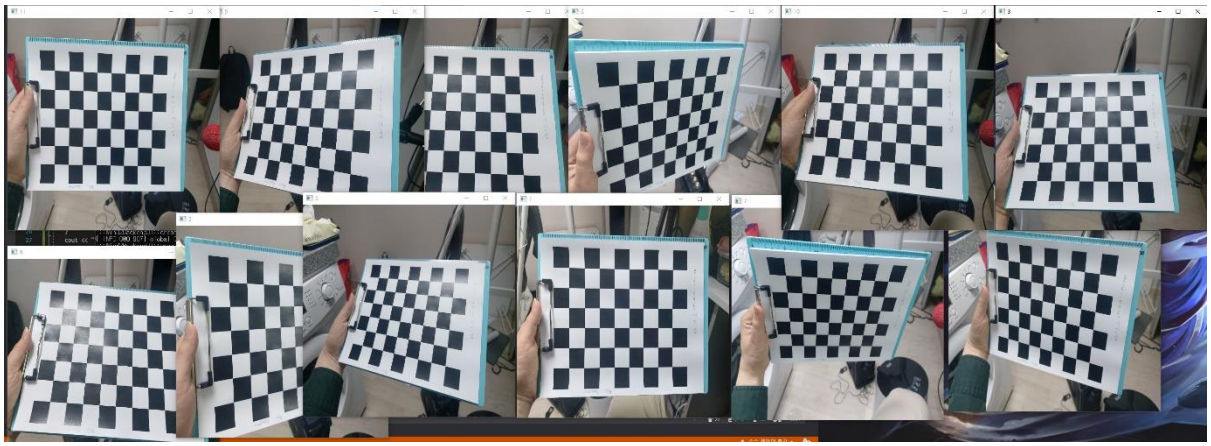
멀티미디어시스템 과제#2

2016114463 이인영

1. chessboard 10장 영상을 opencv로 10개의 윈도우로 출력하고 화면 capture - chessboard 아래쪽에 본인의 이름이 적혀져 있어야 함 - 10개 윈도우를 컴퓨터 화면을 동시에 출력하면 글자가 작게 보일 수 있느나 식별 가능하면 됨



2. program source file 1개, 보고서에 capture 또는 ctrl^c, ctrl^v (.h 파일 등은 필요없음)

(다음장 참고)

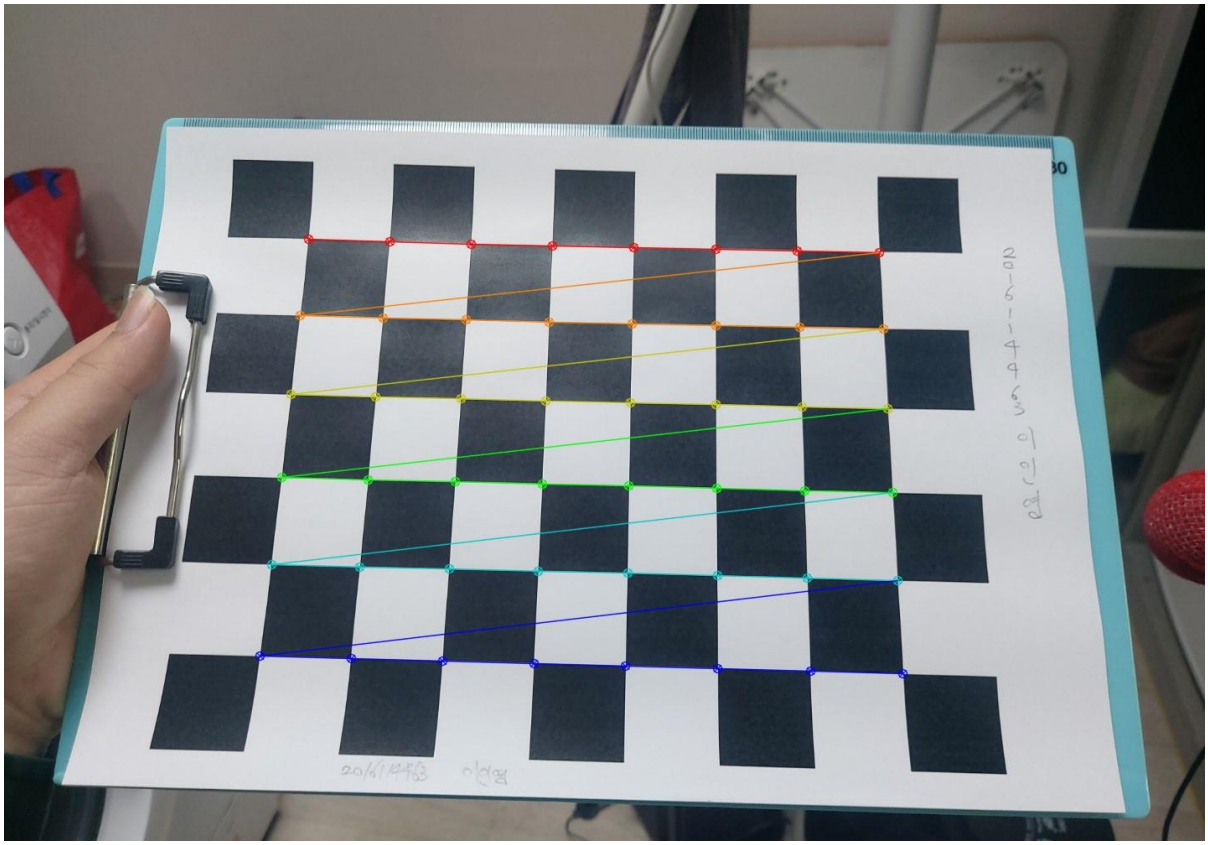
3) 보정결과 화면 capture (image file) (내부값, 왜곡값, 보정오차값)

```
------------------------
cameraMatrix : [1079.202790164676, 0, 725.4320454978556;
 0, 1080.952529327568, 536.9088678614257;
 0, 0, 1]
-----------

distCoeffs : [-0.03284694677907634, 0.6240409995573205, 0.002956633483741982, 0.008409533190210517, -1.967767309711385]
-----------

rvecs : [0.00809515026654148, -0.05873888734182337, 0.009919701969504452;
 0.008505879869603741, -0.4064147615352987, -0.05213407164646773;
 0.2540393579276745, 0.2835820422095629, 0.1202656009912893;
 -0.3981263825062115, 0.2750262491252882, 0.1710843143191968;
 0.5053113478476663, -0.5099085141385107, -0.04188727151741442;
 0.6771212854917478, -0.1234986967297477, 0.06941341585795907;
 -0.2182658040327461, 0.3721125987119364, 0.08782920097728329;
 -0.2250030572832916, 0.1390125243152997, 0.06027630616817576;
 0.05465839178067, -0.04157504892034888, 0.02117492182460531;
 -0.2514908199852683, 0.0008796144538049979, 0.02467590209991481]
-----------

tvecs : [-97.53421016660101, -58.11505937927104, 285.1508505539224;
 -81.13888093156375, -54.24658204718946, 238.0223594511801;
 -99.98122059684917, -70.16957044533375, 338.3287742547222;
 -81.14315426128742, -62.82832963683275, 414.5802314056893;
 -82.85619883370141, -42.27378736955595, 233.3315498545618;
 -91.84477517095226, -49.84031773452759, 278.5206461767751;
 -108.6862141774932, -79.44144777883537, 351.1101914910828;
 -111.4842001729432, -84.58422588452719, 334.5246062912623;
 -112.6945872689217, -75.29911643427525, 302.05723672982;
 -99.73301237996283, -71.14860710864869, 300.4785937821625]
-----------
```

```cpp
#include <string>
#include <vector>
#include <iostream>
#include "opencv2/opencv.hpp"

using namespace std;
using namespace cv;

// This program was made along with c++ 20 rule

int main() {
    Mat p_img;
    Mat p_imgsmall;
    vector<string> images;
    string path = "./images/";
    glob(path, images, false);
    cout << "로드 갯수 : " << images.size() << endl;
    if (images.size() == 0)
        cout << "이미지가 존재하지 않습니다.\n" << endl;

    for (int cnt = 0; cnt < images.size(); cnt++)
    {
        p_img = imread(images[cnt]);
        resize(p_img, p_imgsmall, Size(200, 200));
        imshow(to_string(cnt + 1), p_imgsmall);
        printf("%d 번째 이미지 실행 완료. \n", cnt + 1);
    }
    cout << "\n" << endl;
    waitKey();
    return 0;
}
```

```
success
success
false
success
success
success
success
false
success
success
success
success
----------------------
정상 검출된 이미지 개수 :10
----------------------
```

12개의 이미지 中, 10개의 이미지만 정상적으로 검출이 되었다.



4) 보정결과 .txt - 카메라 내부값, 렌즈왜곡값, 보정오차값 - 카메라외부값 (rotation, translation)
(보정판 영상 1개당 외부가 1개씩 있음. 모든 보정판 에 대한 결과값 출력)

---

cameraMatrix : [1079.202790164676, 0, 725.4320454978556;

0, 1080.952529327568, 536.9088678614257;

0, 0, 1]

distCoeffs : [-0.03284694677907634, 0.6240409995573205, 0.002956633483741982, 0.003409533190210517, -1.967767309711385]

rvecs : [0.00809515026654148, -0.05873888734182337, 0.003919701969504452;
0.00850587986960374 1, -0.4064147615352987, -0.05213407164646773;
0.2540393579276745, 0.2835820422095629, 0.1202656009912893;
-0.3981263825062115, 0.2750262491252882, 0.1710843143191968;
0.5053113478476663, -0.5099085141385107, -0.04188727151741442;
0.6771212854917478, -0.1234986967297477, 0.06941341585795907;
-0.2182658040327461, 0.3721125987119364, 0.08782920097728329;
-0.2250030572832916, 0.1390125243152997, 0.06027630616817576;
0.054658391978067, -0.0415750489203488, 0.02117492182460531;
-0.2514908199852683, 0.0008796144538049979, 0.02467590209991481]

tvecs : [-97.53421016660101, -58.11505937927104, 285.1503505539224;
-81.13888093156375, -54.24658204718946, 238.0223594511801;
-99.98122059684917, -70.16957044533375, 338.3287742547222;
-81.14315426128742, -62.82832963683275, 414.5802314056893;
-82.85619883370141, -42.27378736955595, 233.3315498545618;
-91.84477517095226, -49.84031773452759, 278.5206461767751;
-108.6862141774932, -79.44144777883537, 351.1101914910828;
-111.4842001729432, -84.58422588452719, 334.5246062912623;
-112.6945872689217, -75.29911643427525, 302.05723672982;
-99.73301237996283, -71.14860710864869, 300.4785937821625]

5) 계산된 스마트폰의 focal length와 스펙 상의 focal length 비교 (iphone 등 focal length 스펙을 찾을 수 없으면 비교부분 제외)

S는 x,y관계없이 1250;

FxS 1079.202790164676, 1080.952529327568 근사값 약 1080으로 잡으면

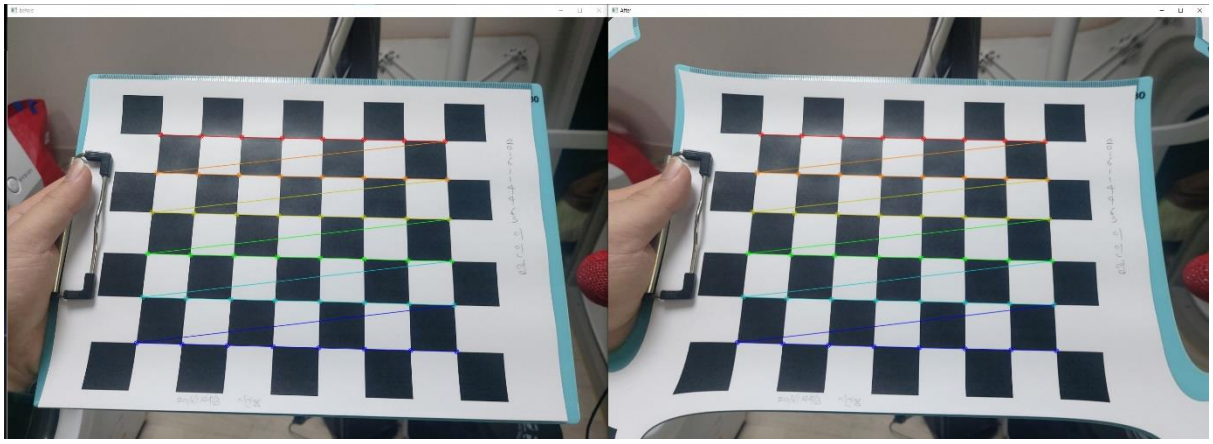
Focal length = 0.86mm이 나온다.

실제 focal length는 4.36mm이므로 엄청난 차이가 난다. 초점거리가 짧을수록 왜곡이 심하므로

즉, 프로그램은 심한 왜곡이 걸려 있다고 판단한 것이다.

이러한 이유는 체스보드의 A4용지가 완벽한 평면위에 있는 것이 아닌, 구겨진 평면위에 있어 심한 오차가 나타났다.

6) 원영상 및 왜곡 보정 영상 (1 pair)



보정 전보다 오히려 더 왜곡된 결과가 나왔다.

오차의 이유는 아마 체스보드가 얇은 A4용지라서 조금 구김이 있어, 완벽하게 평면위에 있지 않았기 때문 이라고 생각한다.

(마지막페이지는 사용한 코드입니다.)

```cpp
#include <opencv2/opencv.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/calib3d/calib3d_c.h>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <stdio.h>
#include <opencv2/opencv_modules.hpp>
#include <iostream>
#include <vector>


using namespace std;
using namespace cv;

// This program was made along with c++ 20 rule

int main() {
    Mat p_img, gray;
    Mat p_imgsmall;
    vector<string> images;
    vector<Point2f> corners;
    vector<Point3f> objp_t;
    vector<vector<Point3f>> objp;
    vector<vector<Point2f>> imgp;
    Size patternsize(8, 6);
    for (int i = 0; i < 6; i++)
    {
        for (int j = 0; j < 8; j++)
        {
            objp_t.push_back(Point3f(27*j, 27*i, 0));
        }
    }

    Mat cameraMatrix, distCoeffs, rvecs, tvecs;

    string path = "./images/";
    glob(path, images, false);
    cout << "로드 갯수 : " << images.size() << endl;
    if (images.size() == 0)
        cout << "이미지가 존재하지 않습니다.\n" << endl;

    cout << "----------------------" << endl;
    for (int cnt = 0; cnt < images.size(); cnt++)
    {
        p_img = imread(images[cnt]);
        cvtColor(p_img, gray, COLOR_BGR2GRAY);
        bool patternfound = findChessboardCorners(gray, patternsize, corners,
            CALIB_CB_ADAPTIVE_THRESH + CALIB_CB_NORMALIZE_IMAGE
            + CALIB_CB_FAST_CHECK);
        if (patternfound) {
            cornerSubPix(gray, corners, Size(11, 11), Size(-1, -1),
                TermCriteria(TermCriteria::EPS + TermCriteria::MAX_ITER, 30, 0.001));
            imgp.push_back(corners);
            drawChessboardCorners(p_img, patternsize, Mat(corners), patternfound);
```

```
55:             cout << "success" << endl;
56:         }
57:         else {
58:             cout << "false" << endl;
59:         }
60:     }
61:     cout << "----------------------" << endl;
62:     cout << "정상 검출된 이미지 개수 :" << imgp.size() << endl;
63:     for (int i = 0; i < imgp.size(); i++) {
64:         objp.push_back(objp_t);
65:     }
66:     cout << "----------------------" << endl;
67:     calibrateCamera(objp, imgp, Size(gray.rows, gray.cols), cameraMatrix, distCoeffs
68:     cout << "\n" << endl;
69:     cout << "----------------------" << endl;
70:     cout << "cameraMatrix : " << cameraMatrix << "\n-----------\n" << endl;
71:     cout << "distCoeffs : " << distCoeffs << "\n-----------\n" << endl;
72:     cout << "rvecs : " << rvecs << "\n-----------\n" << endl;
73:     cout << "tvecs : " << tvecs << "\n-----------\n" << endl;
74:     return 0;
75: }
```

```cpp
#include <opencv2/opencv.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/calib3d/calib3d_c.h>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <stdio.h>
#include <opencv2/opencv_modules.hpp>
#include <iostream>
#include <vector>


using namespace std;
using namespace cv;

// This program was made along with c++ 20 rule

int main() {
    Mat p_img, gray, undt;
    Mat p_imgsmall;
    vector<string> images;
    vector<Point2f> corners;
    vector<Point3f> objp_t;
    vector<vector<Point3f>> objp;
    vector<vector<Point2f>> imgp;
    Size patternsize(8, 6);
    for (int i = 0; i < 6; i++)
    {
        for (int j = 0; j < 8; j++)
        {
            objp_t.push_back(Point3f(27*j, 27*i, 0));
        }
    }

    Mat cameraMatrix, distCoeffs, rvecs, tvecs;

    string path = "./images/";
    glob(path, images, false);
    cout << "로드 갯수 : " << images.size() << endl;
    if (images.size() == 0)
        cout << "이미지가 존재하지 않습니다.\n" << endl;

    cout << "----------------------" << endl;
    for (int cnt = 0; cnt < images.size(); cnt++)
    {
        p_img = imread(images[cnt]);
        cvtColor(p_img, gray, COLOR_BGR2GRAY);
        bool patternfound = findChessboardCorners(gray, patternsize, corners,
            CALIB_CB_ADAPTIVE_THRESH + CALIB_CB_NORMALIZE_IMAGE
            + CALIB_CB_FAST_CHECK);
        if (patternfound) {
            cornerSubPix(gray, corners, Size(11, 11), Size(-1, -1),
                TermCriteria(TermCriteria::EPS + TermCriteria::MAX_ITER, 30, 0.001));
            imgp.push_back(corners);
            drawChessboardCorners(p_img, patternsize, Mat(corners), patternfound);
```

```
55:             cout << "success" << endl;
56:         }
57:         else {
58:             cout << "false" << endl;
59:         }
60:     }
61:     cout << "----------------------" << endl;
62:     cout << "정상 검출된 이미지 개수 :" << imgp.size() << endl;
63:     for (int i = 0; i < imgp.size(); i++) {
64:         objp.push_back(objp_t);
65:     }
66:     imshow("before", p_img);
67:     cout << "----------------------" << endl;
68:     calibrateCamera(objp, imgp, Size(gray.rows, gray.cols), cameraMatrix, distCoeffs
69:     cout << "\n" << endl;
70:     cout << "----------------------" << endl;
71:     cout << "cameraMatrix : " << cameraMatrix << "\n----------\n" << endl;
72:     cout << "distCoeffs : " << distCoeffs << "\n----------\n" << endl;
73:     cout << "rvecs : " << rvecs << "\n----------\n" << endl;
74:     cout << "tvecs : " << tvecs << "\n----------\n" << endl;
75:     undistort(p_img, undt, cameraMatrix, distCoeffs);
76:     imshow("After", undt);
77:     waitKey();
78:     return 0;
79: }
```