

# Tugas HW1 Vision

## Identitas Mahasiswa

Nama: Reza Budiawan  
NIM: 33221040

## Deskripsi Umum Tugas 1

Pada homework 1 (HW1), dilakukan operasi untuk mengubah ukuran dari citra (resizing image). Terdapat 4 method yang harus dituliskan body-nya dalam file "resize\_image.c".

### 1.1 Nearest Neighbor Interpolation

#### Deskripsi

Tuliskan body method dari "nn\_interpolate". Kode yang diharapkan adalah pengubahan ukuran citra menggunakan nearest neighbour interpolation.

#### Solusi

Nearest neighbour interpolate akan mengisi nilai piksel dari nilai titik terdekatnya:

$$f(x, y, z) = im(round(x), round(y), z)$$

Method "nn\_interpolate" memiliki 4 parameter input: citra, x, y, dan channel. Nilai yang dikembalikan oleh method ini merupakan nilai piksel hasil nearest neighbour interpolation. Untuk mengambil closest int dari x & y, digunakan function roundf. Nilai piksel ini didapat menggunakan method get\_pixel dari tugas sebelumnya (HW0).

```
float nn_interpolate(image im, float x, float y, int c)
{
    return get_pixel(im, roundf(x), roundf(y), c);
}
```

#### Hasil

Menggunakan command "uwimg test hw1", scenario testing dijalankan. Gambar di bawah ini memperlihatkan hasil scenario test "test\_nn\_interpolate".

```
C:\Users\rezab\Documents\Studi Lanjut\Sem 3\Vision - IF6083\If6083ComputerVision>uwimg test hw1
passed: [test_nn_interpolate] testing [within_eps(nn_interpolate(im, -.5, -.5, 0) , 0.231373, EPS)]
e 202
passed: [test_nn_interpolate] testing [within_eps(nn_interpolate(im, -.5, .5, 1) , 0.239216, EPS)]
e 203
passed: [test_nn_interpolate] testing [within_eps(nn_interpolate(im, .499, .5, 2) , 0.207843, EPS)]
e 204
passed: [test_nn_interpolate] testing [within_eps(nn_interpolate(im, 14.2, 15.9, 1), 0.690196, EPS)]
e 205
```

Berdasarkan hasil di atas, terlihat bahwa nilai yang diuji telah sesuai, sehingga hasil uji memiliki status “passed”.

## 1.2 Nearest Neighbor Resizing

### Deskripsi

Tuliskan body method “nn\_resize”. Method ini memiliki 4 parameter: citra, width, height. Hasil balikan method adalah citra yang sudah di-resize. Hal yang dilakukan pada method ini yaitu:

- Membuat citra berukuran sama seperti citra input
- Melakukan looping pada tiap pixel & memetakannya
- Menggunakan method “nn\_interpolate” untuk mengisi image

### Solusi

```
image nn_resize(image im, int w, int h)
{
    image im_nn = make_image(w, h, im.c);

    float w_scale = (float)im.w / w;
    float h_scale = (float)im.h / h;
    for (int i = 0; i < im.c; ++i){
        for (int j = 0; j < h; ++j){
            for (int k = 0; k < w; ++k){
                float w_mapped = w_scale * (k + 0.5) - 0.5;
                float h_mapped = h_scale * (j + 0.5) - 0.5;
                float v = nn_interpolate(im, w_mapped, h_mapped, i);
                set_pixel(im_nn, k, j, i, v);
            }
        }
    }
    return im_nn;
}
```

### Hasil



Pengujian scenario test dari “test\_nn\_resize” memiliki status “passed”.

```
C:\Users\rezab\Documents\Studi Lanjut\Sem 3\Vision - IF6083\If6083ComputerVision>uwimg test
passed: [test_nn_interpolate] testing [within_eps(nn_interpolate(im, -.5, -.5, 0) , 0.231373
e 202
passed: [test_nn_interpolate] testing [within_eps(nn_interpolate(im, -.5, .5, 1) , 0.239216,
203
passed: [test_nn_interpolate] testing [within_eps(nn_interpolate(im, .499, .5, 2) , 0.207843
e 204
passed: [test_nn_interpolate] testing [within_eps(nn_interpolate(im, 14.2, 15.9, 1), 0.69019
ne 205
passed: [test_nn_resize] testing [same_image(resized, gt, EPS)] in ./src/test.c, line 225
passed: [test_nn_resize] testing [same_image(resized2, gt2, EPS)] in ./src/test.c, line 233
```

Selain itu, hasil eksekusi command “python tryhw1.py”, membentuk sebuah file “dog4x-nn”. Gambar dari file ini merupakan hasil resize dari dogsmall.jpg (width=192, height=144).



Hasil resize (perbesaran 4x) dari citra di atas sudah sama/sesuai dengan harapan/tujuan.

Hasil	Tujuan
 <p>Width = 768 Height = 576</p>	 <p>Width = 768 Height = 576</p>

Hasil dari nearest neighbor interpolation ini terlihat blocky.

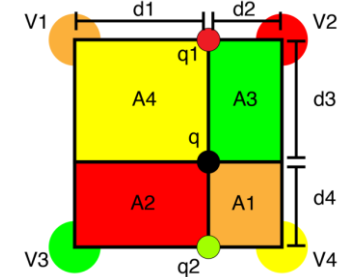
## 1.3 Bilinear Interpolation

### Deskripsi

Tambahkan body method untuk “bilinear\_interpolate”.

### Solusi

Bilinear interpolation diilustrasikan dengan menggunakan titik berikut:

	<p>Perhitungan untuk mendapatkan nilai q:</p> <ul style="list-style-type: none"> <li>• <math>q1 = V1 * d2 + V2 * d1</math></li> <li>• <math>q2 = V3 * d2 + V4 * d1</math></li> <li>• <math>q = q1 * d4 + q2 * d3</math></li> </ul>
---	--

Untuk mendapatkan nilai v1-v4, dapat digunakan fungsi floor dan ceil. Fungsi floor akan mengembalikan nilai int tertinggi yang tidak lebih dari x. Contoh,  $x=1.6 \rightarrow 1.0$ . Sedangkan fungsi ceil akan mengembalikan nilai int tertinggi yang tidak kurang dari x. Contoh,  $x=2.3 \rightarrow 3.0$ .

```
float bilinear_interpolate(image im, float x, float y, int c)
{
    float v1 = get_pixel(im, floorf(x), floorf(y), c);
    float v2 = get_pixel(im, ceilf(x), floorf(y), c);
    float v3 = get_pixel(im, floorf(x), ceilf(y), c);
    float v4 = get_pixel(im, ceilf(x), ceilf(y), c);

    float q1, q2, q;

    if (floorf(x) == ceilf(x)){
        q1 = v1;
    }else{
        q1 = v1 * (ceilf(x) - x) + v2 * (x - floorf(x));
    }

    if (floorf(x) == ceilf(x)){
        q2 = v3;
    }else{
        q2 = v4 * (x - floorf(x)) + v3 * (ceilf(x) - x);
    }

    if (floorf(y) == ceilf(y)){
        q = q1;
    }else{
        q = q2 * (y - floorf(y)) + q1 * (ceilf(y) - y);
    }
    return q;
}
```

## Hasil

Status passed pada scenario testing “test\_bl\_interpolate”.

```
passed: [test_bl_interpolate] testing [within_eps(bilinear_interpolate(im, -.5, .5, 1) , 0.237255, EPS)] :
, line 213
passed: [test_bl_interpolate] testing [within_eps(bilinear_interpolate(im, .499, .5, 2) , 0.206861, EPS)]
c, line 214
passed: [test_bl_interpolate] testing [within_eps(bilinear_interpolate(im, 14.2, 15.9, 1), 0.678588, EPS)
.c, line 215
```

## 1.4 Bilinear Resizing

### Deskripsi

Isi method “bilinear\_resize” yang memanggil method sebelumnya (bilinear\_interpolate).

## Solusi

Kode ini hampir sama dengan `nn_resize`, hanya saja memanggil method `bilinear_interpolate`.

```
image bilinear_resize(image im, int w, int h)
{
    image im_bilinear = make_image(w, h, im.c);

    float w_scale = (float)im.w / w;
    float h_scale = (float)im.h / h;
    for (int i = 0; i < im.c; ++i){
        for (int j = 0; j < h; ++j){
            for (int k = 0; k < w; ++k){
                float w_mapped = w_scale * (k + 0.5) - 0.5;
                float h_mapped = h_scale * (j + 0.5) - 0.5;
                float v = bilinear_interpolate(im, w_mapped, h_mapped, i);
                set_pixel(im_bilinear, k, j, i, v);
            }
        }
    }
    return im_bilinear;
}
```

## Hasil

Skenario test untuk “`test_bl_resize`” memiliki status “passed”.



```
C:\Users\rezab\Documents\Studi Lanjut\Sem 3\Vision - IF6083\If6083ComputerVision>uwimg test
passed: [test_nn_interpolate] testing [within_eps(nn_interpolate(im, -.5, -.5, 0) , 0.231373
e 202
passed: [test_nn_interpolate] testing [within_eps(nn_interpolate(im, -.5, .5, 1) , 0.239216,
203
passed: [test_nn_interpolate] testing [within_eps(nn_interpolate(im, .499, .5, 2) , 0.207843
e 204
passed: [test_nn_interpolate] testing [within_eps(nn_interpolate(im, 14.2, 15.9, 1), 0.69019
ne 205
passed: [test_nn_resize] testing [same_image(resized, gt, EPS)] in ./src/test.c, line 225
passed: [test_nn_resize] testing [same_image(resized2, gt2, EPS)] in ./src/test.c, line 233
passed: [test_bl_interpolate] testing [within_eps(bilinear_interpolate(im, -.5, -.5, 0) , 0.
c, line 212
passed: [test_bl_interpolate] testing [within_eps(bilinear_interpolate(im, -.5, .5, 1) , 0.2
, line 213
passed: [test_bl_interpolate] testing [within_eps(bilinear_interpolate(im, .499, .5, 2) , 0.
c, line 214
passed: [test_bl_interpolate] testing [within_eps(bilinear_interpolate(im, 14.2, 15.9, 1), 0
.c, line 215
passed: [test_bl_resize] testing [same_image(resized, gt, EPS)] in ./src/test.c, line 244
passed: [test_bl_resize] testing [same_image(resized2, gt2, EPS)] in ./src/test.c, line 252
passed: [test_multiple_resize] testing [same_image(im, gt, EPS)] in ./src/test.c, line 270
13 tests, 13 passed, 0 failed
```

Selain itu, jika menambahkan kode berikut pada “`tryhw1`” dan mengeksekusinya:

```
im = load_image("data/dogsmall.jpg")
b = bilinear_resize(im, im.w*4, im.h*4)
```

```
save_image(b, "dog4x-bl")
```

Terbentuk sebuah file "dog4x-bl" yang sudah sesuai dengan harapan/tujuan.

Hasil	Tujuan
	
Width = 768 Height = 576	Width = 768 Height = 576

Operasi ini dapat diaplikasikan pada citra dengan perubahan yang tidak terlalu besar. Jika perubahannya terlalu besar, maka citra yang dihasilkan akan bersifat noisy.

## File `resize_image.c`

```
#include <math.h>
#include "image.h"

float nn_interpolate(image im, float x, float y, int c)
{
    return get_pixel(im, roundf(x), roundf(y), c);
}

image nn_resize(image im, int w, int h)
{
    image im_nn = make_image(w, h, im.c);

    float w_scale = (float)im.w / w;
    float h_scale = (float)im.h / h;
    for (int i = 0; i < im.c; ++i){
        for (int j = 0; j < h; ++j){
            for (int k = 0; k < w; ++k){
                float w_mapped = w_scale * (k + 0.5) - 0.5;
                float h_mapped = h_scale * (j + 0.5) - 0.5;
                float v = nn_interpolate(im, w_mapped, h_mapped, i);
                set_pixel(im_nn, k, j, i, v);
            }
        }
    }
}
```

```

    }
    return im_nn;
}

float bilinear_interpolate(image im, float x, float y, int c)
{
    float v1 = get_pixel(im, floorf(x), floorf(y), c);
    float v2 = get_pixel(im, ceilf(x), floorf(y), c);
    float v3 = get_pixel(im, floorf(x), ceilf(y), c);
    float v4 = get_pixel(im, ceilf(x), ceilf(y), c);

    float q1, q2, q;

    if (floorf(x) == ceilf(x)){
        q1 = v1;
    }else{
        q1 = v1 * (ceilf(x) - x) + v2 * (x - floorf(x));
    }

    if (floorf(x) == ceilf(x)){
        q2 = v3;
    }else{
        q2 = v4 * (x - floorf(x)) + v3 * (ceilf(x) - x);
    }

    if (floorf(y) == ceilf(y)){
        q = q1;
    }else{
        q = q2 * (y - floorf(y)) + q1 * (ceilf(y) - y);
    }
    return q;
}

image bilinear_resize(image im, int w, int h)
{
    image im_bilinear = make_image(w, h, im.c);

    float w_scale = (float)im.w / w;
    float h_scale = (float)im.h / h;
    for (int i = 0; i < im.c; ++i){
        for (int j = 0; j < h; ++j){
            for (int k = 0; k < w; ++k){
                float w_mapped = w_scale * (k + 0.5) - 0.5;
                float h_mapped = h_scale * (j + 0.5) - 0.5;
                float v = bilinear_interpolate(im, w_mapped, h_mapped, i);
                set_pixel(im_bilinear, k, j, i, v);
            }
        }
    }
}

```

```
}  
    return im_bilinear;  
}
```