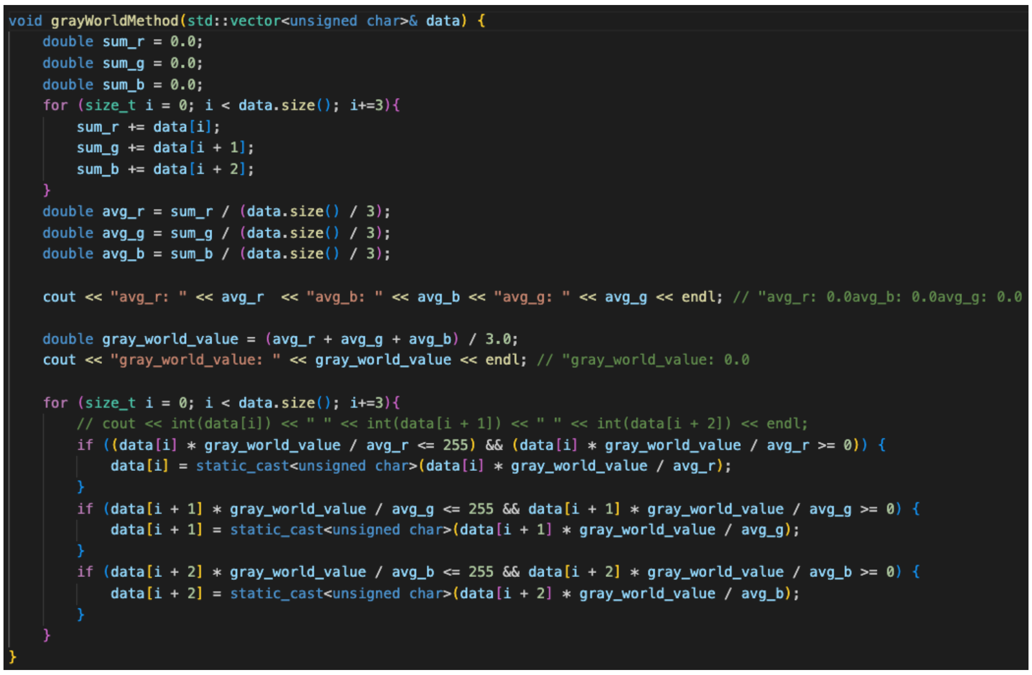
Digital Image Processing: Homework 3 Report

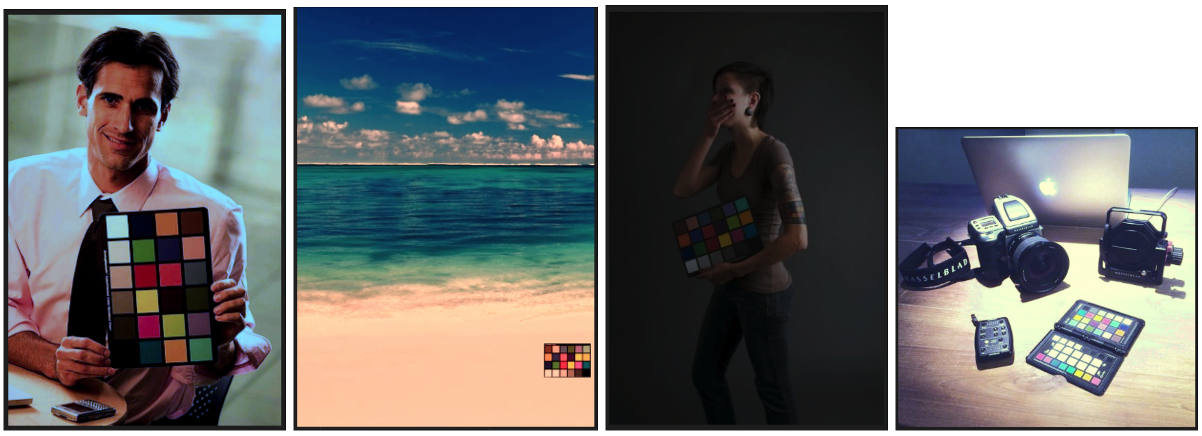
Task1: Chromatic Adaptation

I apply the Gray *World* method to adjust the color temperature of the input images. Simply Calculate the averages of each channel from all pixels, and use above three averages to get the gray world value. For each pixel, multiply the ratio of gray world value and average value of each channel with RGB value respectively.



```cpp
void grayWorldMethod(std::vector<unsigned char>& data) {
    double sum_r = 0.0;
    double sum_g = 0.0;
    double sum_b = 0.0;
    for (size_t i = 0; i < data.size(); i+=3){
        sum_r += data[i];
        sum_g += data[i + 1];
        sum_b += data[i + 2];
    }
    double avg_r = sum_r / (data.size() / 3);
    double avg_g = sum_g / (data.size() / 3);
    double avg_b = sum_b / (data.size() / 3);

    cout << "avg_r: " << avg_r  << "avg_b: " << avg_b << "avg_g: " << avg_g << endl; // "avg_r: 0.0avg_b: 0.0avg_g: 0.0

    double gray_world_value = (avg_r + avg_g + avg_b) / 3.0;
    cout << "gray_world_value: " << gray_world_value << endl; // "gray_world_value: 0.0

    for (size_t i = 0; i < data.size(); i+=3){
        // cout << int(data[i]) << " " << int(data[i + 1]) << " " << int(data[i + 2]) << endl;
        if ((data[i] * gray_world_value / avg_r <= 255) && (data[i] * gray_world_value / avg_r >= 0)) {
            data[i] = static_cast<unsigned char>(data[i] * gray_world_value / avg_r);
        }
        if (data[i + 1] * gray_world_value / avg_g <= 255 && data[i + 1] * gray_world_value / avg_g >= 0) {
            data[i + 1] = static_cast<unsigned char>(data[i + 1] * gray_world_value / avg_g);
        }
        if (data[i + 2] * gray_world_value / avg_b <= 255 && data[i + 2] * gray_world_value / avg_b >= 0) {
            data[i + 2] = static_cast<unsigned char>(data[i + 2] * gray_world_value / avg_b);
        }
    }
}
```

Results of task1:

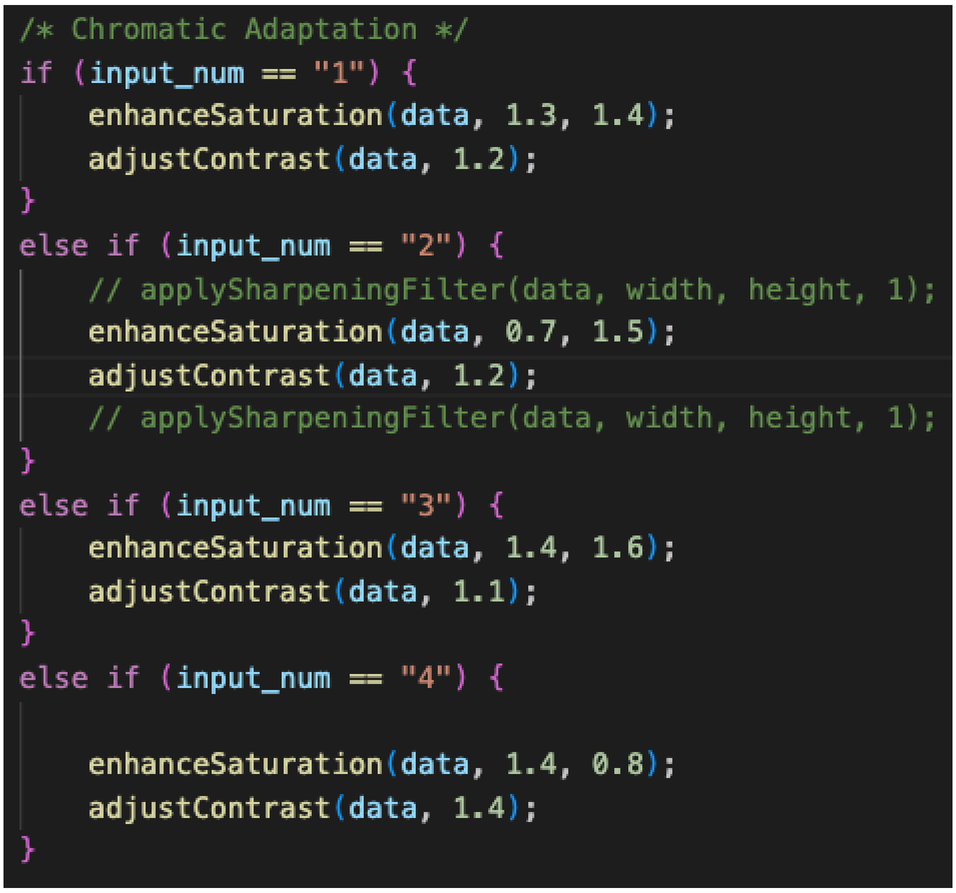


Discussions:

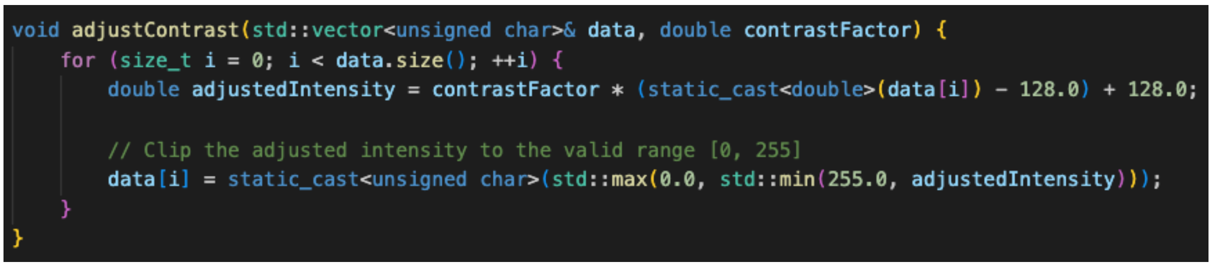
I've also tried the maxRGB method, but it turned out to be a failure.

Task2: Image Enhancement

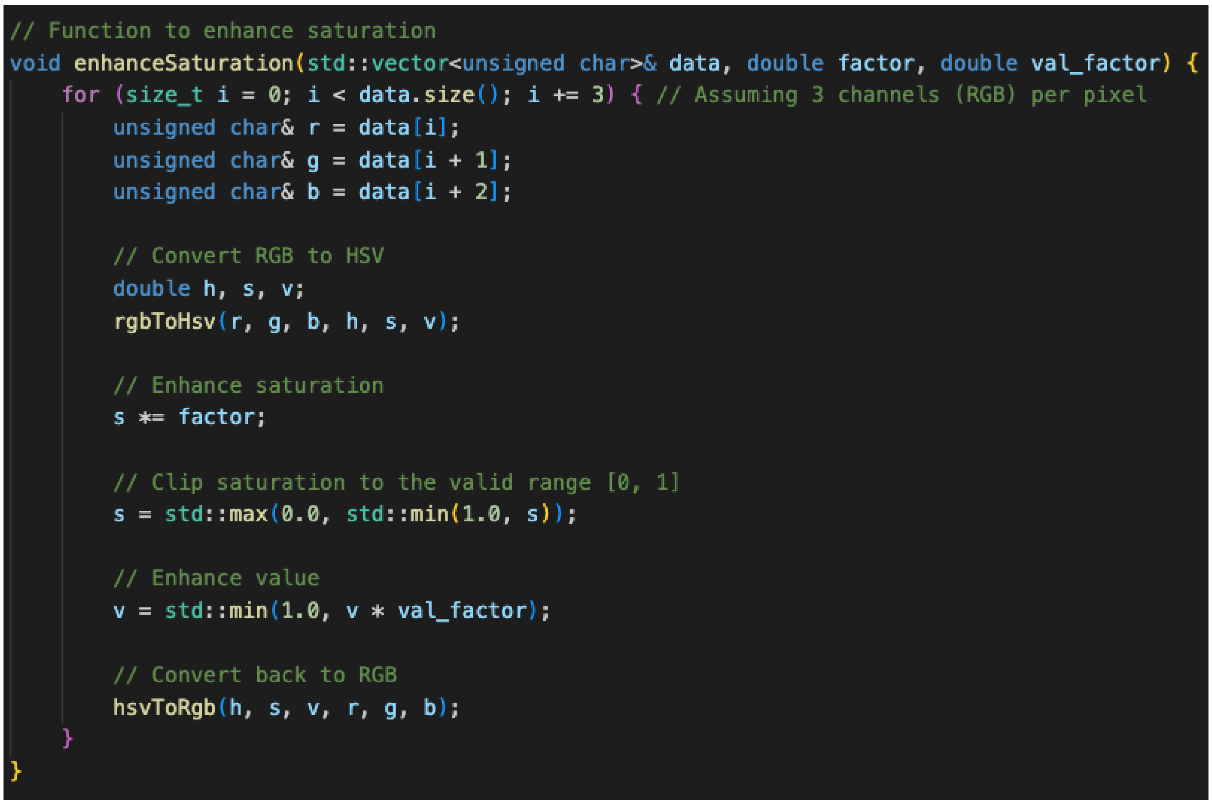
In Task2, I adjust the Saturations, luminous intensity, and contrast of the images to improve the images quality. I convert the images from RGB to HSV, and use HSV value to adjust Saturations and luminous intensity. Below is the config, which for recovering to target images.



```cpp
/* Chromatic Adaptation */
if (input_num == "1") {
    enhanceSaturation(data, 1.3, 1.4);
    adjustContrast(data, 1.2);
}
else if (input_num == "2") {
    // applySharpeningFilter(data, width, height, 1);
    enhanceSaturation(data, 0.7, 1.5);
    adjustContrast(data, 1.2);
    // applySharpeningFilter(data, width, height, 1);
}
else if (input_num == "3") {
    enhanceSaturation(data, 1.4, 1.6);
    adjustContrast(data, 1.1);
}
else if (input_num == "4") {

    enhanceSaturation(data, 1.4, 0.8);
    adjustContrast(data, 1.4);
}
```

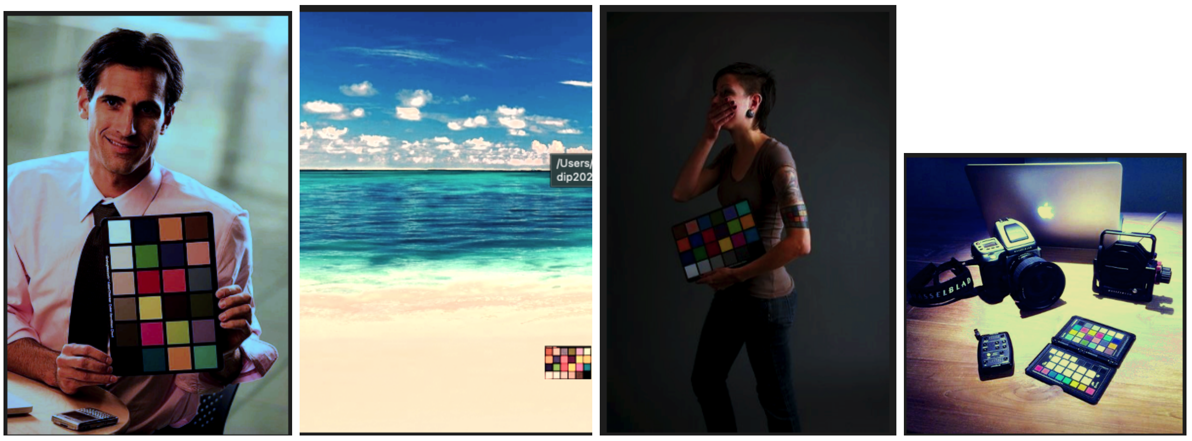


```cpp
void adjustContrast(std::vector<unsigned char>& data, double contrastFactor) {
    for (size_t i = 0; i < data.size(); ++i) {
        double adjustedIntensity = contrastFactor * (static_cast<double>(data[i]) - 128.0) + 128.0;

        // Clip the adjusted intensity to the valid range [0, 255]
        data[i] = static_cast<unsigned char>(std::max(0.0, std::min(255.0, adjustedIntensity)));
    }
}
```



```cpp
// Function to enhance saturation
void enhanceSaturation(std::vector<unsigned char>& data, double factor, double val_factor) {
    for (size_t i = 0; i < data.size(); i += 3) { // Assuming 3 channels (RGB) per pixel
        unsigned char& r = data[i];
        unsigned char& g = data[i + 1];
        unsigned char& b = data[i + 2];

        // Convert RGB to HSV
        double h, s, v;
        rgbToHsv(r, g, b, h, s, v);

        // Enhance saturation
        s *= factor;

        // Clip saturation to the valid range [0, 1]
        s = std::max(0.0, std::min(1.0, s));

        // Enhance value
        v = std::min(1.0, v * val_factor);

        // Convert back to RGB
        hsvToRgb(h, s, v, r, g, b);
    }
}
```

Results of task2:



Discussions:

Above results still miss some details compared with target images, such as the sand details and the wood textures. But I thought that the details may already be broken in the input images.