영상처리와 패턴인식

- 함수기반 소스코드

소스코드

```
#include <stdio.h>
#include <stdlib.h>
#include <Windows.h>
#include <math.h>
void swap(BYTE * a, BYTE * b)
{
    BYTE temp = *a;
    *a = *b;
    *b = temp;
}
```

소스코드

```
void swap(BYTE * a, BYTE * b)
{
    BYTE temp = *a;
    *a = *b;
    *b = temp;
}
```

소스코드

```
void Sorting(BYTE * Arr, int Size) // 버블소팅(오름차순)
{
    for (int i = 0; i < Size - 1; i++){ // Pivot 인덱스
        for (int j = i + 1; j < Size; j++){ // 비교대상 인덱스
            if (Arr[i] > Arr[j]) // 비교대상이 더 작다면
                swap(&Arr[i], &Arr[j]); // 두 값을 교체(swap)
        }
    }
}
```

소스코드

```
void ImageTranslation(BYTE * Image, BYTE * Output, int W, int H)
{
    int Tx, Ty;
    printf("X 이동량: ");
    scanf("%d", &Tx);
    printf("Y 이동량: ");
    scanf("%d", &Ty);
    Ty = -Ty;
    int CTx, CTy;
    for (int i = 0; i < H; i++){
        for (int j = 0; j < W; j++){
            CTy = i + Ty;
            CTx = j + Tx;
            if (CTy >= H) CTy = H - 1;
            if (CTy < 0) CTy = 0;
            if (CTx >= W) CTx = W - 1;
            if (CTx < 0) CTx = 0;
            Output[CTy*W + CTx] = Image[i*W + j];
        }
    }
}
```

소스코드

```
void ImageRotation(BYTE * Image, BYTE * Output, int W, int H)
{
    int Theta;
    printf("회전각을 정수로 입력: ");
    scanf("%d", &Theta);
    double Radian = 3.141592 / 180.0 * Theta;
    int Rx, Ry;
    for (int i = 0; i < H; i++){
        for (int j = 0; j < W; j++){
            Rx = (int)(j*cos(Radian) + i*sin(Radian));
            Ry = (int)(-j*sin(Radian) + i*cos(Radian));
            if (Rx >= W || Rx < 0 || Ry >= H || Ry < 0) Output[i*W + j] = 0;
            else Output[i*W + j] = Image[Ry*W + Rx];
        }
    }
}
```

소스코드

```
void Thresholding(BYTE * Image, BYTE *Output, int ImgSize, int Th)
{
    for (int i = 0; i < ImgSize; i++){
        Image[i] > Th ? Output[i] = 0 : Output[i] = 255;
    }
}
```

소스코드

```
int CntPixel(BYTE * BinImage, int Cx, int Cy, int W){
    int cnt = 0;
    for (int i = Cy - 1; i <= Cy + 1; i++){ //중심 기준 위~아래
        for (int j = Cx - 1; j <= Cx + 1; j++){ // 중심 기준 좌~우
            if (BinImage[i*W + j] == 255) cnt++; // 주변화소가 255면 카운트
        }
    }
    return cnt-1; // 255인 주변화소 개수 return
}
```

소스코드

```
void EdgeDetection(BYTE * BinImage, BYTE * EdgeImage, int W, int H)
{
    // 경계영상 배열 모두 블랙으로 초기화
    for (int i = 0; i < W*H; i++) EdgeImage[i] = 0;
    // 이진영상으로부터 경계검출
    for (int i = 1; i < H-1; i++){
        for (int j = 1; j < W - 1; j++){
            if (BinImage[i*W + j] == 255){
                if (CntPixel(BinImage, j, i, W) != 8) // 경계라고 판단되면...
                    EdgeImage[i*W + j] = 255; // 경계영상배열 해당 화소를 마킹
            }
        }
    }
}
```

소스코드

```
void Color2Gray(BYTE * Color, BYTE * Gray, int W, int H){
    int temp;
    for (int i = 0; i < H; i++){
        for (int j = 0; j < W; j++){
            temp = i*W + j;
            Gray[temp] =
                (BYTE)((Color[temp * 3] +  // B
                Color[temp * 3 + 1] + // G
                Color[temp * 3 + 2]) / 3.0); // R
        }
    }
}
```

소스코드

```
void Gray2Color(BYTE * Gray, BYTE * Color, int W, int H)
{
    int temp;
    for (int i = 0; i < H; i++){
        for (int j = 0; j < W; j++){
            temp = i*W + j;
            Color[temp * 3] =
                Color[temp * 3 +1] =
                Color[temp * 3 + 2] =
                Gray[temp];
        }
    }
}
```

소스코드

```
int push(short *stackx, short *stacky, int arr_size, short vx, short vy, int *top)
{
    if (*top >= arr_size) return(-1);
    (*top)++;
    stackx[*top] = vx;
    stacky[*top] = vy;
    return(1);
}
```

소스코드

```
// 스택 자료삭제 함수
int pop(short *stackx, short *stacky, short *vx, short *vy, int *top)
{
    if (*top == 0) return(-1);
    *vx = stackx[*top];
    *vy = stacky[*top];
    (*top)--;
    return(1);
}
```

소스코드

```
// GlassFire 알고리즘을 이용한 라벨링 함수
void m_BlobColoring(BYTE* CutImage, int height, int width)
{
    int i, j, m, n, top, area, Out_Area, index, BlobArea[1000];
    long k;
    short curColor = 0, r, c;
    // BYTE** CutImage2;
    Out_Area = 1;
```

소스코드

```
// 스택으로 사용할 메모리 할당
short* stackx = new short[height*width];
short* stacky = new short[height*width];
short* coloring = new short[height*width];

int arr_size = height * width;
```

소스코드

```
// 라벨링된 픽셀을 저장하기 위해 메모리 할당
for (k = 0; k<height*width; k++) coloring[k] = 0;  // 메모리 초기화

for (i = 0; i<height; i++)
{
    index = i*width;
    for (j = 0; j<width; j++)
    {
```

소스코드

```
// 이미 방문한 점이거나 픽셀값이 255가 아니라면 처리 안함
if (coloring[index + j] != 0 || CutImage[index + j] != 255) continue;
r = i; c = j; top = 0; area = 1;
curColor++;

while (1)
{
GRASSFIRE:
    for (m = r - 1; m <= r + 1; m++)
    {
        index = m*width;
        for (n = c - 1; n <= c + 1; n++)
        {
```

소스코드

```
                    //관심 픽셀이 영상경계를 벗어나면 처리 안함
                    if (m<0 || m >= height || n<0 || n >= width) continue;

                    if ((int)CutImage[index + n] == 255 && coloring[index + n] == 0)
                    {
                        coloring[index + n] = curColor; // 현재 라벨로 마크
                        if (push(stackx, stacky, arr_size, (short)m, (short)n, &top) == -1)
                            continue;
                        r = m; c = n; area++;
                        goto GRASSFIRE;
                    }
                }
            }
            if (pop(stackx, stacky, &r, &c, &top) == -1) break;
        }
        if (curColor<1000) BlobArea[curColor] = area;
    }
}
```

소스코드

```
// float grayGap = 250.0f / (float)curColor;
// 가장 면적이 넓은 영역을 찾아내기 위함
for (i = 1; i <= curColor; i++)
{
    if (BlobArea[i] >= BlobArea[Out_Area]) Out_Area = i;
}
```

소스코드

```
    // CutImage 배열 클리어~
    for (k = 0; k < width*height; k++) CutImage[k] = 0;
    // coloring에 저장된 라벨링 결과중 (Out_Area에 저장된) 영역이 가장 큰 것만 CutImage에 저장
    for (k = 0; k < width*height; k++)
    {
        if (coloring[k] == Out_Area) CutImage[k] = 255;  // 가장 큰 것만 저장
    }
    delete[] coloring;
    delete[] stackx;
    delete[] stacky;
    // UscFree(CutImage2, 300);
}
```

소스코드

```
// 라벨링 후 가장 넓은 영역에 대해서만 뽑아냄

void DrawCross(BYTE * In, int W, int H, int Cx, int Cy, int R, int G, int B)
{
    for (int i = 0; i < W; i++){
        In[(Cy * W + i) * 3] = B;
        In[(Cy * W + i) * 3 + 1] = G;
        In[(Cy * W + i) * 3 + 2] = R;
    }
    for (int i = 0; i < H; i++){
        In[(i * W + Cx) * 3] = B;
        In[(i * W + Cx) * 3 + 1] = G;
        In[(i * W + Cx) * 3 + 2] = R;
    }
}
```

소스코드

```
void main()
{
    BITMAPFILEHEADER hf;
    BITMAPINFOHEADER hInfo;
    RGBQUAD hRGB[256];
    FILE * fp;
    fp = fopen("eyeimage.bmp", "rb");
    if (fp == NULL)
    {
        printf("File not found!₩n");
        return;
    }
```

소스코드

```
fread(&hf, sizeof(BITMAPFILEHEADER), 1, fp);
fread(&hInfo, sizeof(BITMAPINFOHEADER), 1, fp);
if (hInfo.biBitCount == 8) // Index(Gray) Image
    fread(hRGB, sizeof(RGBQUAD), 256, fp);
int W, H, ImgSize;
W = hInfo.biWidth;
H = hInfo.biHeight;
ImgSize = W*H;
BYTE * Image;
BYTE * Output;
```

소스코드

```
if (hInfo.biBitCount == 8){
    Image = (BYTE *)malloc(ImgSize);
    Output = (BYTE *)malloc(ImgSize);
    fread(Image, sizeof(BYTE), ImgSize, fp);
}
else if (hInfo.biBitCount == 24){
    Image = (BYTE *)malloc(ImgSize * 3);
    Output = (BYTE *)malloc(ImgSize * 3);
    fread(Image, sizeof(BYTE), ImgSize * 3, fp);
}
fclose(fp);
```

소스코드

```
BYTE * Gray = (BYTE *)malloc(ImgSize);
BYTE * Label = (BYTE *)malloc(ImgSize);
// 영상처리
/*Thresholding(Image, Image, ImgSize, 100);
EdgeDetection(Image, Output, W, H);*/
Color2Gray(Image, Gray, W, H);
Thresholding(Gray, Gray, ImgSize, 80);
m_BlobColoring(Gray, H, W);
```

소스코드

```
int SumX = 0, SumY = 0, Cnt = 0;
for (int i = 0; i < H; i++){
    for (int j = 0; j < W; j++){
        if (Gray[i*W + j] == 255){ // 동공영역이라면..
            SumX += j;
            SumY += i;
            Cnt++;
        }
    }
}
```

소스코드

```
int Cx = SumX / Cnt;
int Cy = SumY / Cnt;
for (int i = 0; i < ImgSize * 3; i++) Output[i] = Image[i];
DrawCross(Output, W, H, Cx, Cy, 255, 0, 255);
```

소스코드

```
    //Gray2Color(Gray, Output, W, H);

    // 결과를 파일로 출력
    fp = fopen("output.bmp", "wb");
    fwrite(&hf, sizeof(BYTE), sizeof(BITMAPFILEHEADER), fp);
    fwrite(&hInfo, sizeof(BYTE), sizeof(BITMAPINFOHEADER), fp);
```

소스코드

```
    if (hInfo.biBitCount == 8)
        fwrite(hRGB, sizeof(RGBQUAD), 256, fp);
    if (hInfo.biBitCount == 8)
        fwrite(Output, sizeof(BYTE), ImgSize, fp);
    else if (hInfo.biBitCount == 24)
        fwrite(Output, sizeof(BYTE), ImgSize*3, fp);
    fclose(fp);
    free(Image);
    free(Output);
    free(Gray);
    free(Label);
}
```