

Tugas HW0 Vision

Identitas Mahasiswa

Nama: Reza Budiawan
NIM: 33221040

Deskripsi Umum Tugas 0

Tugas yang diberikan melakukan manipulasi/pemrosesan citra menggunakan library (yang ditulis dengan bahasa C). Manipulasi yang dilakukan dilakukan dengan mengubah nilai piksel dari citra. Struktur data dari citra tersebut adalah:

```
typedef struct {  
    int h, w, c;  
    float * data;  
} image;
```

Terdiri dari 3 nilai: h menyatakan tinggi citra; w lebar citra; dan c untuk channel. Nilai citra disimpan dalam bentuk float (pointer). Nilai piksel pertama pada data menyatakan nilai pada channel 0, baris 0, kolom 1. Berikutnya, nilai yang tersimpan merupakan nilai pada channel 0, baris 0, kolom 1; dan seterusnya.

0.0: Getting and setting pixels

Deskripsi

Pada tugas ini diminta untuk mengisi body method pada “get_pixel” & “set_pixel” dalam file “/src/hw0/process_image.c”.

get_pixel

Tujuan dari method “get_pixel” adalah mengambil nilai piksel yang sesuai berdasarkan nilai parameter inputnya. Method get_pixel memiliki 4 parameter input: im, x, y, c. Setiap parameter ini merepresentasikan:

1. im: citra yang diambil nilai pikselnya.
2. x: nilai kolom
3. y: nilai baris
4. c: nilai channel

Jadi ketika method ini dipanggil dengan kode “get_pixel(im, 0, 5, 1)” artinya mengambil nilai piksel dari image “im” pada kolom 0, baris 5, channel 1.

set_pixel

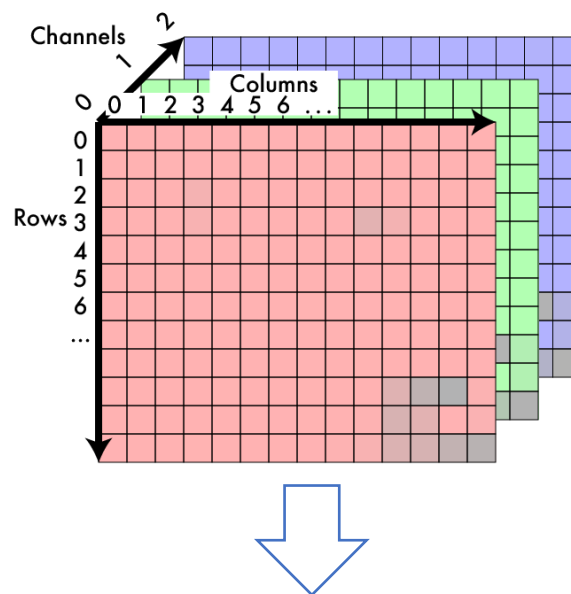
Method “set_pixel” memiliki 5 parameter input: im, x, y, c, dan v. Setiap parameter ini merepresentasikan:

1. im: citra yang diubah nilai pikselnya.
2. x: nilai kolom
3. y: nilai baris
4. c: nilai channel
5. v: nilai piksel yang baru

Penggunaan “set_pixel” dapat dilihat pada tryhw0.py yang menge-set nilai piksel menjadi 0 pada indeks 0 hingga height×width (mengeset semua nilai piksel di channel 0 menjadi 0).

Solusi

Representasi nilai piksel, disimpan dalam bentuk array. Sehingga, diperlukan perhitungan nilai indeks yang sesuai dari array tersebut. Berdasarkan struktur data yang diberikan, diilustrasikan penyimpanan nilai piksel pada array sebagai berikut:



Channel 0 Baris 0 Kolom 0	Channel 0 Baris 0 Kolom 1	Channel 0 Baris 0 Kolom 2	Channel 0 Baris 0 Kolom 3	Channel 0 Baris 0 Kolom 4	Channel 0 Baris 0 Kolom 5	Channel 0 Baris 0 Kolom 6	...	Channel 2 Baris n Kolom n
0	1	2	3	4	5	6	...	

Berdasarkan hal di atas, perlu dilakukan perhitungan nilai indeks yang tepat untuk mengembalikan nilai piksel (im.data). Perhitungannya adalah:

$$\text{Nilai indeks} = (\text{channel} \times \text{image height} \times \text{image width}) + (y \times \text{image width}) + x$$

Hal ini dilakukan karena nilai channel berikutnya disimpan setelah 1 channel selesai disimpan, yaitu setelah sebanyak nilai (height×width) tersimpan. Sedangkan untuk menuju ke titik yang diinginkan, tergantung nilai kolom dan nilai baris yang diberikan.

Contohnya, untuk mendapatkan piksel baris (y = 3) kolom (x = 2) dari citra berukuran 5×5, maka nilai indeks dapat diakses setelah “melewati” 5 kolom beberapa kali sesuai jumlah baris, yaitu 3, dan ditambah jumlah kolom, yaitu 2.

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

Indeks 17, menyimpan nilai baris ke-3 kolom ke-2.

Dikarenakan parameter input kolom (x), baris (y), dan nilai channel (c) dapat menerima input bernilai negative atau melebihi nilai dari image, maka diperlukan pengecekan terhadap nilai yang tidak bersesuaian. Sehingga, kode dapat dituliskan sebagai berikut:

```
float get_pixel(image im, int x, int y, int c)
{
    if (x < 0){
        x = 0;
    }else if (x >= im.w){
        x = im.w - 1;
    }

    if (y < 0){
        y = 0;
    }else if (y >= im.h){
        y = im.h - 1;
    }

    if (c >= 0 && c < im.c){
        return im.data[c*im.h*im.w + y*im.w + x];
    }else{
        return 0;
    }
}
```

Untuk method set_pixel, perhitungan yang sama juga dilakukan. Perbedaannya, pada method ini, indeks yang ditentukan akan diset nilainya dengan parameter input "v". Berikut kodenya:

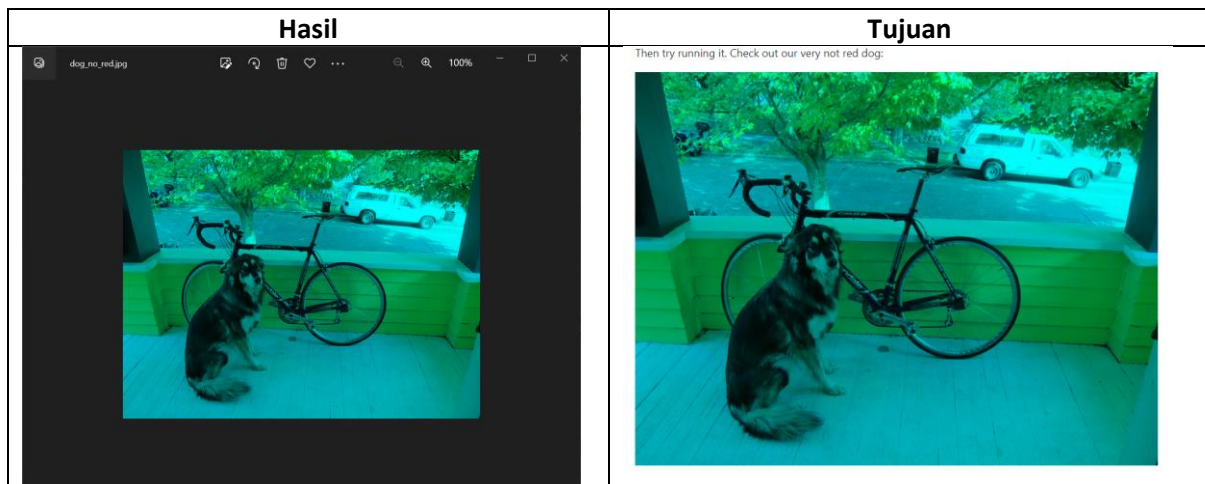
```
void set_pixel(image im, int x, int y, int c, float v)
{
    if (x >= 0 && x < im.w && y >= 0 && y < im.h && c >= 0 && c < im.c){
        im.data[c*im.h*im.w + y*im.w + x] = v;
    }
}
```

Hasil

Berdasarkan hasil test yang dijalankan untuk hw0, test_get_pixel & test_set_pixel sudah memiliki status "passed".

```
C:\Users\rezab\Documents\Studi Lanjut\Sem 3\Vision - IF6083\If6083ComputerVision>uwimg test hw0
passed: [test_get_pixel] testing [within_eps(0, get_pixel(im, 0,0,0), EPS)] in ./src/test.c, line 95
passed: [test_get_pixel] testing [within_eps(1, get_pixel(im, 1,0,1), EPS)] in ./src/test.c, line 96
passed: [test_get_pixel] testing [within_eps(0, get_pixel(im, 2,0,1), EPS)] in ./src/test.c, line 97
passed: [test_get_pixel] testing [within_eps(1, get_pixel(im, 0,3,1), EPS)] in ./src/test.c, line 100
passed: [test_get_pixel] testing [within_eps(1, get_pixel(im, 7,8,0), EPS)] in ./src/test.c, line 101
passed: [test_get_pixel] testing [within_eps(0, get_pixel(im, 7,8,1), EPS)] in ./src/test.c, line 102
passed: [test_get_pixel] testing [within_eps(1, get_pixel(im, 7,8,2), EPS)] in ./src/test.c, line 103
passed: [test_set_pixel] testing [same_image(d, gt, EPS)] in ./src/test.c, line 121
```

Jika menjalankan “uwimg tryhw0.py”, maka akan terbentuk file “dog_no_red.jpg” yang memperlihatkan gambar anjing yang nilai piksel pada channel 0-nya telah diubah menjadi 0. Hal ini sesuai dengan tujuan pada tugas 0.0.



0.1 Copying images

Deskripsi

Isilah body method “copy_image”. Method ini mengembalikan image hasil copy dari original image yang diberikan pada parameter input. Gunakan memcpy untuk mengisi body method “copy_image”.

Solusi

Function “memcpy” pada bahasa C memiliki 3 parameter: variable destination, variable original, dan jumlah byte yang dicopy. Untuk itu, body method pada copy_image dapat dituliskan sebagai berikut:

```
image copy_image(image im)
{
    image copy = make_image(im.w, im.h, im.c);
    memcpy(copy.data, im.data, im.c*im.h*im.w*sizeof(float));
    return copy;
}
```

Hasil

Berikut hasil test dari hw0, scenario “test_copy” menunjukkan status “passed”.

```
C:\Users\rezab\Documents\Studi Lanjut\Sem 3\Vision - IF6083\If6083ComputerVision>uwimg test hw0
passed: [test_get_pixel] testing [within_eps(0, get_pixel(im, 0,0,0), EPS)] in ./src/test.c, line 95
passed: [test_get_pixel] testing [within_eps(1, get_pixel(im, 1,0,1), EPS)] in ./src/test.c, line 96
passed: [test_get_pixel] testing [within_eps(0, get_pixel(im, 2,0,1), EPS)] in ./src/test.c, line 97
passed: [test_get_pixel] testing [within_eps(1, get_pixel(im, 0,3,1), EPS)] in ./src/test.c, line 100
passed: [test_get_pixel] testing [within_eps(1, get_pixel(im, 7,8,0), EPS)] in ./src/test.c, line 101
passed: [test_get_pixel] testing [within_eps(0, get_pixel(im, 7,8,1), EPS)] in ./src/test.c, line 102
passed: [test_get_pixel] testing [within_eps(1, get_pixel(im, 7,8,2), EPS)] in ./src/test.c, line 103
passed: [test_set_pixel] testing [same_image(d, gt, EPS)] in ./src/test.c, line 121
passed: [test_copy] testing [same_image(c, gt, EPS)] in ./src/test.c, line 141
```

0.2 Grayscale image

Deskripsi

Pada tugas ini, dilakukan pengubahan bentuk warna (RGB) menjadi grayscale. Nilai piksel baru didapat dari persamaan berikut:

$$Y' = 0.299 R' + 0.587 G' + .114 B'$$

Berdasarkan hal di atas, isilah body method “rgb_to_grayscale”.

Solusi

Method “rgb_to_grayscale” mengembalikan image 1 channel dengan sebuah parameter input (original image). Pada method ini dapat dilakukan nested looping (berdasarkan *image-per-channel*). Perulangan dilakukan sebanyak jumlah channel & ukuran image dari citra original yang akan diubah ke bentuk grayscale. Di dalam perulangan tersebut, data piksel citra original dikalikan dengan intensitas sesuai rumus. Sehingga, kodenya dapat dituliskan sebagai berikut:

```
image rgb_to_grayscale(image im)
{
    assert(im.c == 3);
    image gray = make_image(im.w, im.h, 1);

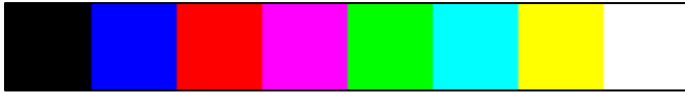
    float scale[] = {0.299, 0.587, 0.114};
    for (int i = 0; i < im.c; ++i){
        for (int j = 0; j < im.h*im.w; ++j){
            gray.data[j] += scale[i] * im.data[i*im.h*im.w+j];
        }
    }
    return gray;
}
```

Hasil



Jika hasil test_grayscale dijalankan, maka statusnya adalah passed.

```
passed: [test_grayscale] testing [same_image(gray, gt, EPS)] in ./src/test.c, line 131
```

Untuk melihat apakah kode sudah berjalan seperti yang diinginkan, dapat dijalankan command “python tryhw0.py”. Jika tidak ada error, maka akan terbentuk sebuah file baru Graybar. File ini merupakan pengubahan dari colorbar.png, sebuah gambar yang terdiri dari beberapa warna.



Gambar di atas (citra ori tidak mengandung frame hitam), akan diubah menjadi dalam bentuk citra grayscale. Hal ini sudah sesuai dengan tujuan akhir dari tugas.

Hasil	Tujuan
	<pre># 3. Grayscale image im = load_image("data/colorbar.png") graybar = rgb_to_grayscale(im) save_image(graybar, "graybar")</pre> 

0.3 Shifting the image colors

Deskripsi

Pada tugas ini dilakukan “shifting” citra untuk menambah atau mengurangi pewarnaan pada citra. Body method yang harus diisi adalah method “shift_image” dengan 3 parameter input. Parameter ini terdiri dari citra im, channel c, dan data v. Nilai v merupakan nilai shifting. Jika nilai .4, maka shifting dilakukan 40% (menambah nilai 0.4 dari nilai awal).

Solusi

Untuk melakukan shifting, hal yang dilakukan adalah menambah 0.4 pada nilai piksel original di channel sesuai parameter input. Sehingga, dapat dituliskan kode sebagai berikut:

```
void shift_image(image im, int c, float v)
{
    for (int i = 0; i < im.h*im.w; ++i){
        im.data[c*im.h*im.w+i] += v;
    }
}
```

Hasil

Setelah melakukan make & menjalankan test terhadap method, didapat scenario testing “test_shift” sudah memiliki status “passed”.

```
passed: [test_copy] testing [same_image(c, gt, EPS)] in ./src/test.c, line 141
passed: [test_shift] testing [within_eps(c.data[0], im.data[0], EPS)] in ./src/test.c, line 170
passed: [test_shift] testing [within_eps(c.data[im.w*im.h + 13], im.data[im.w*im.h+13] + .1, EPS)] in
171
passed: [test_shift] testing [within_eps(c.data[2*im.w*im.h + 72], im.data[2*im.w*im.h+72], EPS)] in
172
passed: [test_shift] testing [within_eps(c.data[im.w*im.h + 47], im.data[im.w*im.h+47] + .1, EPS)] in
```

Selain itu, jika dijalankan command “python tryhw0.py”, maka citra anjing berikut, akan mengalami perubahan warna.

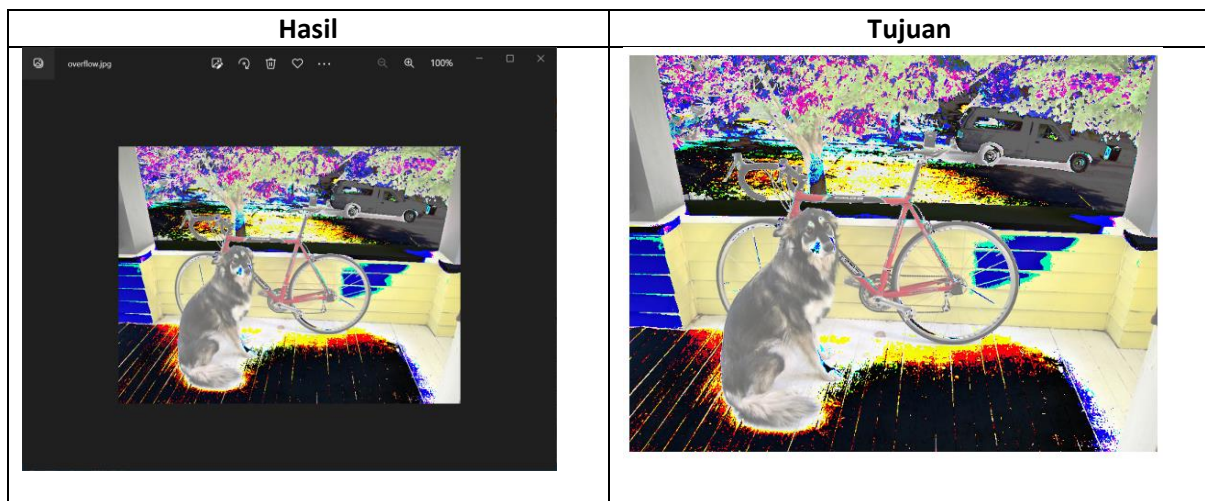


Figure 1: Citra Original Anjing

Kode shifting:

```
im = load_image("data/dog.jpg")
shift_image(im, 0, .4)
shift_image(im, 1, .4)
shift_image(im, 2, .4)
save_image(im, "overflow")
```

Citra hasil shifting:



0.4 Clamping the image values

Deskripsi

Berdasarkan hasil dari shifting sebelumnya, nilai piksel dapat melebihi yang seharusnya (nilai maksimal adalah 1). Hal ini berdampak pada gambar hasil yang memperlihatkan hasil yang bagus untuk menambah intensitas warna/cahaya. Nilai di atas 1, akan di-convert ke byte array & akan berubah

menjadi nilai yang kecil (mendekati 0). Hal ini menyebabkan area yang seharusnya terang menjadi hitam & dark.

Operasi clamping dapat diterapkan pada gambar hasil di atas. Hal ini dilakukan dengan mengubah nilai negative (kurang dari 0) menjadi 0, dan nilai piksel yang melebihi 1 menjadi 1. Hal ini menjadikan nilai piksel pada citra tetap dalam rentang nilai 0-1. Pada tugas ini, akan diisi body dari method “clamp_image”. Method ini memiliki 1 parameter masukan berupa citra yang akan di-clamping.

Solusi

Operasi clamping akan menormalkan nilai yang melebihi ambang batas. Kode yang diberikan adalah melakukan looping untuk semua nilai piksel pada image & mengecek apakah nilainya melebihi ambang. Jika melebihi, maka ubah nilai tersebut. Kode yang dituliskan adalah sebagai berikut:

```
void clamp_image(image im)
{
    for (int i = 0; i < im.c*im.h*im.w; ++i){
        if (im.data[i] > 1){
            im.data[i] = 1;
        }else if (im.data[i] < 0){
            im.data[i] = 0;
        }
    }
}
```

Hasil

Berikut hasil test dari scenario “test_clamp”:

```
passed: [test_shift] testing [within_eps(c.data[im.w*im.h + 47], im.data[im.w*im.h + 47], EPS)] in ./src/test.c, line 173
passed: [test_clamp] testing [same_image(c, im, EPS)] in ./src/test.c, line 160
```

Hasilnya adalah “passed”. Selain itu, jika command “python tryhw0.py” dieksekusi, maka gambar dog hasil clamping dari overflow akan menjadi seperti gambar di bawah ini.



Terlihat area yang awalnya hitam pada operasi sebelumnya sudah menjadi cerah (intensitas warna naik) seperti yang diinginkan.

0.5 RGB to Hue, Saturation, Value

Deskripsi

Ada variasi colorspace selain RGB, salah satunya adalah HSV (Hue, Saturation, dan Value). Tugas kali ini akan mentranslasikan kubik sRGB menjadi silinder HSV. Hue merupakan base dari warna piksel, saturation menyatakan intensitas, dan value merepresentasikan kecerahan. Body method “rgb_to_hsv” akan diisi menggunakan rumus yang ada.

Solusi

Untuk mengisi body method “rgb_to_hsv”, dapatkan nilai r, g, dan b terlebih dahulu. Berikutnya, dapatkan nilai v, dengan mengambil nilai maksimum dari r, g, b. Nilai m, didapat dari nilai minimum r, g, b. Dapatkan selisih keduanya untuk nilai c. Nilai saturation diambil dari rasio c terhadap v. Selanjutnya, nilai h dihitung berdasarkan ketentuan.

Kode dari transformasi ini dituliskan sebagai berikut:

```
void rgb_to_hsv(image im)
{
    float r, g, b, h, s, v;
    float m, c;

    for (int i = 0; i < im.h*im.w; ++i){
        r = im.data[i];
        g = im.data[i + im.h * im.w];
        b = im.data[i + 2 * im.h * im.w];
        v = three_way_max(r, g, b);
        m = three_way_min(r, g, b);
        c = v - m;

        if (v == 0){
            s = 0;
        }else{
            s = c / v;
        }

        if (v == m){
            h = 0;
        }else{
            if (v == r){
                h = (g - b) / c;
            }else if (v == g){
                h = (b - r) / c + 2;
            }else{
                h = (r - g) / c + 4;
            }
        }
    }
}
```

```

        if (h < 0){
            h = h / 6 + 1;
        }else{
            h /= 6;
        }

        im.data[i] = h;
        im.data[i + im.h*im.w] = s;
        im.data[i + 2*im.h*im.w] = v;
    }
}

```

Hasil

Hasil dari scenario test “test_rgb_to_hsv” memiliki status passed.

```

passed: [test_grayscale] testing [same_image(gray, gt, EPS)] in ./src/test.c, line 131
passed: [test_rgb_to_hsv] testing [same_image(im, hsv, EPS)] in ./src/test.c, line 183

```

0.6 HSV to RGB

Deskripsi

Isi body method “hsv_to_rgb”.

Solusi

```

void hsv_to_rgb(image im)
{
    float h, s, v, r, g, b;
    float dst, mid_pos, mid_neg, m;
    for (int i = 0; i < im.h*im.w; ++i){
        h = im.data[i];
        s = im.data[i + im.h*im.w];
        v = im.data[i + 2*im.h*im.w];
        h *= 6;
        m = v - s * v;
        dst = h - floor(h);
        mid_pos = s * v * dst + m;
        mid_neg = s * v * (1 - dst) + m;
        if (h >= 0 && h < 1){
            r = v;
            g = mid_pos;
            b = m;
        }
    }
}

```

```

    }else if (h >= 1 && h < 2){
        g = v;
        r = mid_neg;
        b = m;
    }else if (h >= 2 && h < 3){
        g = v;
        b = mid_pos;
        r = m;
    }else if (h >= 3 && h < 4){
        b = v;
        g = mid_neg;
        r = m;
    }else if (h >= 4 && h < 5){
        b = v;
        r = mid_pos;
        g = m;
    }else{
        r = v;
        b = mid_neg;
        g = m;
    }
    im.data[i] = r;
    im.data[i + im.h*im.w] = g;
    im.data[i + 2*im.h*im.w] = b;
}
}

```

Hasil

Status passed didapatkan untuk scenario test “test_hsv_to_rgb”.

```

C:\Users\rezab\Documents\Studi Lanjut\Sem 3\Vision - IF6083\If6083ComputerVision>uwimg test hw0
passed: [test_get_pixel] testing [within_eps(0, get_pixel(im, 0,0,0), EPS)] in ./src/test.c, line 95
passed: [test_get_pixel] testing [within_eps(1, get_pixel(im, 1,0,1), EPS)] in ./src/test.c, line 96
passed: [test_get_pixel] testing [within_eps(0, get_pixel(im, 2,0,1), EPS)] in ./src/test.c, line 97
passed: [test_get_pixel] testing [within_eps(1, get_pixel(im, 0,3,1), EPS)] in ./src/test.c, line 100
passed: [test_get_pixel] testing [within_eps(1, get_pixel(im, 7,8,0), EPS)] in ./src/test.c, line 101
passed: [test_get_pixel] testing [within_eps(0, get_pixel(im, 7,8,1), EPS)] in ./src/test.c, line 102
passed: [test_get_pixel] testing [within_eps(1, get_pixel(im, 7,8,2), EPS)] in ./src/test.c, line 103
passed: [test_set_pixel] testing [same_image(d, gt, EPS)] in ./src/test.c, line 121
passed: [test_copy] testing [same_image(c, gt, EPS)] in ./src/test.c, line 141
passed: [test_shift] testing [within_eps(c.data[0], im.data[0], EPS)] in ./src/test.c, line 170
passed: [test_shift] testing [within_eps(c.data[im.w*im.h + 13], im.data[im.w*im.h+13] + .1, EPS)] in
171
passed: [test_shift] testing [within_eps(c.data[2*im.w*im.h + 72], im.data[2*im.w*im.h+72], EPS)] in
172
passed: [test_shift] testing [within_eps(c.data[im.w*im.h + 47], im.data[im.w*im.h+47] + .1, EPS)] in
173
passed: [test_clamp] testing [same_image(c, im, EPS)] in ./src/test.c, line 160
passed: [test_grayscale] testing [same_image(gray, gt, EPS)] in ./src/test.c, line 131
passed: [test_rgb_to_hsv] testing [same_image(im, hsv, EPS)] in ./src/test.c, line 183
passed: [test_hsv_to_rgb] testing [same_image(im, c, EPS)] in ./src/test.c, line 194
17 tests, 17 passed, 0 failed

```

Hasil di atas memperlihatkan semua scenario test telah memiliki status passed.

Selain itu, hasil pengubahan dari RGB \rightarrow HSV, melakukan shifting & clamp, lalu HSV \rightarrow RGB akan menghasilkan gambar di bawah ini, yang sesuai dengan hasil yang ditunjukkan pada persoalan.



0.7 A small amount of extra credit

Deskripsi

Implementasikan “void scale_image(image im, int c, float v);” untuk melakukan penskalaan sebuah channel. Kode yang dilakukan hampir sama seperti shifting (tugas 0.3).

Solusi

Kode untuk penskalaan, hampir sama seperti shifting. Hanya saja, shifting melakukan penambahan. Sedangkan penskalaan melakukan perkalian.

```
void scale_image(image im, int c, float v)
{
    for (int i = 0; i < im.h*im.w; ++i){
        im.data[c*im.h*im.w+i] = im.data[c*im.h*im.w+i] * v;
    }
}
```

File process_image.c

```
#include <stdio.h>
#include <string.h>
#include <assert.h>
#include <math.h>
#include "image.h"

float get_pixel(image im, int x, int y, int c)
{
    if (x < 0){
        x = 0;
    }else if (x >= im.w){
        x = im.w - 1;
    }

    if (y < 0){
        y = 0;
    }else if (y >= im.h){
        y = im.h - 1;
    }

    if (c >= 0 && c < im.c){
        return im.data[c*im.h*im.w + y*im.w + x];
    }else{
        return 0;
    }
}

void set_pixel(image im, int x, int y, int c, float v)
{
    if (x >= 0 && x < im.w && y >= 0 && y < im.h && c >= 0 && c < im.c){
        im.data[c*im.h*im.w + y*im.w + x] = v;
    }
}

image copy_image(image im)
{
    image copy = make_image(im.w, im.h, im.c);
    memcpy(copy.data, im.data, im.c*im.h*im.w*sizeof(float));
    return copy;
}

image rgb_to_grayscale(image im)
{
    assert(im.c == 3);
    image gray = make_image(im.w, im.h, 1);

    float scale[] = {0.299, 0.587, 0.114};
    for (int i = 0; i < im.c; ++i){
        for (int j = 0; j < im.h*im.w; ++j){
            gray.data[j] += scale[i] * im.data[i*im.h*im.w+j];
        }
    }
    return gray;
}

void shift_image(image im, int c, float v)
{
    for (int i = 0; i < im.h*im.w; ++i){
        im.data[c*im.h*im.w+i] = im.data[c*im.h*im.w+i] + v;
    }
}
```

```

void clamp_image(image im)
{
    for (int i = 0; i < im.c*im.h*im.w; ++i){
        if (im.data[i] > 1){
            im.data[i] = 1;
        }else if (im.data[i] < 0){
            im.data[i] = 0;
        }
    }
}

// These might be handy
float three_way_max(float a, float b, float c)
{
    return (a > b) ? ( (a > c) ? a : c ) : ( (b > c) ? b : c ) ;
}

float three_way_min(float a, float b, float c)
{
    return (a < b) ? ( (a < c) ? a : c ) : ( (b < c) ? b : c ) ;
}

void rgb_to_hsv(image im)
{
    float r, g, b, h, s, v;
    float m, c;

    for (int i = 0; i < im.h*im.w; ++i){
        r = im.data[i];
        g = im.data[i + im.h * im.w];
        b = im.data[i + 2 * im.h * im.w];
        v = three_way_max(r, g, b);
        m = three_way_min(r, g, b);
        c = v - m;

        if (v == 0){
            s = 0;
        }else{
            s = c / v;
        }

        if (v == m){
            h = 0;
        }else{
            if (v == r){
                h = (g - b) / c;
            }else if (v == g){
                h = (b - r) / c + 2;
            }else{
                h = (r - g) / c + 4;
            }
        }

        if (h < 0){
            h = h / 6 + 1;
        }else{
            h /= 6;
        }

        im.data[i] = h;
        im.data[i + im.h*im.w] = s;
        im.data[i + 2*im.h*im.w] = v;
    }
}

```

```

}

void hsv_to_rgb(image im)
{
    float h, s, v, r, g, b;
    float dst, mid_pos, mid_neg, m;
    for (int i = 0; i < im.h*im.w; ++i){
        h = im.data[i];
        s = im.data[i + im.h*im.w];
        v = im.data[i + 2*im.h*im.w];
        h *= 6;
        m = v - s * v;
        dst = h - floor(h);
        mid_pos = s * v * dst + m;
        mid_neg = s * v * (1 - dst) + m;
        if (h >= 0 && h < 1){
            r = v;
            g = mid_pos;
            b = m;
        }else if (h >= 1 && h < 2){
            g = v;
            r = mid_neg;
            b = m;
        }else if (h >= 2 && h < 3){
            g = v;
            b = mid_pos;
            r = m;
        }else if (h >= 3 && h < 4){
            b = v;
            g = mid_neg;
            r = m;
        }else if (h >= 4 && h < 5){
            b = v;
            r = mid_pos;
            g = m;
        }else{
            r = v;
            b = mid_neg;
            g = m;
        }
        im.data[i] = r;
        im.data[i + im.h*im.w] = g;
        im.data[i + 2*im.h*im.w] = b;
    }
}

void scale_image(image im, int c, float v)
{
    for (int i = 0; i < im.h*im.w; ++i){
        im.data[c*im.h*im.w+i] = im.data[c*im.h*im.w+i] * v;
    }
}

```