**Lab 9: Stereo Matching**

**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>

#include <cmath>

using namespace cv;

using namespace std;

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

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

//Lab09: Stereo Matching

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

//Load image

if (argc > 1) {

src\_Path = argv[1];

}

Mat src\_left = imread((src\_Path + "scene1.row3.col2.png").c\_str(), IMREAD\_GRAYSCALE);

Mat src\_right = imread((src\_Path + "scene1.row3.col3.png").c\_str(), IMREAD\_GRAYSCALE);

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

if (src\_left.empty() || src\_right.empty()) {

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

return -1;

}

//Image load end result: src

//Lab09

Mat dst=Mat::zeros(src\_left.size(),CV\_8U);

//int h = src\_left.rows; int w = src\_left.cols;

int win\_size = 3;

int srange = 16;

int best\_disp;

float cost, min\_cost;

std::cout << "src image size: " << src\_left.size << std::endl;//288\*384

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

//std::cout << "src info: " << h << ", " << w << std::endl;

for (int y = win\_size / 2; y < src\_left.rows - win\_size / 2; y++) {

for (int x = win\_size / 2; x < src\_left.cols - win\_size / 2; x++) {

best\_disp = 0;

min\_cost = INT\_MAX;

for (int d = 0; d <= srange; d++) {

cost = 0;

if (x - d - win\_size / 2 < 0)

break;//check if out of image boundary

for (int i = 0; i < win\_size; i++) {

for (int j = 0; j < win\_size; j++) {

cost += pow(src\_left.at<uchar>(y + (i - win\_size / 2), x + (j - win\_size / 2))-

src\_right.at<uchar>(y+(i-win\_size/2),x-d+(j-win\_size/2)),2);

}

}

if (min\_cost > cost) {

best\_disp = d;

min\_cost = cost;

}

}

dst.at<uchar>(y, x) = best\_disp;

}

}

dst = dst \* 10;

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

//Display results

imshow("source\_left Image", src\_left);

imshow("disparity map", dst);//6-4

//Wait until user exits program

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

waitKey(0);

return 0;

}

**[Result]**

