Sift함수를 사용하여 동영상에서 추출된이미지와 매칭시키기
전체 코드

```
#include "ofApp.h"


int main(int argc, char *argv[]) {

                                                                        //
------------------------------------------------------------------------
-----
    // 동영상에서 추출된이미지와 매칭시키기
    // opencv의 VideoCapture capture를 이용하여 동영상을 가져옴.

    VideoCapture capture;
     capture.open("3411.mp4");   //캡쳐할 동영상을 가져옴. 윗줄VideoCapture
capture(0);으로 하고 웹캠으로 가져올 수도 있음.
    if (!capture.isOpened()) {
         std::cerr << "Could not open video" << std::endl;  //동영상을
못 가져올시
        return 0;
    }

    // 새창이름
    cv::namedWindow("video", 1);



                                                                        //
------------------------------------------------------------------------
-----


      cv::Mat db_original = cv::imread("313131.jpg",
CV_LOAD_IMAGE_GRAYSCALE);//동영상의 일부이미지와 매칭될 jpg를 가져옴.이미지를 그
레이스케일로 변환
    cv::Mat db;

    cv::resize(db_original, db, cv::Size(db_original.cols / 2,

db_original.rows / 2), 0, 0, CV_INTER_NN);


    // SiftFeatureDetector를 이용해 특징점 추출및 계산해서 읽어오기
          Ptr<SiftFeatureDetector>    detector    =
SiftFeatureDetector::create(0.05,5.0);
       Ptr<SiftDescriptorExtractor>  extractor  =
SiftDescriptorExtractor::create(3.0);


    // 입력받은 이미지에서 특징점 추출
```

```
    std::vector<cv::KeyPoint> kps_db;
    detector->detect(db, kps_db);

    // 입력받은 이미지에서 특징점을 계산해서 사용
    cv::Mat dscr_db;
    extractor->compute(db, kps_db, dscr_db);

    while (true) {
        bool frame_valid = true;

        cv::Mat frame_original;
        cv::Mat frame;

        try {
            capture >> frame_original; // 동영상으로부터 매 프레임마다 이미지
를 캡쳐함
                    cv::resize(frame_original, frame,
cv::Size(frame_original.cols / 2,
                    frame_original.rows / 2), 0, 0, CV_INTER_NN); //
downsample 1/2x
        }
        catch (cv::Exception& e) {
            std::cerr << "Exception occurred. Ignoring frame... " <<
e.err
            << std::endl;
            frame_valid = false;
        }

        if (frame_valid) {
            try {
                // 캡쳐된 이미지를 프레임마다 이퀄라이즈 & 그레이 스케일로 변환시킴
                cv::Mat grayframe;
                cv::cvtColor(frame, grayframe, CV_BGR2GRAY);
                cv::equalizeHist(grayframe, grayframe);

                                                                    //
------------------------------------------------------------
                // 이미지 추출

                // 입력받은 동영상의 그레이프레임영상에서 특징점 추출
                std::vector<cv::KeyPoint> kps_frame;
                detector->detect(grayframe, kps_frame);

                // 위의 특징점 계산해서 사용
                cv::Mat dscr_frame;
                       extractor->compute(grayframe, kps_frame,
dscr_frame);

                // FLANN함수로  이미지 매칭함
                cv::FlannBasedMatcher matcher;
                std::vector<cv::DMatch> matches;
                matcher.match(dscr_db, dscr_frame, matches);

                double max_dist = 0.0, min_dist = 100.0;
```

```
                for (int i = 0; i < matches.size(); i++) {
                    double dist = matches[i].distance;
                    if (dist < min_dist) min_dist = dist;
                    if (dist > max_dist) max_dist = dist;
                }

                // matches에서 매칭 결과사이즈가 2배 미만의 적은 결과값을 갖는다
                std::vector<cv::DMatch> good_matches;

                for (int i = 0; i < matches.size(); i++) {
                    if (matches[i].distance <= 2 * min_dist) {
                        good_matches.push_back(matches[i]);
                    }
                }

                cv::Mat img_matches;
                     cv::drawMatches(db,  kps_db,  frame,  kps_frame,
good_matches,
                                     img_matches, cv::Scalar::all(-1),
cv::Scalar::all(-1),
                                    std::vector<char>(),

cv::DrawMatchesFlags::NOT_DRAW_SINGLE_POINTS);

                // 화면에 매칭되는 이미지 출력
                cv::imshow("webcam", img_matches);

            }
            catch (cv::Exception& e) {
                 std::cerr << "Exception occurred. Ignoring frame...
" << e.err
                << std::endl;
            }
        }
        if (cv::waitKey(30) >= 0) break;
    }

    // 캡쳐된 비디오에서 자동으로 할당해제
    return 0;
}
```