**Lab 6: Bilinear Interpolation**

**2020 Spring**

**Multi-Media & Lab**

**Name/ID: 201533661 이승수**

**Dept: Software**

**[My Code]**

//#include <opencv2/core/core.hpp>

//#include <opencv2/imgcodecs.hpp>

//#include <opencv2/highgui/highgui.hpp>

#include <opencv2/opencv.hpp>

#include <iostream>

#include <string>

using namespace cv;

using namespace std;

Mat bilinearInterpolation(const Mat& imageSrc,double scaleX, double scaleY);

int main(int argc, char\*\* argv) {

//Lab6: Image Warping

string src\_Path = "D:\\repos\_VS\\Project\_sourceIMG\\";//image source file path(folder)

//Load image

if (argc > 1) {

src\_Path = argv[1];

}

Mat src = imread((src\_Path + "Lena\_color.png").c\_str(), IMREAD\_COLOR);

//Mat backgroundImage = imread((src\_Path + "sky\_background.jpg").c\_str(), IMREAD\_COLOR); //Mat::zeros(src.size(), src.type());

if (src.empty()) {

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

return -1;

}

//Image load end result: src

//Lab6-4

Mat dst;

//resize(src,dst,Size(),1.5,1.5,INTER\_LINEAR);

dst = bilinearInterpolation(src,1.5,1.5);

std::cout << "src image size: " << src.size << std::endl;

std::cout << "dst image size: " << dst.size << std::endl;

imwrite(src\_Path + "lena\_resized\_nearestNeighbor.png", dst);

//Display results

imshow("source Image", src);

imshow("bilinear interpolated Image", dst);//6-4

//Wait until user exits program

std::cout << "program Ended. press any Key. ";

waitKey(0);

return 0;

}

Mat bilinearInterpolation(const Mat& imageSrc,double scaleX, double scaleY)

{

assert(!imageSrc.empty());

Mat output = Mat::zeros(imageSrc.rows \* scaleX, imageSrc.cols \* scaleY, imageSrc.type());

for (int y = 0; y < imageSrc.rows; y++) {

for (int x = 0; x < imageSrc.cols; x++) {

double alpha = x / scaleX;

double beta = 1 - alpha;

int pfx = floor(x / scaleX); int pcx = ceil(x / scaleX);

int pfy = floor(y / scaleY); int pcy = ceil(y / scaleY);

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

output.at<Vec3b>(y, x)[c] = alpha\*imageSrc.at<Vec3b>(pfy, pfx)[c]+ beta \* imageSrc.at<Vec3b>(pcy, pcx)[c];

}

}

}

resize(imageSrc, output, Size(imageSrc.cols \* scaleX, imageSrc.rows \* scaleY));

return output;

}

**[Result]**

**case(scaleX:1.5, scaleY:1.5)**

