Open Software Project

Assignment 01 Technical Report

1976235 오진솔

1. Rotate.cpp

1) Mat myrotate(Mat input, float angle, char* opt)

```
int row = input.rows;
int col = input.cols;
```

row, col 은 각각 원래 input 이미지의 행(세로) 열(가로) 크기를 의미한다

```
float radia n = angle * CV_PI / 180;
```

degree 값으로 받은 각도를 radian으로 변환한다.

```
float sq_row = ceil(row * sin(radian) + col * cos(radian));
float sq_col = ceil(col * sin(radian) + row * cos(radian));

Mat output = Mat::zeros(sq_row, sq_col, input.type());
```

sq_row, sq_col은 각각 output 이미지의 행(세로) 열(가로) 크기를 의미한다. 이 크기를 갖는 Output 이미지의 Mat를 생성하고 0으로 초기화한다.

```
for (int i = 0; i < sq_row; i++) {
    for (int j = 0; j < sq_col; j++) {
        float x = (j - sq_col / 2) * cos(radian) - (i - sq_row / 2) *
sin(radian) + col / 2;
        float y = (j - sq_col / 2) * sin(radian) + (i - sq_row / 2) *
cos(radian) + row / 2;
```

-output matrix에서의 픽셀 좌표는 0~sq_row, 0~sq_col 사이의 수인데, 이미지의 중심을 기준으로 회전시킬 것이므로 이미지의 중심을 원점으로 하는 좌표를 새롭게 구한다. -Output의 각 픽셀에서 inverse warping을 적용하여 대응되는 input 픽셀을 찾는다. -좌표를 다시 되돌려서 input matrix 에서의 픽셀 위치를 구한다. -output matrix의 모든 픽셀에 대해 위 과정을 반복한다.

```
if ((y >= 0) && (y <= (row - 1)) && (x >= 0) && (x <= (col - 1))) {
    if (!strcmp(opt, "nearest")) {
        x = round(x);
        y = round(y);
        output.at<T>(i, j) = input.at<T>(y, x);
    }
```

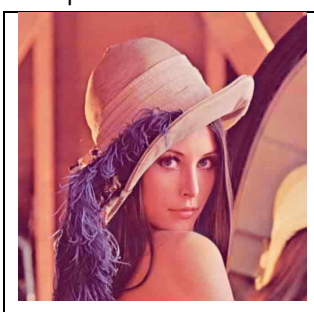
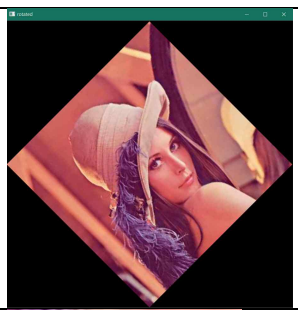
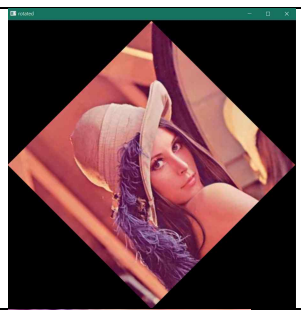
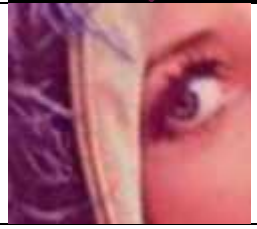
- 결과가 input image의 범위 안에 있고 nearest interpolation을 사용한다면,
- 실수인 x와 y를 반올림하여 가장 가까이 있는 정수 좌표를 구한다.(nearest neighbor interpolation)
- 해당 픽셀의 값을 output 에 복사한다.

```
    else if (!strcmp(opt, "bilinear")) {
        float Xdivide = x - floor(x);
        float Ydivide = y – floor(y);
        output.at<T>(i, j) = (Ydivide) * (input.at<T>(ceil(y), ceil(x))
* Xdivide + input.at<T>(ceil(y), floor(x)) * (1 - Xdivide))
                + (1 - Ydivide) * (input.at<T>(floor(y), ceil(x)) *
Xdivide + input.at<T>(floor(y), floor(x)) * (1 - Xdivide));
    }
```

- 결과가 input image의 범위 안에 있고 bilinear interpolation을 사용한다면,
- x와 y의 소수점 이하 자리를 계산해 Xdivide, Ydivide를 구한다
- Xdivide, Ydivide를 이용해 output 픽셀 값을 계산한다.

```
    }
}
```

2) Examples

<table>
<tr><td>  </td><td>  </td><td>  </td></tr>
<tr><td></td><td>  </td><td>  </td></tr>
<tr><td>Original</td><td>Nearest</td><td>bilinear</td></tr>
</table>

2. Stitching.cpp

1) main

- image1, image2 를 읽고 CV_32FC3 형식으로 변환한다.
- image stitching의 기준이 되는 대응점의 좌표를 각각 배열에 저장한다.

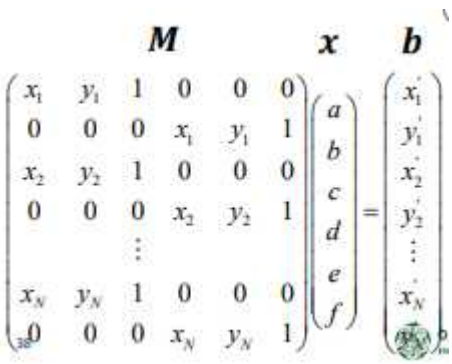
  ptl_x, ptl_y, ptr_x, ptr_y
- cal_affine 함수를 이용해 affine matrix를 구한다.
- I_f 이미지의 크기와 모서리 좌표를 구하고 초기화한다.
- inverse warping과 bilinear interpolation을 사용해서 I2를 I_f 에 알맞은 각도로 복사한다.
- blend_stitching을 이용해서 I1 을 I_f에 붙인다.

2) cal_affine(int ptl_x[], int ptl_y[], int ptr_x[], int ptr_y[], int number_of_points)

```
Mat M(2 * number_of_points, 6, CV_32F, Scalar(0));
Mat b(2 * number_of_points, 1, CV_32F);
Mat M_trans, temp, affineM;

// initialize matrix
for (int i = 0; i < number_of_points; i++) {
        M.at<T>(2 * i, 0) = ptl_x[i];        M.at<T>(2 * i, 1) = ptl_y[i];        M.at<T>(2 * i, 2) = 1;
        M.at<T>(2 * i + 1, 3) = ptl_x[i];        M.at<T>(2 * i + 1, 4) = ptl_y[i];
        M.at<T>(2 * i + 1, 5) = 1;
        b.at<T>(2 * i) = ptr_x[i];        b.at<T>(2 * i + 1) = ptr_y[i];
}
```

오른쪽 그림과 같은 행렬 M, b를 만들고 초기화한다.



```
transpose(M, M_trans);
invert(M_trans * M, temp);
affineM = temp * M_trans * b;

return affineM;
```

$(M^{T}M)^{-1}M^{T}b$ 를 계산하고 그 값을 반환한다.

3) blend_stitching(const Mat I1, const Mat I2, Mat &I_f, int bound_l, int bound_u, float alpha)

```
for (int i = 0; i < I1.rows; i++) {
    for (int j = 0; j < I1.cols; j++) {
        bool cond_I2 = I_f.at<Vec3f>(i - bound_u, j - bound_l) != Vec3f(0, 0, 0) ? true : false;

        if (cond_I2)
            I_f.at<Vec3f>(i - bound_u, j - bound_l) = alpha * I1.at<Vec3f>(i, j) + (1 - alpha) *
            I_f.at<Vec3f>(i - bound_u, j - bound_l);
        else
            I_f.at<Vec3f>(i - bound_u, j - bound_l) = I1.at<Vec3f>(i, j);
```

I1과 겹치는 범위의 I_f의 각 픽셀에 대해,
그 자리에 이미 I2가 존재한다면 $\alpha I_1 + (1-\alpha)I_2'$ 를 계산한다.
I2가 존재하지 않는다면 I1의 값을 복사한다.

4) Example

Image1 (left)

Image2 (right)

result