

영상처리 n주차 과제

학번: 201404376

이름: 정성욱

1. 과제 내용

목표 Image rotation으로 회전 시킨 후에 bilinear과 nearest의 방법으로 회전시킨 이미지를 처리한다.

```
re_img = zeros(row, col);

x_center = round(row/2);
y_center = round(col/2);
```

회전 조정을 위해서 x_center, y_center의 변수를 두어 회전시켜놓은 이미지를 붙여넣을 가운데 인덱스를 정의 합니다.

Nearest

```
if strcmp(interpolation, 'nearest')
|   for i = 1:row
|       for j = 1:col
|           v = f * double([(i-x_center);(j-y_center)]);

|           v(1) = v(1)+round(x/2);
|           v(2) = v(2)+round(y/2);
|           if v(1) < 1 || v(1) > x || v(2) < 1 || v(2) > y
|               continue;
|           end
|           re_img(i, j) = img(int64(v(1)), int64(v(2)));
|       end
|   end
end
```

nearest입니다. 루프는 기울어진 이미지를 채워야하기 때문에, 패딩된 이미지의 가로세로인 row와 col을 줍니다. 그리고 그대로 $f(i,j)$ 하게 되면 이미지가



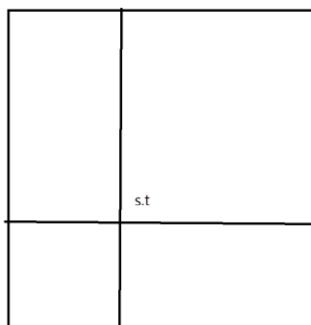
이런식으로 밖으로 튀어 나가게 되는데, 여기서 밖으로 튀어나간 회전된 이미지를 다시

넣어줘야 합니다. 그에 대한 작업입니다. Re_이미지의 가운데 좌표들에 대해 현재 좌표를 빼준 후에 $f = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$ 를 곱해줍니다. 여기서 f 는 일반적인 로테이션공식과 다른데 그 이유는 f 의 역함수를 이용하여 연산을 진행했을때의 최종인덱스가 있어야할 위치가 이미지의 위치가 원래이미지가 되어야하는 것입니다. 그렇기 위한 조정으로 우선 기존의 x, y 에 대해 re_img 의 중앙 인덱스 (반값 씩) 만큼씩 빼줍니다. 그리고 여기에 원본 이미지의 반값씩 더해주면 됩니다. 이것은 re_img 의 x, y 가 f 연산을 했을 때 이동했을 위치와 원본이미지의 위치의 차를 구해서 더해주는 것입니다. 그리고 그 연산의 결과가 가르키는 값이 $1, x, 1, y$ 사이에 있지 않을 경우 0으로 채워넣는 연산을 하는 것입니다. 그리고 인덱스의 근사값을 취해준 뒤에 넣어주면 됩니다.

Bilinear

```
elseif strcmp(interpolation, 'bilinear')
    pad_img = zeros(x+2,y+2);
    pad_img(2:x+1,2:y+1) = img;
    for i = 1:row
        for j = 1:col
            v = f * double([(i-x_center);(j-y_center)]);
            v(1) = v(1)+round(x/2);
            v(2) = v(2)+round(y/2);
            if v(1) < 1 || v(1) > x || v(2) < 1 || v(2) > y
                continue;
            end
            s = abs(v(1)-round(v(1)));
            t = abs(v(2)-round(v(2)));
            re_img(i, j) = (1-s)*(1-t)*pad_img(floor(v(1)),floor(v(2)))+(s)*(t)*pad_img(floor(v(1))+1,floor(v(2))+1)+
                (1-s)*(t)*pad_img(floor(v(1)),floor(v(2))+1)+(s)*(1-t)*pad_img(floor(v(1))+1,floor(v(2)));
        end
    end
end
```



Bilinear입니다. 이 역시 이미지를 돌리는 매커니즘은 위의nearest와 같은데 로테이션한 이미지를 넣는 과정이 조금 다릅니다. 우선 bilinear연산을 해야하므로, 원본이미지를 한 칸씩 제로 패딩 시킵니다. s 와 t 를 정의하는데 여기서는 앞서 정의한 $v(1), v(2)$ 에 대해 round를 취한 값을 빼므로, 소수점을 넣어서 절대값을 취한 것을 넣어주고 이것에 대해 $(1-s) * (1-t) * \text{pad_img}(\text{floor}(v(1)), \text{floor}(v(2))) + (s) * (t) * \text{pad_img}(\text{floor}(v(1))+1, \text{floor}(v(2))+1) + (1-s) * (t) * \text{pad_img}(\text{floor}(v(1)), \text{floor}(v(2))+1) + (s) * (1-t) * \text{pad_img}(\text{floor}(v(1))+1, \text{floor}(v(2)))$ 를 수행한 결과를



bilinear연산을 한 결과를 최종 값에 넣어줍니다. 다만 여기서 쓰이는 이미지는 가중치에 대한 픽셀 값을 비교해야하므로, 가로세로 1칸씩 제로패딩된

이미지를 사용합니다.

구현 결과

Nearest	Bilinear
	

2. 느낀 점

:: 구현하면서 느낀 점, 어려웠던 점, 혹은 설명이 필요하다고 느낀 부분

너무 너무너무 진짜 제일 힘들었습니다. Rotation 원래를 조금 더 자세히 설명해주셨으면, 더 좋았으리라는 아쉬움이 많이 남습니다. 여러가지 방법들이 있는데 한가지 방법을 예시로 들어줬으면, 조금 좋았으리라 하는 생각이 듭니다....