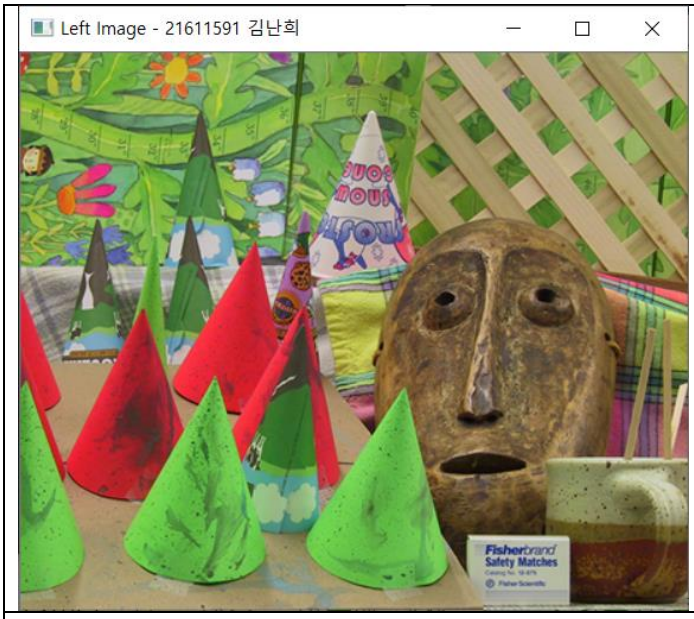
[Computer Vision Programming]

Lab 9. Stereo Matching and Rendering

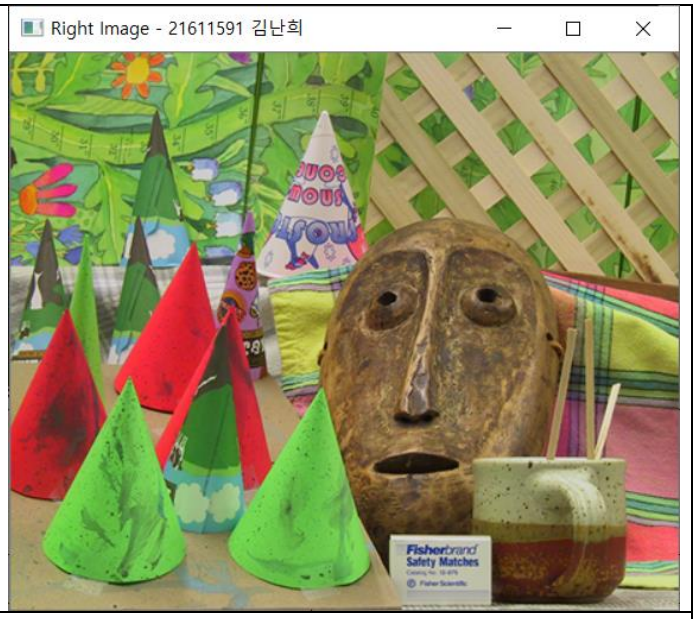
전자공학과 21611591 김난희

Lab. 1. Block matching based Disparity

-결과 사진

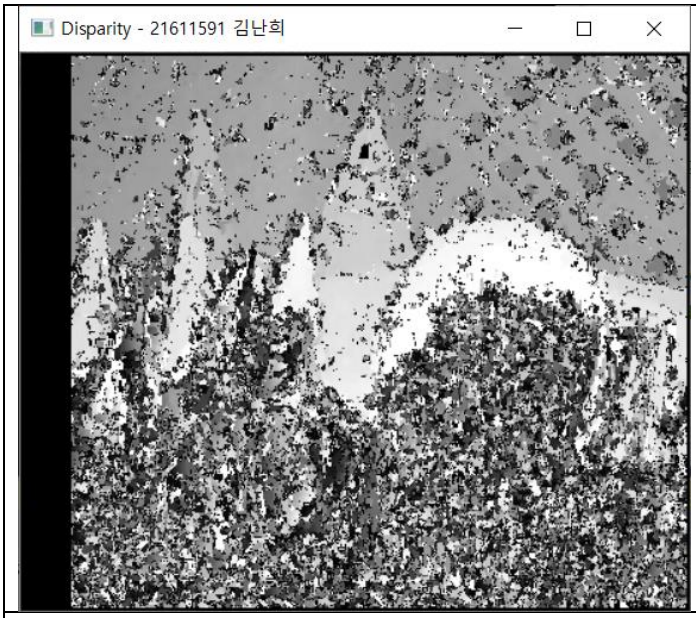


Left Image

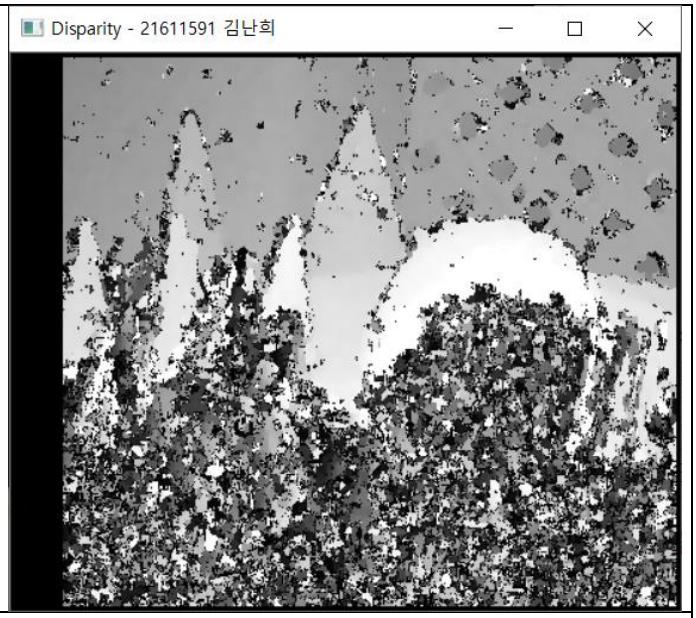


Right Image

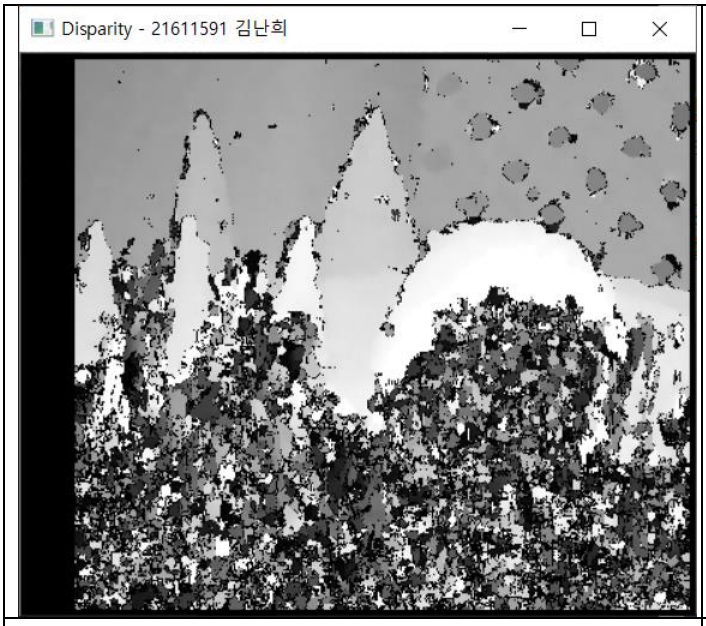
Middlebury Stereo Datasets에서 가져온 Stereo image를 input으로 넣은 후,
Support 영역에 따라 결과로 나온 Disparity 사진이다.



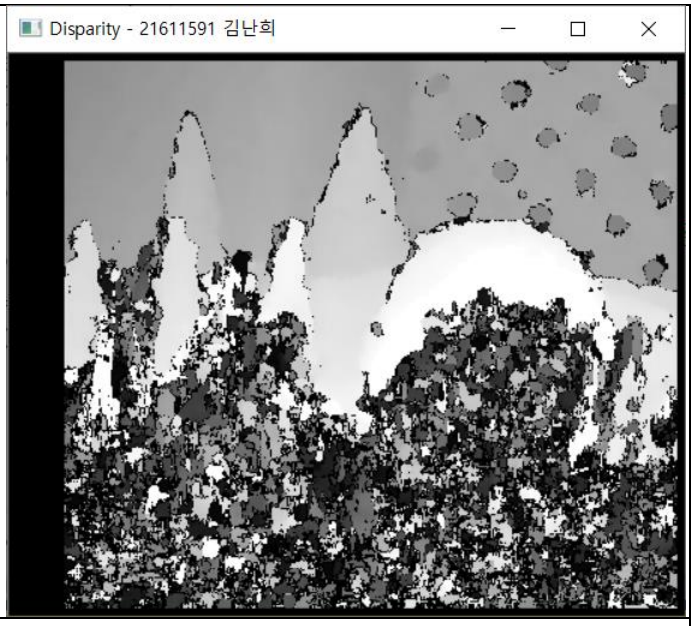
Support: 5



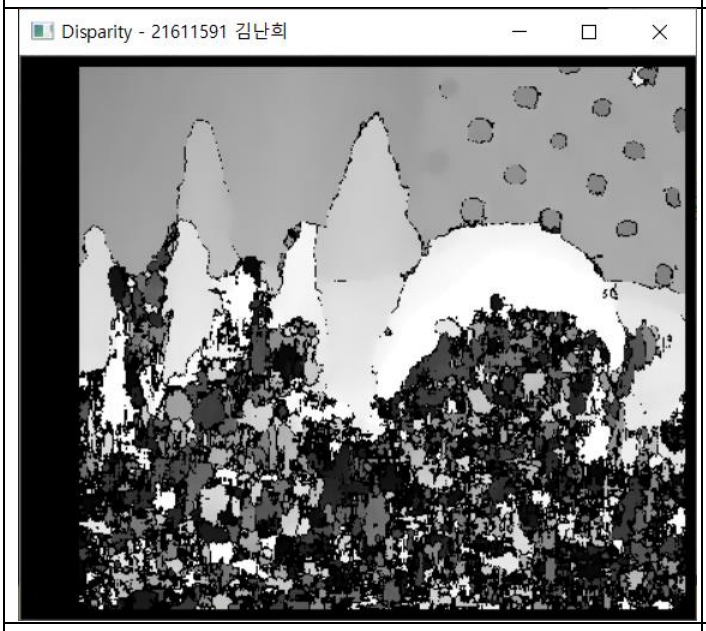
Support: 7



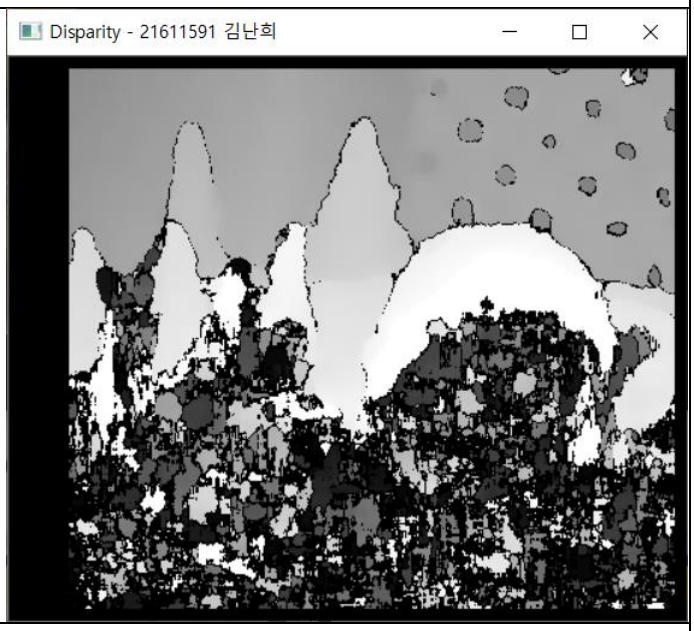
Support: 9



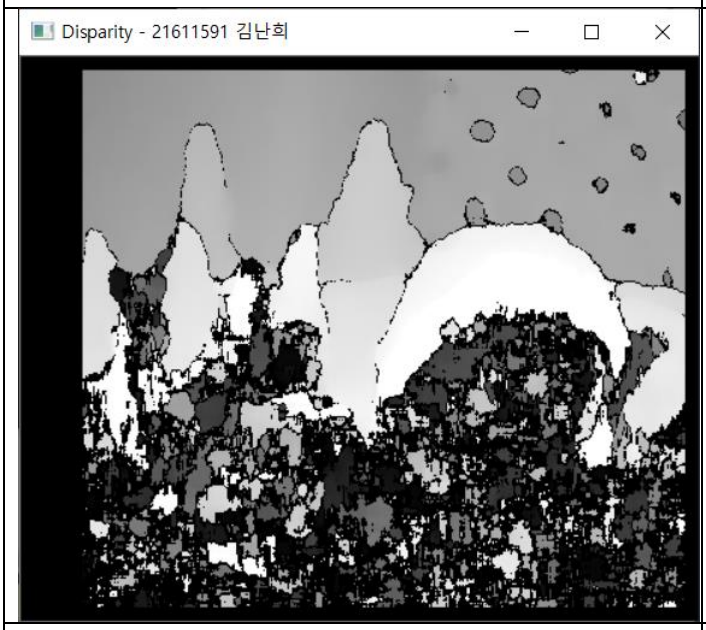
Support: 11



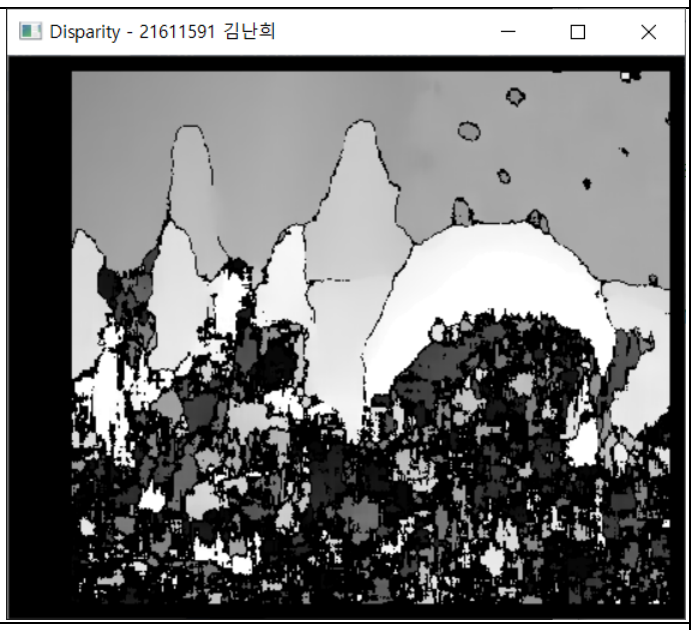
Support: 15



Support: 17



Support: 19



Support: 21

SAD: sum of absolute difference, window size로
BMState->SADWindowSize를 수정하였다.
사이즈는 5x5부터 21x21까지 수정할 수 있다.

-discursion

Support 영역이 11에서 가장 결과가 좋았다. Support 영역이 11->5로 변하면, 모호성이 커져서 오히려 잘 안 나오게 된다. 21은 두루뭉실하지만(정밀도 떨어짐) matching은 더 잘 나왔다.

여러 이미지를 테스트 해보았을 때, 이 이미지가 변화가 가장 잘 보였다.
우선 stereo 이미지를 구하기 위해서는 수평축 각도가 변경된 곳에서 사진을 찍는 것보단, 한 번 찍어서 왼쪽으로 치우친 사진과 오른쪽으로 치우친 사진으로 잘라 생성하는 것이 더 좋은 결과를 나타냈다. 이는 stereo 원리를 바탕으로 생각할 수 있었다.

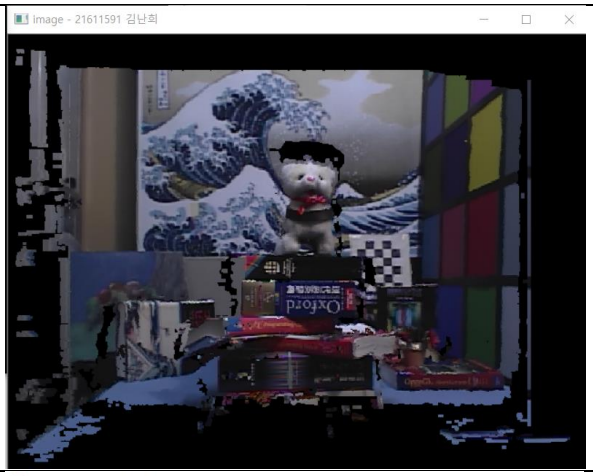
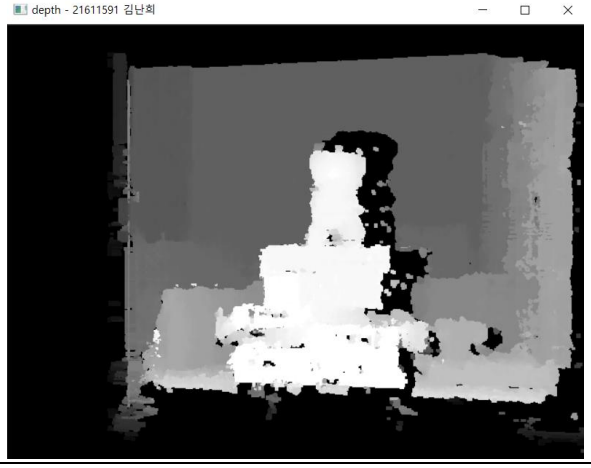
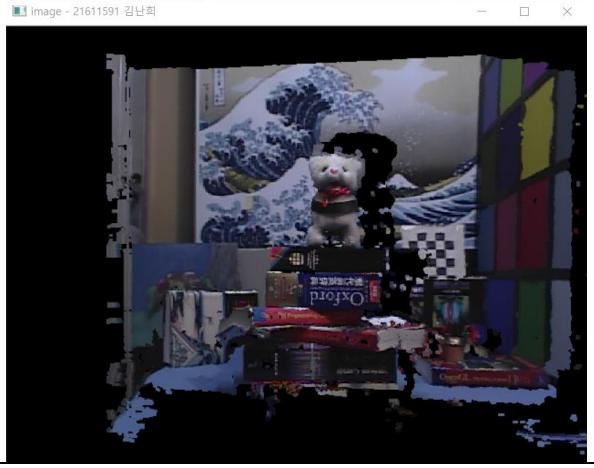
소스 코드

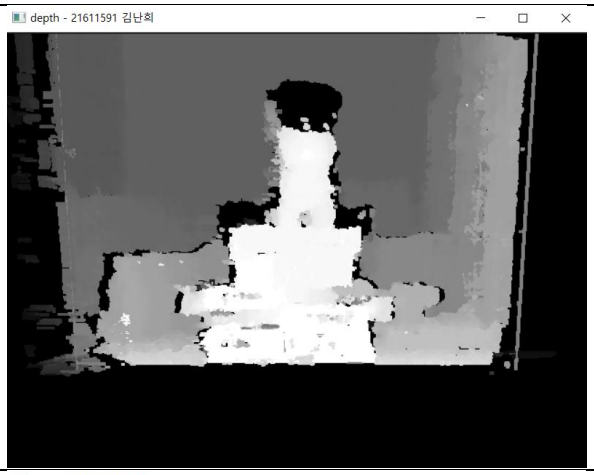
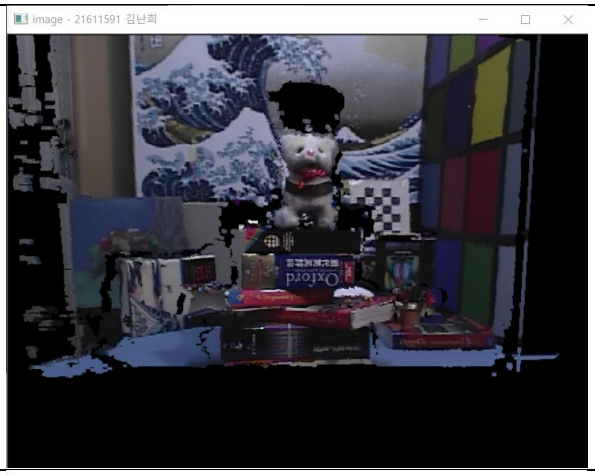
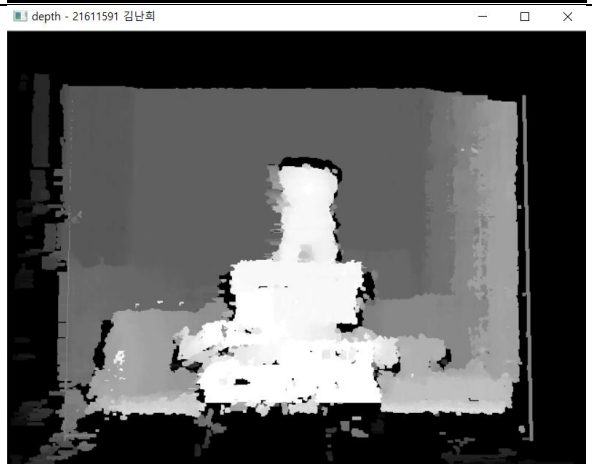
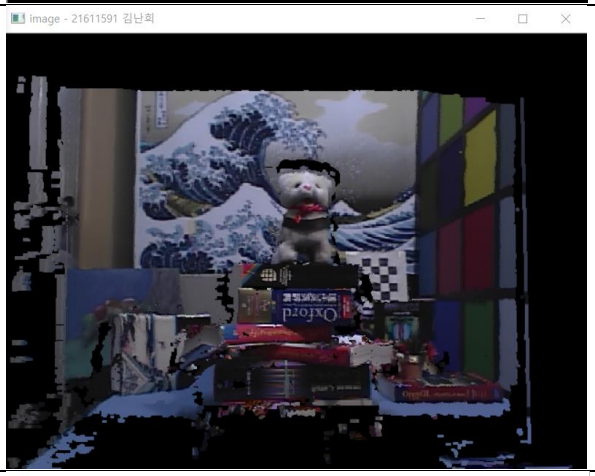
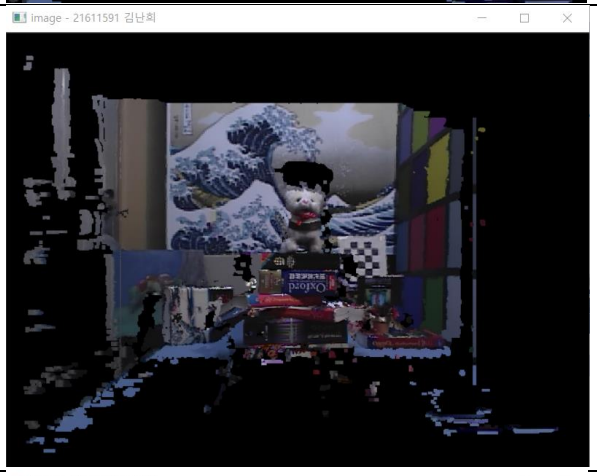
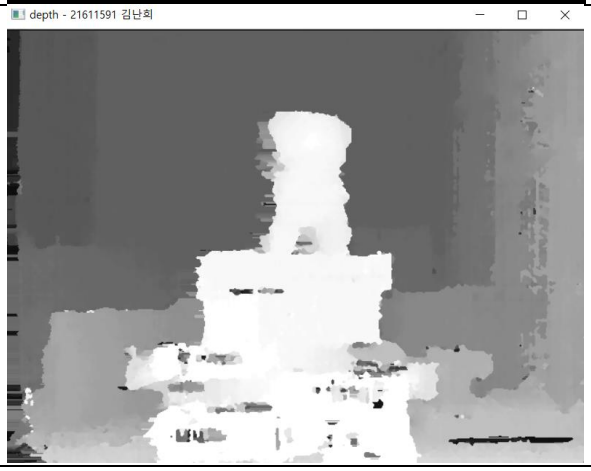
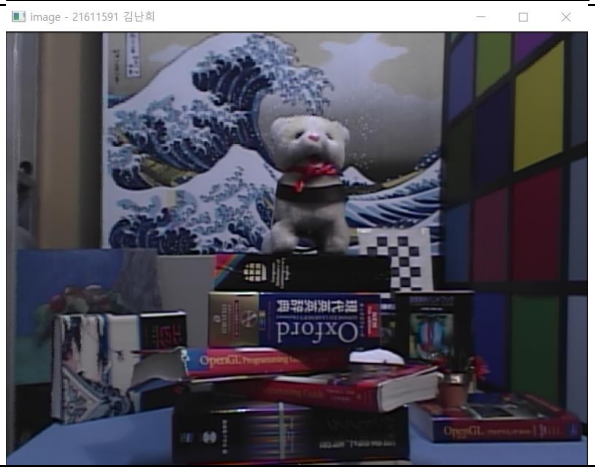
```
1   #include "opencv2/highgui/highgui.hpp"
2   #include "opencv2/imgproc/imgproc.hpp"
3   #include "opencv2/core/core.hpp"
4   #include "opencv2/calib3d/calib3d.hpp"
5   #include "opencv2/opencv.hpp"
6
7   using namespace cv;
652 int main(int argc, char** argv)
653 {
654     //* * * * * * * * * * lab 1. block matching based disparity * * * * * * * * * * *//
655     //load stereo images -> 영상 2장
656     Mat l = imread("im2.ppm"); // left  영상
657     Mat r = imread("im6.ppm"); // right 영상 right_un.png
658
659     //image resize because big size
660     //resize(l, l, size(l.size().width / 4, l.size().height / 4));
661     //resize(r, r, size(r.size().width / 4, r.size().height / 4));
662
663     imshow("Left image - 21611591 김난희", l);
664     imshow("Right image - 21611591 김난희", r);
665
666     // set block matching parameters // 파라미터 세팅
667     CvStereoBMState *BMState = cvCreateStereoBMState(); // 클래스를 이용해 객체 만듦 -> 함수가 주소 return한 것을 받음
668     BMState->preFilterSize = 5;  // brightness correction, 5x5 ~ 21x21 // 객체는 ., pointer는 그 주소로 가서 화살표로 함
669     BMState->preFilterCap = 5;  // removed region after prefilter
670     BMState->SADWindowSize = 17; // sad: sum of absolute difference, window size (5x5.... 21x21) // 윈도우 사이즈 11(support 영역) 좀 크게함
671     //support 영역 11->5 모호성이 커져서 오히려 잘 안나옴 // 25는 두루뭉실하지만(정밀도 떨어짐) matching은 더 잘함
672     BMState->minDisparity = 1;   // minimum disparity [pixel] // 찾고자하는 minimum 값
673     BMState->numberOfDisparities = 32; // maximum disparity [pixel] // 1~32까지 pixel 봄
674     BMState->textureThreshold = 10;  // minimum allowed
675     BMState->uniquenessRatio = 5;    // uniqueness  (removing false matching)
676
677     // convert color space
678     Mat Left, Right;
679     cvtColor(l, Left, CV_RGB2GRAY); // gray로 변환
680     cvtColor(r, Right, CV_RGB2GRAY);
681
682     // type conversion: mat to iplimage
683     IplImage *left, *right;
684     left = &IplImage(Left);
685     right = &IplImage(Right);
686
687     CvMat* disparity = cvCreateMat(Left.rows, Left.cols, CV_16SC1);
688     CvMat* disparity_img = cvCreateMat(Left.rows, Left.cols, CV_8UC1); // 1~32 -> 0~255 밝기를 눈으로 볼 수 있게 바꿔줌
689
690     // run algorithm
691     cvFindStereoCorrespondenceBM(left, right, disparity, BMState);
692     cvNormalize(disparity, disparity_img, 0, 255, CV_MINMAX);  //normalize to display
693
694     // show results // stereo matching show
695     cvShowImage("disparity - 21611591 김난희", disparity_img);
696
697     waitKey(0);
```

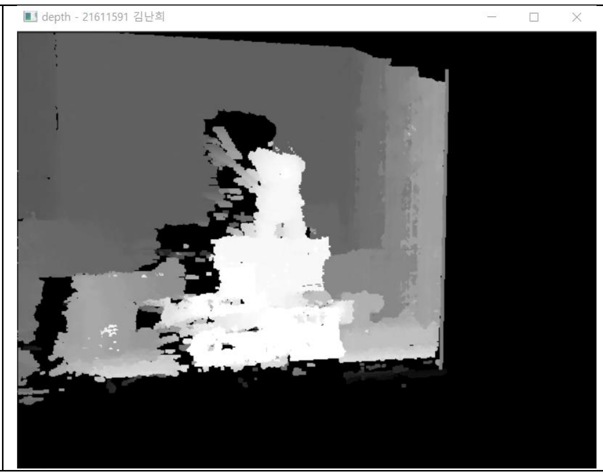
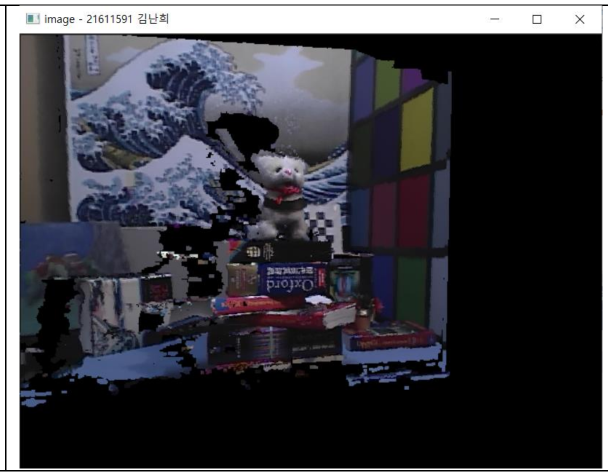
Lab. 2. 3D Rendering

- Render the disparity in 3D space (point cloud)

-결과 사진

시점의 변화 (중심의 변화) [x;y;z]	depth image (렌더링된 시차 이미지)	picture (렌더링 결과)
[0; 0; 140] 시작 시점		
[56; 0; 140] z key x 증가		
[-42; 0; 140] x key x 감소		

[0; 42; 140]		
a key y 증가		
[0; -14; 140]		
s key y 감소		
[0; 0; 336]		
q key z 증가		
[0; 0; 0]		
w key z 감소		

 x, y, z 모두 이동		

-discursion

테스트를 하면서 이전 lab 1에서 실험을 하며 깨달은 것을 적용시켜 lab 2를 진행하였다. 하지만, lab 2에서 stereo left image와 right image를 여러 사진을 하며 테스트해보았지만 저자가 사용한 사진에 비해 결과가 많이 좋지 않았다. 저자가 사용한 사진을 테스트 결과로 사용하였다.

여러 input 사진으로 테스트하며 알게 된 점은,

1. input 사진은 lab 1에서처럼 한 사진을 찍어 왼쪽, 오른쪽 각각으로 치우친 사진을 만들어 실험하는 것이 결과가 잘 나왔다.
2. 밝은 배경에 물체 하나만 있는다고 해서 좋은 결과가 나오지 않았다.
3. 물체 주변이 그림자가 있을수록 더 좋은 결과를 나타냈는데, 오히려 물체가 없이 그림자만 있다면 depth 추정을 더욱 못한다.
4. 물체는 윤곽이 있을수록 더욱 좋다
5. 배경과 물체가 비슷한 색깔이라면 depth 결과가 좋지 못하다.

사이트에서 소개하는 유튜브 영상처럼 카메라 구도가 빠르게 움직이지 않았다. key 버튼을 수정하며 여러 구도에서 볼 수 있었는데, 결과 사진은 천천히 움직였다. 이미지를 작게 변형할수록 계산량이 줄어들어 빠른 구도 변화를 할 수 있었다. 그러나, 유튜브 영상처럼 아주 멀리서, 다양한 각도로, 3D 모습이 더욱 잘 보이게 하는 것은 불가능했다. OpenMP를 이용하여 병렬처리를 했다고 하는데 소스에서 omp와 관련된 부분인 것 같다.

저자가 사용한 key 버튼이 불편해서 다른 key 버튼으로 변경하였다. 전체적인 소스코드는 구도 변화에 대한 소스 코드로, 어떻게 변화하는 지에 대한 개념을 더 공부한 후에, 다시 본다면 도움될 것이다.

소스 코드(Render the disparity in 3D space (point cloud))

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/core/core.hpp"
#include "opencv2/calib3d/calib3d.hpp"
#include "opencv2/opencv.hpp"

using namespace cv;
using namespace std;

//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * /
//* * * * * * * * * * for lab2 - Render the disparity in 3D space (point cloud) * * * * * * * * * * poing cloud를 사용하려면 여기서부터 주석 해제 * * * * * * * * /
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * /
```

```cpp
template <class T>
static void fillOcclusionInv_(Mat& src, T invalidvalue)
{
    int bb = 1;
    const int MAX_LENGTH = src.cols*0.8;
#pragma omp parallel for // openMP를 이용한 병렬 처리
    for (int j = bb; j < src.rows - bb; j++)
    {
        T* s = src.ptr<T>(j);
        const T st = s[0];
        const T ed = s[src.cols - 1];
        s[0] = 0;
        s[src.cols - 1] = 0;
        for (int i = 0; i < src.cols; i++)
        {
            if (s[i] == invalidvalue)
            {
                int t = i;
                do
                {
                    t++;
                    if (t > src.cols - 1)break;
                } while (s[t] == invalidvalue);

                const T dd = max(s[i - 1], s[t]);
                    if (t - i > MAX_LENGTH)
                    {
                        for (int n = 0; n < src.cols; n++)
                        {
                            s[n] = invalidvalue;
                        }
                    }
                    else
                    {
                        for (; i < t; i++)
                        {
                            s[i] = dd;
                        }
                    }
                }
            }
        }
    }
template <class T>  <T>
static void fillOcclusion_(Mat& src, T invalidvalue)
{
    int bb = 1;
    const int MAX_LENGTH = src.cols*0.5;
#pragma omp parallel for // openMP를 이용한 병렬 처리
    for (int j = bb; j < src.rows - bb; j++)
    {
        T* s = src.ptr<T>(j);
        const T st = s[0];
        const T ed = s[src.cols - 1];
        s[0] = 255;
        s[src.cols - 1] = 255;
        for (int i = 0; i < src.cols; i++)
        {
            if (s[i] <= invalidvalue)
            {
                int t = i;
                do
                {
                    t++;
                    if (t > src.cols - 1)break;
                } while (s[t] <= invalidvalue);

                const T dd = min(s[i - 1], s[t]);
                if (t - i > MAX_LENGTH)
                {
                    for (int n = 0; n < src.cols; n++)
                    {
                        s[n] = invalidvalue;
                    }
                }
                else
                {
                    for (; i < t; i++)
                    {
                        s[i] = dd;
                    }
                }
            }
        }
    }
```

```cpp
void fillOcclusion(Mat& src, int invalidvalue, bool isInv = false)
{   // 불규칙한 값을 조작하여 불규칙한 것을 최소화하게 다시 채움
    if (isInv)
    {
        if (src.type() == CV_8U)
        {
            fillOcclusionInv_<uchar>(src, (uchar)invalidvalue);
        }
        else if (src.type() == CV_16S)
        {
            fillOcclusionInv_<short>(src, (short)invalidvalue);
        }
        else if (src.type() == CV_16U)
        {
            fillOcclusionInv_<unsigned short>(src, (unsigned short)invalidvalue);
        }
        else if (src.type() == CV_32F)
        {
            fillOcclusionInv_<float>(src, (float)invalidvalue);
        }
    }
    else
    {
        if (src.type() == CV_8U)
        {
            fillOcclusion_<uchar>(src, (uchar)invalidvalue);
        }
        else if (src.type() == CV_16S)
        {
            fillOcclusion_<short>(src, (short)invalidvalue);
        }
        else if (src.type() == CV_16U)
        {
            fillOcclusion_<unsigned short>(src, (unsigned short)invalidvalue);
        }
        else if (src.type() == CV_32F)
        {
            fillOcclusion_<float>(src, (float)invalidvalue);
        }
    }
}
void eular2rot(double yaw, double pitch, double roll, Mat& dest)
{
    double theta = yaw / 180.0*CV_PI;
    double pusai = pitch / 180.0*CV_PI;
    double phi = roll / 180.0*CV_PI;

    double datax[3][3] = { {1.0,0.0,0.0},
    {0.0,cos(theta),-sin(theta)},
    {0.0,sin(theta),cos(theta)} };
    double datay[3][3] = { {cos(pusai),0.0,sin(pusai)},
    {0.0,1.0,0.0},
    {-sin(pusai),0.0,cos(pusai)} };
    double dataz[3][3] = { {cos(phi),-sin(phi),0.0},
    {sin(phi),cos(phi),0.0},
    {0.0,0.0,1.0} };
    Mat Rx(3, 3, CV_64F, datax);
    Mat Ry(3, 3, CV_64F, datay);
    Mat Rz(3, 3, CV_64F, dataz);
    Mat rr = Rz * Rx*Ry;
    rr.copyTo(dest);
}

void lookat(Point3d from, Point3d to, Mat& destR)
{
    double x = (to.x - from.x);
    double y = (to.y - from.y);
    double z = (to.z - from.z);
```

```
167
168         double pitch = asin(x / sqrt(x*x + z * z)) / CV_PI * 180.0;
169         double yaw = asin(-y / sqrt(y*y + z * z)) / CV_PI * 180.0;
170
171         eular2rot(yaw, pitch, 0, destR);
172     }
173     template <class T>
174     static void projectImagefromXYZ_(Mat& image, Mat& destimage, Mat& disp, Mat& destdisp, Mat& xyz, Mat& R, Mat& t, Mat& K, Mat& dist, Mat& mask, bool isSub)
175     {
176         if (destimage.empty())destimage = Mat::zeros(Size(image.size()), image.type());
177         if (destdisp.empty())destdisp = Mat::zeros(Size(image.size()), disp.type());
178
179         vector<Point2f> pt;
180         if (dist.empty()) dist = Mat::zeros(Size(5, 1), CV_32F);
181         cv::projectPoints(xyz, R, t, K, dist, pt);
182
183         destimage.setTo(0);
184         destdisp.setTo(0);
185
186     #pragma omp parallel for
187         for (int j = 1; j < image.rows - 1; j++)
188         {
189             int count = j * image.cols;
190             uchar* img = image.ptr<uchar>(j);
191             uchar* m = mask.ptr<uchar>(j);
192             for (int i = 0; i < image.cols; i++, count++)
193             {
194                 int x = (int)(pt[count].x + 0.5);
195                 int y = (int)(pt[count].y + 0.5);
196                 if (m[i] == 255)continue;
197                 if (pt[count].x >= 1 && pt[count].x < image.cols - 1 && pt[count].y >= 1 && pt[count].y < image.rows - 1)
198                 {
199                     short v = destdisp.at<T>(y, x);
200                     if (v < disp.at<T>(j, i))
201                     {
202                         destimage.at<uchar>(y, 3 * x + 0) = img[3 * i + 0];
203                         destimage.at<uchar>(y, 3 * x + 1) = img[3 * i + 1];
204                         destimage.at<uchar>(y, 3 * x + 2) = img[3 * i + 2];
205                         destdisp.at<T>(y, x) = disp.at<T>(j, i);
206
207                         if (isSub)
208                         {
209                             if ((int)pt[count + image.cols].y - y > 1 && (int)pt[count + 1].x - x > 1)
210                             {
211                                 destimage.at<uchar>(y, 3 * x + 3) = img[3 * i + 0];
212                                 destimage.at<uchar>(y, 3 * x + 4) = img[3 * i + 1];
213                                 destimage.at<uchar>(y, 3 * x + 5) = img[3 * i + 2];
214
215                                 destimage.at<uchar>(y + 1, 3 * x + 0) = img[3 * i + 0];
216                                 destimage.at<uchar>(y + 1, 3 * x + 1) = img[3 * i + 1];
217                                 destimage.at<uchar>(y + 1, 3 * x + 2) = img[3 * i + 2];
218
219                                 destimage.at<uchar>(y + 1, 3 * x + 3) = img[3 * i + 0];
220                                 destimage.at<uchar>(y + 1, 3 * x + 4) = img[3 * i + 1];
221                                 destimage.at<uchar>(y + 1, 3 * x + 5) = img[3 * i + 2];
222
223                                 destdisp.at<T>(y, x + 1) = disp.at<T>(j, i);
224                                 destdisp.at<T>(y + 1, x) = disp.at<T>(j, i);
225                                 destdisp.at<T>(y + 1, x + 1) = disp.at<T>(j, i);
226                             }
227                             else if ((int)pt[count - image.cols].y - y < -1 && (int)pt[count - 1].x - x < -1)
```

```
228                     {
229                         destimage.at<uchar>(y, 3 * x - 3) = img[3 * i + 0];
230                         destimage.at<uchar>(y, 3 * x - 2) = img[3 * i + 1];
231                         destimage.at<uchar>(y, 3 * x - 1) = img[3 * i + 2];
232 
233                         destimage.at<uchar>(y - 1, 3 * x + 0) = img[3 * i + 0];
234                         destimage.at<uchar>(y - 1, 3 * x + 1) = img[3 * i + 1];
235                         destimage.at<uchar>(y - 1, 3 * x + 2) = img[3 * i + 2];
236 
237                         destimage.at<uchar>(y - 1, 3 * x - 3) = img[3 * i + 0];
238                         destimage.at<uchar>(y - 1, 3 * x - 2) = img[3 * i + 1];
239                         destimage.at<uchar>(y - 1, 3 * x - 1) = img[3 * i + 2];
240 
241                         destdisp.at<T>(y, x - 1) = disp.at<T>(j, i);
242                         destdisp.at<T>(y - 1, x) = disp.at<T>(j, i);
243                         destdisp.at<T>(y - 1, x - 1) = disp.at<T>(j, i);
244                     }
245                     else if ((int)pt[count + 1].x - x > 1)
246                     {
247                         destimage.at<uchar>(y, 3 * x + 3) = img[3 * i + 0];
248                         destimage.at<uchar>(y, 3 * x + 4) = img[3 * i + 1];
249                         destimage.at<uchar>(y, 3 * x + 5) = img[3 * i + 2];
250 
251                         destdisp.at<T>(y, x + 1) = disp.at<T>(j, i);
252                     }
253                     else if ((int)pt[count - 1].x - x < -1)
254                     {
255                         destimage.at<uchar>(y, 3 * x - 3) = img[3 * i + 0];
256                         destimage.at<uchar>(y, 3 * x - 2) = img[3 * i + 1];
257                         destimage.at<uchar>(y, 3 * x - 1) = img[3 * i + 2];
258 
259                         destdisp.at<T>(y, x - 1) = disp.at<T>(j, i);
260                     }
261                     else if ((int)pt[count + image.cols].y - y > 1)
262                     {
263                         destimage.at<uchar>(y + 1, 3 * x + 0) = img[3 * i + 0];
264                         destimage.at<uchar>(y + 1, 3 * x + 1) = img[3 * i + 1];
265                         destimage.at<uchar>(y + 1, 3 * x + 2) = img[3 * i + 2];
266 
267                         destdisp.at<T>(y + 1, x) = disp.at<T>(j, i);
268                     }
269                     else if ((int)pt[count - image.cols].y - y < -1)
270                     {
271                         destimage.at<uchar>(y - 1, 3 * x + 0) = img[3 * i + 0];
272                         destimage.at<uchar>(y - 1, 3 * x + 1) = img[3 * i + 1];
273                         destimage.at<uchar>(y - 1, 3 * x + 2) = img[3 * i + 2];
274 
275                         destdisp.at<T>(y - 1, x) = disp.at<T>(j, i);
276                     }
277                 }
278             }
279         }
280     }
281 }
282 
283 if (isSub)
284 {
285     Mat image2;
286     Mat disp2;
287     destimage.copyTo(image2);
288     destdisp.copyTo(disp2);
289     const int BS = 1;
290 #pragma omp parallel for
291     for (int j = BS; j < image.rows - BS; j++)
292     {
```

```
291             for (int j = BS; j < image.rows - BS; j++)
292             {
293                 uchar* img = destimage.ptr<uchar>(j);
294                 T* m = disp2.ptr<T>(j);
295                 T* dp = destdisp.ptr<T>(j);
296                 for (int i = BS; i < image.cols - BS; i++)
297                 {
298                     if (m[i] == 0)
299                     {
300                         int count = 0;
301                         int d = 0;
302                         int r = 0;
303                         int g = 0;
304                         int b = 0;
305                         for (int l = -BS; l <= BS; l++)
306                         {
307                             T* dp2 = disp2.ptr<T>(j + l);
308                             uchar* img2 = image2.ptr<uchar>(j + l);
309                             for (int k = -BS; k <= BS; k++)
310                             {
311                                 if (dp2[i + k] != 0)
312                                 {
313                                     count++;
314                                     d += dp2[i + k];
315                                     r += img2[3 * (i + k) + 0];
316                                     g += img2[3 * (i + k) + 1];
317                                     b += img2[3 * (i + k) + 2];
318                                 }
319                             }
320                         }
321                         if (count != 0)
322                         {
323                             double div = 1.0 / count;
324                             dp[i] = d * div;
325                             img[3 * i + 0] = r * div;
326                             img[3 * i + 1] = g * div;
327                             img[3 * i + 2] = b * div;
328                         }
329                     }
330                 }
331             }
332         }
333     }
334     void projectImagefromXYZ(Mat& image, Mat& destimage, Mat& disp, Mat& destdisp, Mat& xyz, Mat& R, Mat& t, Mat& K, Mat& dist, bool isSub = true, Mat& mask = Mat())
335     {
336         if (mask.empty())mask = Mat::zeros(image.size(), CV_8U);
337         if (disp.type() == CV_8U)
338         {
339             projectImagefromXYZ_<unsigned char>(image, destimage, disp, destdisp, xyz, R, t, K, dist, mask, isSub);
340         }
341         else if (disp.type() == CV_16S)
342         {
343             projectImagefromXYZ_<short>(image, destimage, disp, destdisp, xyz, R, t, K, dist, mask, isSub);
344         }
345         else if (disp.type() == CV_16U)
346         {
347             projectImagefromXYZ_<unsigned short>(image, destimage, disp, destdisp, xyz, R, t, K, dist, mask, isSub);
348         }
349         else if (disp.type() == CV_32F)
350         {
351             projectImagefromXYZ_<float>(image, destimage, disp, destdisp, xyz, R, t, K, dist, mask, isSub);
352         }
353         else if (disp.type() == CV_64F)
354         {
355             projectImagefromXYZ_<double>(image, destimage, disp, destdisp, xyz, R, t, K, dist, mask, isSub);
```

```
356         }
357     }
358 Mat makeQMatrix(Point2d image_center, double focal_length, double baseline)
359 {
360     Mat Q = Mat::eye(4, 4, CV_64F);
361     Q.at<double>(0, 3) = -image_center.x;
362     Q.at<double>(1, 3) = -image_center.y;
363     Q.at<double>(2, 3) = focal_length;
364     Q.at<double>(3, 3) = 0.0;
365     Q.at<double>(2, 2) = 0.0;
366     Q.at<double>(3, 2) = 1.0 / baseline;
367 
368     return Q;
369 }
370 void stereoTest()
371 {
372     //(1) Reading L&R images and estimating disparity map by semi-global block matching.
373     Mat image = imread("l.png", 1); // 스테레오 이미지를 읽어드림
374     Mat imageR = imread("r.png", 1);
375     Mat destimage;
376 
377     /*int resize_k = 4; // 사진 크기가 크면 사이즈 바꾸기
378     resize(image, image, Size(image.size().width / resize_k, image.size().height / resize_k));
379     resize(imageR, imageR, Size(imageR.size().width / resize_k, imageR.size().height / resize_k));*/
380 
381     StereoSGBM sgbm(1, 16 * 2, 3, 200, 255, 1, 0, 0, 0, 0, true); //SGBM으로 시차 화상을 계산함
382     Mat disp;
383     Mat destdisp;
384     Mat dispshow;
385     sgbm(image, imageR, disp); // 계산 결과의 시차는 최소 시차 -16의 값이 불규칙으로 할당됨
386     fillOcclusion(disp, 16); // 불규칙 값을 그럴싸한 값으로 재할당하는 함수
387 
388     //(2)make Q matrix and reproject pixels into 3D space // 매트릭스의 구축과 시차 이미지의 3차원 공간에 투영
389     const double focal_length = 598.57;
390     const double baseline = 14.0;
391     // 3차원 공간에 투영하기 위해 Q 행렬을 사용 // 스테레오 카메라를 보정하여 Q 행렬을 얻기 위해 CV::streoRevtify에서 보완함
392     Mat Q = makeQMatrix(Point2d((image.cols - 1.0) / 2.0, (image.rows - 1.0) / 2.0), focal_length, baseline * 16);
393 
394     Mat depth;
395     cv::reprojectImageTo3D(disp, depth, Q); // 시차를 3차원 공간에 투영함
396     Mat xyz = depth.reshape(3, depth.size().area()); // 3채널의 float 배열에 저장됨
397 
398     //(3) camera setting // 카메라 설정 // 행렬 설정
399     Mat K = Mat::eye(3, 3, CV_64F);
400     K.at<double>(0, 0) = focal_length;
401     K.at<double>(1, 1) = focal_length;
402     K.at<double>(0, 2) = (image.cols - 1.0) / 2.0;
403     K.at<double>(1, 2) = (image.rows - 1.0) / 2.0;
404 
405     Mat dist = Mat::zeros(5, 1, CV_64F);
406     Mat R = Mat::eye(3, 3, CV_64F);
407     Mat t = Mat::zeros(3, 1, CV_64F);
408 
409     Point3d viewpoint(0.0, 0.0, baseline * 10); // 관점
410     Point3d lookatpoint(0.0, 0.0, -baseline * 10.0); // 주시점
411     const double step = baseline;
412     int key = 0;
413     bool isSub = true;
414     //(4) rendering loop // 3차원 좌표에서 새로운 관점의 이미지에 점군을 투영
415     while (key != 27)
416     {
417         lookat(viewpoint, lookatpoint, R); // 가상 카메라의 방향이 업데이트됨
418         t.at<double>(0, 0) = viewpoint.x;
419         t.at<double>(1, 0) = viewpoint.y;
420         t.at<double>(2, 0) = viewpoint.z;
421 
```

```
            cout << t << endl;  // 좌표 출력
            t = R * t;

            //(5) projecting 3D point cloud to image.
            projectImagefromXYZ(image, destimage, disp, destdisp, xyz, R, t, K, dist, isSub);

            destdisp.convertTo(dispshow, CV_8U, 0.5);
            imshow("depth - 21611591 김난희", dispshow);
            imshow("image - 21611591 김난희", destimage);

            // 위치 방향 컨트롤
            if (key == 'f') // 양자화 억제 필터의 온 오프 전환
            {               // default 값은 ON
                isSub = isSub ? false : true;
            }
            if (key == 'a') // 아래로
            {
                viewpoint.y += step;
            }
            if (key == 's') // 위로
            {
                viewpoint.y -= step;
            }
            if (key == 'z') //오른쪽으로
            {
                viewpoint.x += step;
            }
            if (key == 'x') //왼쪽으로
            {
                viewpoint.x -= step;
            }
            if (key == 'q') // 멀리
            {
                viewpoint.z += step;
            }
            if (key == 'w') // 가까이
            {
                viewpoint.z -= step;
            }
            key = waitKey(1); // 실시간으로 key 값을 출력하기 위해 1이 들어감
                              // 0은 바뀔 때마다 키값을 출력하도록 함
        }
}
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * /
//* * * * * * * * * * for lab2 - Render the disparity in 3D space (point cloud) * * * * * * * * * * poing cloud를 사용하려면 여기까지 주석 끝 * * * * * * * * * * * /
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * /
```

- Texture mapping on the 3D shape

과제에 texture mapping과 관련된 부분이 있길래 조사해보니, OpenGL을 사용해야한다고 되어있었다.

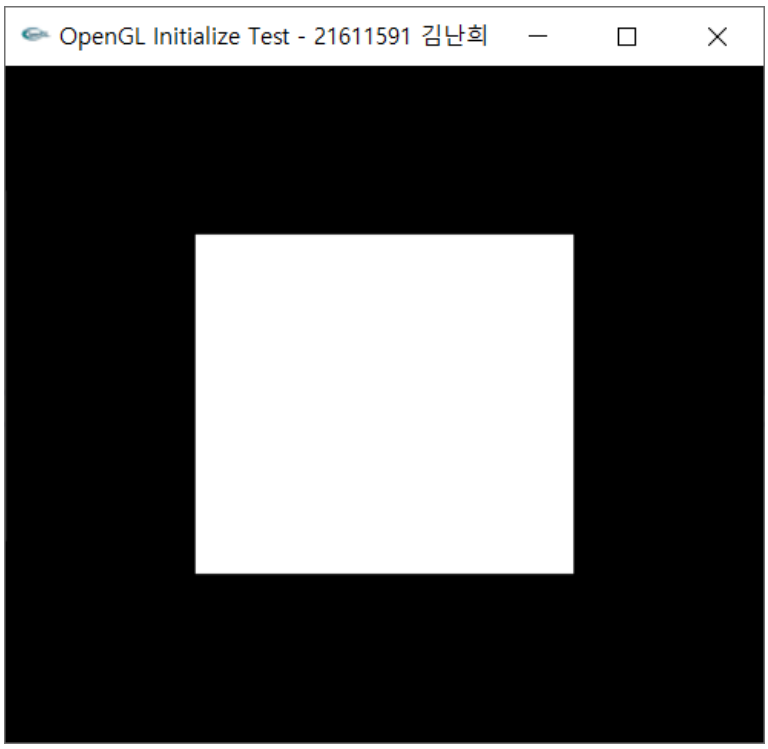
texture mapping 개념
텍스쳐매핑 [texture mapping] 텍스쳐 매핑이란 사실감 있는 입체 화상을 제작하기 위해 이미지와 같은 2차원 무늬를 점, 선, 면으로 구성된 3차원 오브젝트 표면에 적용시키는 방법.

texture mapping을 위해 opengl을 설치하였다. opengl을 설치하는 것은 opencv의 설치와 거의 동일했다. 참고 사이트의 파일을 다운로드 받아 bin(dll 파일), include(header 파일), lib(lib 파일)로 분류한 후, 로컬 디스크 C로 옮긴다. 그 후, 다음 과정을 따라한다.

1. 환경 변수 설정 -> opengl의 bin 폴더 설정 -> dll 설정 완료
2. visual studio의 설정 -> 프로젝트 속성 -> VC++디렉터리 -> 포함 디렉터리에는 opengl의 include 폴더로, 라이브러리 디렉터리는 opengl의 lib 폴더로 설정
3. visual studio의 설정 -> 프로젝트 속성 -> 링커 -> 입력 -> 추가종속성에 opengl의 lib 폴더에서 lib 파일 전체를 입력

이렇게 해서 opengl 테스트를 해보았다.

-결과 사진(opengl 테스트 결과)

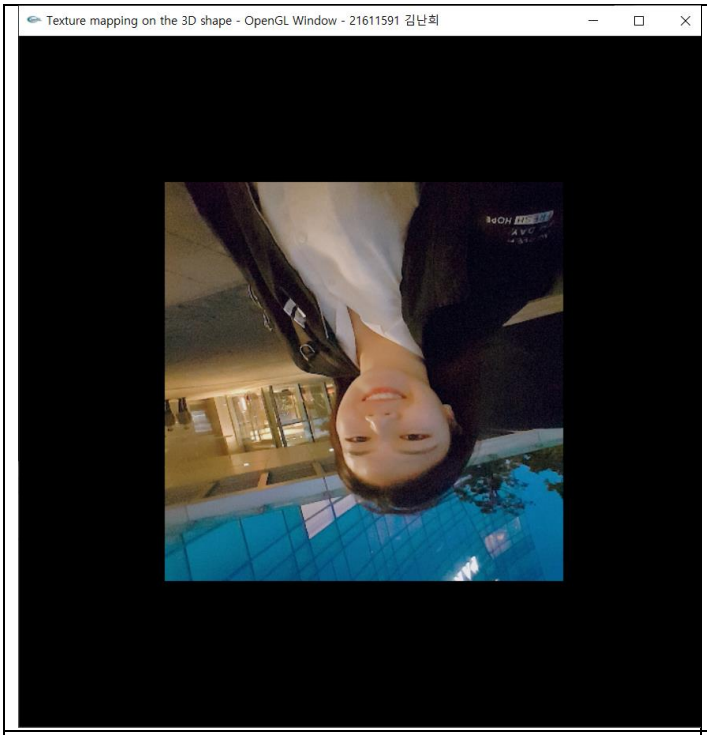


opengl 테스트 코드

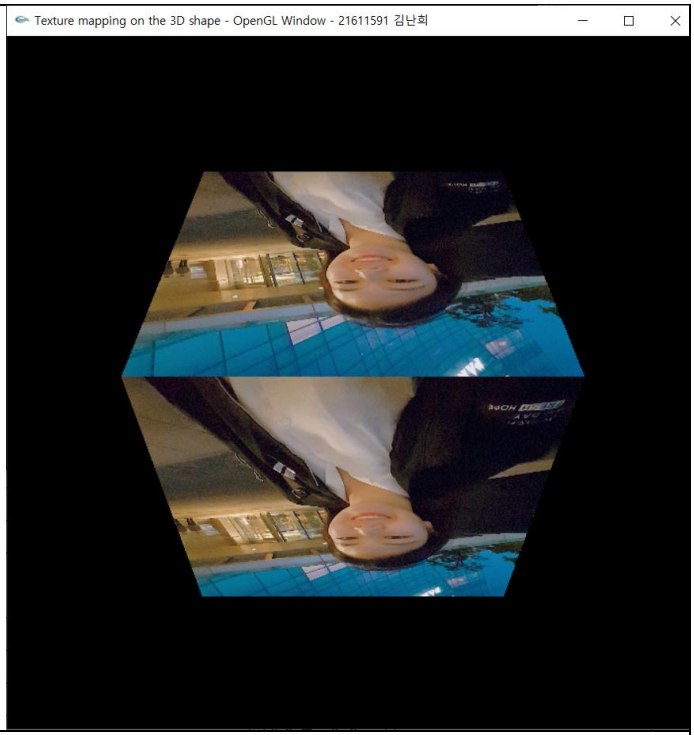
openGL_test

```
1   #include <glut.h>
2   #include <glu.h>
3
4   void MyDisplay() {
5       glClear(GL_COLOR_BUFFER_BIT);
6       glBegin(GL_POLYGON);
7       glVertex3f(-0.5, -0.5, 0.0);
8       glVertex3f(0.5, -0.5, 0.0);
9       glVertex3f(0.5, 0.5, 0.0);
10      glVertex3f(-0.5, 0.5, 0.0);
11      glEnd();
12      glFlush();
13  }
14  int main() {
15      glutCreateWindow("OpenGL Initialize Test");
16      glutDisplayFunc(MyDisplay);
17      glutMainLoop();
18      return 0;
19  }
```

- Texture mapping 결과 사진(결과 동영상 함께 첨부)

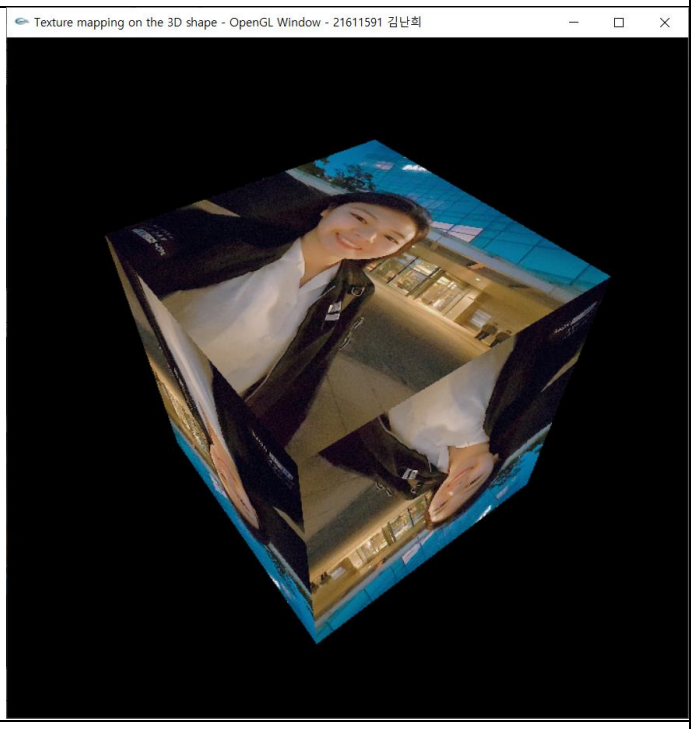


original

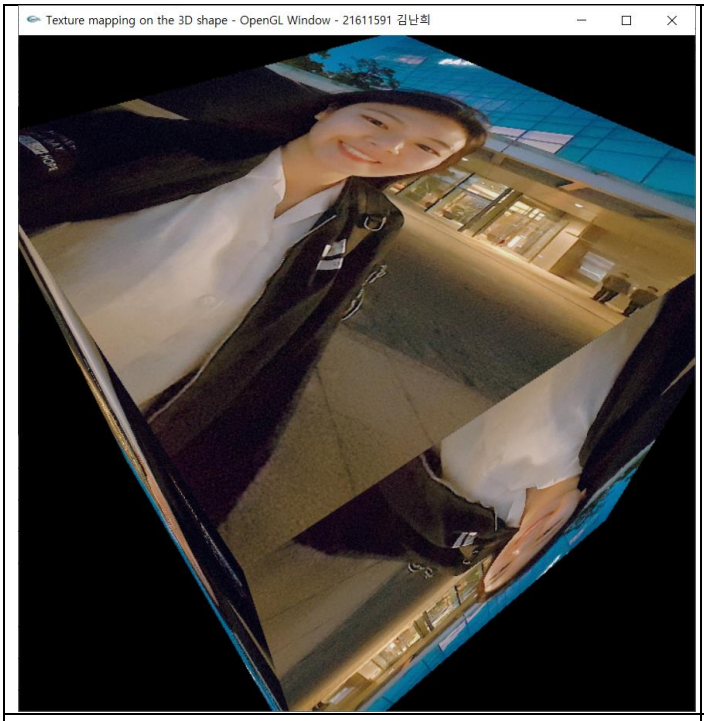


상향 키(위로 회전)

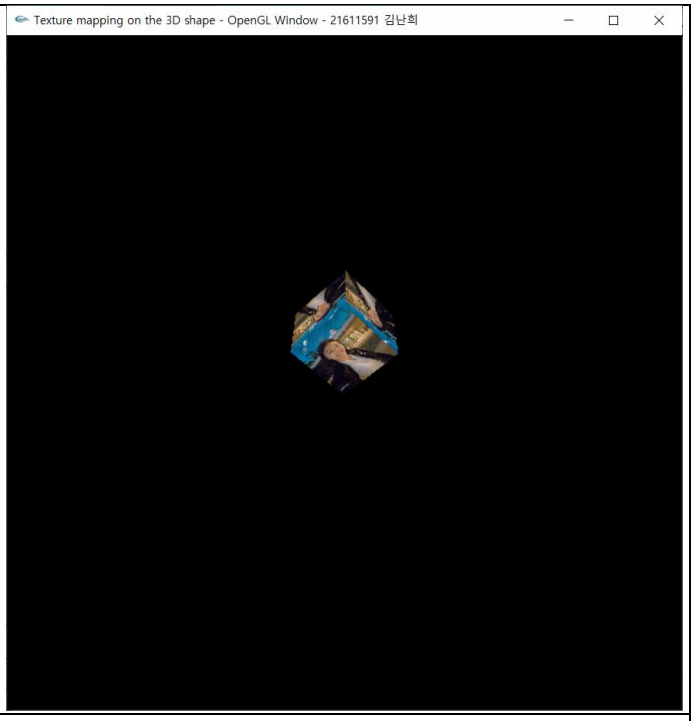
우향 키(오른쪽으로 회전)



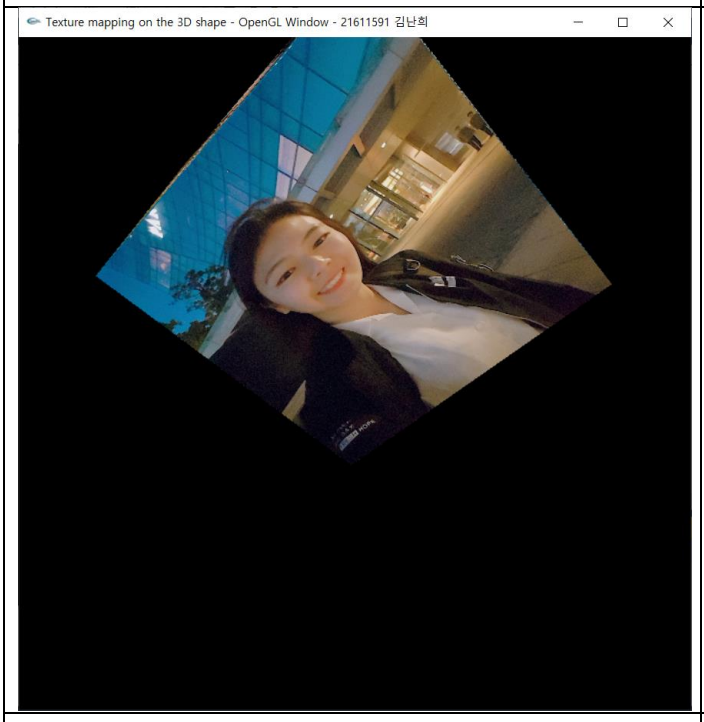
하향 키(아래로 회전)



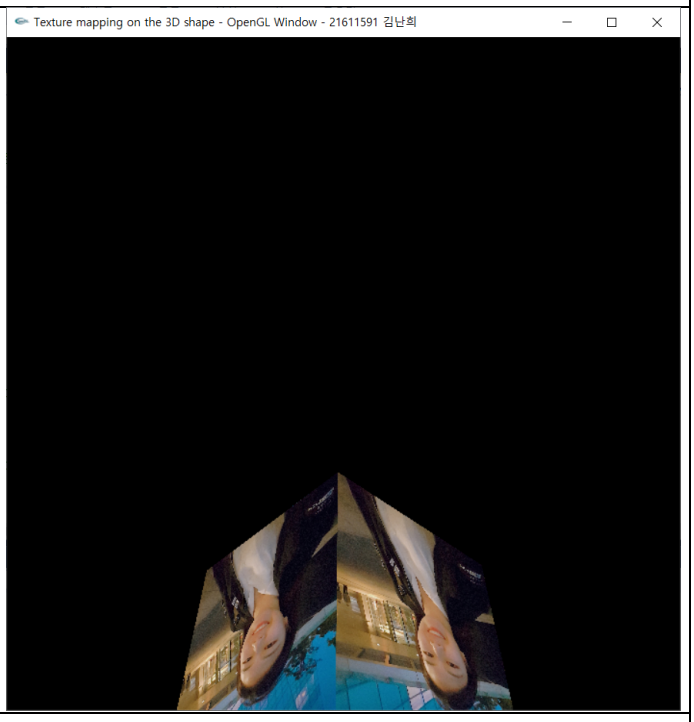
q 키(가깝게 이동)



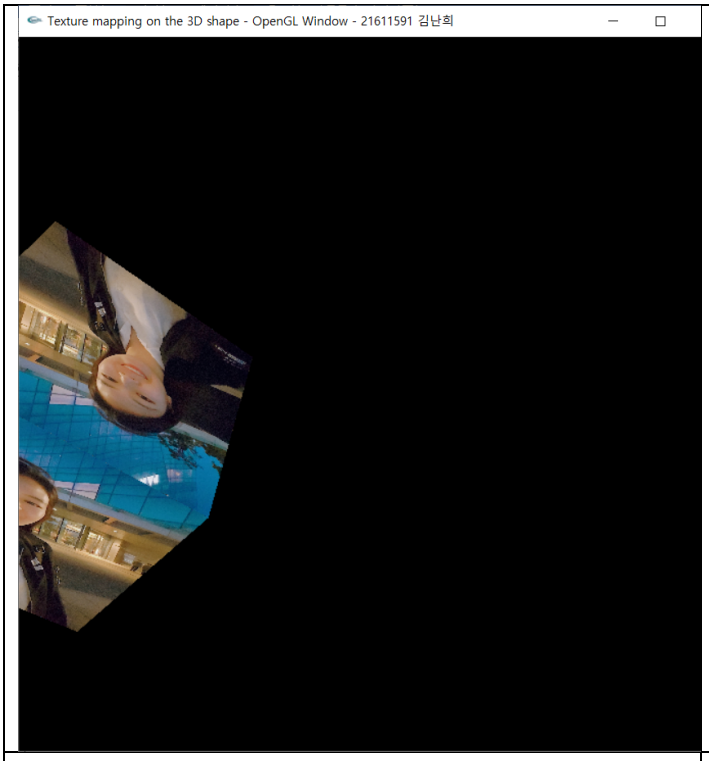
w 키(멀리 이동)



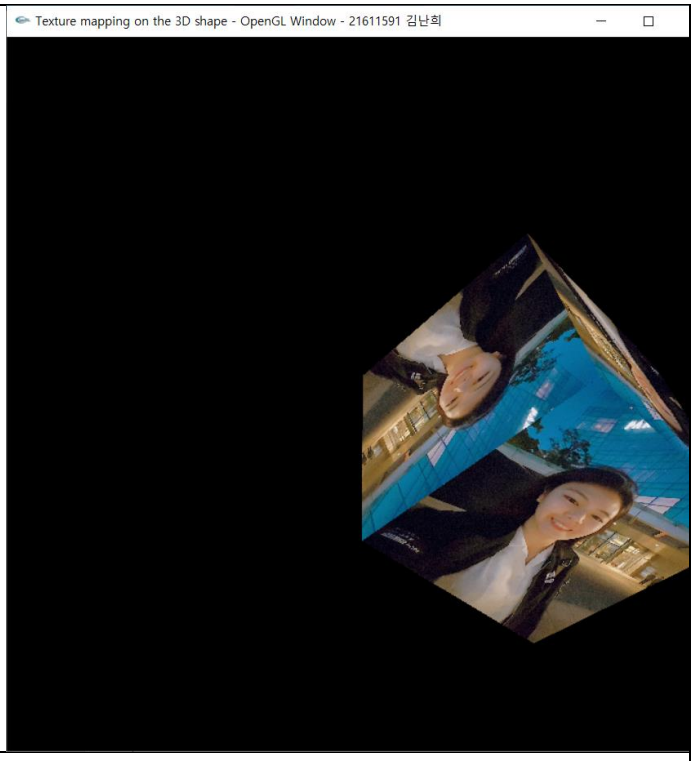
a키(위로 이동)



s 키(아래로 이동)

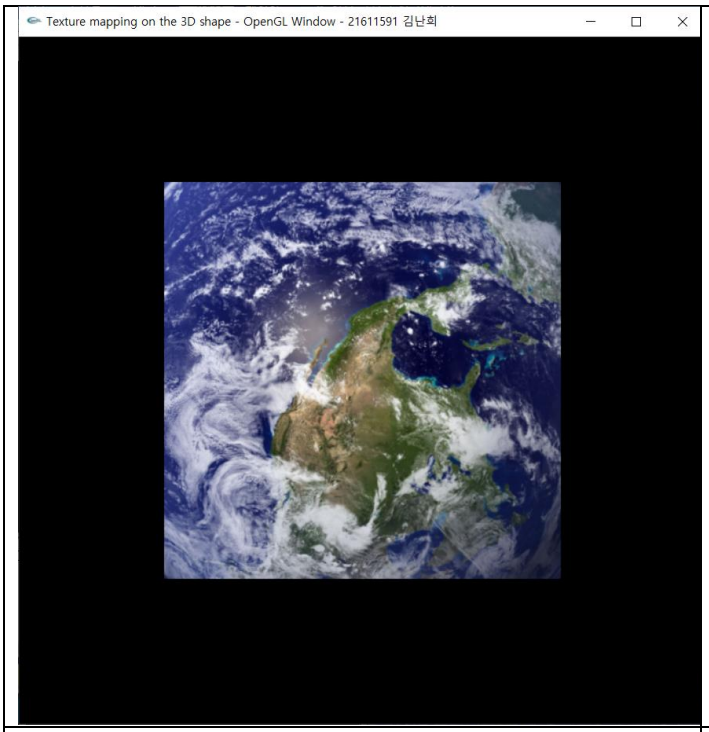


x 키(왼쪽 이동)

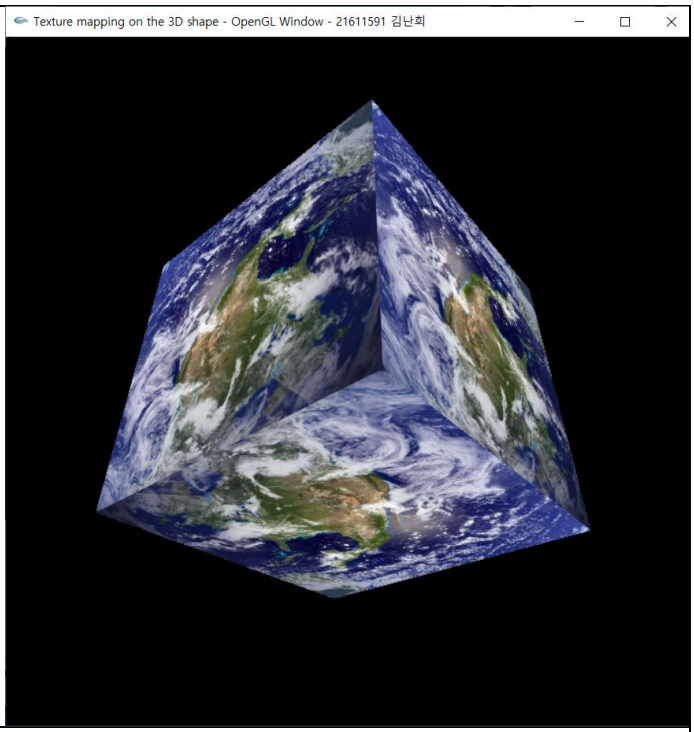


z 키(오른쪽 이동)

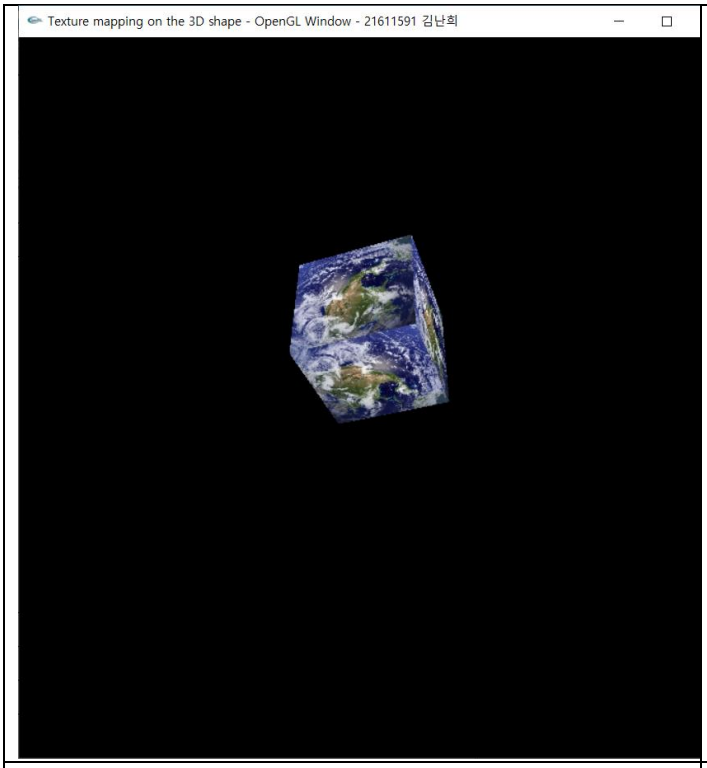
-추가 실험(earth) 결과 사진



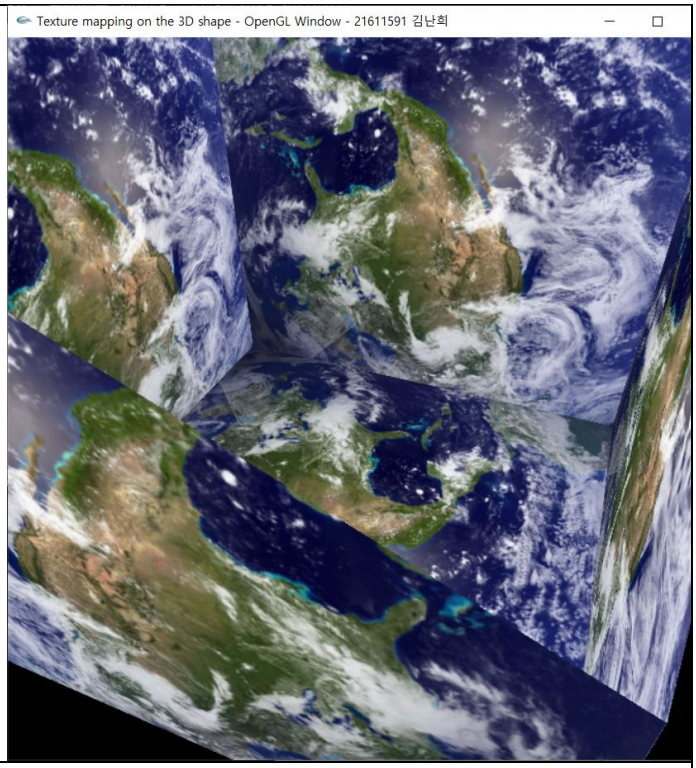
original



방향키 이동



w 키로 멀리 이동



q 키로 가깝게 이동(내부가 보임)

-discursion

opengl을 처음 사용해보아 gpu를 사용해야하는지 등 모르는 점이 많았다. 생각보다 구글에는 설명이 잘 되어있었다. opencv에서 cloud point를 이용한 렌더링보다 opengl로 texture mapping을 한 것은 훨씬 빨랐다. opengl의 내부 소스가 자세하게 어떻게 이루어진지는 잘 모르겠지만, opencv는 픽셀 한 점, 한 점을 계산하다보니 긴 계산 시간이 필요했다.

처음 사이트에서 찾은 texture mapping 소스는 움직일 수 없었다. 다른 사이트를 통해 키보드를 이용하여 움직이도록 소스를 변형해주었다. 생각과 다르게 opengl로 요리조리 움직임은 빠르게 잘 되었다. opengl에 대해 더 공부가 필요하지만, 그래픽적으로 opencv보다 더욱 뛰어난 것 같다.

소스 코드(Texture mapping on the 3D shape)

```
473  ///openGL 관련
474 #include <windows.h>
475  #include <glut.h>
476
477  ///openCV 관련
478 #include "opencv2/core/core.hpp"
479  #include "opencv2/highgui/highgui.hpp"
480  #include "opencv2/imgproc/imgproc.hpp"
481
482  #include <strstream>
483  #include <functional>
484  #include <stdio.h>
485
486  using namespace cv;
487
488  GLuint TextureIdx[1]; //the array for our TextureIdx
489  Mat mSource_Bgr;      //texture mapping할 image
490  GLfloat angle = 45.0; //original source의 cube 각도
491
492  /// keyboard로 3d 큐브 움직이기
493  // (1) 큐브 위치
494  float cubeX = 0.0;
495  float cubeY = 0.0;
496  float cubeZ = -4.0;
497  // (2) 회전
498  float pitch = 0.0;
499  float yaw = 0.0;
500  float roll = 0.0;
501
502
503 int UploadTexture(Mat image, GLuint &TextureIdx) ///The my Image to OpenGl TextureIdx function
504  {
505      if (image.empty())
506          return -1;
507      glGenTextures(1, &TextureIdx);
508      glBindTexture(GL_TEXTURE_2D, TextureIdx); //bind the TextureIdx to it's array
509      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
510      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
511      glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
512      glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, image.cols, image.rows, 0, GL_BGR_EXT, GL_UNSIGNED_BYTE, image.data);
513      return 0;
514  }
515 void FreeTexture(GLuint TextureIdx) ///texture mapping 후 동적할당 해제와 같음
516  {
517      glDeleteTextures(1, &TextureIdx);
518  }
519 void special(int key, int x, int y) // 방향키에 의해 회전을 입력
520  {
521      if (key == GLUT_KEY_UP)
522      {
523          pitch += 1.0;
524      }
525      else if (key == GLUT_KEY_DOWN)
526      {
527          pitch -= 1.0;
528      }
529      else if (key == GLUT_KEY_RIGHT)
530      {
531          yaw += 1.0;
532      }
```

```
533     else if (key == GLUT_KEY_LEFT)
534     {
535         yaw -= 1.0;
536     }
537 }
538 void keyboard(unsigned char key, int x, int y) // 상하 좌우 가깝게멀리 입력 //* 주의할 점: 대문자 영문은 안됨
539 {
540     //cout << "다음 키가 눌러졌습니다. ₩"" << key << "₩" ASCII: " << (int)key << endl;
541
542     //ESC 키가 눌러졌다면 프로그램 종료
543     if (key == 27)
544     {
545         exit(0);
546     }
547     else if (key == 43) // 방향키 +키
548     {
549         roll += 1.0;
550     }
551     else if (key == 45) // 방향키 -키
552     {
553         roll -= 1.0;
554     }
555     else if (key == 'q') // q key - 크게, 가깝게
556     {
557         cubeZ += 0.1;
558     }
559     else if (key == 'w') // w key - 작게 ,멀리
560     {
561         cubeZ -= 0.1;
562     }
563     else if (key == 'a') // a key - 위로
564     {
565         cubeY += 0.1;
566     }
567     else if (key == 's') // s key - 아래로
568     {
569         cubeY -= 0.1;
570     }
571     else if (key == 'z') // z key - 오른쪽으로
572     {
573         cubeX += 0.1;
574     }
575     else if (key == 'x') // x key - 왼쪽으로
576     {
577         cubeX -= 0.1;
578     }
579 }
580 void plane(void)
581 {
582     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // Clear The Screen And The Depth Buffer
583     glLoadIdentity();                                   // Reset The View
584
585     //아래 두 줄은 original source, 회전 불가 //angle에 의한 변화는 관찰 가능
586     //glTranslatef(0.0f, .0f, -5.0f);
587     //glRotatef(angle, 1.0f, 1.0f, 0.0f);
588
589     // 이동과 회전을 적용
590     glTranslatef(cubeX, cubeY, cubeZ);
591     glRotatef(pitch, 1.0, 0.0, 0.0);  //x축에 대해 회전
592     glRotatef(yaw, 0.0, 1.0, 0.0);    //y축에 대해 회전
593     glRotatef(roll, 0.0, 0.0, 1.0);   //z축에 대해 회전
594
```

```
        glBindTexture(GL_TEXTURE_2D, TextureIdx[0]); //unbind the TextureIdx

        glBegin(GL_QUADS);
        // Front Face
        glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
        glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);
        glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, 1.0f, 1.0f);
        glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);
        // Back Face
        glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
        glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);
        glTexCoord2f(0.0f, 1.0f); glVertex3f(1.0f, 1.0f, -1.0f);
        glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);
        // Top Face
        glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);
        glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, 1.0f, 1.0f);
        glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, 1.0f, 1.0f);
        glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, 1.0f, -1.0f);
        // Bottom Face
        glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
        glTexCoord2f(0.0f, 1.0f); glVertex3f(1.0f, -1.0f, -1.0f);
        glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);
        glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
        // Right face
        glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);
        glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, 1.0f, -1.0f);
        glTexCoord2f(0.0f, 1.0f); glVertex3f(1.0f, 1.0f, 1.0f);
        glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);
        // Left Face
        glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
        glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
        glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);
        glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);
        glEnd();


        glBindTexture(GL_TEXTURE_2D, 0); //unbind the TextureIdx
}
void display(void)
{
        glClearColor(0.0, 0.0, 0.0, 1.0);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //화면을 지운다. (컬러버퍼와 깊이버퍼)
        glLoadIdentity();
        gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

        plane(); //큐브를 그림
        glutSwapBuffers();
}
void reshape(int w, int h)
{
        glViewport(0, 0, (GLsizei)w, (GLsizei)h); //윈도우 크기로 뷰포인트 설정
        glMatrixMode(GL_PROJECTION);             //이후 연산은 Projection Matrix에 영향을 줌. 카메라로 보이는 장면 설정
        glLoadIdentity();
        gluPerspective(60, (GLfloat)w / (GLfloat)h, 1.0, 100.0);  //Field of view angle(단위 degrees), 윈도우의 aspect ratio, Near와 Far Plane 설정
        glMatrixMode(GL_MODELVIEW);                               //이후 연산은 ModelView Matirx에 영향을 줌. 객체 조작
}
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * /
//* * * * * * * * * * for lab2 - Texture mapping on the 3D shape * * * * * * * * * * // Texture mapping을 사용하려면 여기까지 주석 끝 * * * * * * * * * * /
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * /
```

main()

```
    /////* * * * * (2) Texture mapping on the 3D shape with openGL
    glutInit(&argc, argv);                              // GLUT 초기화
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);       // 더블 버퍼와 깊이 버퍼를 사용하도록 설정, GLUT_RGB=0x00임
    glutInitWindowSize(700, 700);                       // 윈도우의 width와 height
    glutInitWindowPosition(100, 100);                   // 윈도우의 위치 (x,y)

    glutCreateWindow("Texture mapping on the 3D shape - OpenGL Window - 21611591 김난희"); // 윈도우 생성
    glEnable(GL_TEXTURE_2D);                            // Enable Texture Mapping ( NEW )
    glShadeModel(GL_SMOOTH);                            // Enable Smooth Shading
    glClearColor(0.0f, 0.0f, 0.0f, 0.5f);               // Black Background
    glClearDepth(1.0f);                                 // Depth Buffer Setup
    glEnable(GL_DEPTH_TEST);                            // Enables Depth Testing
    glDepthFunc(GL_LEQUAL);                             // The Type Of Depth Testing To Do
    glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);          // Really Nice Perspective

    glutDisplayFunc(display); // 디스플레이 콜백 함수 등록
                              // display함수는 윈도우 처음 생성할 때와 화면 다시 그릴 필요 있을때 호출됨
    glutIdleFunc(display);

    glutReshapeFunc(reshape); // reshape 콜백 함수 등록
                              // reshape함수는 윈도우 처음 생성할 때와 윈도우 크기 변경시 호출된다.

    mSource_Bgr = imread("myimg2.jpg", 1);  //The load my image to opengl TextureIdx

    UploadTexture(mSource_Bgr, TextureIdx[0]); // 직육면체에 그려질 사진을 load texture

    //키보드 콜백 함수 등록, 키보드가 눌러지면 호출된다.
    glutKeyboardFunc(keyboard);
    glutSpecialFunc(special);

    // GLUT event processing loop에 진입함
    // 이 함수는 리턴되지 않기 때문에 다음줄에 있는 코드가 실행되지 않음
    glutMainLoop();

    // ESC Key를 눌러나감 // 실제로는 사용하지 않는 아래 2줄 //동적할당 해제와 같은 역할
    FreeTexture(TextureIdx[0]);           //Free our TextureIdx
    return 0;
}
```

-소스 코드

전체 소스 코드

```
//#include "opencv2/highgui/highgui.hpp"
//#include "opencv2/imgproc/imgproc.hpp"
//#include "opencv2/core/core.hpp"
//#include "opencv2/calib3d/calib3d.hpp"
//#include "opencv2/opencv.hpp"
//
//using namespace cv;
//using namespace std;
//
////* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
////* * * * * * * * * * * for lab2 - Render the disparity in 3D space (point cloud) * * * * * * * * * * * poing cloud를 사용하려면 여기서부터 주석 해제 * * * * * * * * * //
////* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
//template <class T>
//static void fillOcclusionInv_(Mat& src, T invalidvalue)
//{
//      int bb = 1;
//      const int MAX_LENGTH = src.cols*0.8;
//#pragma omp parallel for // openMP를 이용한 병렬 처리
//      for (int j = bb; j < src.rows - bb; j++)
//      {
//              T* s = src.ptr<T>(j);
//              const T st = s[0];
//              const T ed = s[src.cols - 1];
```

```
//              s[0] = 0;
//              s[src.cols - 1] = 0;
//              for (int i = 0; i < src.cols; i++)
//              {
//                      if (s[i] == invalidvalue)
//                      {
//                              int t = i;
//                              do
//                              {
//                                      t++;
//                                      if (t > src.cols - 1)break;
//                              } while (s[t] == invalidvalue);
//
//                              const T dd = max(s[i - 1], s[t]);
//                              if (t - i > MAX_LENGTH)
//                              {
//                                      for (int n = 0; n < src.cols; n++)
//                                      {
//                                              s[n] = invalidvalue;
//                                      }
//                              }
//                              else
//                              {
//                                      for (; i < t; i++)
//                                      {
//                                              s[i] = dd;
//                                      }
//                              }
//                      }
//              }
//      }
//}
//template <class T>
//static void fillOcclusion_(Mat& src, T invalidvalue)
//{
//      int bb = 1;
//      const int MAX_LENGTH = src.cols*0.5;
//#pragma omp parallel for // openMP를 이용한 병렬 처리
//      for (int j = bb; j < src.rows - bb; j++)
//      {
//              T* s = src.ptr<T>(j);
//              const T st = s[0];
//              const T ed = s[src.cols - 1];
//              s[0] = 255;
//              s[src.cols - 1] = 255;
//              for (int i = 0; i < src.cols; i++)
//              {
//                      if (s[i] <= invalidvalue)
//                      {
//                              int t = i;
//                              do
//                              {
//                                      t++;
//                                      if (t > src.cols - 1)break;
//                              } while (s[t] <= invalidvalue);
//
//                              const T dd = min(s[i - 1], s[t]);
//                              if (t - i > MAX_LENGTH)
//                              {
//                                      for (int n = 0; n < src.cols; n++)
//                                      {
//                                              s[n] = invalidvalue;
```

```
//                                      }
//                              }
//                              else
//                              {
//                                      for (; i < t; i++)
//                                      {
//                                              s[i] = dd;
//                                      }
//                              }
//                      }
//              }
//      }
//}
//void fillOcclusion(Mat& src, int invalidvalue, bool isInv = false)
                  //{     // 불규칙한 값을 조작하여 불규칙한 것을 최소화하게 다시 채움
//      if (isInv)
//      {
//              if (src.type() == CV_8U)
//              {
//                      fillOcclusionInv_<uchar>(src, (uchar)invalidvalue);
//              }
//              else if (src.type() == CV_16S)
//              {
//                      fillOcclusionInv_<short>(src, (short)invalidvalue);
//              }
//              else if (src.type() == CV_16U)
//              {
//                      fillOcclusionInv_<unsigned short>(src, (unsigned short)invalidvalue);
//              }
//              else if (src.type() == CV_32F)
//              {
//                      fillOcclusionInv_<float>(src, (float)invalidvalue);
//              }
//      }
//      else
//      {
//              if (src.type() == CV_8U)
//              {
//                      fillOcclusion_<uchar>(src, (uchar)invalidvalue);
//              }
//              else if (src.type() == CV_16S)
//              {
//                      fillOcclusion_<short>(src, (short)invalidvalue);
//              }
//              else if (src.type() == CV_16U)
//              {
//                      fillOcclusion_<unsigned short>(src, (unsigned short)invalidvalue);
//              }
//              else if (src.type() == CV_32F)
//              {
//                      fillOcclusion_<float>(src, (float)invalidvalue);
//              }
//      }
//}
//void eular2rot(double yaw, double pitch, double roll, Mat& dest)
//{
//      double theta = yaw / 180.0*CV_PI;
//      double pusai = pitch / 180.0*CV_PI;
//      double phi = roll / 180.0*CV_PI;
//
//      double datax[3][3] = { {1.0,0.0,0.0},
//      {0.0,cos(theta),-sin(theta)},
```

```
//	{0.0,sin(theta),cos(theta)} };
//	double datay[3][3] = { {cos(pusai),0.0,sin(pusai)},
//	{0.0,1.0,0.0},
//	{-sin(pusai),0.0,cos(pusai)} };
//	double dataz[3][3] = { {cos(phi),-sin(phi),0.0},
//	{sin(phi),cos(phi),0.0},
//	{0.0,0.0,1.0} };
//	Mat Rx(3, 3, CV_64F, datax);
//	Mat Ry(3, 3, CV_64F, datay);
//	Mat Rz(3, 3, CV_64F, dataz);
//	Mat rr = Rz * Rx*Ry;
//	rr.copyTo(dest);
//}
//
//void lookat(Point3d from, Point3d to, Mat& destR)
//{
//	double x = (to.x - from.x);
//	double y = (to.y - from.y);
//	double z = (to.z - from.z);
//
//	double pitch = asin(x / sqrt(x*x + z * z)) / CV_PI * 180.0;
//	double yaw = asin(-y / sqrt(y*y + z * z)) / CV_PI * 180.0;
//
//	eular2rot(yaw, pitch, 0, destR);
//}
//template <class T>
//static void projectImagefromXYZ_(Mat& image, Mat& destimage, Mat& disp, Mat& destdisp, Mat& xyz, Mat& R,
Mat& t, Mat& K, Mat& dist, Mat& mask, bool isSub)
//{
//	if (destimage.empty())destimage = Mat::zeros(Size(image.size()), image.type());
//	if (destdisp.empty())destdisp = Mat::zeros(Size(image.size()), disp.type());
//
//	vector<Point2f> pt;
//	if (dist.empty()) dist = Mat::zeros(Size(5, 1), CV_32F);
//	cv::projectPoints(xyz, R, t, K, dist, pt);
//
//	destimage.setTo(0);
//	destdisp.setTo(0);
//
//#pragma omp parallel for
//	for (int j = 1; j < image.rows - 1; j++)
//	{
//		int count = j * image.cols;
//		uchar* img = image.ptr<uchar>(j);
//		uchar* m = mask.ptr<uchar>(j);
//		for (int i = 0; i < image.cols; i++, count++)
//		{
//			int x = (int)(pt[count].x + 0.5);
//			int y = (int)(pt[count].y + 0.5);
//			if (m[i] == 255)continue;
//			if (pt[count].x >= 1 && pt[count].x < image.cols - 1 && pt[count].y >= 1 &&
pt[count].y < image.rows - 1)
//			{
//				short v = destdisp.at<T>(y, x);
//				if (v < disp.at<T>(j, i))
//				{
//					destimage.at<uchar>(y, 3 * x + 0) = img[3 * i + 0];
//					destimage.at<uchar>(y, 3 * x + 1) = img[3 * i + 1];
//					destimage.at<uchar>(y, 3 * x + 2) = img[3 * i + 2];
//					destdisp.at<T>(y, x) = disp.at<T>(j, i);
//
//					if (isSub)
```

```
//                                      {
//                                              if ((int)pt[count + image.cols].y - y > 1 &&
(int)pt[count + 1].x - x > 1)
//                                              {
//                                                      destimage.at<uchar>(y, 3 * x + 3) = img[3 * i +
0];
//                                                      destimage.at<uchar>(y, 3 * x + 4) = img[3 * i +
1];
//                                                      destimage.at<uchar>(y, 3 * x + 5) = img[3 * i +
2];
//
//                                                      destimage.at<uchar>(y + 1, 3 * x + 0) = img[3 *
i + 0];
//                                                      destimage.at<uchar>(y + 1, 3 * x + 1) = img[3 *
i + 1];
//                                                      destimage.at<uchar>(y + 1, 3 * x + 2) = img[3 *
i + 2];
//
//                                                      destimage.at<uchar>(y + 1, 3 * x + 3) = img[3 *
i + 0];
//                                                      destimage.at<uchar>(y + 1, 3 * x + 4) = img[3 *
i + 1];
//                                                      destimage.at<uchar>(y + 1, 3 * x + 5) = img[3 *
i + 2];
//
//                                                      destdisp.at<T>(y, x + 1) = disp.at<T>(j, i);
//                                                      destdisp.at<T>(y + 1, x) = disp.at<T>(j, i);
//                                                      destdisp.at<T>(y + 1, x + 1) = disp.at<T>(j, i);
//                                              }
//                                              else if ((int)pt[count - image.cols].y - y < -1 &&
(int)pt[count - 1].x - x < -1)
//                                              {
//                                                      destimage.at<uchar>(y, 3 * x - 3) = img[3 * i +
0];
//                                                      destimage.at<uchar>(y, 3 * x - 2) = img[3 * i +
1];
//                                                      destimage.at<uchar>(y, 3 * x - 1) = img[3 * i +
2];
//
//                                                      destimage.at<uchar>(y - 1, 3 * x + 0) = img[3 *
i + 0];
//                                                      destimage.at<uchar>(y - 1, 3 * x + 1) = img[3 *
i + 1];
//                                                      destimage.at<uchar>(y - 1, 3 * x + 2) = img[3 *
i + 2];
//
//                                                      destimage.at<uchar>(y - 1, 3 * x - 3) = img[3 *
i + 0];
//                                                      destimage.at<uchar>(y - 1, 3 * x - 2) = img[3 *
i + 1];
//                                                      destimage.at<uchar>(y - 1, 3 * x - 1) = img[3 *
i + 2];
//
//                                                      destdisp.at<T>(y, x - 1) = disp.at<T>(j, i);
//                                                      destdisp.at<T>(y - 1, x) = disp.at<T>(j, i);
//                                                      destdisp.at<T>(y - 1, x - 1) = disp.at<T>(j, i);
//                                              }
//                                              else if ((int)pt[count + 1].x - x > 1)
//                                              {
//                                                      destimage.at<uchar>(y, 3 * x + 3) = img[3 * i +
0];
//                                                      destimage.at<uchar>(y, 3 * x + 4) = img[3 * i +
```

```
1];
//                                                      destimage.at<uchar>(y, 3 * x + 5) = img[3 * i +
2];
//
//                                                      destdisp.at<T>(y, x + 1) = disp.at<T>(j, i);
//                                              }
//                                              else if ((int)pt[count - 1].x - x < -1)
//                                              {
//                                                      destimage.at<uchar>(y, 3 * x - 3) = img[3 * i +
0];
//                                                      destimage.at<uchar>(y, 3 * x - 2) = img[3 * i +
1];
//                                                      destimage.at<uchar>(y, 3 * x - 1) = img[3 * i +
2];
//
//                                                      destdisp.at<T>(y, x - 1) = disp.at<T>(j, i);
//                                              }
//                                              else if ((int)pt[count + image.cols].y - y > 1)
//                                              {
//                                                      destimage.at<uchar>(y + 1, 3 * x + 0) = img[3 *
i + 0];
//                                                      destimage.at<uchar>(y + 1, 3 * x + 1) = img[3 *
i + 1];
//                                                      destimage.at<uchar>(y + 1, 3 * x + 2) = img[3 *
i + 2];
//
//                                                      destdisp.at<T>(y + 1, x) = disp.at<T>(j, i);
//                                              }
//                                              else if ((int)pt[count - image.cols].y - y < -1)
//                                              {
//                                                      destimage.at<uchar>(y - 1, 3 * x + 0) = img[3 *
i + 0];
//                                                      destimage.at<uchar>(y - 1, 3 * x + 1) = img[3 *
i + 1];
//                                                      destimage.at<uchar>(y - 1, 3 * x + 2) = img[3 *
i + 2];
//
//                                                      destdisp.at<T>(y - 1, x) = disp.at<T>(j, i);
//                                              }
//                                      }
//                              }
//                      }
//              }
//      }
//
//      if (isSub)
//      {
//              Mat image2;
//              Mat disp2;
//              destimage.copyTo(image2);
//              destdisp.copyTo(disp2);
//              const int BS = 1;
//#pragma omp parallel for
//              for (int j = BS; j < image.rows - BS; j++)
//              {
//                      uchar* img = destimage.ptr<uchar>(j);
//                      T* m = disp2.ptr<T>(j);
//                      T* dp = destdisp.ptr<T>(j);
//                      for (int i = BS; i < image.cols - BS; i++)
//                      {
//                              if (m[i] == 0)
//                              {
```

```
//                                      int count = 0;
//                                      int d = 0;
//                                      int r = 0;
//                                      int g = 0;
//                                      int b = 0;
//                                      for (int l = -BS; l <= BS; l++)
//                                      {
//                                              T* dp2 = disp2.ptr<T>(j + l);
//                                              uchar* img2 = image2.ptr<uchar>(j + l);
//                                              for (int k = -BS; k <= BS; k++)
//                                              {
//                                                      if (dp2[i + k] != 0)
//                                                      {
//                                                              count++;
//                                                              d += dp2[i + k];
//                                                              r += img2[3 * (i + k) + 0];
//                                                              g += img2[3 * (i + k) + 1];
//                                                              b += img2[3 * (i + k) + 2];
//                                                      }
//                                              }
//                                      }
//                                      if (count != 0)
//                                      {
//                                              double div = 1.0 / count;
//                                              dp[i] = d * div;
//                                              img[3 * i + 0] = r * div;
//                                              img[3 * i + 1] = g * div;
//                                              img[3 * i + 2] = b * div;
//                                      }
//                              }
//                      }
//              }
//      }
//}
//void projectImagefromXYZ(Mat& image, Mat& destimage, Mat& disp, Mat& destdisp, Mat& xyz, Mat& R, Mat& t,
Mat& K, Mat& dist, bool isSub = true, Mat& mask = Mat())
//{
//      if (mask.empty())mask = Mat::zeros(image.size(), CV_8U);
//      if (disp.type() == CV_8U)
//      {
//              projectImagefromXYZ_<unsigned char>(image, destimage, disp, destdisp, xyz, R, t, K, dist,
mask, isSub);
//      }
//      else if (disp.type() == CV_16S)
//      {
//              projectImagefromXYZ_<short>(image, destimage, disp, destdisp, xyz, R, t, K, dist, mask,
isSub);
//      }
//      else if (disp.type() == CV_16U)
//      {
//              projectImagefromXYZ_<unsigned short>(image, destimage, disp, destdisp, xyz, R, t, K, dist,
mask, isSub);
//      }
//      else if (disp.type() == CV_32F)
//      {
//              projectImagefromXYZ_<float>(image, destimage, disp, destdisp, xyz, R, t, K, dist, mask,
isSub);
//      }
//      else if (disp.type() == CV_64F)
//      {
//              projectImagefromXYZ_<double>(image, destimage, disp, destdisp, xyz, R, t, K, dist, mask,
isSub);
```

```
//      }
//}
//Mat makeQMatrix(Point2d image_center, double focal_length, double baseline)
//{
//      Mat Q = Mat::eye(4, 4, CV_64F);
//      Q.at<double>(0, 3) = -image_center.x;
//      Q.at<double>(1, 3) = -image_center.y;
//      Q.at<double>(2, 3) = focal_length;
//      Q.at<double>(3, 3) = 0.0;
//      Q.at<double>(2, 2) = 0.0;
//      Q.at<double>(3, 2) = 1.0 / baseline;
//
//      return Q;
//}
//void stereoTest()
//{
//      //(1) Reading L&R images and estimating disparity map by semi-global block matching.
//      Mat image = imread("l.png", 1); // 스테레오 이미지를 읽어드림
//      Mat imageR = imread("r.png", 1);
//      Mat destimage;
//
//      /*int resize_k = 4; // 사진 크기가 크면 사이즈 바꾸기
//      resize(image, image, Size(image.size().width / resize_k, image.size().height / resize_k));
//      resize(imageR, imageR, Size(imageR.size().width / resize_k, imageR.size().height / resize_k));*/
//
//      StereoSGBM sgbm(1, 16 * 2, 3, 200, 255, 1, 0, 0, 0, 0, true); //SGBM으로 시차 화상을 계산함
//      Mat disp;
//      Mat destdisp;
//      Mat dispshow;
//      sgbm(image, imageR, disp); // 계산 결과의 시차는 최소 시차 -16의 값이 불규칙으로 할당됨
//      fillOcclusion(disp, 16); // 불규칙 값을 그럴싸한 값으로 재할당하는 함수
//
//      //(2)make Q matrix and reproject pixels into 3D space // 매트릭스의 구축과 시차 이미지의 3차원
공간에 투영
//      const double focal_length = 598.57;
//      const double baseline = 14.0;
//      // 3차원 공간에 투영하기 위해 Q 행렬을 사용 // 스테레오 카메라를 보정하여 Q 행렬을 얻기 위해
CV::streoRevtify에서 보완함
//      Mat Q = makeQMatrix(Point2d((image.cols - 1.0) / 2.0, (image.rows - 1.0) / 2.0), focal_length,
baseline * 16);
//
//      Mat depth;
//      cv::reprojectImageTo3D(disp, depth, Q); // 시차를 3차원 공간에 투영함
//      Mat xyz = depth.reshape(3, depth.size().area()); // 3채널의 float 배열에 저장됨
//
//      //(3) camera setting // 카메라 설정 // 행렬 설정
//      Mat K = Mat::eye(3, 3, CV_64F);
//      K.at<double>(0, 0) = focal_length;
//      K.at<double>(1, 1) = focal_length;
//      K.at<double>(0, 2) = (image.cols - 1.0) / 2.0;
//      K.at<double>(1, 2) = (image.rows - 1.0) / 2.0;
//
//      Mat dist = Mat::zeros(5, 1, CV_64F);
//      Mat R = Mat::eye(3, 3, CV_64F);
//      Mat t = Mat::zeros(3, 1, CV_64F);
//
//      Point3d viewpoint(0.0, 0.0, baseline * 10); // 관점
//      Point3d lookatpoint(0.0, 0.0, -baseline * 10.0); // 주시점
//      const double step = baseline;
//      int key = 0;
//      bool isSub = true;
//      //(4) rendering loop // 3차원 좌표에서 새로운 관점의 이미지에 점군을 투영
```

```
//      while (key != 27)
//      {
//              lookat(viewpoint, lookatpoint, R); // 가상 카메라의 방향이 업데이트됨
//              t.at<double>(0, 0) = viewpoint.x;
//              t.at<double>(1, 0) = viewpoint.y;
//              t.at<double>(2, 0) = viewpoint.z;
//
//              cout << t << endl;  // 좌표 출력
//              t = R * t;
//
//              //(5) projecting 3D point cloud to image.
//              projectImagefromXYZ(image, destimage, disp, destdisp, xyz, R, t, K, dist, isSub);
//
//              destdisp.convertTo(dispshow, CV_8U, 0.5);
//              imshow("depth - 21611591 김난희", dispshow);
//              imshow("image - 21611591 김난희", destimage);
//
//              // 위치 방향 컨트롤
//              if (key == 'f') // 양자화 억제 필터의 온 오프 전환
//              {                                // default 값은 ON
//                      isSub = isSub ? false : true;
//              }
//              if (key == 'a') // 아래로
//              {
//                      viewpoint.y += step;
//              }
//              if (key == 's') // 위로
//              {
//                      viewpoint.y -= step;
//              }
//              if (key == 'z') //오른쪽으로
//              {
//                      viewpoint.x += step;
//              }
//              if (key == 'x') //왼쪽으로
//              {
//                      viewpoint.x -= step;
//              }
//              if (key == 'q') // 멀리
//              {
//                      viewpoint.z += step;
//              }
//              if (key == 'w') // 가까이
//              {
//                      viewpoint.z -= step;
//              }
//              key = waitKey(1); // 실시간으로 key 값을 출력하기 위해 1이 들어감
//                                                // 0은 바뀔 때마다 키값을 출력하도록 함
//      }
//}
////* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
////* * * * * * * * * * for lab2 - Render the disparity in 3D space (point cloud) * * * * * * * * * * poing cloud를 사용하려면 여기까지 주석 끝 * * * * * * * * * * //
////* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
//* * * * * * * * * * for lab2 - Texture mapping on the 3D shape * * * * * * * * * * Texture mapping을 사용하려면 여기서부터 주석해제 * * * * * * * * * * * * * * //
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * //

///openGL 관련
#include <windows.h>
#include <glut.h>

///openCV 관련
#include "opencv2/core/core.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"

#include <strstream>
#include <functional>
#include <stdio.h>

using namespace cv;

GLuint TextureIdx[1]; //the array for our TextureIdx
Mat mSource_Bgr;   //texture mapping할 image
GLfloat angle = 45.0; //original source의 cube 각도

/// keyboard로 3d 큐브 움직이기
// (1) 큐브 위치
float cubeX = 0.0;
float cubeY = 0.0;
float cubeZ = -4.0;
// (2) 회전
float pitch = 0.0;
float yaw = 0.0;
float roll = 0.0;

int UploadTexture(Mat image, GLuint &TextureIdx) ///The my Image to OpenGl TextureIdx function
{
	if (image.empty())
		return -1;
	glGenTextures(1, &TextureIdx);
	glBindTexture(GL_TEXTURE_2D, TextureIdx); //bind the TextureIdx to it's array
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
	glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
	glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
	glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, image.cols, image.rows, 0, GL_BGR_EXT, GL_UNSIGNED_BYTE, 
image.data);
	return 0;
}
void FreeTexture(GLuint TextureIdx) ///texture mapping 후 동적할당 해제와 같음
{
	glDeleteTextures(1, &TextureIdx);
}
void special(int key, int x, int y) // 방향키에 의해 회전을 입력
{
	if (key == GLUT_KEY_UP)
	{
		pitch += 1.0;
	}
	else if (key == GLUT_KEY_DOWN)
	{
		pitch -= 1.0;
	}
	else if (key == GLUT_KEY_RIGHT)
	{
		yaw += 1.0;
	}
```

```
        else if (key == GLUT_KEY_LEFT)
        {
                yaw -= 1.0;
        }
}
void keyboard(unsigned char key, int x, int y) // 상하 좌우 가깝게멀리 입력 //* 주의할 점: 대문자 영문은 안됨
{
        //cout << "다음 키가 눌러졌습니다. ₩"" << key << "₩" ASCII: " << (int)key << endl;

        //ESC 키가 눌러졌다면 프로그램 종료
        if (key == 27)
        {
                exit(0);
        }
        else if (key == 43) // 방향키 +키
        {
                roll += 1.0;
        }
        else if (key == 45) // 방향키 -키
        {
                roll -= 1.0;
        }
        else if (key == 'q') // q key - 크게, 가깝게
        {
                cubeZ += 0.1;
        }
        else if (key == 'w') // w key - 작게 ,멀리
        {
                cubeZ -= 0.1;
        }
        else if (key == 'a') // a key - 위로
        {
                cubeY += 0.1;
        }
        else if (key == 's') // s key - 아래로
        {
                cubeY -= 0.1;
        }
        else if (key == 'z') // z key - 오른쪽으로
        {
                cubeX += 0.1;
        }
        else if (key == 'x') // x key - 왼쪽으로
        {
                cubeX -= 0.1;
        }
}
void plane(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // Clear The Screen And The Depth Buffer
        glLoadIdentity();                                   // Reset The View

        //아래 두 줄은 original source, 회전 불가 //angle에 의한 변화는 관찰 가능
        //glTranslatef(0.0f, .0f, -5.0f);
        //glRotatef(angle, 1.0f, 1.0f, 0.0f);

        // 이동과 회전을 적용
        glTranslatef(cubeX, cubeY, cubeZ);
        glRotatef(pitch, 1.0, 0.0, 0.0);  //x축에 대해 회전
        glRotatef(yaw, 0.0, 1.0, 0.0);    //y축에 대해 회전
        glRotatef(roll, 0.0, 0.0, 1.0);   //z축에 대해 회전
```

```
	glBindTexture(GL_TEXTURE_2D, TextureIdx[0]); //unbind the TextureIdx

	glBegin(GL_QUADS);
	// Front Face
	glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
	glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);
	glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, 1.0f, 1.0f);
	glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);
	// Back Face
	glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
	glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);
	glTexCoord2f(0.0f, 1.0f); glVertex3f(1.0f, 1.0f, -1.0f);
	glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);
	// Top Face
	glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);
	glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, 1.0f, 1.0f);
	glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, 1.0f, 1.0f);
	glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, 1.0f, -1.0f);
	// Bottom Face
	glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
	glTexCoord2f(0.0f, 1.0f); glVertex3f(1.0f, -1.0f, -1.0f);
	glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);
	glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
	// Right face
	glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);
	glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, 1.0f, -1.0f);
	glTexCoord2f(0.0f, 1.0f); glVertex3f(1.0f, 1.0f, 1.0f);
	glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);
	// Left Face
	glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
	glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
	glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);
	glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);
	glEnd();


	glBindTexture(GL_TEXTURE_2D, 0); //unbind the TextureIdx
}
void display(void)
{
	glClearColor(0.0, 0.0, 0.0, 1.0);
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //화면을 지운다. (컬러버퍼와 깊이버퍼)
	glLoadIdentity();
	gluLookAt(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

	plane(); //큐브를 그림
	glutSwapBuffers();
}
void reshape(int w, int h)
{
	glViewport(0, 0, (GLsizei)w, (GLsizei)h); //윈도우 크기로 뷰포인트 설정
	glMatrixMode(GL_PROJECTION);			//이후 연산은 Projection Matrix에 영향을 줌.
카메라로 보이는 장면 설정
	glLoadIdentity();
	gluPerspective(60, (GLfloat)w / (GLfloat)h, 1.0, 100.0);  //Field of view angle(단위 degrees),
윈도우의 aspect ratio, Near와 Far Plane 설정
	glMatrixMode(GL_MODELVIEW);
//이후 연산은 ModelView Matirx에 영향을 줌. 객체 조작
}
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * //
```

```
//* * * * * * * * * * * for lab2 - Texture mapping on the 3D shape * * * * * * * * * * // Texture mapping을 
사용하려면 여기까지 주석 끝 * * * * * * * * * * //
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * //


int main(int argc, char** argv)
{
	////* * * * * * * * * * lab 1. block matching based disparity * * * * * * * * * *//
	////load stereo images -> 영상 2장
	//Mat l = imread("im2.ppm"); // left  영상
	//Mat r = imread("im6.ppm"); // right 영상 right_un.png

	////image resize because big size
	////resize(l, l, size(l.size().width / 4, l.size().height / 4));
	////resize(r, r, size(r.size().width / 4, r.size().height / 4));

	//imshow("Left image - 21611591 김난희", l);
	//imshow("Right image - 21611591 김난희", r);

	//// set block matching parameters // 파라미터 세팅
	//CvStereoBMState *BMState = cvCreateStereoBMState(); // 클래스를 이용해 객체 만듦 -> 함수가 주소 
return한 것을 받음
	//BMState->preFilterSize = 5;  // brightness correction, 5x5 ~ 21x21 // 객체는 ., pointer는 그 
주소로 가서 화살표로 함
	//BMState->preFilterCap = 5;      // removed region after prefilter
	//BMState->SADWindowSize = 17; // sad: sum of absolute difference, window size (5x5.... 21x21) // 
윈도우 사이즈 11(support 영역) 좀 크게함
	////support 영역 11->5 모호성이 커져서 오히려 잘 안나옴 // 25는 두루뭉실하지만(정밀도 떨어짐) 
matching은 더 잘함
	//BMState->minDisparity = 1;   // minimum disparity [pixel] // 찾고자하는 minimum 값
	//BMState->numberOfDisparities = 32; // maximum disparity [pixel] // 1~32까지 pixel 봄
	//BMState->textureThreshold = 10;  // minimum allowed
	//BMState->uniquenessRatio = 5;    // uniqueness  (removing false matching)

	//// convert color space
	//Mat Left, Right;
	//cvtColor(l, Left, CV_RGB2GRAY); // gray로 변환
	//cvtColor(r, Right, CV_RGB2GRAY);

	//// type conversion: mat to iplimage
	//IplImage *left, *right;
	//left = &IplImage(Left);
	//right = &IplImage(Right);

	//CvMat* disparity = cvCreateMat(Left.rows, Left.cols, CV_16SC1);
	//CvMat* disparity_img = cvCreateMat(Left.rows, Left.cols, CV_8UC1); // 1~32 -> 0~255 밝기를 눈으로 
볼 수 있게 바꿔줌

	//// run algorithm
	//cvFindStereoCorrespondenceBM(left, right, disparity, BMState);
	//cvNormalize(disparity, disparity_img, 0, 255, CV_MINMAX);  //normalize to display

	//// show results // stereo matching show
	//cvShowImage("disparity - 21611591 김난희", disparity_img);

	//waitKey(0);

	//* * * * * * * * * * lab 2. 3d rendering * * * * * * * * * *//
	///* * * * * (1) render the disparity in 3d space (point cloud)
	// sample of stereo matching and 3d point cloud rendering
	//stereoTest();
```

```
        /////* * * * * (2) Texture mapping on the 3D shape with openGL
        glutInit(&argc, argv);                                                  // GLUT 초기화
        glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);   // 더블 버퍼와 깊이 버퍼를 사용하도록 설정,
GLUT_RGB=0x00임
        glutInitWindowSize(700, 700);                                   // 윈도우의 width와 height
        glutInitWindowPosition(100, 100);                       // 윈도우의 위치 (x,y)

        glutCreateWindow("Texture mapping on the 3D shape - OpenGL Window - 21611591 김난희"); // 윈도우
생성
        glEnable(GL_TEXTURE_2D);                        // Enable Texture Mapping ( NEW )
        glShadeModel(GL_SMOOTH);                        // Enable Smooth Shading
        glClearColor(0.0f, 0.0f, 0.0f, 0.5f);           // Black Background
        glClearDepth(1.0f);                             // Depth Buffer Setup
        glEnable(GL_DEPTH_TEST);                        // Enables Depth Testing
        glDepthFunc(GL_LEQUAL);                         // The Type Of Depth Testing To Do
        glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);          // Really Nice Perspective

        glutDisplayFunc(display); // 디스플레이 콜백 함수 등록
                                                                // display함수는 윈도우 처음 생성할 때와 화면
다시 그릴 필요 있을때 호출됨
        glutIdleFunc(display);

        glutReshapeFunc(reshape); // reshape 콜백 함수 등록
                                                                // reshape함수는 윈도우 처음 생성할 때와 윈도우
크기 변경시 호출된다.

        mSource_Bgr = imread("earth.png", 1);  //The load my image to opengl Textureldx

        UploadTexture(mSource_Bgr, Textureldx[0]); // 직육면체에 그려질 사진을 load texture

        //키보드 콜백 함수 등록, 키보드가 눌러지면 호출된다.
        glutKeyboardFunc(keyboard);
        glutSpecialFunc(special);

        // GLUT event processing loop에 진입함
        // 이 함수는 리턴되지 않기 때문에 다음줄에 있는 코드가 실행되지 않음
        glutMainLoop();

        // ESC Key를 눌러나감 // 실제로는 사용하지 않는 아래 2줄 //동적할당 해제와 같은 역할
        FreeTexture(Textureldx[0]);          //Free our Textureldx
        return 0;
}
```

opengl 테스트 소스 코드

```
#include <glut.h>
#include <glu.h>

void MyDisplay() {
        glClear(GL_COLOR_BUFFER_BIT);
        glBegin(GL_POLYGON);
        glVertex3f(-0.5, -0.5, 0.0);
        glVertex3f(0.5, -0.5, 0.0);
        glVertex3f(0.5, 0.5, 0.0);
        glVertex3f(-0.5, 0.5, 0.0);
        glEnd();
        glFlush();
}
int main() {
        glutCreateWindow("OpenGL Initialize Test - 21611591 김난희");
```

```
        glutDisplayFunc(MyDisplay);
        glutMainLoop();
        return 0;
}
```

참고 사이트

Lab1

Stereo datasets - 2003 datasets의 우산 사진

http://vision.middlebury.edu/stereo/data/

http://vision.middlebury.edu/stereo/data/scenes2003/

Lab2

-Render the disparity in 3D space (point cloud)

http://opencv.jp/opencv2-x-samples/point-cloud-rendering

-Texture mapping on the 3D shape

opengl – texture mapping

https://answers.opencv.org/question/54499/place-texture-using-a-given-uv-buffer/

http://cgvr.cs.uni-bremen.de/teaching/cg_literatur/Cube_map_tutorial/cube_map.html

opengl – keyboard로 3d 큐브 움직이기

https://webnautes.tistory.com/1096

opengl 설치

https://eestrella.tistory.com/22

opengl 설치 파일

https://koreatech.tistory.com/entry/opengl-%ED%99%98%EA%B2%BD%EC%84%A4%EC%A0%95%ED%95%98%EA%B8%B0-%EC%B5%9C%EC%8B%A0%EB%B2%84%EC%A0%84?category=717951

opengl 설치 테스트 소스코드

https://eine.tistory.com/entry/Visual-Studio-2017%EC%97%90%EC%84%9C-openGL-%EC%84%A4%EC%A0%95-%ED%9B%84-%EC%98%88%EC%A0%9C-%EC%8B%A4%ED%96%89%ED%95%B4%EB%B3%B4%EA%B8%B0