전기전자심화설계 및 소프트웨어실습 〈Assignment 2〉

담당교수 : 김원준
이름 : 201810528 고려욱

1. Histogram of oriented gradients

1) Source Code

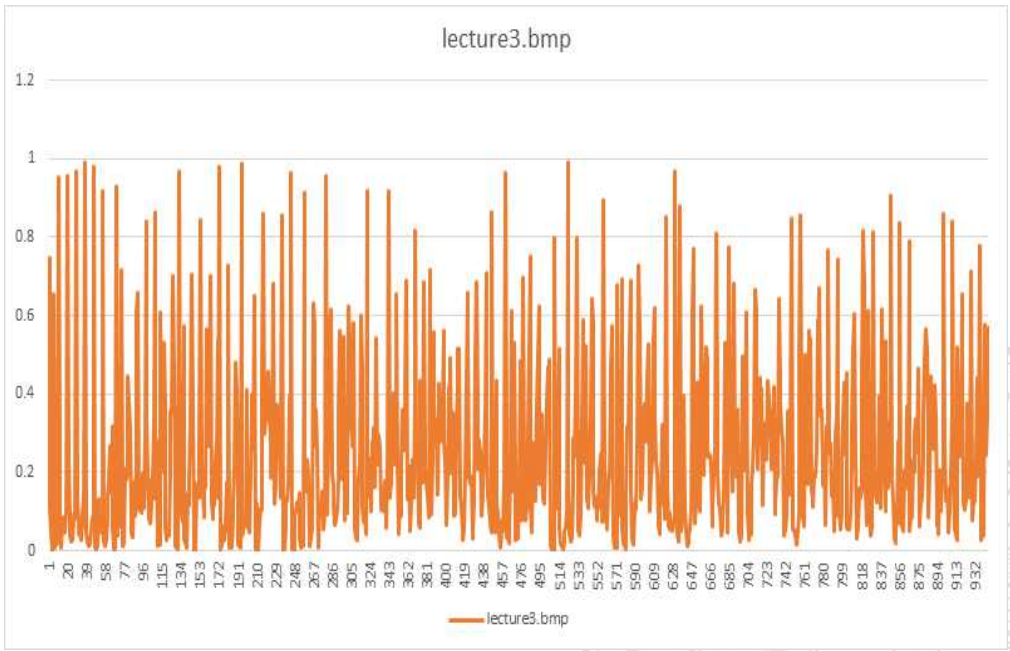
```
#pragma warning(disable:4996)
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include <cmath>

#include <stdio.h>
#include <stdlib.h>
#define PI 3.141592
#define bl_size 16
#define quan 9

using namespace cv;

void gradient_computation(Mat input)
{
	int x, y, xx, yy;
	int height = input.rows;
	int width = input.cols;
	float conv_x, conv_y;
	float dir;
	int deg;
	float max = -1.0;
	float min = 10000000.0;
	float* mag = (float*)calloc(height * width, sizeof(float));
	float* o_dir = (float*)calloc(height * width, sizeof(float));
	Mat imgEdge(height, width, CV_8UC1);
	int mask_x[9] = { -1,-1,-1,0,0,0,1,1,1 };
	int mask_y[9] = { -1,0,1,-1,0,1,-1,0,1 };

	//compute fx and fy
	for (y = 0; y < height; y++) {
	 for (x = 0; x < width; x++) {
	  conv_x = 0;
	  conv_y = 0;


	  //inner product
	  for (yy = y - 1; yy <= y + 1; yy++) {
	   for (xx = x - 1; xx <= x + 1; xx++) {
	    if (yy >= 0 && yy < height && xx >= 0 && xx < width) {
	     conv_x += input.at<uchar>(yy, xx) * mask_x[(yy - (y - 1)) * 3 + (xx - (x - 1))];
	     conv_y += input.at<uchar>(yy, xx) * mask_y[(yy - (y - 1)) * 3 + (xx - (x - 1))];
	    }
	   }
	  }

	  mag[y * width + x] = sqrt(conv_x * conv_x + conv_y * conv_y);
	  dir = atan2(conv_y, conv_x);
	  dir = dir * 180.0 / PI;
	  o_dir[y * width + x] = dir;

	  if (max < mag[y * width + x]) max = mag[y * width + x];
	  if (min > mag[y * width + x]) min = mag[y * width + x];
	 }
	}
	int size = ((width - 16) / 8 + 1) * ((height - 16) / 8 + 1);
	float** dir_mag = (float**)calloc(size, sizeof(float*));
	int idx_x = 0;
```

```
        for (x = 0; x < size; x++) {
         dir_mag[x] = (float*)calloc(quan, sizeof(float));
        }
        for (y = 0; y <= height - bl_size; y += bl_size/2) {
         for (x = 0; x <= width - bl_size; x += bl_size / 2) {
          for (yy = y; yy < y + bl_size; yy++) {
           for (xx = x; xx < x + bl_size; xx++) {
            deg = o_dir[yy * width + xx] / 20.0;
            dir_mag[idx_x][deg] += mag[yy * width + xx];
           }
          }
          idx_x++;
         }
        }

        FILE* fp = fopen("HOG.csv", "w");
        fprintf(fp, "xy,lecture3.bmp₩n");
        float normal_pow;
        for (x = 0; x < size; x++) {
         normal_pow = 0.0;
         for (y = 0; y < quan; y++) {
          normal_pow += (dir_mag[x][y] * dir_mag[x][y]);
         }
         for (y = 0; y < quan; y++) {
          dir_mag[x][y] = dir_mag[x][y] / (sqrt(normal_pow)+0.0000001);
          fprintf(fp, "%d%d,%f₩n", x, y, dir_mag[x][y]);
         }
        }
        fclose(fp);
}

void main()
{
        int height, width;
        Mat imgGray = imread("lecture3.bmp", CV_LOAD_IMAGE_GRAYSCALE);

        gradient_computation(imgGray);
}
```

2) 결과 사진

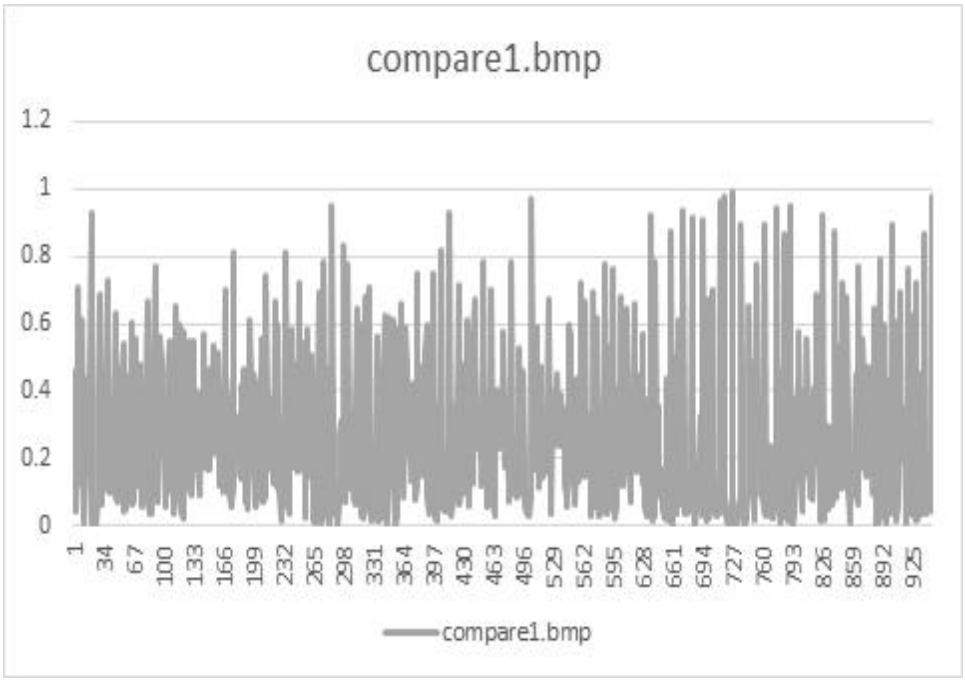
(1) lecture3.bmp

	A	B
1	xy	lecture3.bmp
2	0	0.747342
3	1	0.108644
4	2	0.044095
5	3	0
6	4	0.65293
7	5	0.006625
8	6	0.025231
9	7	0.016276
10	8	0.021666
11	10	0.95138
12	11	0.263181
13	12	0.02895
14	13	0.008034
15	14	0.084688
16	15	0.053243
17	16	0.045981
18	17	0.050962
19	18	0.09995
20	20	0.958343



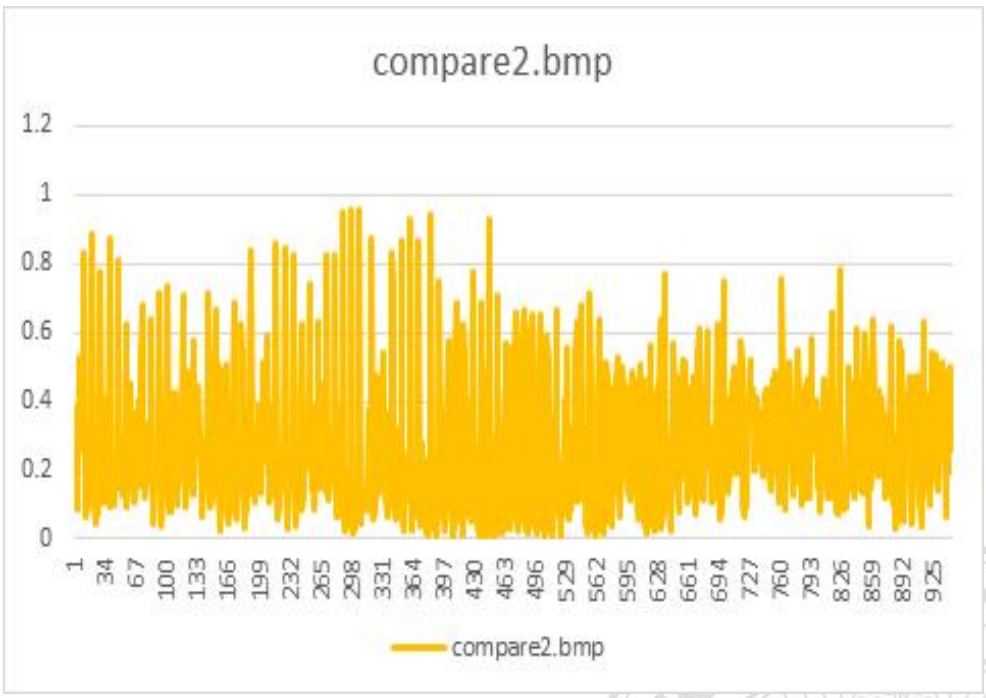
(2) compare1.bmp

	A	C
1	xy	compare1.bmp
2	0	0.45973
3	1	0.109705
4	2	0.043094
5	3	0.214216
6	4	0.709691
7	5	0.150575
8	6	0.124039
9	7	0.300763
10	8	0.310948
11	10	0.608248
12	11	0.125712
13	12	0.009691
14	13	0.164034
15	14	0.438225
16	15	0.152402
17	16	0.156541
18	17	0.359636
19	18	0.467033
20	20	0.929371
21	21	0.089386
22	22	0.010265
23	23	0.020877



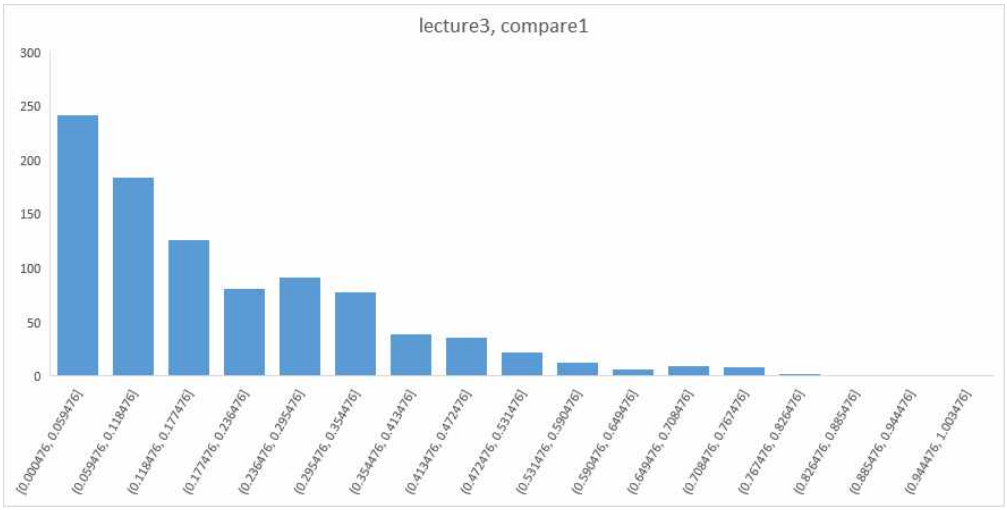
(3) compare3.bmp

	A	D
1	xy	compare2.bmp
2	0	0.386393
3	1	0.0892
4	2	0.144228
5	3	0.179702
6	4	0.460414
7	5	0.53186
8	6	0.373675
9	7	0.256215
10	8	0.299191
11	10	0.831693
12	11	0.278395
13	12	0.066528
14	13	0.092298
15	14	0.172571
16	15	0.234813
17	16	0.15595
18	17	0.258811
19	18	0.204003
20	20	0.884791
21	21	0.249805
22	22	0.099575

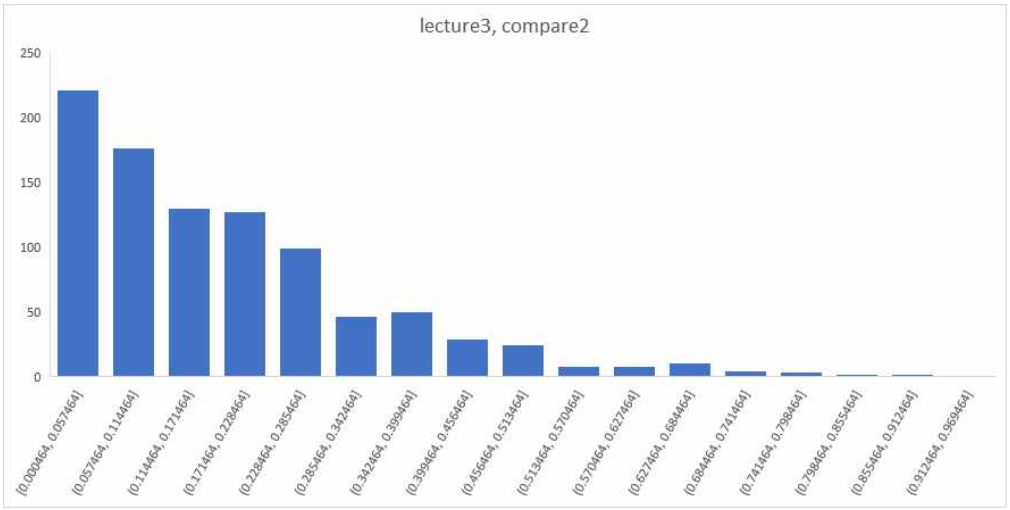


3) 결과 분석

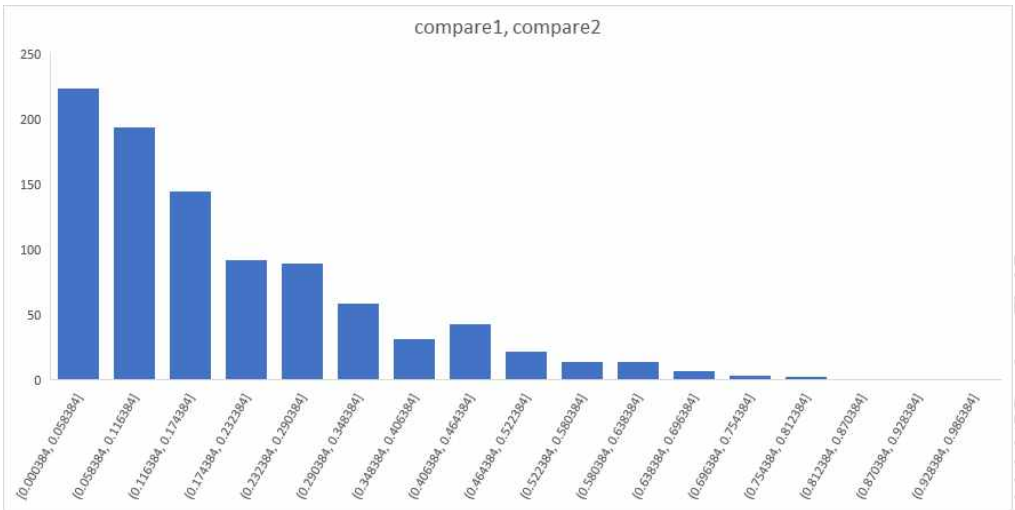
(1) lecture3 – compare2



(2) lecture3, compare1



(3) compare1, compare2



각 세 개의 값과 히스토그램을 비교해보면 히스토그램상에서는 각 사진 3개의 차이가 정확한 수치로 나타나지 않아 비교하기 어렵다. 따라서, 사진3개 각각을 비교해서 각 사진의 차이가 0.2 간격으로 945개의 블록들 중 블록들이 얼마나 들어가 있는지 각각 세어보았다.

먼저, 각 차이가 0.2보다 작은 경우는 lecture3 - compare2가 616개, lecture3 - compare1이 565개, comapre1 - compare2가 576개로 나타났다.

두 번째로, 각 차이가 0.2보다 크고 0.4보다 작은 경우는 lecture3 - compare2가 218개, lecture3 - compare1이 254개, compare1 - compare2가 235개로 나타났다.

세 번째로, 각 차이가 0.4보다 크고 0.6보다 작은 경우는 lecture3 - compare2가 55개, lecture3 - compare1이 70개, compare1 - compare2가 73개로 나타났다.

마지막으로, 각 차이가 0.6보다 큰 경우는 lecture3 - compare2가 29개, lecture3 - compare1이 29개, compare1 - compare2가 34개로 나타났다.

결과적으로, lecture3과 compare2의 경우가 값의 차이가 가장 작은 구간에서 많은 수의 block들이 포진해 있음을 알 수 있다. 값의 차이가 커질수록 그에 해당하는 block의 수가 다른 두 비교에 비해 상대적으로 적다. 따라서, 값의 차이가 가장 적은 lecture3와 compare2가 유사한 그림이라고 볼 수 있다. 실제로 해당그림을 보았을 때, lecture3와 compare2는 각각 사람의 형태를 사진으로 담은 것이기에 비슷한 사진이라고 볼 수 있다. 하지만, 동일한 사람이라고 할 수 없고, 주위 배경과 포즈, 얼굴의 형태 및 크기 등이 다르기 때문에 edge의 유사함을 기준으로 하였기 때문에 다른 두 사진들과 비교해서 완연히 차이가 큰 값을 얻진 않았지만, 각 사진의 gradient에 대한 차이이기 때문에 조금의 다른 값의 차이도 높은 유사도의 기준이 될 수 있다고 생각한다.

또한, lecture3와 compare1의 비교, compare1과 comapre2의 비교는 대체로 그 값이 비슷하게 나타는 것 또한, compare1과 lecture3, compare2는 다른 사진이며, compare1과 lecture3이 비슷한 사진이라는 또 하나의 증거가 될 수 있다고 판단된다.