IMAGE STABILIZATION

폴라개발, 전은철

LOOPIC

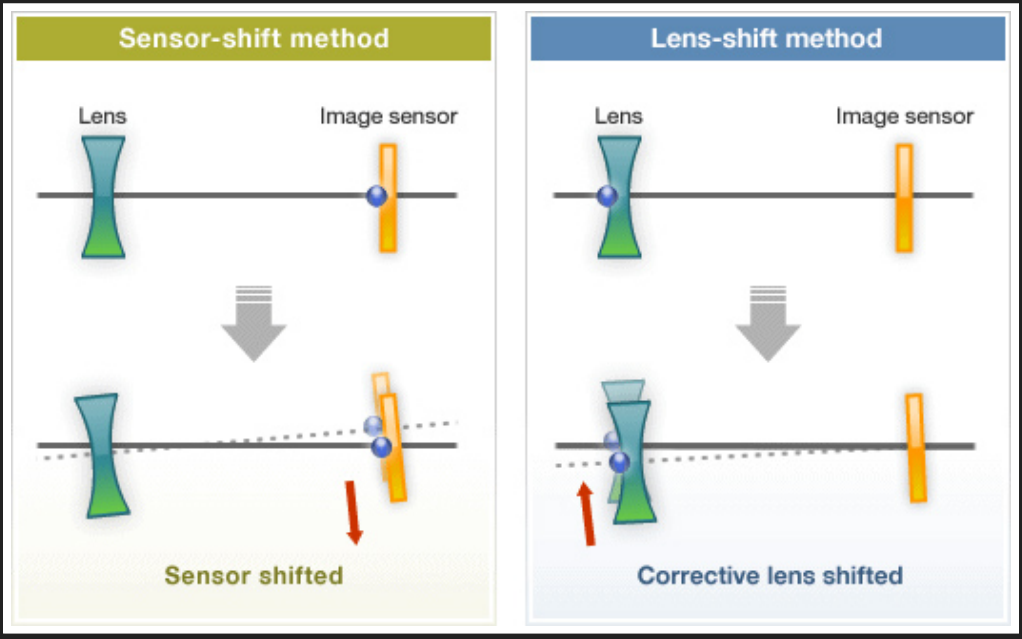
0.25 sec, 7장의 연속사진

LOOPIC

... 근데 떨림이 심하다.

IMAGE STABILIZATION

Hardware, Software



제약사항 파악

Real-time ?

Iterative ?

떨림 정도 ?

개발 기간 ?

제약사항 파악

Real-time -> 필요 없음.

Iterative -> 필요 없음.

떨림 정도 -> 신경 안씀.

개발 기간 -> 아무도 신경안씀.

제약사항 파악

최대한 간단하게 구현해보자!

간단한건 아는게 많을 때 가능하다.

PROCESS

이미지가 떨린다??

각 이미지가 이동한 만큼 되돌려주면 되지 않는가?

얼마나 이동했는지 어떻게 아는가?

전에 나왔던 것이 다음장에 어디에 나오는지 보면 되지않나?

PROCESS

- 각 이미지에서 특징점을 뽑는다.
- 특징점들 중 같은 특징점을 찾는다.
- 같은 특징점들이 얼마나 이동했나를 가지고 이동을 추정 (estimate) 한다

주의사항

이것은 최선이 아닐 수 있습니다.

특징점 뽑기(FEATURE EXTRACTION)

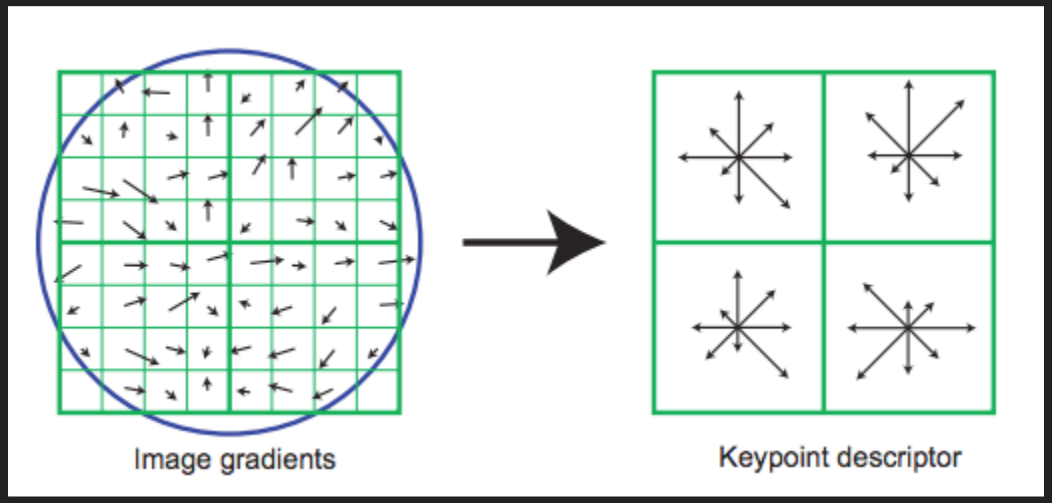
Union Station, https://en.wikipedia.org/wiki/Union_Station_(St._Louis)

SSD(SUM OF SQUARED DIFFERENCE), MORAVEC80

$$S(x,y) = \sum_u \sum_v w(u,v)\ (I(u+x, v+y) - I(u,v))^2$$

Moravec80 -> Harris Corner(Harris88) -> SIFT(Scale invariant feature Transform, Lowe2004)

DESCRIPTOR 생성



(SIFT Descriptor, Lowe 2004)

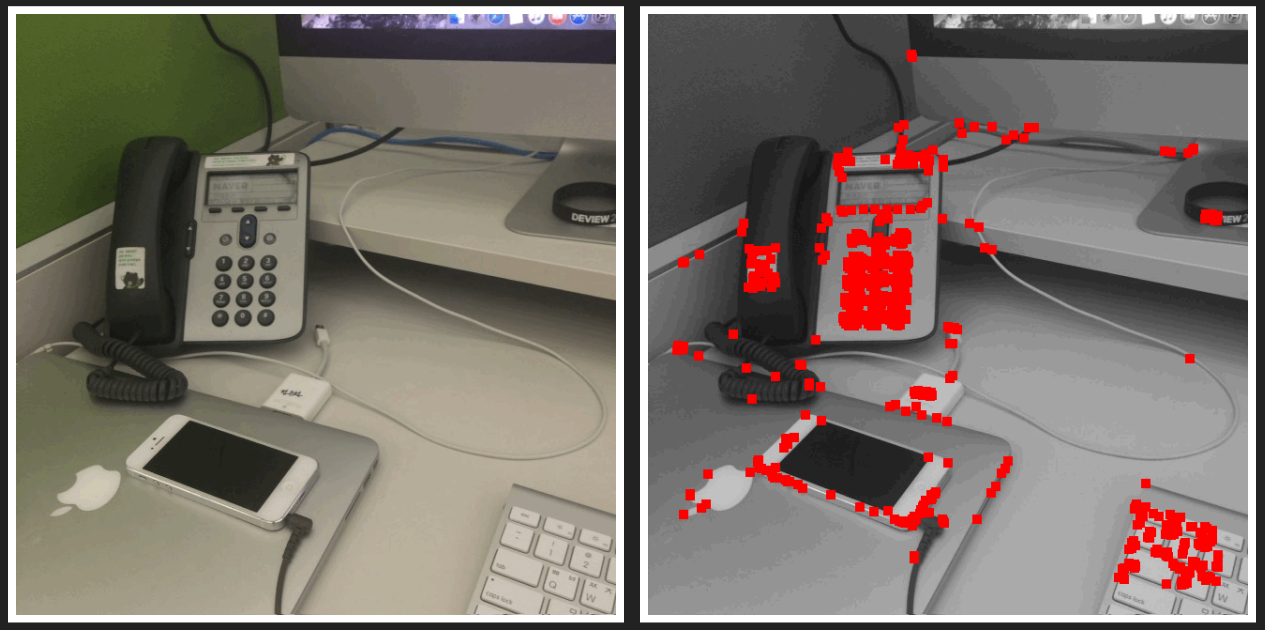
SIFT라는걸 구해서 쓰자.....?

특허가 있음. 일단 더 발전된 BRISK 라는걸 써보자.

BRISK, OPENCV

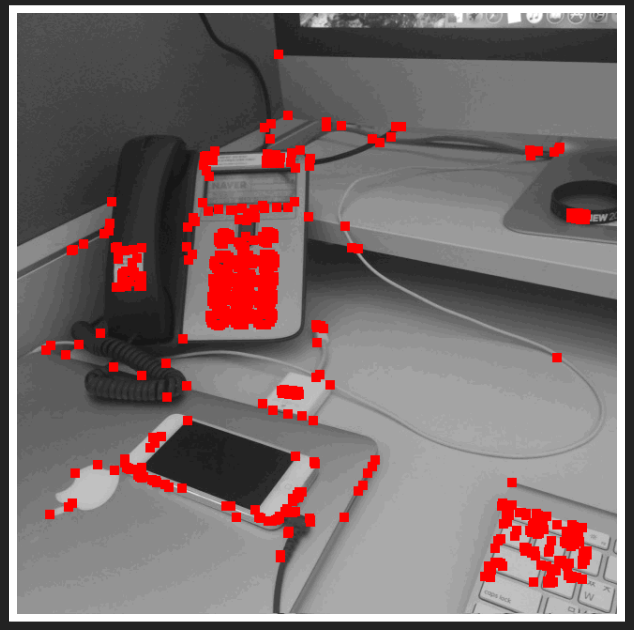
```
void extractFeatureUsingBRISK(Mat& imageMat, vector<KeyPoint>& keyPoi
    // Set brisk parameters
    int Threshl=20;
    int Octaves=3; //(pyramid layer) from which the keypoint has beer
    float PatternScales=1.0f;

    cv::Ptr<cv::FeatureDetector> detector = cv::BRISK::create(Threshl
    detector->detectAndCompute(imageMat, noArray(), keyPoints, descri
}
```

FEATURE EXTRACTION 결과



FEATURE MATCHING

어떤 점이 다음 이미지 들에서 어디에 위치하는가.



FEATURE MATCHING

Descriptor, Euclidean Distance, Mahalanobis Distance, Kd-Tree, Hashing, etc....

OPENCV를 이용한 FEATURE MATCHING

```
cv::BFMatcher matcher(cv::NORM_HAMMING);
std::vector< std::vector<DMatch> > nn_matches;
matcher.knnMatch(descriptorsA, descriptorsB, nn_matches, 2);
```

FEATURE MATCHING 결과

이제 이미지가 얼마나 이동했는지 찾자.

- 에러 값을 최소로 하는 Geometric transformation matrix.
- Ransac

GEOMETRIC TRANSFORMATION MATRIX 를 찾자

```
29
30 Mat getGeometricTransformMat(std::vector<KeyPoint>& aList, std::vector<KeyPoint>& bList){
31
32 // Geometric Transformation Matrix
33 // T = t11 t12 0
34 //     t21 t22 0
35 //     t31 t32 1
36 //
37 // M1 * T = M2
38 // M1 = SumOfAxSquare   SumOfAxAy       SumOfAx     0              0              0
39 //      SumOfAxAy       SumofAySqaure   SumOfAy     0              0              0
40 //      SumOfAx         SumOfAy         SumOf 1     0              0              0
41 //      0               0               0           SumOfAxSquare  SumOfAxAy      SumOfAx
42 //      0               0               0           SumOfAxAy      SumofAySqaure  SumOfAy
43 //      0               0               0           SumOfAx        SumOfAy        SumOf 1
44 //
45 // M2 = SumOfAxBx
46 //      SumOfAyBx
47 //      SumOfBx
48 //      SumOfAxBy
49 //      SumOfAyBy
50 //      SumOfBy
51
52
53     double sumOfAxSquare = 0;
54     double sumOfAxAy = 0;
55     double sumOfAx = 0;
56     double sumOfAySquare = 0;
```

ESTIMATERIGIDTRANSFORM

```
Mat R = estimateRigidTransform(p2, p1, true);
```

교훈 : 구현하기 전에 잘 찾아보자.

WARP PERSPECTIVE

```
warpPerspective(targetImageMats[i], res, prevH, cv::Size(cols, rows))
```

여기까지의 결과

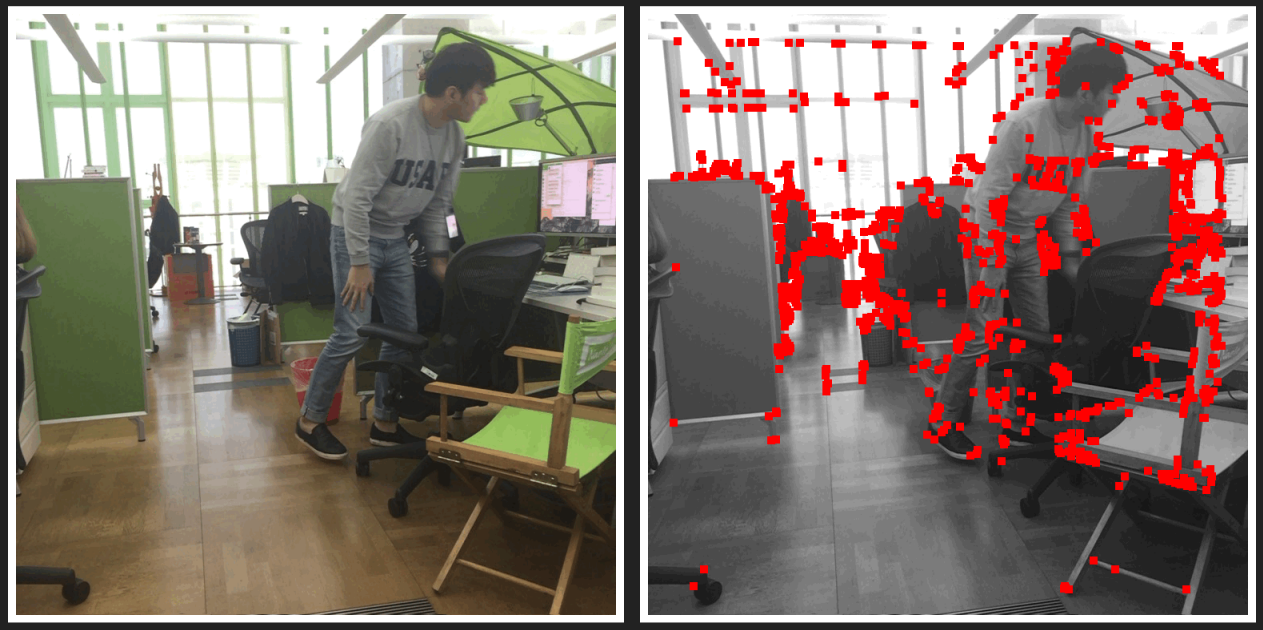
화면에 움직이는 물체가 있으면 어떨까?

ㅜ_ㅠ

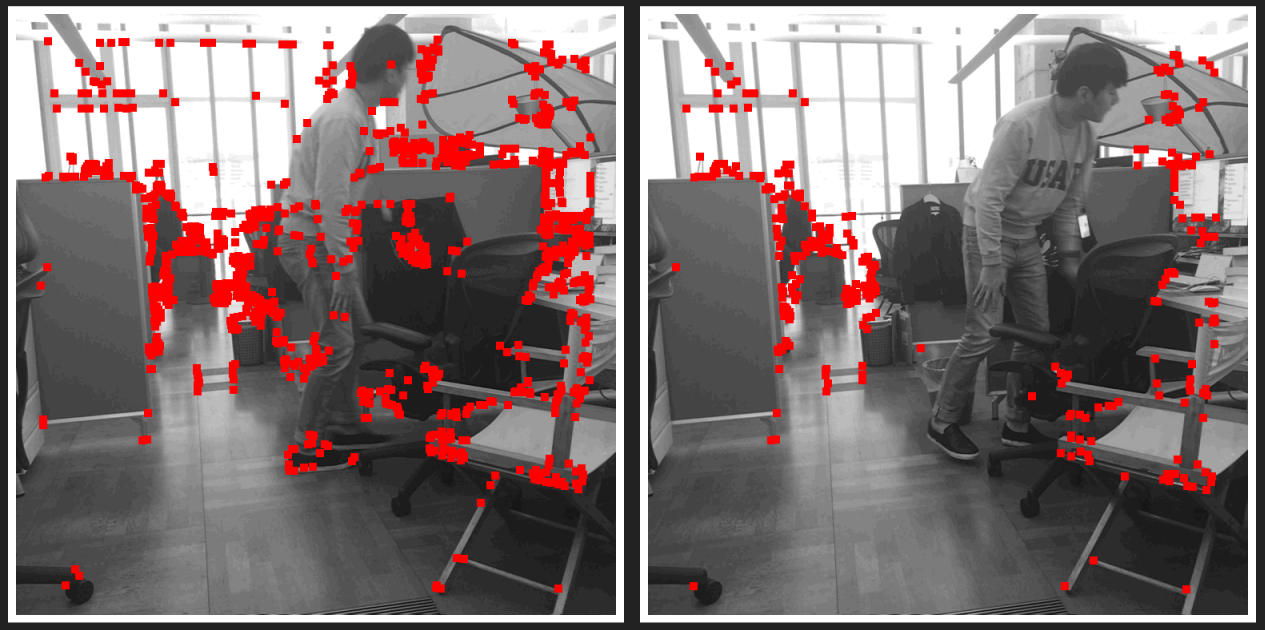
전체 프레임에서 공통적으로 잡히는 점들만 남겨보면 어떨까?

^ _ ^ b

움직이는 물체에서 점 추출



공통으로 나오는 애들만 매칭



결과

속도 문제

iPhone5..... 18secs

jsharp83 commented on 12 Oct

Device : iphone5

Dataset : DATASET_4

Total time : about 18 secs

extract features : about 11 secs, (One image spends about 2secs)

Matching : 0.8secs

Find inliner : 5.5 secs

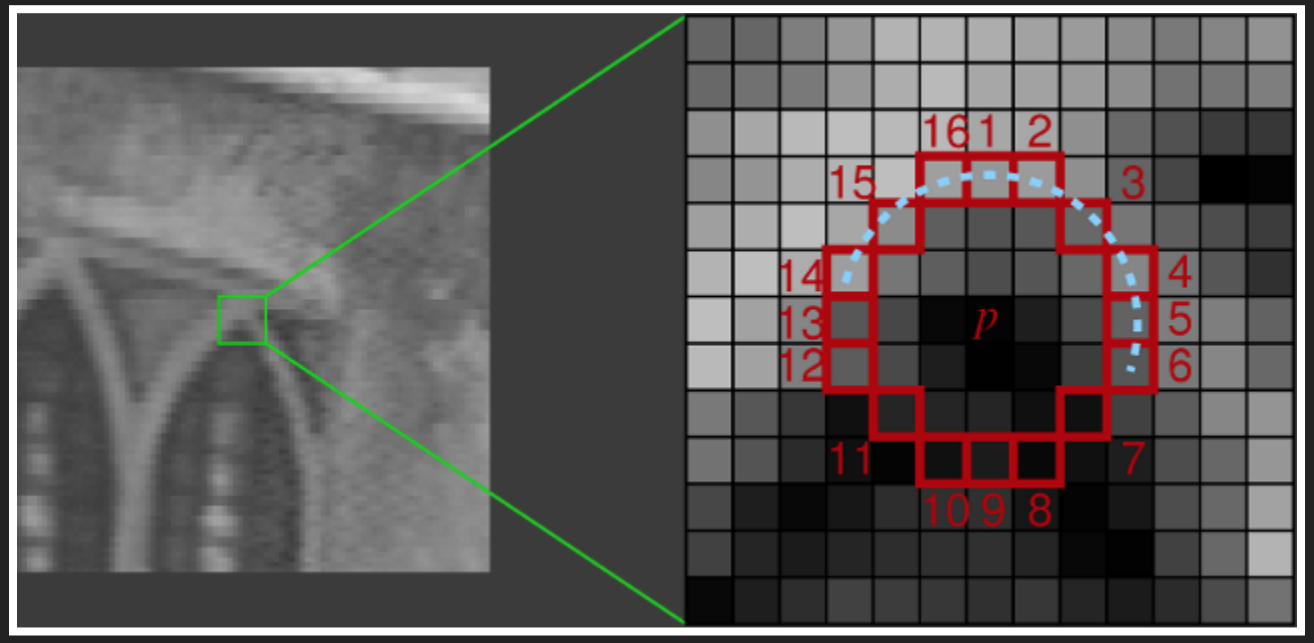
estimate homography matrix = : 0.5sec

속도 문제

extract features : about 11 secs.

```
cv::Ptr<cv::FeatureDetector> detector = cv::BRISK::create(Threshl
detector->detectAndCompute(imageMat, noArray(), keyPoints, descri
```

BRISK -> FAST+ORB



11 secs -> 0.5 secs

FIND INLINER

```
for(int j = 1; j < matchedList.size(); j++){
    std::vector< std::vector<DMatch> > nn_matcher = matchedList[j];
    int nextTrainIdx = nn_matcher[trainIdx][0].trainIdx;

    if(isInliner(nn_matcher, trainIdx)){
    int nextTrainIdx = matchedList[j][trainIdx][0].trainIdx;
    if(isInliner(matchedList[j], trainIdx)){
```

Copy vector.... 5.5 secs -> 0.5 secs

남은 이야기

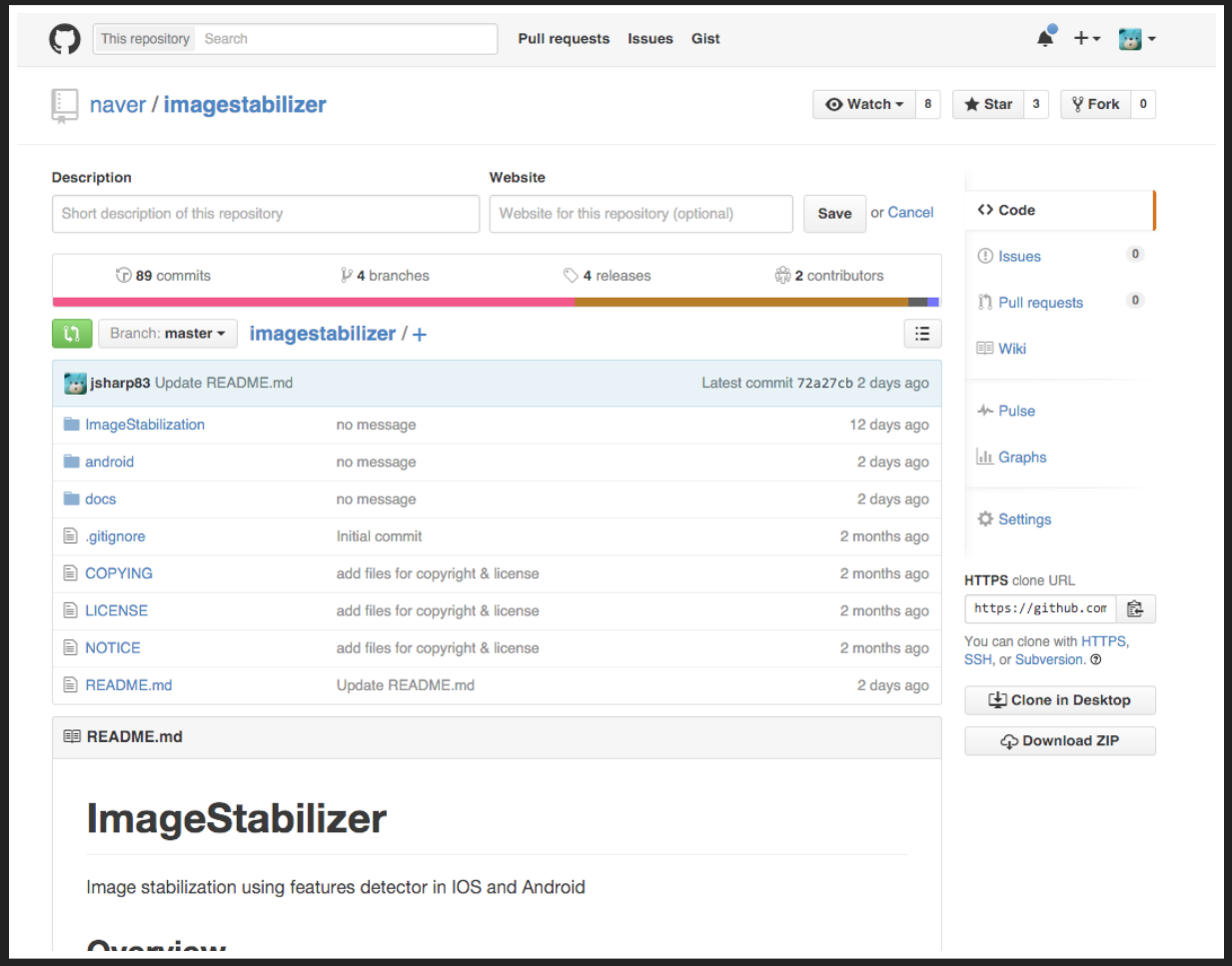
안드로이드에서 OpenCV사용하기 -> JavaCV 말고 NDK사용.

그리고

남은 문제들

속도개선(GPU 사용), 화면 울렁임(특징점을 고르게 분포시킴), 흔들림 정확도 향상(Kalman filter)

OPEN SOURCE



https://github.com/naver/imagestabilizer