ADVENTURE DESIGN

FINAL REPORT.

김 다 은
컴퓨터 공학과 20200123

1. 문제

1.1 문제 소개

- PPM

PPM(raw portable pixel map) 은 매우 간단한 헤더를 갖는 구조로 컬러를 저장한다. 처음 2바이트는 PPM파일의 식별자 (magic number) "P6"의 값을 갖는다. 그리고 순서대로 영상의 가로 및 세로 길이 최대 밝기 값을 나타낸다.

- 문제

PPM 이미지 파일을 읽고 색상 반전, 상하/좌우 반전, 영상 회전, 채널 추출, 디더링을 포함한 그레이 스케일y scale 변환, 하프톤 이진 영상, crop, 밝기 조절, 대비 조절, 워터마크 추가를 사용자가 선택하여 적용한 후 저장한다. 선택한 작용을 미리보기를 통해 확인할 수 있고 원본 이미지를 확인할 수 있으며 이를 GUI 로 제공한다.

1.2 주의 사항 및 고려할 점

PPM 이미지를 회전한 뒤 결과물에 구멍이 발생하지 않도록 한다. 이미지를 변형 시킨 것을 미리 보여주고 사용자가 마음에 들지 않을 경우 설정을 초기화 시킬 수 있어야 한다. ppm 파일이 아닌 경우에 오류 메시지를 띄우고 이미지 입력을 받지 않았을 때에는 작업을 클릭해도 실행이 되지 않도록 한다.

2. 기능 및 요구 사항 분석

2.1 입출력 정의

- 입력 정보

ppm 파일 입력

- 출력 정보

영상 처리 작업을 완료한 ppm 파일

2.2 핵심 기능과 그들 간의 관계

- 이미지 로드

이미지를 불러들여 PPMimage 로 저장한다.

- 이미지 저장

SAVE 버튼을 누르면 현재까지 작업한 이미지를 지정한 경로로 저장한다.

- 디더링을 포함한 그레이스케일 변환

이미지에 노이즈를 포함한 디더링을 적용한 이미지 변환

- 하프톤 이진 영상 변환

원으로 이루어진 그레이 스케일 이미지로 변환

- 채널 추출
라디오 버튼을 통해 R G B 중 어떤 채널을 추출할지 결정할 수 있다.
채널을 추출하면 선택한 채널로 다른 채널 값도 다 통일되게 된다.

- 밝기 조절

- 대비 조절
각각 R G B 마다 값이 큰 경우 그 값을 더 크게, 작은 경우 그 값을 더 작게 만들어 대비효과를 준다.

- 자르기
좌표를 입력하면 해당 좌표만큼의 이미지를 잘라낸다.

- 워터마크 추가
시작 좌표를 입력하고 불투명도를 입력하면 워터마크를 추가할 수 있다.

- 회전
회전 각도를 입력하면 이미지를 회전시킬 수 있다.

- 상하 / 좌우 반전

- 색상 반전

- 이미지 로드와 이미지 저장
이미지 로드가 정상적으로 이루어졌을 때 이미지 저장이 가능하도록 한다.

- 워터마크와 이미지 로드
워터마크를 로드할 때 이미지 로드를 실행하여 준다.

2.3 요구 사항
- 사용자가 PPM 파일을 선택한다. 파일을 선택하면 파일의 해상도와 파일의 이름을 보여준다.

- PPM 파일이 아니라면 오류 메시지를 출력하고 작업을 눌렀을 경우 아무일도 발생하지 않도록 한다.

- 프로그램 종료 전까지 계속해서 과정을 반복할 수 있도록 한다.

- 작업도중 이미지를 초기화 시킬 수 있도록 한다.

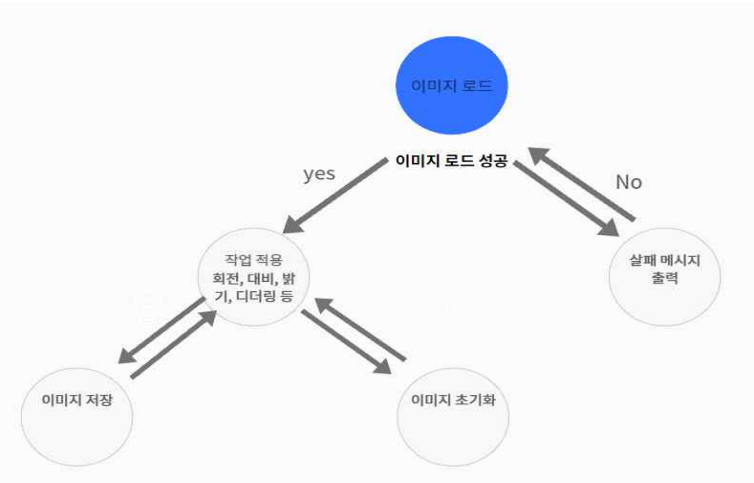
- 새로운 이미지를 불러들일 수 있도록 한다.

3. 설 계

3-1 자료 구조 설계

- 영상 처리 프로그램은 PPM 클래스와 작업 클래스로 구성되어있다. PPM 클래스는 생성자가 실행될 때 가로 세로 각각 pixel의 R,G,B 값을 담고 있는 구조체들을 갖고있는 2차원 배열과 Magic number 문자열 변수, 최대밝기, 가로, 세로 값을 담고 있는 변수가 생성된다.
- 작업 클래스에서는 PPM 이미지 객체를 생성자에서 받고 각각의 작업을 처리해준다.
- 작업 클래스에서는 초기화를 위한 원본 이미지 img 와 작업을 적용시켜주기 위한 prev_img img_new 가 존재한다. img_new 는 목적 이미지 prev_img 는 입력 이미지로 작동한다.

3-2 알고리즘 설계 (pseudo code 또는 flowchart)



4. 구 현

4.1 자료 구조 및 클래스 코드

- Pixel struct

```
typedef struct Pixel {
    unsigned char R;
    unsigned char G;
    unsigned char B;
}Pixel;
```

- Pixel 구조체 배열을 가지는 PPMimage 구조체

```
typedef struct PPMimage {
    char M, N;
    int width;
    int height;
    int max;
    struct Pixel** pixels;
}PPMimage;
```

- Process 클래스

```
class Process {
protected:
    PPMimage img_new;
    PPMimage img;
    PPMimage waterMark;
    PPMimage prev;
    bool isPPM;
    void createNewImg(int width = NULL, int height = NULL);
public:
    Process(char* FileName = NULL);
    ~Process();
    void SavePPM(char* filename = NULL);
    int GetWidth();
    int GetHeight();
    void updown_reverse();
    void leftright_reverse();
    void negative();
    bool isPPMf();
    void halfton();
    void extract_R(); //R_color

    void extract_G(); //G_color

    void extract_B(); //B_color

    void brightnessControl(int brightness); //brightness_submit

    double rotateDegree(double degree); // for uclid

    void ResetBackground(); //for rotate
```

```
    void RGBcorrectN(int &R, int &G, int &B);

    void UclidRotate(double seta); //for rotateControl

    void rotateControl(double degree);  //degree_submit

    void cropControl(int y1, int y2,int x1, int x2);

    void contrast(int cont);

    void waterMarkInput(int x, int y, int opacity);
    void waterMarkload(char* fileName);

    void RandomDithering();
    void grayscale();
    void SavePrevPPM(char* filename);
    bool getProcessIsPPM();
    void clear();
};
```

4.2 핵심 알고리즘의 코드 부분

- 파일읽기

파일 읽기는 Process 라는 클래스를 생성할 때 시작되며 파일 경로를 입력하면 이미지를 생성한다.

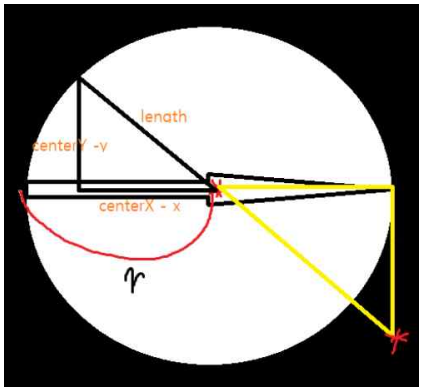
```
fscanf(fp, "%c%c\n", &img.M, &img.N);
fscanf(fp, "%d %d\n", &img.width, &img.height);
fscanf(fp, "%d\n", &img.max);
img.pixels = (Pixel**)calloc(img.height, sizeof(Pixel*));
for (int i = 0; i < img.height; i++) {
    img.pixels[i] = (Pixel*)calloc(img.width, sizeof(Pixel));
}
for (int i = 0; i < img.height; i++) {
    for (int j = 0; j < img.width; j++) {
        fread(&img.pixels[i][j], sizeof(char), 3, fp);
    }
}
```

- 색상 반전

이미지가 가질 수 있는 최대 밝기에서 픽셀마다 RGB 값을 각각 빼준다.

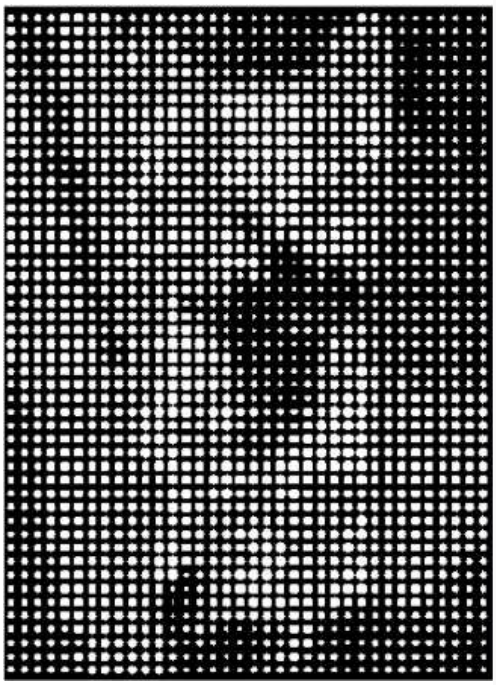
```
for (int i = 0; i < prev.height; i++) {
        for (int j = 0; j < prev.width; j++) {
            img_new.pixels[i][j].R = prev.max - prev.pixels[i][j].R;
            img_new.pixels[i][j].G = prev.max - prev.pixels[i][j].G;
            img_new.pixels[i][j].B = prev.max - prev.pixels[i][j].B;
        }
    }
```

- Half tone




```
            for(int x=centerX-radius; x < centerX+radius; x++)
                for(int y=centerY-radius; y < centerY+radius; y++)
                {
                    length = sqrt(pow((double)(centerY - y), 2) + pow((double)
(centerX - x), 2));
                    if (length > tradius) img_new.pixels[y][x] = {0, 0, 0};
                    else img_new.pixels[y][x] = {(unsigned char)prev.max,
(unsigned char)prev.max,(unsigned char)prev.max};
                }
```

tradius 는 대비를 반지름의 길이로 압축시킨 것이다. 현재 픽셀의 대비를 길이로 압축시킨 tradius 보다 현재 픽셀이 위치한 곳에서 중심까지 거리가 더 긴 경우는 검은색 짧은 경우는 흰색으로 채워준다.



```
for (int row=startRow; row < endRow; row+=diameter)
    {
        for(int col=startCol; col < endCol; col+=diameter)
        {
            int mean = 0;
            for(int x=col; x < col+diameter; x++)
                for(int y = row; y < row+diameter; y++)
                    mean += (prev.pixels[y][x].R + prev.pixels[y][x].G +
prev.pixels[y][x].B) / 3;
            mean = mean / (diameter*diameter);

            tradius = mean / (prev.max / radius);

            centerX = col + radius;
            centerY = row + radius;
            for(int x=centerX-radius; x < centerX+radius; x++)
                for(int y=centerY-radius; y < centerY+radius; y++)
                {
                    length = sqrt(pow((double)(centerY - y), 2) + pow((double)
(centerX - x), 2));
                    if (length > tradius) img_new.pixels[y][x] = {0, 0, 0};
                    else img_new.pixels[y][x] = {(unsigned char)prev.max,
(unsigned char)prev.max,(unsigned char)prev.max};
                }
        }
    }
```

각 픽셀마다 RGB의 평균을 다 더해준다. 전체 사각형의 평균 밝기를 구하기 위해 mean 에 사각형의 넓이를 나누어 준다.

그레이 스케일의 밝기 범위를 반지름으로 압축시키기 위해 평균을 radius 로 나누고 그것을 255에 곱한다. 그럼 대비에 따른 길이 tradius 가 정해지게 된다.

- 회전


```
for (int i = 0; i < prev.height; i++) {
        for (int j = 0; j < prev.width; j++) {
            new_x = round((i - center_y) * sin(seta) + (j - center_x) *
cos(seta) + center_x);
            new_y = round((i - center_y) * cos(seta) - (j - center_x) *
sin(seta) + center_y);

            if (new_x < 0)      continue;
            if (new_x >= prev.width)   continue;
            if (new_y < 0)      continue;
            if (new_y >= prev.height)   continue;

            img_new.pixels[i][j] = prev.pixels[new_y][new_x];
        }
    }
```

img 의 크기에 벗어나는 경우는 따로 픽셀을 대입하지 않도록 하여 검게 나오거나 출력되지 않도록 한다.

- 워터마크

PPMimage waterMark를 통해 워터 마크 이미지를 생성하여 img_new 에 저장한다.

투명한 부분을 표현하기 위하여 검은색이 아닌 경우에 img_new 에 넣는다

투명도를 표현하기 위해 1의 비율에 맞게 op를 RGB 값에 곱하여 적용시켜 준다.

만약 현재 이미지 보다 워터 마크의 위치가 벗어나게 될 경우 break를 통해 중지시켜 준다.

```
for (int j = x + 1; j <= x + waterMark.width; j++) {
            if (j > prev.width) {
                break;
            }
            if (waterMark.pixels[y1][x1].R != 0){
                r = (unsigned char)((double)img_new.pixels[i-1][j-1].R*(1-op)
+ (double)waterMark.pixels[y1][x1].R*op);
                g = (unsigned char)((double)img_new.pixels[i-1][j-1].G*(1-op)
+ (double)waterMark.pixels[y1][x1].G*op);
                b = (unsigned char)((double)img_new.pixels[i-1][j-1].B*(1-op)
+ (double)waterMark.pixels[y1][x1].B*op);
                img_new.pixels[i-1][j-1] = {r, g, b};
            }
            x1++;
        }
        y1++;
    }
```

- 대비

각 RGB 마다 넓이에 대해 평균을 구한 뒤 만약 평균 보다 값이 클 경우 입력받은 만큼 cont를 높여주고 평균보다 작을 경우는 cont 값을 낮춰주어 RGB 값을 더 극적으로 나타낸

다. 이렇게 값을 연산하다보면 RGB 값이 최대 밝기를 넘어 서서 이상한 값이 출력될 수 있다. 따라서 RGBcorrectN 함수를 추가적으로 구현하여 RGB 값을 현재 이미지의 최대밝기로 맞출 수 있게 한다.

```
for (int i = y + 1; i <= y + waterMark.height; i++) {
        x1 = 0;
        if (i > prev.height) {
            break;
        }
        for (int j = x + 1; j <= x + waterMark.width; j++) {
            if (j > prev.width) {
                break;
            }
            if (waterMark.pixels[y1][x1].R != 0){
                r = (unsigned char)((double)img_new.pixels[i-1][j-1].R*(1-op)
+ (double)waterMark.pixels[y1][x1].R*op);
                g = (unsigned char)((double)img_new.pixels[i-1][j-1].G*(1-op)
+ (double)waterMark.pixels[y1][x1].G*op);
                b = (unsigned char)((double)img_new.pixels[i-1][j-1].B*(1-op)
+ (double)waterMark.pixels[y1][x1].B*op);
                img_new.pixels[i-1][j-1] = {r, g, b};
            }
            x1++;
        }
        y1++;
    }
```

```
mean_R = mean_R / (prev.height * prev.width);
mean_G = mean_G / (prev.height * prev.width);
mean_B = mean_B / (prev.height * prev.width);

for (int i = 0; i < prev.height; i++) {
    for (int j = 0; j < prev.width; j++) {
        if (prev.pixels[i][j].R > mean_R) {
            R = prev.pixels[i][j].R + cont;   // 픽셀값 색 반전
            RGBcorrectN(R, G, B);
        }else{
            R = prev.pixels[i][j].R - cont;
            RGBcorrectN(R, G, B);
        }
        if (prev.pixels[i][j].G > mean_G) {
            G = prev.pixels[i][j].G + cont;   // 픽셀값 색 반전
            RGBcorrectN(R, G, B);
        }else{
            G = prev.pixels[i][j].G - cont;
            RGBcorrectN(R, G, B);
        }
        if (prev.pixels[i][j].B > mean_B) {
            B = prev.pixels[i][j].B + cont;   // 픽셀값 색 반전
            RGBcorrectN(R, G, B);
        }else{
            B = prev.pixels[i][j].B - cont;
            RGBcorrectN(R, G, B);
        }
        img_new.pixels[i][j].R = R;
        img_new.pixels[i][j].G = G;
        img_new.pixels[i][j].B = B;
        }
}
```

- 잘라내기

사용자가 입력값을 잘못 줬을 경우를 대비하여 y1(x1) 가 더 클 경우 swap을 해준다.

이미지의 크기를 벗어나지 않도록 원본이미지 보다 클 경우 원본 이미지의 좌표를 최대로 하여 새롭게 new_img를 동적할당 해준다.

```
int new_height = y2 - y1 + 1;
    int new_width = x2 - x1 + 1;
    int y = 0;
    int x = 0;


    createNewImg(new_height, new_width);
    for (int i = y1; i <= y2; i++) {
        x = 0;
        for (int j = x1; j <= x2; j++) {
            img_new.pixels[y][x] = prev.pixels[i][j];
            x++;
        }
        y++;
    }
```

- 상하 좌우 반전

이미지가 상하로 뒤집힐 경우 새로운 이미지에 height 부분을 역으로 넣어 간다.

```
for (int i = 0; i < prev.height; i++) {
        for (int j = 0; j < prev.width; j++) {
            img_new.pixels[i][j] = prev.pixels[prev.height - i-1][j];
        }
    }
```

이미지가 좌우로 뒤집힌 경우 새로운 이미지에 width 부분을 역으로 넣어 간다.

```
for (int i = 0; i < prev.height; i++) {
        for (int j = 0; j < prev.width; j++) {
            img_new.pixels[i][j] = prev.pixels[i][prev.width -j -1];
        }
    }
```

- 채널 추출

라디오 버튼을 통해 추출하고 싶은 값을 추출할 수 있도록 하였다.

R을 예시로 하면 픽셀의 다른 RGB 값을 모두 R로 두어 그레이 스케일이 되도록 한다.

```
for (int i = 0; i < prev.height; i++) {
        for (int j = 0; j < prev.width; j++) {
            img_new.pixels[i][j] = {prev.pixels[i][j].R,prev.pixels[i]
[j].R,prev.pixels[i][j].R};    // 픽셀값 색 반전
        }
    }
```

- 밝기 조정

각 픽셀마다 brightness 값을 더해 밝기 값을 더 높여 준다.

값의 초과나 음수를 방지하기 위해 최대 밝기를 넘어설 경우 최소 최대 밝기를 대입해준다.

```
for (int i = 0; i < prev.height; i++) {
       for (int j = 0; j < prev.width; j++) {
           R = prev.pixels[i][j].R + brightness;    // 픽셀값 색 반전
           G = prev.pixels[i][j].G + brightness;
           B = prev.pixels[i][j].B + brightness;
           RGBcorrectN(R,G,B);
           img_new.pixels[i][j].R = R;
           img_new.pixels[i][j].G = G;
           img_new.pixels[i][j].B = B;
           }
    }
```

- 디더링

각 픽셀마다 평균을 계산하고 발생시킨 난수 값 보다 평균값이 클 경우 RGB를 최대 밝기로 두고 평균값이 작을 경우 RGB를 0으로 둔다.

```
for(int i = 0; i < prev.height; i++){
        for(int j = 0; j < prev.width; j++){
            n = rand() % prev.max;
            mean = (prev.pixels[i][j].R + prev.pixels[i][j].G + prev.pixels[i]
[j].B) / 3;
            if(mean < n){
                img_new.pixels[i][j] = {0,0,0};
            }else{
                img_new.pixels[i][j] = {(unsigned char)prev.max,(unsigned
char)prev.max,(unsigned char)prev.max};
            }
        }
```

- 그레이 스케일

각 픽셀마다 평균을 계산하고 RGB 값을 평균으로 둔다.

```
for(int i = 0; i <prev.height; i++){
        for(int j = 0; j < prev.width; j++){
            mean = (prev.pixels[i][j].R + prev.pixels[i][j].G + prev.pixels[i]
[j].B) / 3;
            img_new.pixels[i][j] = {mean, mean, mean};
        }
    }
```

5. 결과 분석 및 검토

5.1 시도할 의미가 있는 기술 요소

- 영상 회전

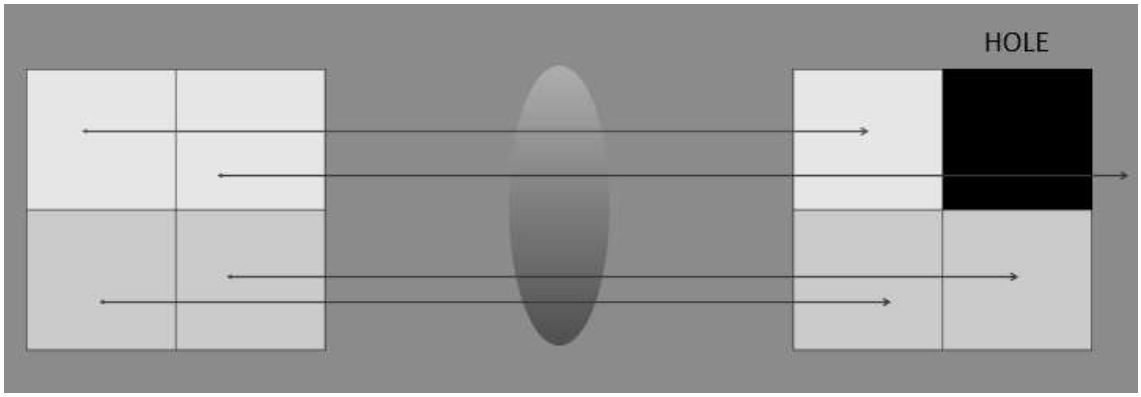
포워드 매핑 : 원래 이미지 좌표 (x , y) 를 특정 각도만큼 회전 시켜 (x′, y′), 그 위치에 대입해야 겠다.

백워드 매핑 : 내가 만들 이미지는 원래 이미지의 특정 각도만큼 회전 후 위치할 (x′, y′)에서 값을 원래 이미지로부터 가져올때 -(특정 각도) 로 변환된 (x, y)에서 가져오자.

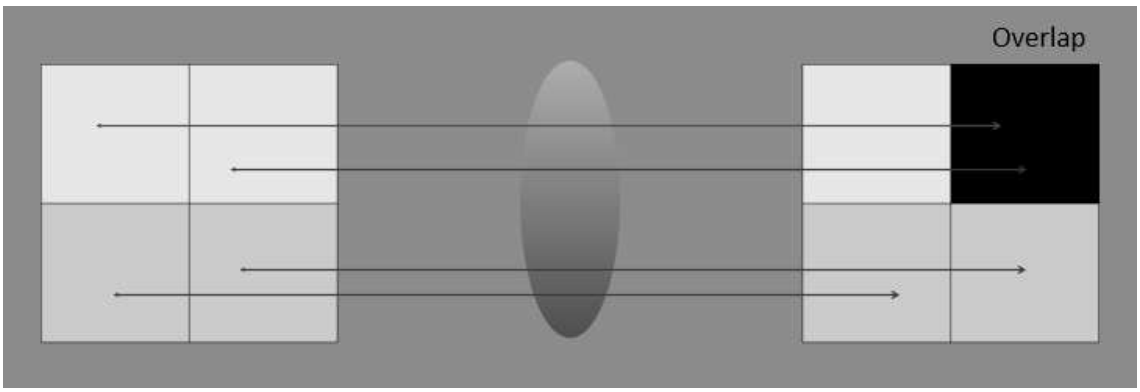
정리하자면 백워드 매핑은 출력 이미지를 대상으로 계산하여 입력 영상으로부터 역으로 회전되어 해당될 이미지를 배껴오는 것이다.

- 포워드 매핑의 문제점

1) Hole 발생



2) Overlap 발생

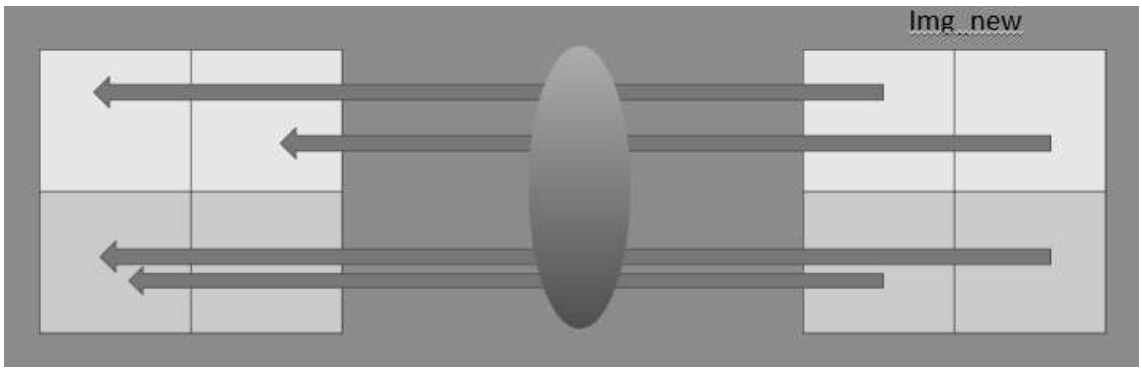


- 백워드 매핑

원본 영상의 한 개의 픽셀이 목적영상에 두 개의 픽셀에 위치하게 된다.

이 경우 홀이나 오버랩은 발생하지 않지만

입력 영상의 한 픽셀이 img_new 에 여러번 사용되게 되면 영상의 품질이 떨어질 수 있다.



- 처음 문제 분석과 달랐던 부분

처음 걱정과 달리 각 픽셀에 대한 값을 배열에 저장하여 영상 처리를 할 때 해상도가 큰 경우에도 c++을 사용하여 시간이 오래 걸리지 않았다.

- 크기 변경

현재 구현한 영상 회전은 회전을 해서 원본 이미지를 넘기는 부분은 자르는 형식이었다.

하지만 크기 변경을 적용하면 이미지 손실 없이 영상을 회전시킬 수 있다.

- 비율을 이용해서 수행하는 경우

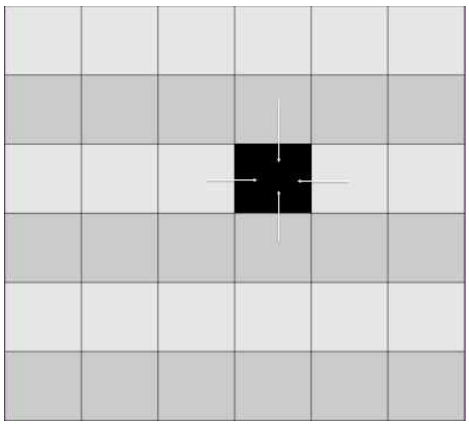
가로와 세로로 변경하고자 하는 비율을 지정하여 원본 이미지의 좌표에 곱하면 img_new 의 좌표를 계산할 수 있다.

- img_new 의 크기를 지정해서 변경하는 방법

입력된 영상과 img_new 의 크기로 비율을 계산하고 계산된 비율을 이용하여 img_new 의 좌표를 계산한다.

- 보간법

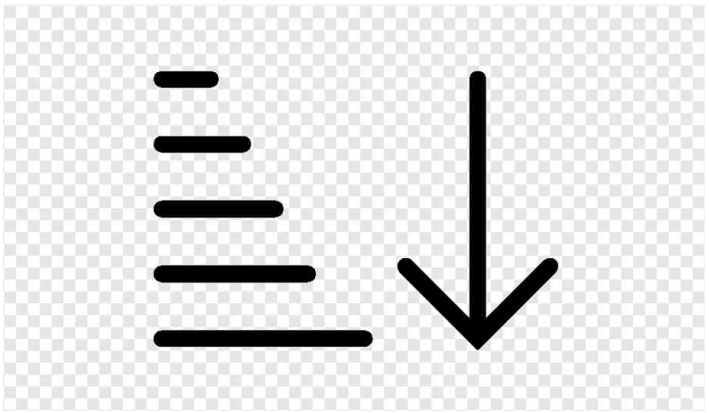
포워드 매핑을 사용하거나 이미지를 확대 축소하는 경우 보간법을 적용하여 hole을 없애주는 방법이 있다.



hole 로 의심되는 주변의 픽셀은 검은색이 아닌데 홀로 검은색인 경우 주변의 픽셀의 값들의 평균을 대입하여 영상의 hole을 없애준다.

- Water Mark

PNG 나 GIF 와 같이 투명한 부분을 지원해주는 형식과 유사하게 ppm 영상 처리를 구현하려면 픽셀의 rgb 값 외에도 이 부분이 투명한지 아닌지를 판단해 주는 변수가 더 필요할 것이다.



위의 이미지와 같이 이미지가 투명한 부분이라면 픽셀들을 이러한 패턴으로 표현하도록 한다. 하지만 이러한 구현은 ppm 파일 자체를 변형시켜야 하기 때문에 어려움이 따른다.

이를 해결하기 위해 프로그램에서 제공하는 water mark 이미지를 선택하여 미리 프로그램에서 픽셀마다 투명도가 필요한 픽셀의 투명 값을 true 아닌 값을 false 로 정해둔다. 이렇게 한다면 투명함을 구현할 수 있을 것이다.

-Half tone

하프톤은 점을 사용하여 크기나 간격에 따라 밝기를 나타낼 수 있다.

공간적 통합 작용 (Spatial Integration)을 이용한 것으로 멀리 떨어져서 보면 검은색과 흰

색 공간이 혼합되어 회색으로 표현 된다.

하프톤은 주로 인쇄와 같은 작업을 할 때 사용하며 이진 영상을 더 화려하게 표현할 수 있다.

-dithering

패턴 디더링

이미지를 n * n 형태로 분할하여 명암 값을 얻어 패턴으로 대체한다.

패턴의 전체 평균 밝기를 계산하여 조건문을 통하여 패턴을 적용시킬 수 있을 것이다.

이런 경우 없던 직선이 생길 수 있다.

오차 확산법

디더링시 발생하는 오류를 줄이는 방법이다.

발생한 오차 값은 이웃 픽셀 값에 분산시키는 방법이다.

- 대비

대비도 디더링과 마찬가지로 평균값에 대해 조건문을 적용하여 그 값보다 작은 경우 더 어둡게 클 경우 더 밝게 만들었다.

그 경계 값에 위치한 경우 오차값이 크게 발생할 수 있다 이런 경우 오차 확산법을 적용시킬 수 있을 것이다.

6. 개발 일지 및 후기

6.1 일지

일자	내용
9월 11일	PPM 영상 처리 자료 조사 및 c++ qt 조사
9월 13일	PPM 영상 처리 프로그램 설계 및 회전 조사
9월 20일	PPM 영상처리 프로그램 구현 (영상 회전, 반전, 상하 좌우 등)
9월 24일	GUI 설계 보충 및 디더링 하프톤 구현
9월 30일	watermark 구현 및 전체 구조에 대한 조사

6.2 후기

- 개발 과정 및 발표 수업에서 느낀 점

개발 과정에서 c++ qt를 써본 경험이 없어 공부하는데 시간이 걸렸던 것 같다. 미리 qt를 공부했었더라면 model view controller를 분리시켜야 한다는 것을 생각했을 텐데 뒤늦게 이에 대해 생각하게 되어 아쉬움이 크게 남았다. 설계와 최종을 포함한 발표 수업에서 똑같은 기능이라도 팀원들 마다 구현한 방법이 다양해서 재미가 있었다. 똑같은 회전이라도 크기에 맞게 이미지를 축소 시킨 사람도 있었고 역방향 매핑을 사용하여 시간 복잡도를 줄인 사람. 포워드 매핑을 사용하고 보간법을 적용한 사람 등 다양했다. 하나의 작업에 대해 다양한 사람들의 방법 및 의견을 들을 수 있어 좋은 기회였다.

만약 디자인 패턴과 같은 과목을 미리 수강하였다면 전체적이 구조에 대해서 이야기를 많이 나누어 보고 더 좋은 프로그램을 만들 수 있었을 것 같은데 아쉬움이 크다.

많은 팀원들이 이미지를 적용하고 다시 이미지를 적용하고자 했을 때 원본 이미지를 파일에서 다시 로드 하는 방법을 선택하였다.

뒤로가기를 구현한 사람은 아무도 없었다. 이 부분을 구현한 사람이 있었다면 전체적인 구조를 만들어나가는데 더 공부가 되었을 것 같다.

구현은 하지 못했지만 내 생각은 전체 프로세스에 대해 큐를 따로 만들어 작업 별로 우선순위를 정해주는 것이다. 어떠한 작업을 취소할 경우 프로세스 큐에서 해당 작업을 빼내고 다시 view를 업데이트 시켜준다. 그럼 우선순위에 맞게 이미지를 작업 시키고 삭제시킨 프로세스는 적용이 되지 않아 원하는 미리보기가 나올 것이다.

또 더 간단한 방법으로는 클래스에 전체 영상 처리 목록에 대한 변수를 정해놓는 것이다. 대비를 위한 변수 밝기를 위한 변수 등을 따로 만들어 값이 업데이트 될 때마다 미리 보기를 통해 이미지를 보여주는 것이다.

이렇게 이야기 해볼 것이 굉장히 많았는데 토의나 회의가 익숙하지 않았던 탓인지 전체적인 구조에 대해 토의를 하지 못한 것이 살짝 아쉽게 느껴진다.

- 자신에 대한 평가

중복되는 코드들이나 전체적인 구조 설계에 대한 부분이 부족하다고 느껴졌다. c++ 이라는 언어에 대한 이해도나 qt를 사용하는 것이 미숙한 탓도 있었던 것 같다. 또한 예외 처리에 대한 부분이 미흡했던 것 같다. 사용자 측에서 잘못된 입력을 할 수 없도록 입력을 제한하는 방식이나 잘못된 입력이 들어와도 어떻게 처리할것인가에 대한 것을 조금더 생각해볼 필요가 있을 것 같다. 다음에는 mvc 패턴을 공부하여 제대로 적용시켜야 겠다는 생각이 들었다. view controller model 이 정확히 분리되어있지 않으니 view에서 model 이 해야할 일을 하고 있는 경우도 있었다. 또한 클래스 내부에서 저렇게 이미지를 만들어 내는 것이 좋은 방법이 아닐 수 있다는 생각이 들었다. prev img img new img 들을 process class 에다가 만드는 방법은 잘못된 것 같다. 처음에는 간단하게 생각하여 대충 설계하였던 것이 나중에는 큰 문제로 다가와서 미리 충분한 조사와 공부가 필요하다는 것을 느꼈고 설계가 얼마나 중요한 것인지 다시한번 알게 되었다.

- 앞으로에 대한 계획

디자인 패턴에 대해서 조금 공부해 보고 mvc 패턴으 적용하여 ui를 구성할 것이다.

하나의 문제에 대해서 여러 가지 해결할 방법이 있단 것을 알고 각 방법을 비교해 보며 현재 구현하고자 하는 프로그램에는 어떠한 방법이 적합할지 생각해 보아야 겠다.

또한 문서화나 구현한 내용을 발표로 표현하는데 익숙하지 않다는 생각이 들었다.

프로그램을 열심히 만들었더라도 사람들에게 내 프로그램에 대해 잘 설명하고 사용하게끔 하는것도 아주 중요한 역량 중 하나인 것 같다. 문서화와 발표를 여러번 도전해 보고 다양한 피드백을 받아 수정해 나가며 역량을 키워나가야 겠다.

7. 참고문헌

[1] 김성영, “멀티 미디어 개론 및 실습”, 카오스 북.

[2] C++, 황준하 교수님 피피티.

[3] ppm 영상 처리, https://oneday0012.tistory.com/19.