과제 보고서

보고서 및 논문 윤리 서약

1. 나는 보고서 및 논문의 내용을 조작하지 않겠습니다.
2. 나는 다른 사람의 보고서 및 논문의 내용을 내 것처럼 무단으로 복사하지 않겠습니다.
3. 나는 다른 사람의 보고서 및 논문의 내용을 참고하거나 인용할 시 참고 및 인용 형식을 갖추고 출처를 반드시 밝히겠습니다.
4. 나는 보고서 및 논문을 대신하여 작성하도록 청탁하지도 청탁받지도 않겠습니다.

나는 보고서 및 논문 작성 시 위법 행위를 하지 않고, 명지인으로서 또한 공학인으로서 나의 양심과 명예를 지킬 것을 약속합니다.

보고서명 : 과제
학　　과 : 전자공학과
과　　목 : 컴퓨터비전시스템
담당교수 : 최 우 영 교수님
학　　번 : 60162460
이　　름 : 고 지 형

1. 프로그램 소스코드

<main.cpp>

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include "main.h"
using namespace cv;
using namespace std;

// x86 mode에서 작동을 보장합니다.

int main()
{
	go();

	return 0;
}
```

<glui.cpp>

```
#pragma warning(disable: 4996)
#include <opencv2/opencv.hpp>
#include <iostream>
#include <GL/glui.h>
#include <GL/glut.h>
#include "OpenFileDialog.h"
#include "SaveFileDialog.h"
#include "main.h"

using namespace cv;
using namespace std;

int main_window;
extern Mat input;
extern Mat frame;
extern int i;
extern int* x_buffer;
extern int* y_buffer;
String FileName;
GLUI_RadioGroup* inputGroup;
GLUI_RadioGroup* frameGroup;
void warp(Mat* src, Mat* dst);

const std::string TCHARToString(const TCHAR* ptsz)
{
	int len = wcslen((wchar_t*)ptsz);

	char* psz = new char[2 * len + 1];

	wcstombs(psz, (wchar_t*)ptsz, 2 * len + 1);
	std::string s = psz;

	delete[] psz;

	return s;
}

void control_rb(int control)
{
	if (control == INPUT)
	{
		if (inputGroup->get_int_val() != 2)		// 캠이 아닐 때
		{
			OpenFileDialog* openFileDialog = new OpenFileDialog();
```

```
                    if (openFileDialog->ShowDialog())
                            FileName = TCHARToString(openFileDialog->FileName);
                }
                initialize();       // 잘 못 눌렀을시..

                switch (inputGroup->get_int_val())
                {
                case 0: callbackImg(); break;
                case 1: callbackVideo(); break;
                case 2: callbackCam(); break;
                }
        }
}
void control_bt(int control)
{
        switch (control)
        {

        case QUIT: exit(0);       break;

        case EXECUTE:
        {
                if (i != 4)
                {
                        printf("출력 포인트를 지정하세요.\n\n");
                        break;
                }

                else if (i == 4)
                {
                        switch (inputGroup->get_int_val())
                        {
                        case 0:
                                printf("종료를 원하시면 QUIT을 클릭하세요.\n");
                                printf("다른 Input을 사용하고 싶으시면 다른 라디오 버튼을 클릭 후
시도하세요.\n\n");

                                input = imread(FileName, IMREAD_COLOR);
                                //imshow("input", input);
                                warp(&input, &frame);
                                imshow("frame", frame);

                                initialize();
                                break;

                        case 1:
                        {
```

```
                printf("종료를 원하시면 아무 키보드 자판이나 누르고 QUIT을 클릭하세요.\n");
                printf("다른 Input을 사용하고 싶으시면 아무 키보드 자판이나 누르고 다른 라디오 버튼을 클릭 후 시도하세요.\n\n");
                VideoCapture cap(FileName);

                while (1)
                {
                        frame = imread(FRAME_NAME, IMREAD_COLOR);
                        cap >> input;   // 동영상에서 하나의 프레임을 추출한다.

                        //imshow("input", input);
                        warp(&input, &frame);
                        imshow("frame", frame);

                        if (waitKey(30) > 0)
                        {
                                initialize();
                                break;
                        }
                }
                break;
        }

        case 2:
        {
                printf("종료를 원하시면 아무 키보드 자판이나 누르고 QUIT을 클릭하세요.\n");
                printf("다른 Input을 사용하고 싶으시면 아무 키보드 자판이나 누르고 다른 라디오 버튼을 클릭 후 시도하세요.\n\n");
                VideoCapture cam;
                cam.open(CAP_NAME);

                while (1)
                {
                        frame = imread(FRAME_NAME, IMREAD_COLOR);
                        cam.read(input);

                        //imshow("input", input);
                        warp(&input, &frame);
                        imshow("frame", frame);

                        if (waitKey(30) > 0)
                        {
                                initialize();
                                break;
```

```
                                        }
                                }
                                break;
                        }

                        default: break;
                        }
                }
        }
        }
}

void go(void)
{
        cout << "Input을 선택하고, Read를 클릭하세요.\n";
        int obj = 0xff;
        GLUI* glui = GLUI_Master.create_glui("GLUI", 0L, 512, 512);
        main_window = glui->get_glut_window_id();
        GLUI_Master.set_glutIdleFunc(NULL);

        GLUI_Panel* inputPanel = new GLUI_Panel(glui, "Input Image");
        inputGroup = new GLUI_RadioGroup(inputPanel, &obj);

        new GLUI_RadioButton(inputGroup, "Image");
        new GLUI_RadioButton(inputGroup, "Video");
        new GLUI_RadioButton(inputGroup, "Cam");

        new GLUI_Button(inputGroup, "Read", INPUT, control_rb);
        new GLUI_Button(inputGroup, "Execute", EXECUTE, control_bt);
        new GLUI_Button(inputGroup, "Quit", QUIT, control_bt);

        glui->set_main_gfx_window(main_window);
        glutMainLoop();
}
```

<callback.cpp>

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include "main.h"
using namespace cv;
using namespace std;

Mat input;
Mat frame;
extern String FileName;

void callbackImg(void)
{
	input = imread(FileName, IMREAD_COLOR);
	if (input.empty()) { cout << "영상 입력 오류\n "; exit(0); }

	frame = imread("frame.jpg", IMREAD_COLOR);
	if (frame.empty()) { cout << "영상 입력 오류\n "; exit(0); }
	imshow("frame", frame);

	cout << "화면에 네 포인트를 선택하고 Excute 버튼을 클릭하세요.\n\n";
	setMouseCallback("frame", onMouse, &frame);
}

void callbackVideo(void)
{
	VideoCapture cap(FileName);	// 동영상 파일인 경우
	if (!cap.isOpened())
		cerr << "ERROR! Unable to open video\n";

	frame = imread(FRAME_NAME, IMREAD_COLOR);
	if (frame.empty()) { cout << "영상 입력 오류\n "; exit(0); }
	imshow("frame", frame);

	cap >> input;	// 동영상에서 하나의 프레임을 추출한다.
	if (input.empty()) { cout << "영상 입력 오류\n "; exit(0); }

	cout << "화면에 네 포인트를 선택하고 Excute 버튼을 클릭하세요.\n";
	setMouseCallback("frame", onMouse, &frame);

	return;
}

void callbackCam(void)
{
	VideoCapture cam;
```

```
        cam.open(CAP_NAME);
        if (!cam.isOpened()) {
                cerr << "ERROR! Unable to open camera\n";
                return;
        }

        cam.read(input);
        if (input.empty()) {
                cerr << "ERROR! blank frame grabbed\n";
                exit(0);
        }

        frame = imread(FRAME_NAME, IMREAD_COLOR);
        if (frame.empty()) { cout << "영상 입력 오류\n "; exit(0); }
        imshow("frame", frame);

        cout << "화면에 네 포인트를 선택하고 Excute 버튼을 클릭하세요.\n";
        setMouseCallback("frame", onMouse, &frame);

        return;

}
```

<mouseEvent.cpp>

```cpp
#include <opencv2/opencv.hpp>
#include <iostream>
#include "main.h"
using namespace cv;
using namespace std;

int x_buffer[4];
int y_buffer[4];
int i = 0;

void initialize(void)
{
    i = 0;
    memset(x_buffer, 0, sizeof(x_buffer));
    memset(y_buffer, 0, sizeof(y_buffer));
}

void onMouse(int event, int x, int y, int flags, void* param)    // 마우스 이벤트가 발생하면 호출되는 콜백 함수이다.
{
    if (event == EVENT_LBUTTONDOWN)
    {  // 좌측 마우스버튼을 눌렀을 때
        Mat& img = *(Mat*)(param);
        if (i < 4)
        {
            circle(img, Point(x, y), 5, Scalar(255, 0, 0), 3);
            x_buffer[i] = x;
            y_buffer[i] = y;
            i++;
            imshow("frame", img);
        }

        if (i == 4) // 다하면 다시 불러와서 circle 삭제
        {
            img = imread("frame.jpg", IMREAD_COLOR);
        }

    }
}

void maskThreshold(Mat* mask, Mat* dst)
{
    Mat thr = Mat::zeros(dst->rows, dst->cols, CV_8UC3); // 마스크 복사 -> threshold로 사용
    Mat img = *dst; // 바깥 프레임 영상
    Mat maskBuffer = mask->clone();
```

```
    Mat maskBuffer_inv = mask->clone();
    Mat fg, bg;

    cvtColor(maskBuffer, maskBuffer, COLOR_BGR2GRAY);    // 1채널
    threshold(maskBuffer, maskBuffer_inv, 0, 255, THRESH_BINARY_INV);    // 1채널
    threshold(maskBuffer_inv, maskBuffer, 0, 255, THRESH_BINARY_INV);    // 1채널
    bitwise_and(img, img, bg, maskBuffer_inv);  // 3채널 저장
    bitwise_and(*mask, *mask, fg, maskBuffer); // 3채널 저장

    add(bg, fg, *dst);

}

void warp(Mat* src, Mat* dst)
{
    Point2f inputp[4], outputp[4];

    inputp[0] = Point2f(0, 0);  //  좌측 맨위
    inputp[1] = Point2f(src->cols, 0);  // 우측 맨위
    inputp[2] = Point2f(0, src->rows);  // 좌측 맨밑
    inputp[3] = Point2f(src->cols, src->rows);  // 우측 맨밑

    outputp[0] = Point2f(x_buffer[0], y_buffer[0]);
    outputp[1] = Point2f(x_buffer[1], y_buffer[1]);
    outputp[2] = Point2f(x_buffer[2], y_buffer[2]);
    outputp[3] = Point2f(x_buffer[3], y_buffer[3]);

    Mat mask;
    Mat dst_buffer = *dst;
    Mat cut = getPerspectiveTransform(inputp, outputp);

    warpPerspective(*src, mask, cut, dst->size());
    maskThreshold(&mask, dst);
    bitwise_or(mask, dst_buffer, *dst);
}
```

<OpenFileDialog.cpp>

```
#include "OpenFileDialog.h"

OpenFileDialog::OpenFileDialog(void)
{
    this->DefaultExtension = 0;
    this->FileName = new TCHAR[MAX_PATH];
    this->Filter = 0;
    this->FilterIndex = 0;
    this->Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;
    this->InitialDir = 0;
    this->Owner = 0;
    this->Title = 0;
}

bool OpenFileDialog::ShowDialog()
{
    OPENFILENAME ofn ;

    ZeroMemory(&ofn, sizeof(ofn));

    ofn.lStructSize = sizeof(ofn);
    ofn.hwndOwner = this->Owner;
    ofn.lpstrDefExt = this->DefaultExtension;
    ofn.lpstrFile = this->FileName;
    ofn.lpstrFile[0] = '\0';
    ofn.nMaxFile = MAX_PATH;
    ofn.lpstrFilter = this->Filter;
    ofn.nFilterIndex = this->FilterIndex;
    ofn.lpstrInitialDir = this->InitialDir;
    ofn.lpstrTitle = this->Title;
    ofn.Flags = this->Flags;

    GetOpenFileName(&ofn);

    if (_tcslen(this->FileName) == 0) return false;

    return true;
}
```

<SaveFIleDialog.cpp>

```
#include "SaveFileDialog.h“

SaveFileDialog::SaveFileDialog(void)
{
    this->DefaultExtension = 0;
    this->FileName = new TCHAR[MAX_PATH];
    this->Filter = 0;
    this->FilterIndex = 0;
    this->Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;
    this->InitialDir = 0;
    this->Owner = 0;
    this->Title = 0;
}

bool SaveFileDialog::ShowDialog()
{
    OPENFILENAME ofn ;

    ZeroMemory(&ofn, sizeof(ofn));

    ofn.lStructSize = sizeof(ofn);
    ofn.hwndOwner = this->Owner;
    ofn.lpstrDefExt = this->DefaultExtension;
    ofn.lpstrFile = this->FileName;
    ofn.lpstrFile[0] = '\0';
    ofn.nMaxFile = MAX_PATH;
    ofn.lpstrFilter = this->Filter;
    ofn.nFilterIndex = this->FilterIndex;
    ofn.lpstrInitialDir = this->InitialDir;
    ofn.lpstrTitle = this->Title;
    ofn.Flags = this->Flags;

    GetSaveFileName(&ofn);

    if (_tcslen(this->FileName) == 0) return false;

    return true;
}
```

<main.h>

```
#ifndef MAIN_H
#define MAIN_H
#define INPUT   0
#define EXECUTE         1
#define QUIT    -1

#define MP4_NAME        "test.mp4"
#define IMG_NAME        "lenna.jpg"
#define FRAME_NAME      "frame.jpg"
#define CAP_NAME        1       // my computer: rear camera 0, front one 1

void onMouse(int event, int x, int y, int flags, void* param);
void go(void);
void callbackImg(void);
void callbackVideo(void);
void callbackCam(void);
void initialize(void);

#endif MAIN_H
```

<OpenFileDialog.h>

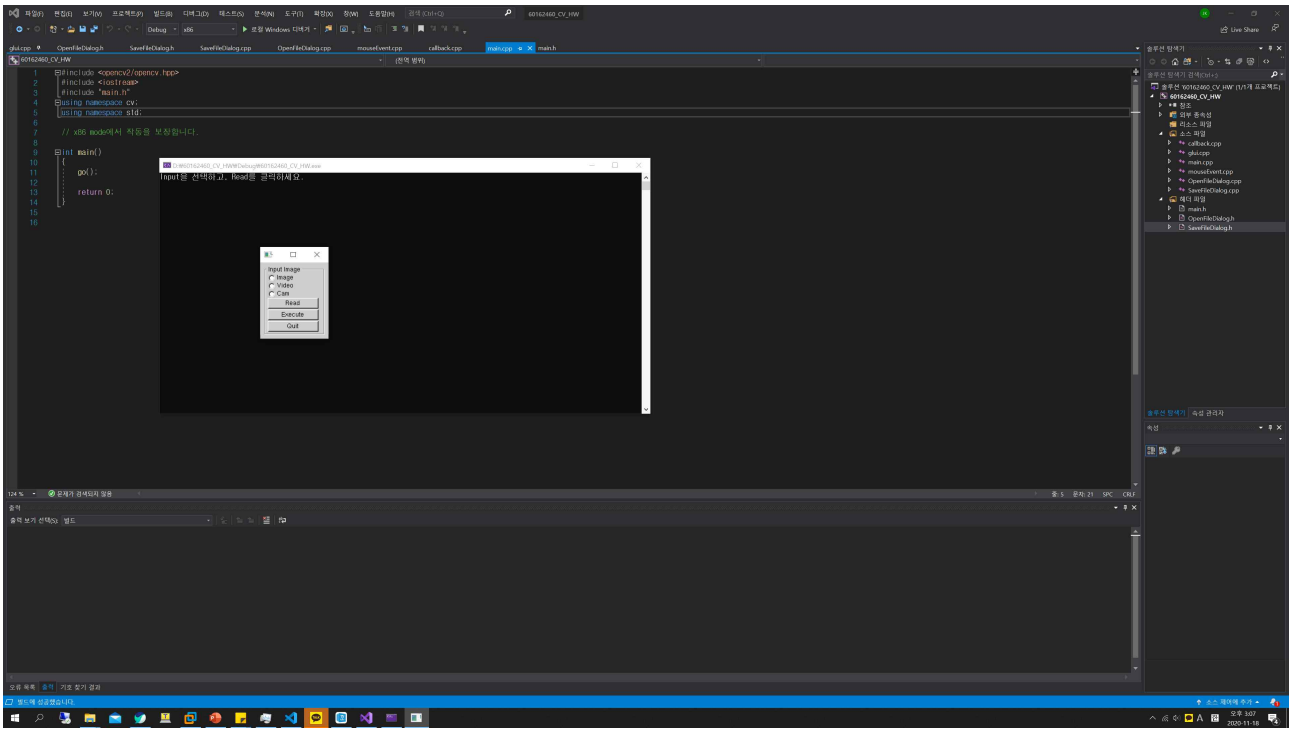
```
#pragma once

#include <Windows.h>
#include <Commdlg.h>
#include <tchar.h>

class OpenFileDialog
{
public:
	OpenFileDialog(void);

	TCHAR *DefaultExtension;
	TCHAR *FileName;
	TCHAR *Filter;
	int FilterIndex;
	int Flags;
	TCHAR *InitialDir;
	HWND Owner;
	TCHAR *Title;

	bool ShowDialog();
};
```

<saveFileDialog.h>

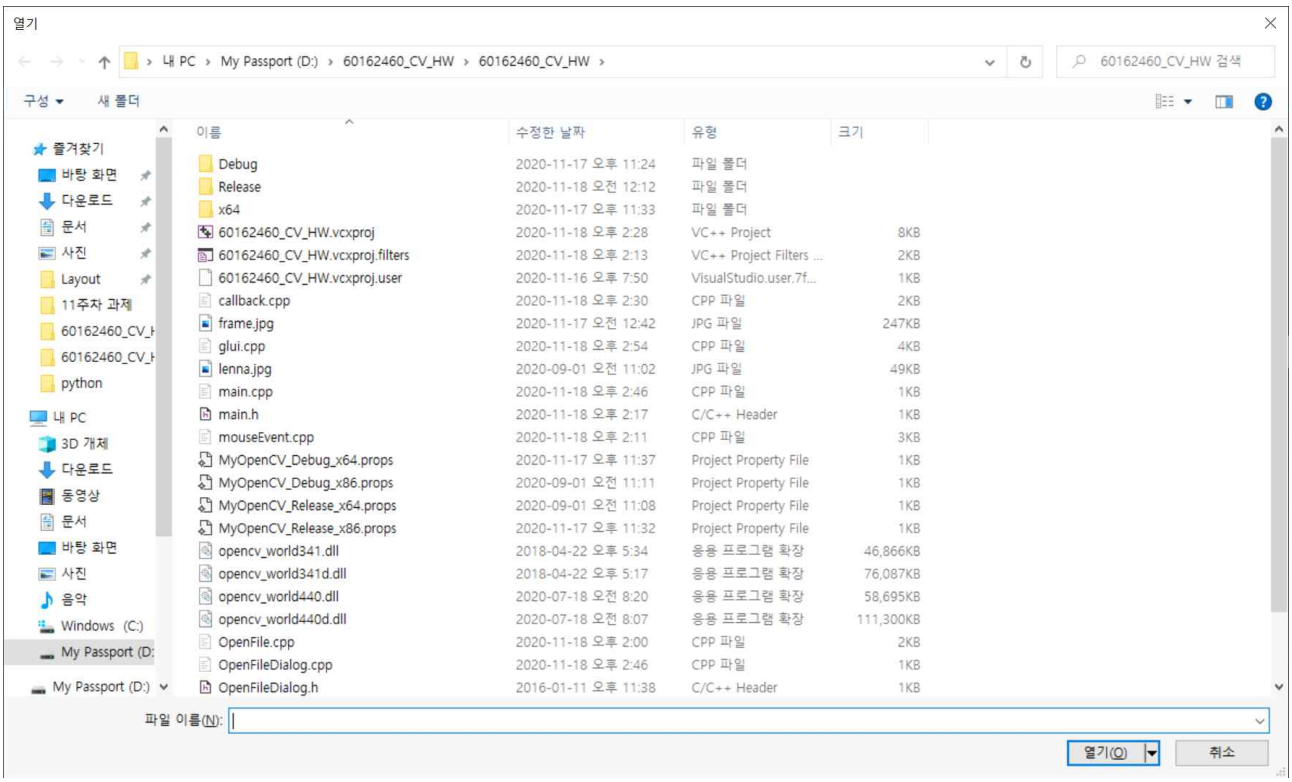
```
#pragma once

#include <Windows.h>
#include <Commdlg.h>
#include <tchar.h>

class SaveFileDialog
{
public:
	SaveFileDialog(void);

	TCHAR *DefaultExtension;
	TCHAR *FileName;
	TCHAR *Filter;
	int FilterIndex;
	int Flags;
	TCHAR *InitialDir;
	HWND Owner;
	TCHAR *Title;

	bool ShowDialog();
};
```

2. 결과영상

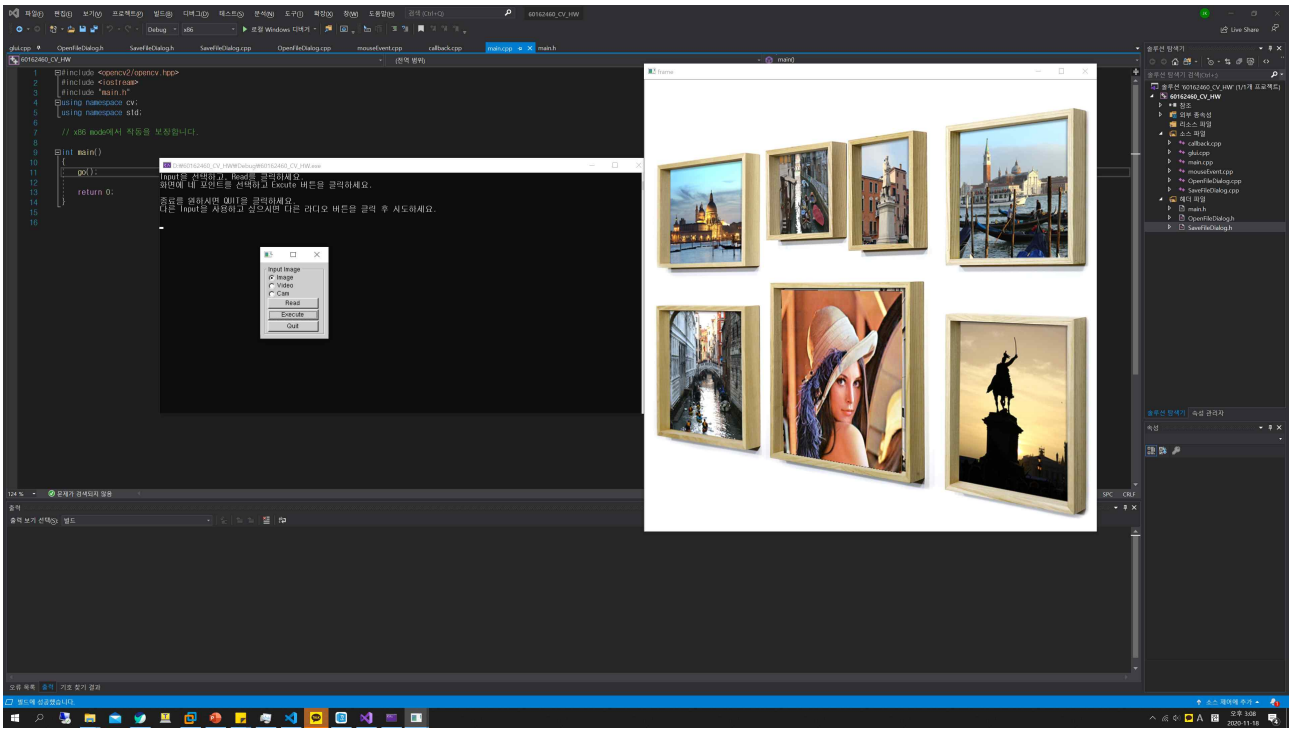
<초기화면>



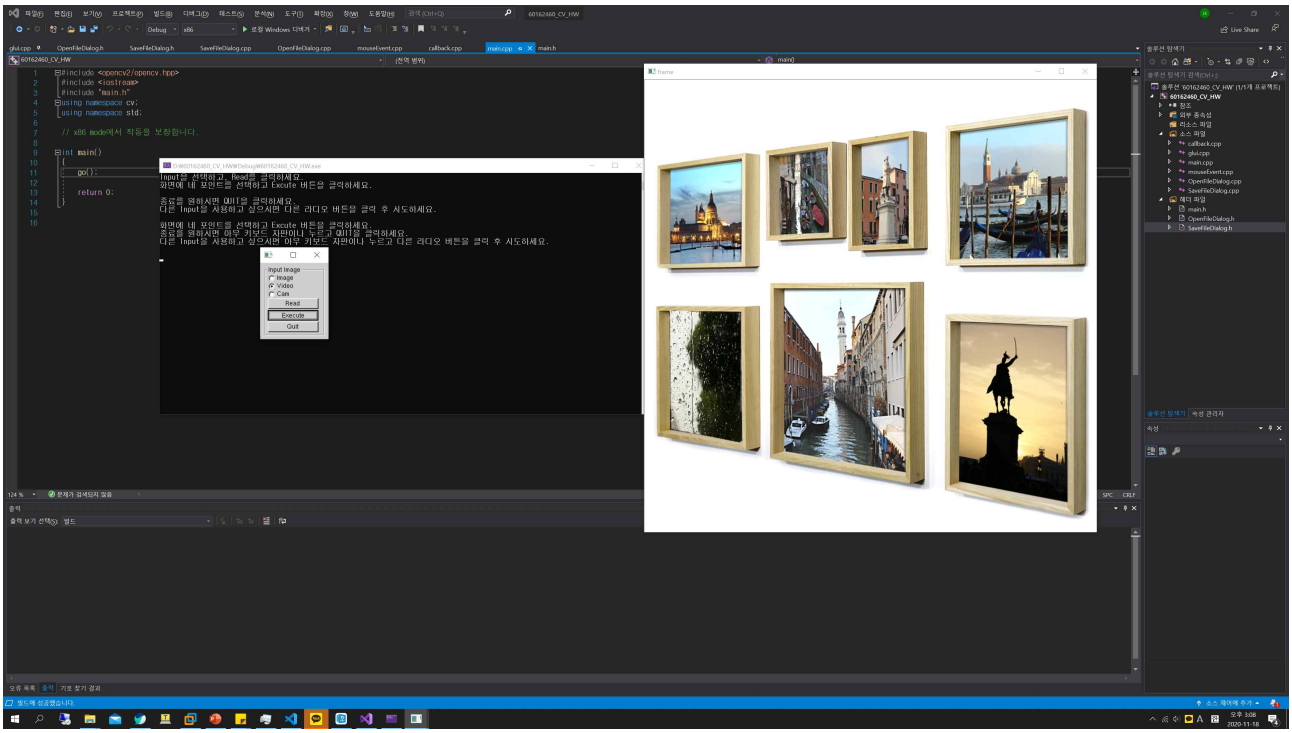
<Image Selection(Image, VIdeo Mode)>



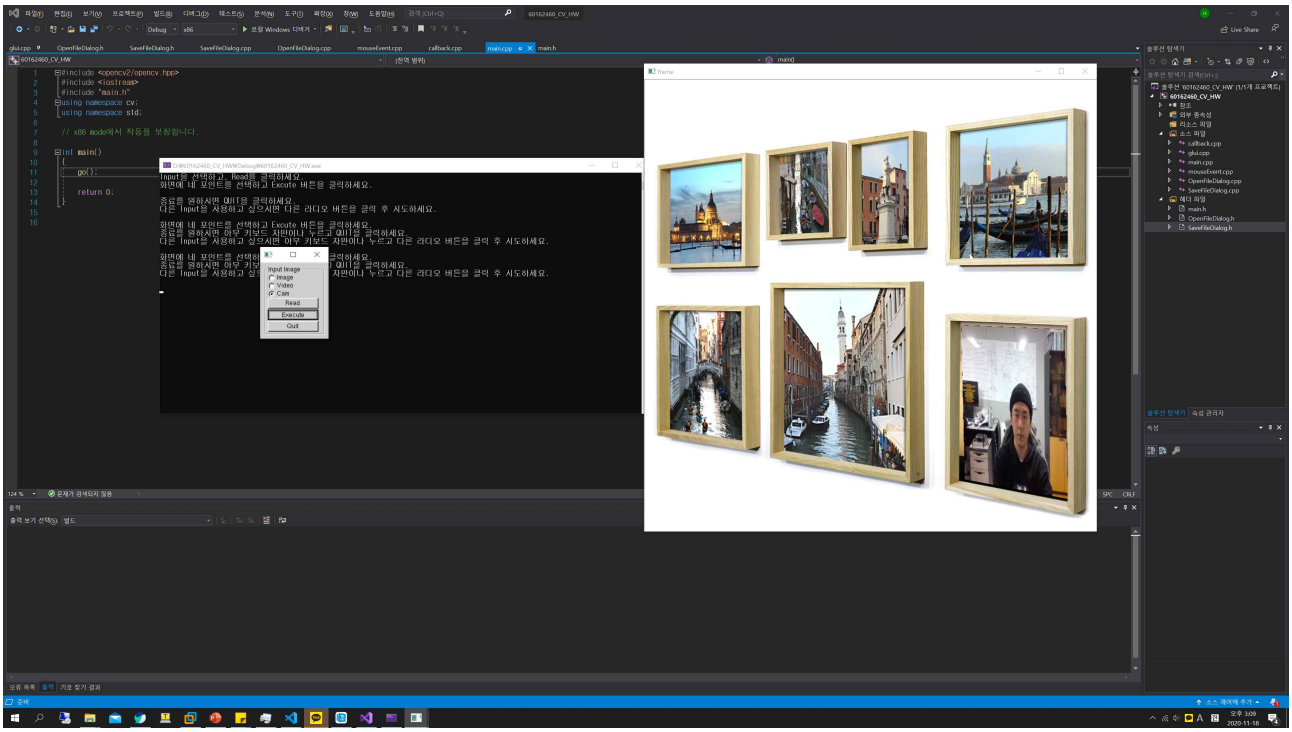
<Image Load>



<Video Load>



<Camera Load>


#include <opencv2/opencv.hpp>
#include <iostream>
#include "main.h"
using namespace cv;
using namespace std;
// x86 mode에서 작동을 보장합니다.
int main()
go();
return 0;
Input을 선택하고 Read를 클릭하세요
와면에 네 포인트를 선택하고 Excute 버튼을 클릭하세요.
종료를 원하시면 QUIT을 클릭하세요.
다른 Input을 사용하고 싶으시면 다른 라디오 버튼을 클릭 후 시도하세요.
Input Image
Image
Video
Cam
Read
Execute
Quit

3. 프로그램 사용방법

1) 프로그램이 실행되는 폴더에 프레임으로 사용될 “frame.jpg” 이미지 파일을 준비한다.

2) 프로그램을 실행하면 아래와 같이 GLUI가 실행되는데, 이 중 원하는 Input을 가리키는 라디오 버튼을 클릭하고, “Read” 버튼을 누른다.

3) Image와 Video mode의 경우 “파일 열기” 창이 나오는데, 적절한 이미지 혹은 동영상 파일을 선택하여 “열기” 버튼을 클릭한다.

4) 프레임 이미지가 나오면 원하는 점 네 곳을 클릭 후 “Execute” 버튼을 클릭한다.

5)
-Image mode: 종료를 원하면 QUIT을 클릭한다.
다른 Input을 사용하고 싶으시면 다른 라디오 버튼을 선택하고 2)부터 다시 실행한다.

-Video mode: 종료를 원하면 아무 키보드 자판이나 눌러 영상을 멈추고 QUIT을 클릭한다.
다른 Input을 사용하고 싶으시면 아무 키보드 자판이나 누르고 다른 라디오 버튼을 선택하고 2)부터 다시 실행한다.

-Camera mode: 종료를 원하면 아무 키보드 자판이나 눌러 영상을 멈추고 QUIT을 클릭한다.
다른 Input을 사용하고 싶으시면 아무 키보드 자판이나 누르고 다른 라디오 버튼을 선택하고 2)부터 다시 실행한다.

참조: https://github.com/JiHyeongKo/tuition_CV_2020