**Open Software Project**

**경영학과 1517034 정성원**

**Assignment 1.**

**1. rotation.cpp**

**1) Idea**

-NN interpolation과 bilinear interpolation을 이용해 주어진 이미지를

**2) Code**

# include <opencv2/opencv.hpp>

# include <iostream>

using namespace cv;

using namespace std;

template <typename T>

Mat myrotate(const Mat input, float angle, const char\* opt);

int nearest(float a);

int bilinear(const Mat input, float x, float y, int ch);

int main() {

Mat input, rotated;

//Read each image

input = imread("lena.jpg");

//Check for invalid input

if (!input.data) {

cout << "Could not open or find the image" << endl;

return -1;

}

//original image

imshow("image", input);

//rotated = myrotate<Vec3b>(input, 45, "nearest");

rotated = myrotate<Vec3b>(input, 45, "bilinear");

//rorated image

imshow("rotated", rotated);

waitKey(0);

return 0;

}

template <typename T>

Mat myrotate(const Mat input, float angle, const char\*opt) {

int row = input.rows;

int col = input.cols;

float radian = angle \* CV\_PI / 180;

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());

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;

if ((y >= 0) && (y <= (row - 1)) && (x >= 0) && (x <= (col - 1))){

if (!strcmp(opt, "nearest")) {

//nearest method

output.at<Vec3b>(i, j)[0] += input.at<Vec3b>(nearest(y), nearest(x))[0];

output.at<Vec3b>(i, j)[1] += input.at<Vec3b>(nearest(y), nearest(x))[1];

output.at<Vec3b>(i, j)[2] += input.at<Vec3b>(nearest(y), nearest(x))[2];

}

else if (!strcmp(opt, "bilinear")) {

//bilinear method

output.at<Vec3b>(i, j)[0] += bilinear(input, y, x, 0);

output.at<Vec3b>(i, j)[1] += bilinear(input, y, x, 1);

output.at<Vec3b>(i, j)[2] += bilinear(input, y, x, 2);

}

}

}

}

return output;

}

int nearest(float a) {

/\*return nearest integer\*/

if (abs(a - (int)a) <= (abs(a + 1 - (int(a)))))

return int(a);

return int(a) + 1;

}

int bilinear(const Mat input, float x, float y, int ch) {

/\*return intensity of floating point using bilinear interpolation\*/

int x1 = int(x); int x2 = int(x) + 1;

int y1 = int(y); int y2 = int(y) + 1;

int temp1 = (x2 - x)\*input.at<Vec3b>(x1, y2)[ch] + (x - x1)\*input.at<Vec3b>(x1, y1)[ch];

int temp2 = (x2 - x)\*input.at<Vec3b>(x2, y2)[ch] + (x - x1)\*input.at<Vec3b>(x2, y1)[ch];

return (x2 - x) \* temp2 + (x - x1) \* temp1;

}

**3) Result**



**2. stitching.cpp**

**1) Idea**

-Affine transformation을 이용한 image stitching

**2) Code**

**A. Code for calculate affine**

template <typename T>

Mat 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++) {

// initialize M

M.at<float>(i \* 2, 0) = M.at<float>(i \* 2 + 1, 3) = ptl\_x[i];

M.at<float>(i \* 2, 1) = M.at<float>(i \* 2 + 1, 4) = ptl\_y[i];

M.at<float>(i \* 2, 2) = M.at<float>(i \* 2 + 1, 5) = 1;

b.at<float>(i \* 2, 0) = ptr\_x[i];

b.at<float>(i \* 2 + 1, 0) = ptr\_y[i];

}

// (M^T \* M)^(−1) \* M^T \* b ( \* : Matrix multiplication)

//cout << "M" << M << endl;

//cout << "M.size()" << M.size() << endl;

M\_trans = M.t();

//cout << M << endl;

affineM = (M\_trans \* M).inv() \* M\_trans \* b;

//cout << affineM << endl;

return affineM;

}

**B. Code for Merge two images**

// initialize merged image

Mat I\_f(bound\_b - bound\_u + 1, bound\_r - bound\_l + 1, CV\_32FC3, Scalar(0));

// inverse warping with bilinear interplolation

for (int i = -diff\_x\_; i < I\_f.rows - diff\_x\_; i++) {

for (int j = -diff\_y\_; j < I\_f.cols - diff\_y\_; j++) {

float x = A12.at<float>(0) \* i + A12.at<float>(1) \* j + A12.at<float>(2) + diff\_x\_;

float y = A12.at<float>(3) \* i + A12.at<float>(4) \* j + A12.at<float>(5) + diff\_y\_;

int y1 = floor(y);

int y2 = ceil(y);

int x1 = floor(x);

int x2 = ceil(x);

float mu = y - y1;

float lambda = x - x1;

if (x1 >= 0 && x2 < I2\_row && y1 >= 0 && y2 < I2\_col){

for (int k = 0; k < 3; k++) {

float f1 = mu \* I2.at<Vec3f>(x1, y2)[k] + (1 - mu) \* I2.at<Vec3f>(x1, y1)[k];

float f2 = mu \* I2.at<Vec3f>(x2, y2)[k] + (1 - mu) \* I2.at<Vec3f>(x2, y1)[k];

I\_f.at<Vec3f>(i + diff\_x\_, j + diff\_y\_)[k] += lambda \* f2 + (1 - lambda) \* f1;

}

}

}

}

**3) Result**

