KAZE Feature

※ Alcantarilla, Pablo Fernández, Adrien Bartoli, and Andrew J. Davison. "KAZE features." *European Conference on Computer Vision*. Springer Berlin Heidelberg, 2012.

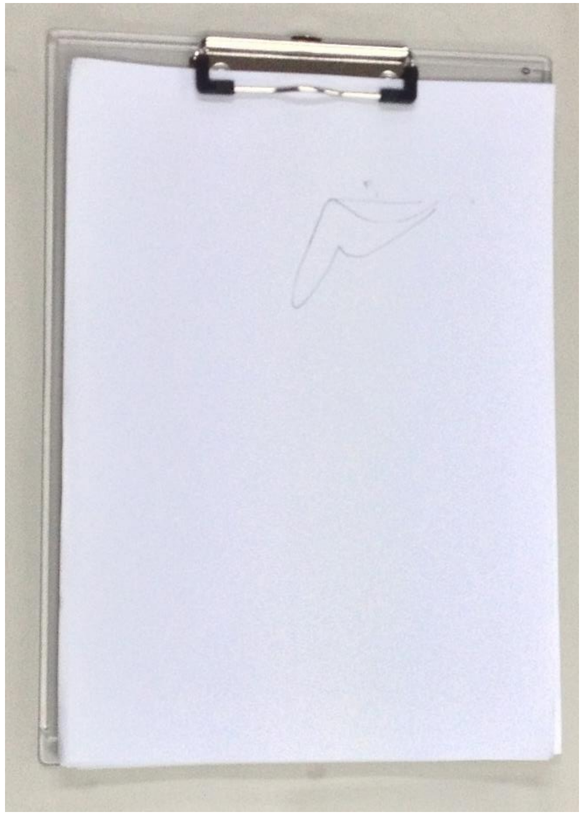
ISL
안재원

INDEX

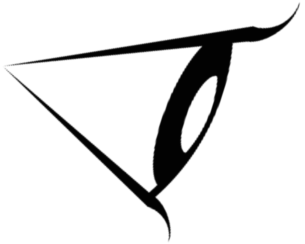
- Feature & Feature detectors
- KAZE Feature
- OpenCV
- Example Video

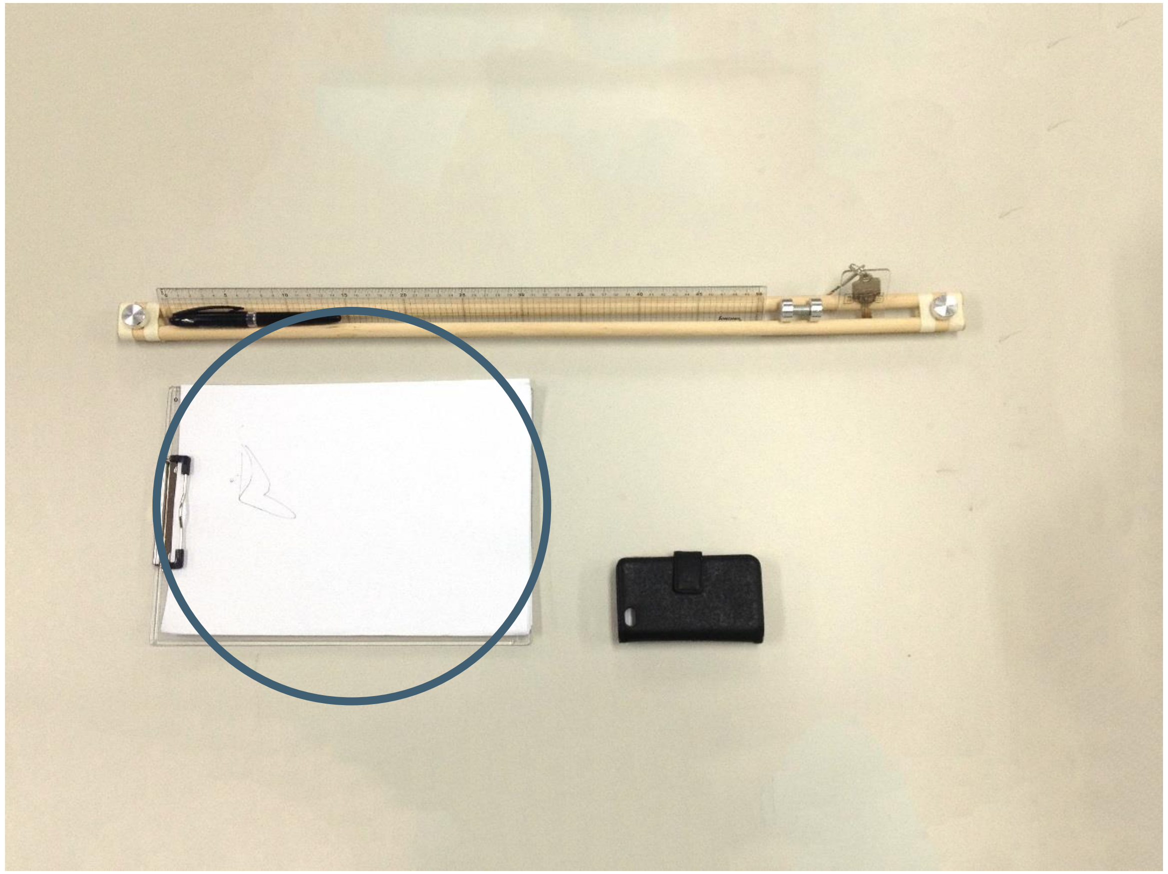
Feature & Feature detectors

Intro



Target Object

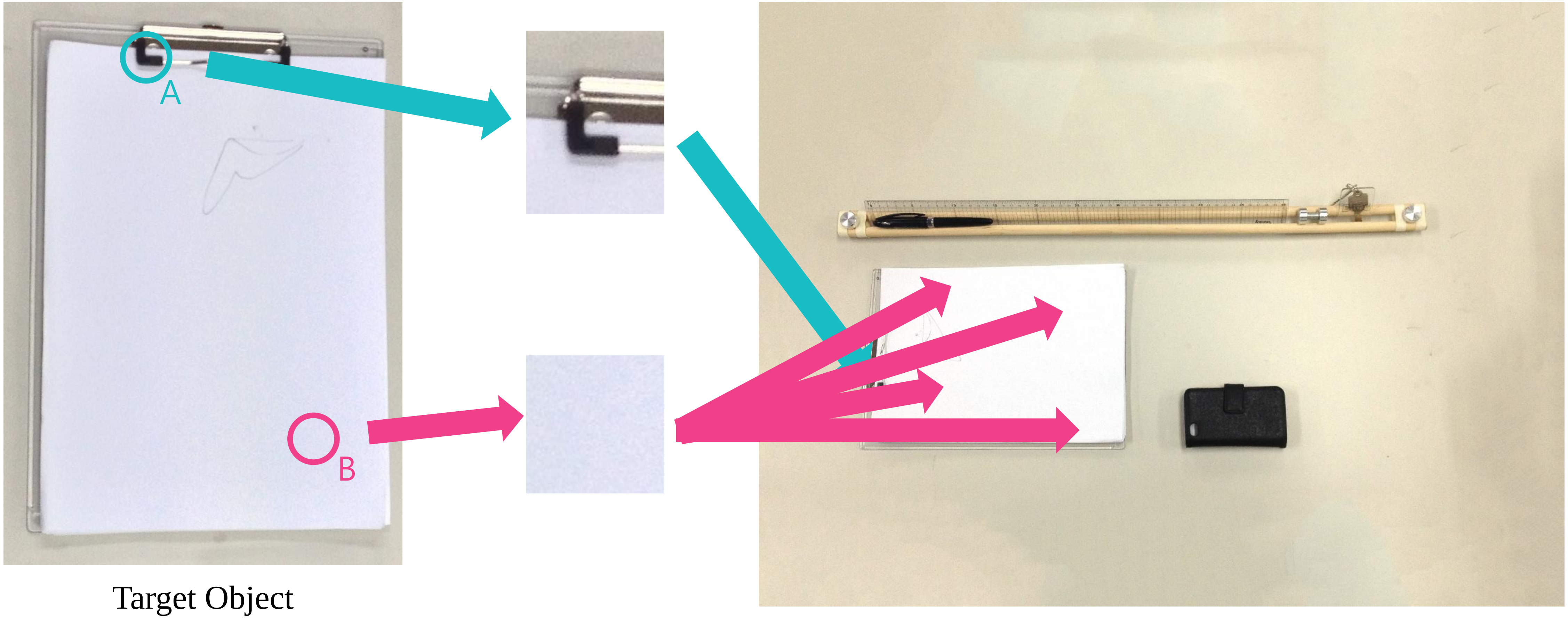




Input Image

Feature & Feature detectors

Intro



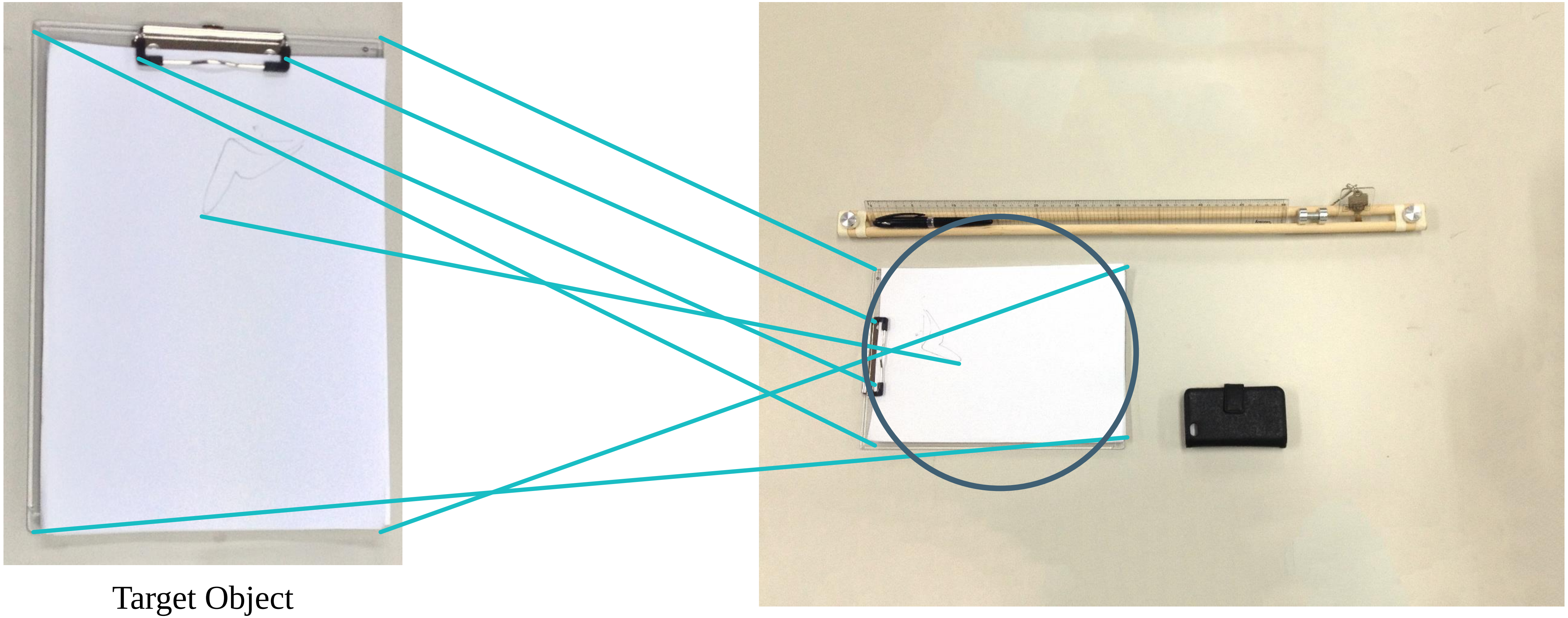
Target Object

Input Image

01

Feature & Feature detectors

Intro



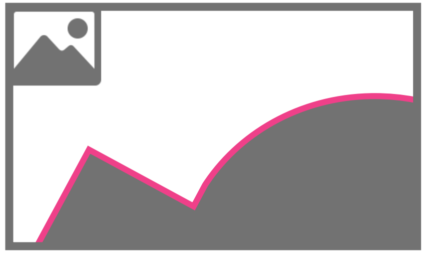
Target Object

Input Image

Feature & Feature detectors

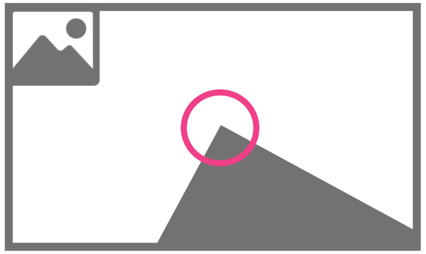
Features

Edge



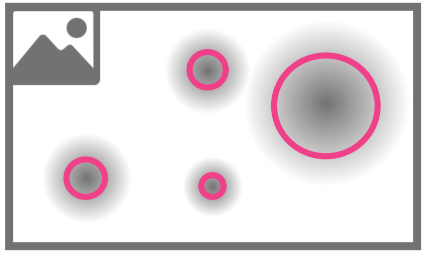
- Canny
- Sobel
- Prewitt

Corner

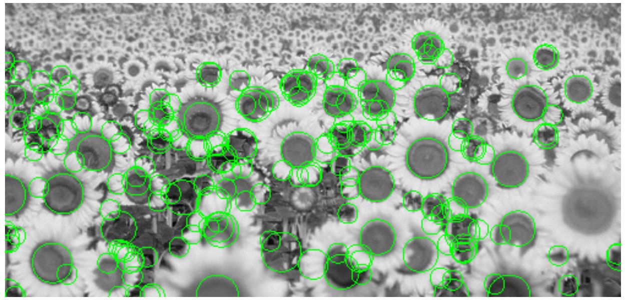


- Harris
- Shi & Tomasi
- SUSAN
- FAST

Blob

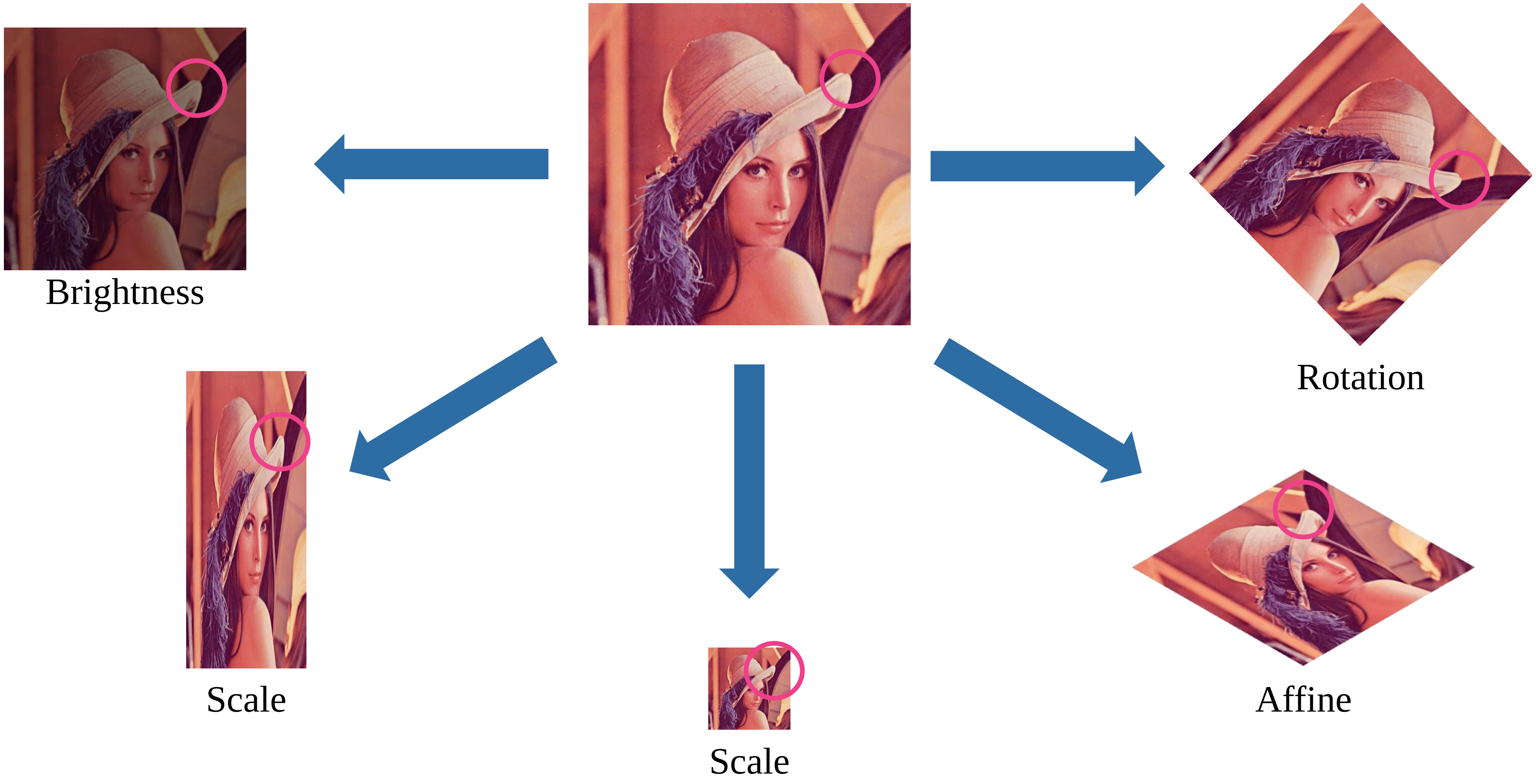


- LoG
- DoG



Feature & Feature detectors

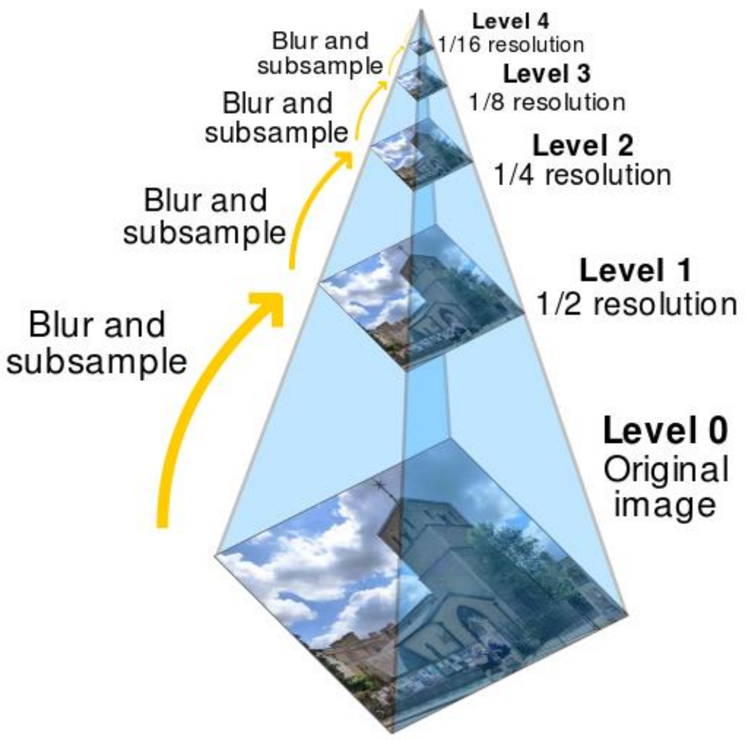
Good Feature



KAZE Feature

Intro

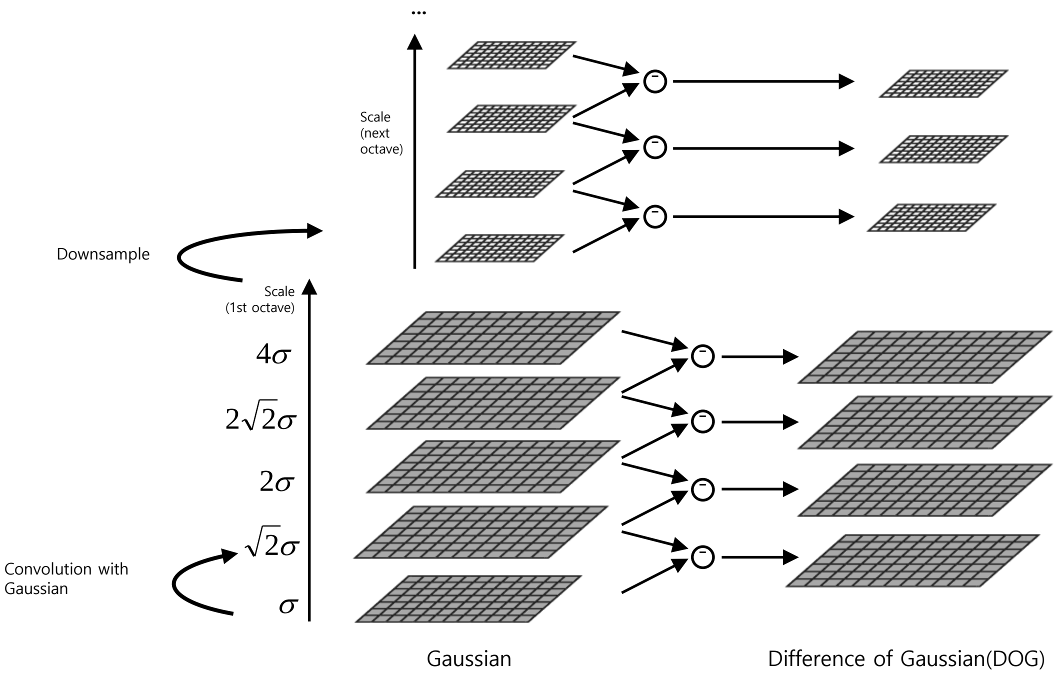
KAZE Feature is multiscale 2D feature detection and description algorithm in nonlinear scale space.



Gaussian Pyramid

Ex) SIFT

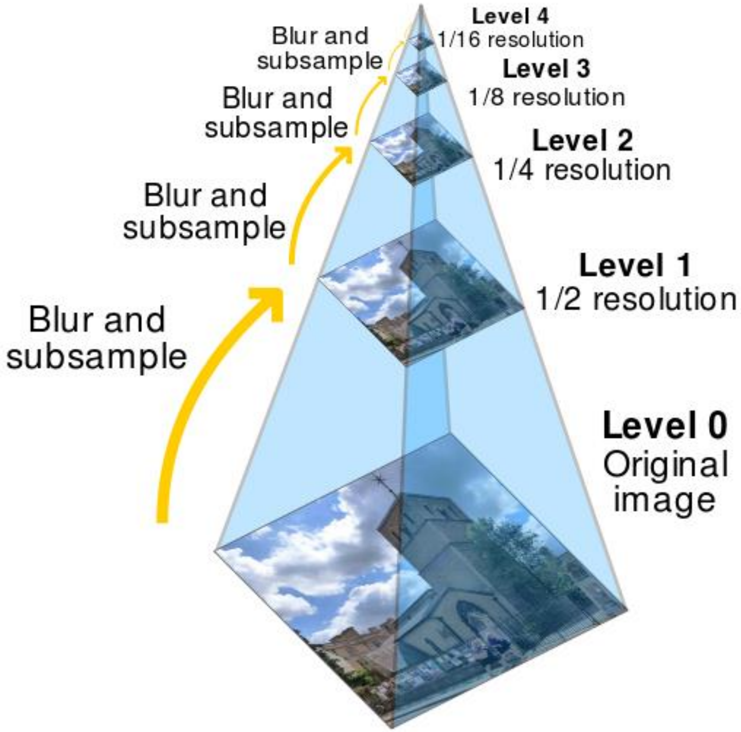
Scale-Space Extrema Detection → Accurate Keypoint Localization → Orientation Assignment → Keypoint Description



KAZE Feature

Intro

KAZE Feature is multiscale 2D feature detection and description algorithm in nonlinear scale space.

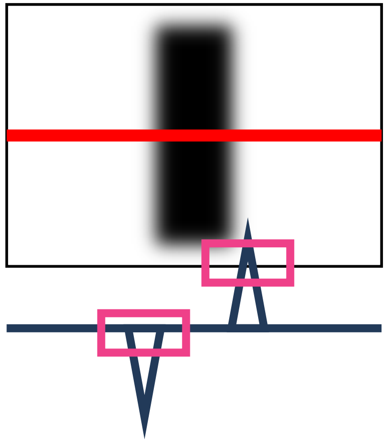


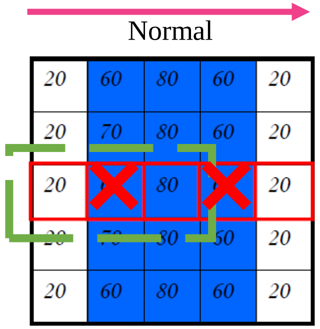
Gaussian Pyramid

- Gaussian blurring → Smooth details
→ Reducing localization accuracy & distinctiveness

Ex) Canny Edge

- Canny Enhancer
  - Smoothing
  - Finding gradients
  - Estimate edge strength & orientaion
- Non-Max Suppression
  - Choose local maxima
- Hysteresis Threshold
  - Double thresholding

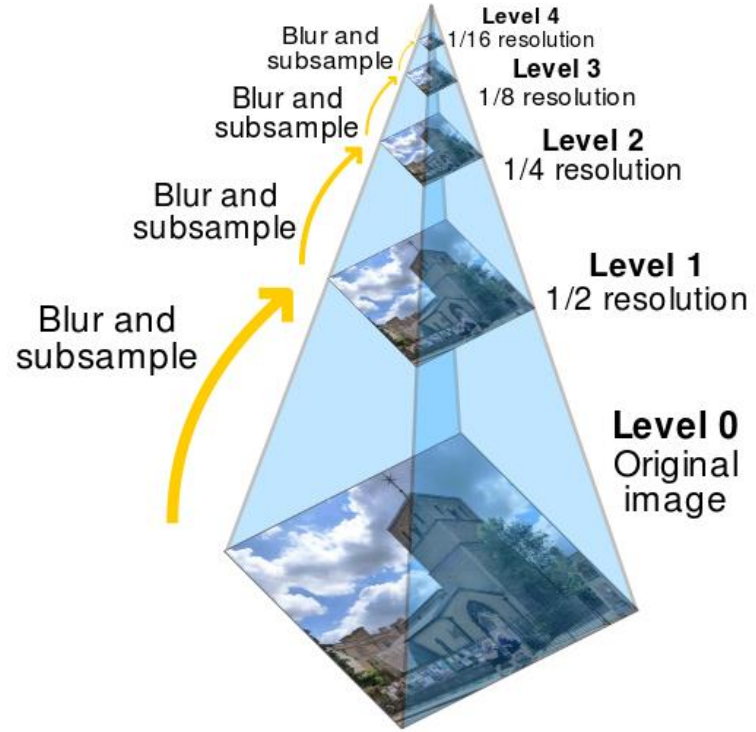




KAZE Feature

Intro

KAZE Feature is multiscale 2D feature detection and description algorithm in nonlinear scale space.



Gaussian Pyramid

- Adaptive blurring(nonlinear diffusion filtering)

KAZE Feature

Nonlinear Diffusion Filtering

※ Perona, Pietro, and Jitendra Malik. "Scale-space and edge detection using anisotropic diffusion." *IEEE Transactions on pattern analysis and machine intelligence* 12.7 (1990): 629-639.

- Perona and Malik diffusion equation

$$\frac{\partial L}{\partial t} = div(c(x,y,t)\cdot \nabla L)$$

→ Scale parameter $t = \frac{1}{2}\sigma^2$

→ Scalar function of image position

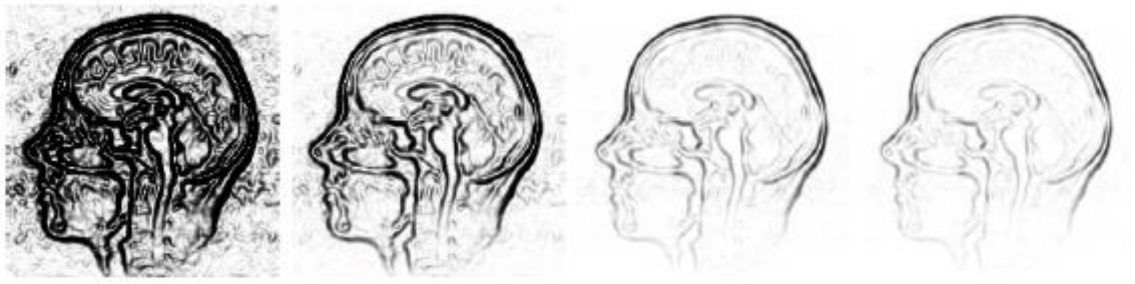
$$c(x,y,t) = g(|\nabla L_\sigma (x,y,t)|)$$

$$g_1 = \exp\left(-\frac{|\nabla L_\sigma|^2}{k^2}\right)$$

→ Contrast factor(Level of diffusion)

$$g_2 = \frac{1}{1+\frac{|\nabla L_\sigma|^2}{k^2}}$$

$$g_3 = \begin{cases} 1 & ,|\nabla L_\sigma|^2 = 0 \\ 1-\exp\left(-\frac{3.315}{(|\nabla L_\sigma|/k)^8}\right) & ,|\nabla L_\sigma|^2 > 0 \end{cases}$$



$k = 5.10$ $k = 12.75$ $k = 38.25$ $k = 89.25$

KAZE Feature

AOS Scheme

※Weickert, Joachim, BM Ter Haar Romeny, and Max A. Viergever. "Efficient and reliable schemes for nonlinear diffusion filtering." *IEEE transactions on image processing* 7.3 (1998): 398-410.

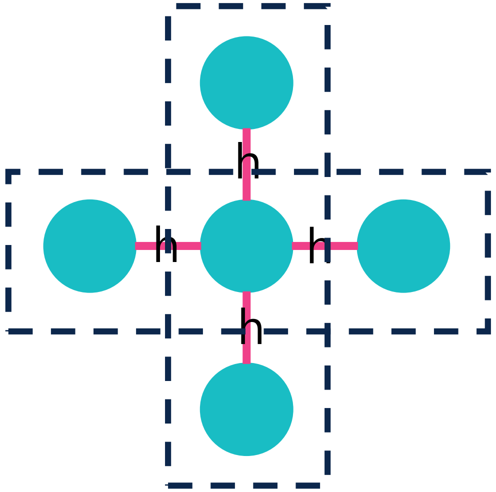
- Discretization of the diffusion equation

$$\frac{\partial L}{\partial t} = div(c(x,y,t)\cdot\nabla L) \;\longrightarrow\; \frac{L^{i+1}-L^{i}}{\tau} = \sum_{l=1}^{m} A_l(L^i)L^{i+1}$$

Image (L^i)

Coordinate의 수(x, y 2개) (l)

$$L^{i+1} = \left(I - \tau\sum_{l=1}^{m} A_l(L^i)\right)^{-1} L^i$$



- Thomas algorithm

$$A\cdot L^{i+1} = L^i$$

LU-decomposition

$$LU\cdot L^{i+1} = L^i$$

$$L\cdot y = L^i$$

$$U\cdot L^{i+1} = y$$

Tridiagonal matrix

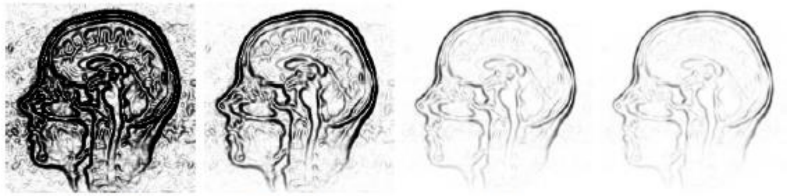
$$\begin{pmatrix} a_1 & b_1 & & & \\ c_1 & a_2 & b_2 & & \\ & c_2 & \ddots & \ddots & \\ & & \ddots & \ddots & b_{n-1} \\ & & & c_{n-1} & a_n \end{pmatrix}$$

$$\begin{cases} -\dfrac{c_i + c_j}{2h^2} & (j \in N(i)), \\ \sum_{n\in N(i)} \dfrac{c_i + c_n}{2h^2} & (j = i), \\ 0 & (else) \end{cases}$$

KAZE Feature

Computation of the Nonlinear Scale Space

Image Input → Gaussian filtering (Reduce noise) → Image gradient histogram for k → Get Multiscale space image



$k = 5.10$ $k = 12.75$ $k = 38.25$ $k = 89.25$

70% Percentile 의 값을 Contrast parameter k로 사용한다.

$$L^{i+1} = \left(I - \tau \sum_{l=1}^{m} A_l(L^i) \right)^{-1} L^i$$

$t_i = 5.12$ $t_i = 20.48$ $t_i = 81.92$ $t_i = 130.04$ $t_i = 206.42$

KAZE Feature

Feature Detection

※Brown, Matthew, and David G. Lowe. "Invariant Features from Interest Point Groups." *BMVC*. No. s 1. 2002.

※Lindeberg, Tony. "Feature detection with automatic scale selection."*International journal of computer vision* 30.2 (1998): 79-116.

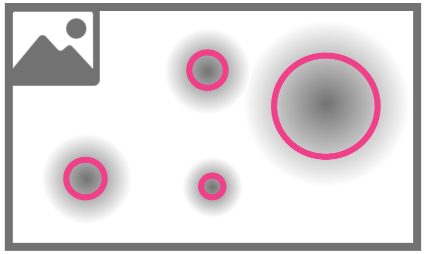
- Hessian Matrix

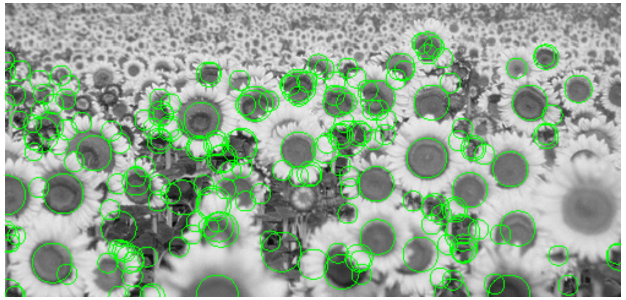
$$H = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{bmatrix}$$

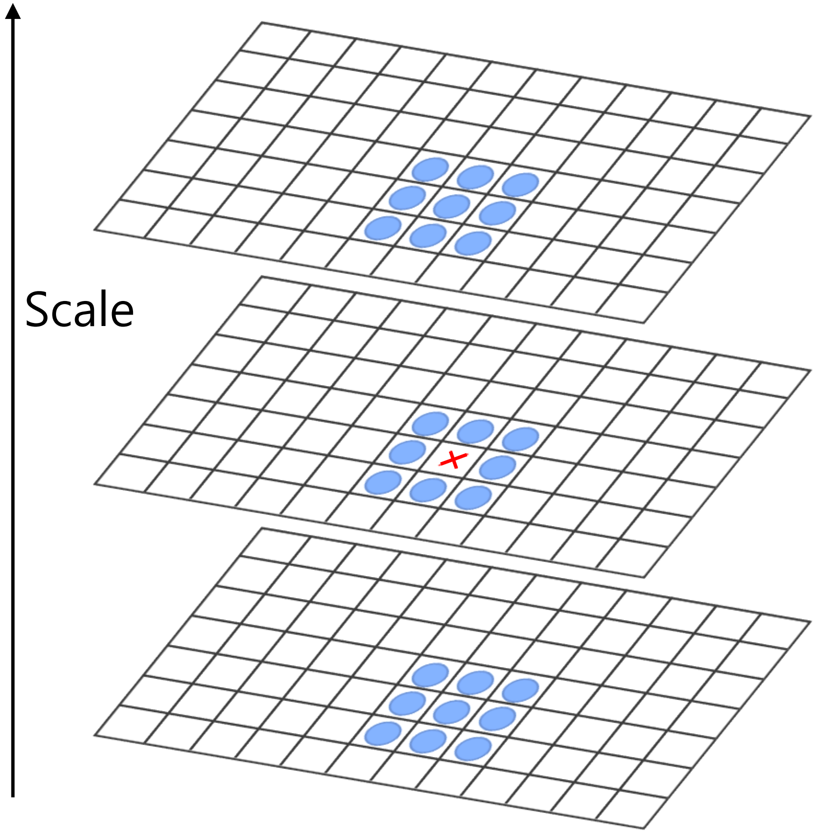
Local minimum : $D > 0$ *and* $L_{xx} > 0$

Local maximum : $D > 0$ *and* $L_{xx} < 0$

Blob







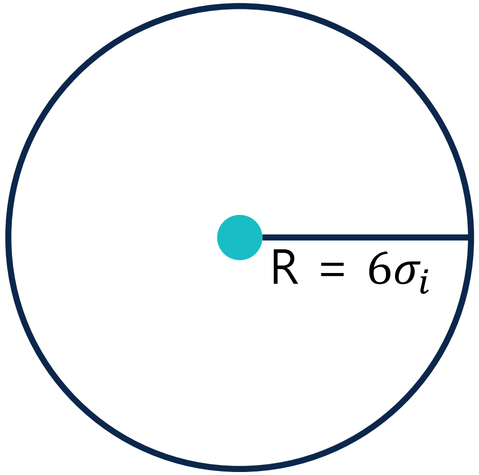
$$L_{Hessian} = \sigma^2 (L_{xx} L_{yy} - L_{xy}^2)$$

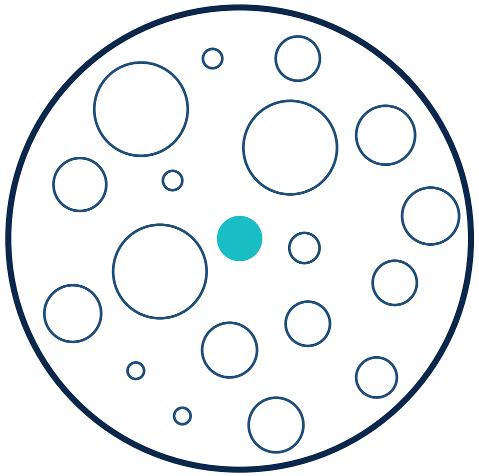
KAZE Feature

Feature Description

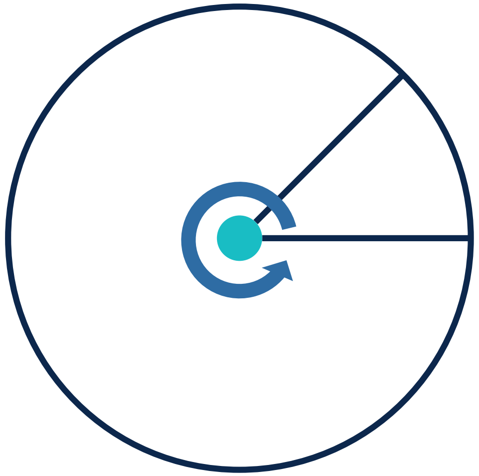
- Finding the Dominant Orientation

Find Circular area of radius 6σ → Get Gaussian weight → Sliding area & To compare value → Finding the dominant orientation

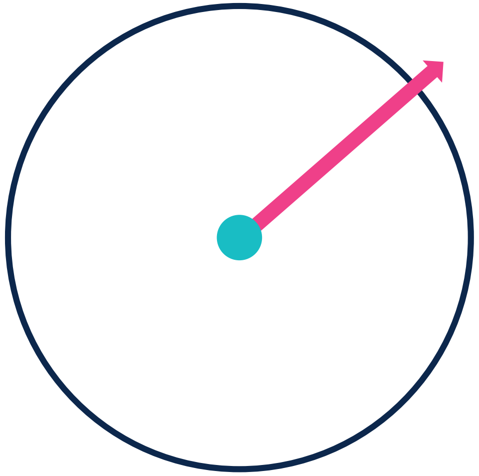




$$G = e^{-\frac{dx^2+dy^2}{2\sigma_i}}$$



$$\left(\sum GL_x\right)^2 + \left(\sum GL_y\right)^2$$



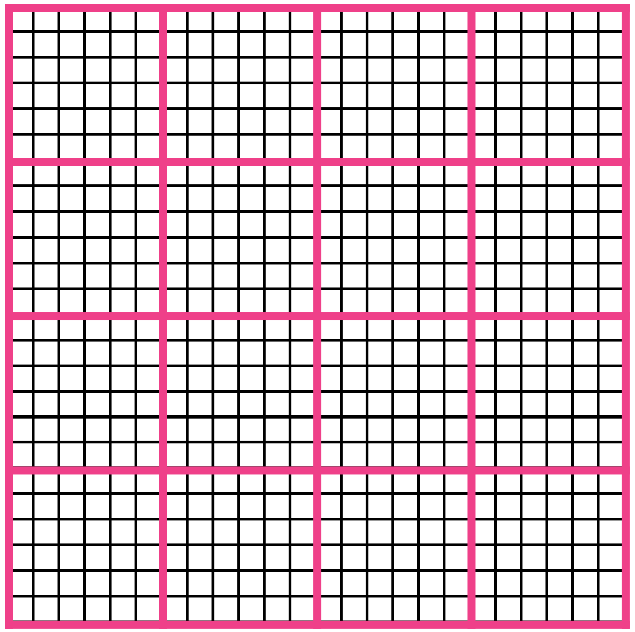
$$O = \tan^{-1}\left(\frac{\sum GL_y}{\sum GL_x}\right)$$

KAZE Feature

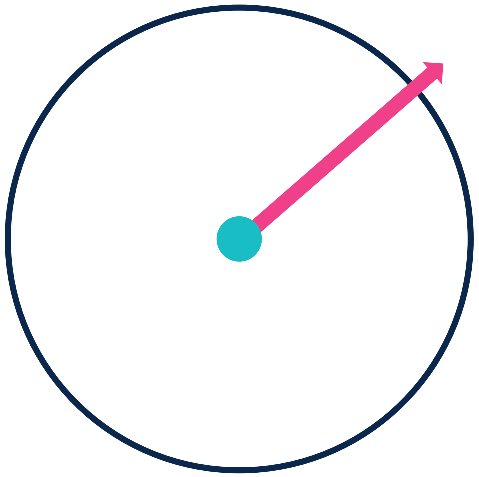
Feature Description

- Building the Descriptor

$L_x \quad L_y$



$24\sigma \times 24\sigma$



- This grid is divided into 4x4 sub-regions.
- Opverlap of 2σ

- Descriptor vector

$$d_v = \left[G_c \sum GL_x, G_c \sum GL_y, G_c \sum \left| GL_x \right|, G_c \sum \left| GL_y \right| \right]$$

02

KAZE Feature

Experimental Result

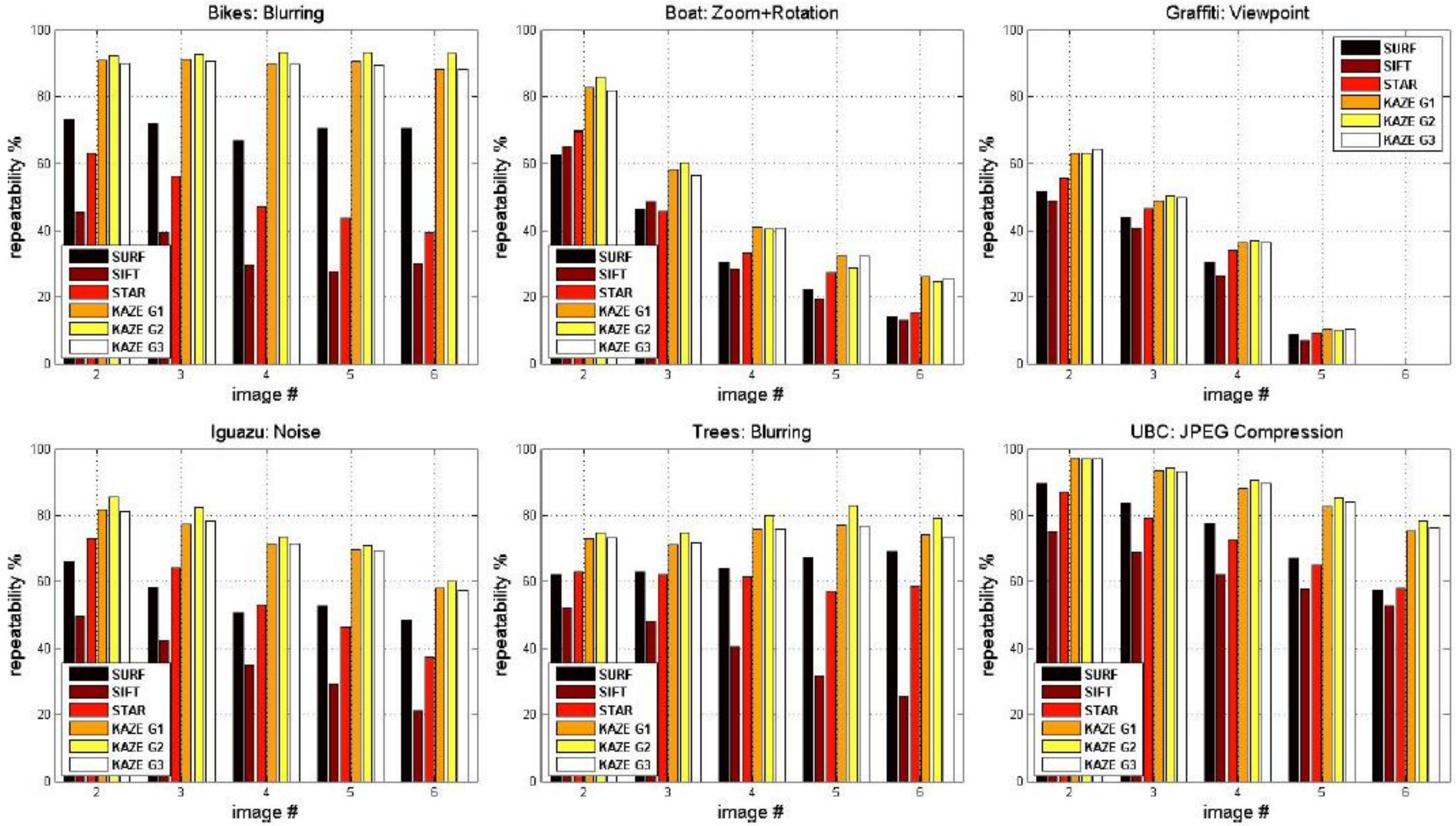
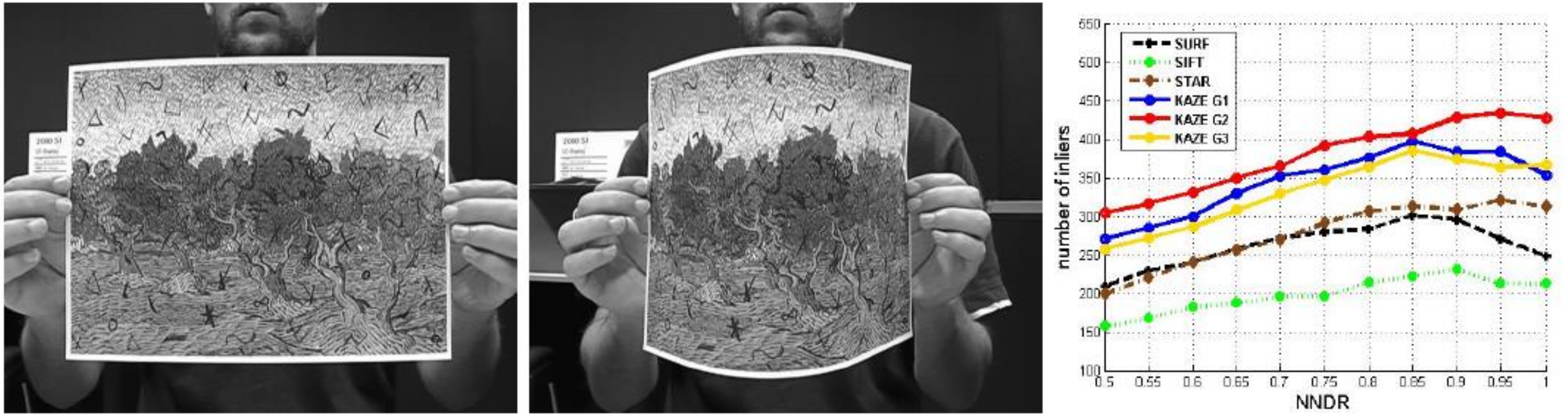


Image System Laboratory

02

KAZE Feature

Experimental Result



(a) (b) (c)

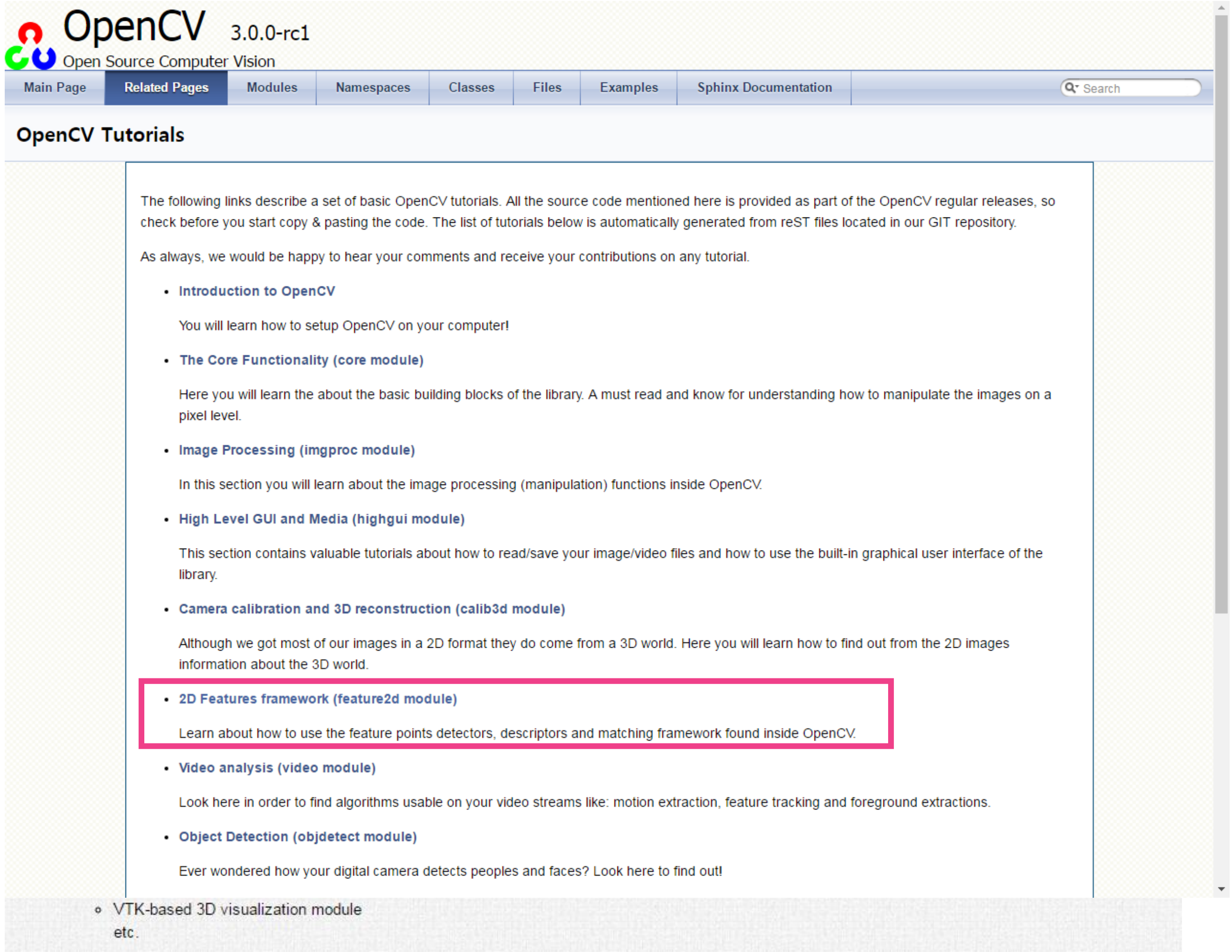
KAZE Feature

Experimental Result

KAZE	UBC 1	Trees 6
Nonlinear Scale Space	1.14	1.53
Feature Detection	0.68	0.93
Feature Description	0.38	0.20
Total Time	2.20	2.66
SURF	0.89	0.63
SIFT	2.66	2.77
STAR	0.25	0.32
Image Resolution	800×640	1000×700
Number of Keypoints	1463	765

OpenCV

OpenCV 3.x


OpenCV 3.0.0-rc1
Open Source Computer Vision

Main Page | Related Pages | Modules | Namespaces | Classes | Files | Examples | Sphinx Documentation | Search

OpenCV Tutorials

The following links describe a set of basic OpenCV tutorials. All the source code mentioned here is provided as part of the OpenCV regular releases, so check before you start copy & pasting the code. The list of tutorials below is automatically generated from reST files located in our GIT repository.

As always, we would be happy to hear your comments and receive your contributions on any tutorial.

- **Introduction to OpenCV**

  You will learn how to setup OpenCV on your computer!
- **The Core Functionality (core module)**

  Here you will learn the about the basic building blocks of the library. A must read and know for understanding how to manipulate the images on a pixel level.
- **Image Processing (imgproc module)**

  In this section you will learn about the image processing (manipulation) functions inside OpenCV.
- **High Level GUI and Media (highgui module)**

  This section contains valuable tutorials about how to read/save your image/video files and how to use the built-in graphical user interface of the library.
- **Camera calibration and 3D reconstruction (calib3d module)**

  Although we got most of our images in a 2D format they do come from a 3D world. Here you will learn how to find out from the 2D images information about the 3D world.
- **2D Features framework (feature2d module)**

  Learn about how to use the feature points detectors, descriptors and matching framework found inside OpenCV.
- **Video analysis (video module)**

  Look here in order to find algorithms usable on your video streams like: motion extraction, feature tracking and foreground extractions.
- **Object Detection (objdetect module)**

  Ever wondered how your digital camera detects peoples and faces? Look here to find out!

  - VTK-based 3D visualization module

  etc.

- features2d.hpp

- KAZE/AKAZE 2종류가 추가됨.

OpenCV

OpenCV 3.x

```cpp
#include <opencv2/features2d.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>
#include <vector>
#include <iostream>

using namespace std;
using namespace cv;

const float inlier_threshold = 2.5f; // Distance threshold to i
const float nn_match_ratio = 0.8f;   // Nearest neighbor matchi

int main(void)
{
    Mat img1 = imread("graf1.png", IMREAD_GRAYSCALE);
    Mat img2 = imread("graf3.png", IMREAD_GRAYSCALE);


    Mat homography;
    FileStorage fs("H1to3p.xml", FileStorage::READ);
    fs.getFirstTopLevelNode() >> homography;

    vector<KeyPoint> kpts1, kpts2;
    Mat desc1, desc2;

    Ptr<AKAZE> akaze = AKAZE::create();
    akaze->detectAndCompute(img1, noArray(), kpts1, desc1);
    akaze->detectAndCompute(img2, noArray(), kpts2, desc2);
```

```cpp
    BFMatcher matcher(NORM_HAMMING);
    vector< vector<DMatch> > nn_matches;
    matcher.knnMatch(desc1, desc2, nn_matches, 2);

    vector<KeyPoint> matched1, matched2, inliers1, inliers2;
    vector<DMatch> good_matches;
    for(size_t i = 0; i < nn_matches.size(); i++) {
        DMatch first = nn_matches[i][0];
        float dist1 = nn_matches[i][0].distance;
        float dist2 = nn_matches[i][1].distance;

        if(dist1 < nn_match_ratio * dist2) {
            matched1.push_back(kpts1[first.queryIdx]);
            matched2.push_back(kpts2[first.trainIdx]);
        }
    }

    for(unsigned i = 0; i < matched1.size(); i++) {
        Mat col = Mat::ones(3, 1, CV_64F);
        col.at<double>(0) = matched1[i].pt.x;
        col.at<double>(1) = matched1[i].pt.y;

        col = homography * col;
        col /= col.at<double>(2);
        double dist = sqrt( pow(col.at<double>(0) - matched2[i].pt.x, 2) +
                            pow(col.at<double>(1) - matched2[i].pt.y, 2));

        if(dist < inlier_threshold) {
            int new_i = static_cast<int>(inliers1.size());
            inliers1.push_back(matched1[i]);
            inliers2.push_back(matched2[i]);
            good_matches.push_back(DMatch(new_i, new_i, 0));
        }
    }

    Mat res;
    drawMatches(img1, inliers1, img2, inliers2, good_matches, res);
    namedWindow("input1", WINDOW_AUTOSIZE);
    namedWindow("input2", WINDOW_AUTOSIZE);


    imshow("input1", img1);
    imshow("input2", img2);

    imshow("res.png", res);

    waitKey();

    return 0;
}
```

- detectAndCompute(input, mask, key-point, descriptor)

OpenCV

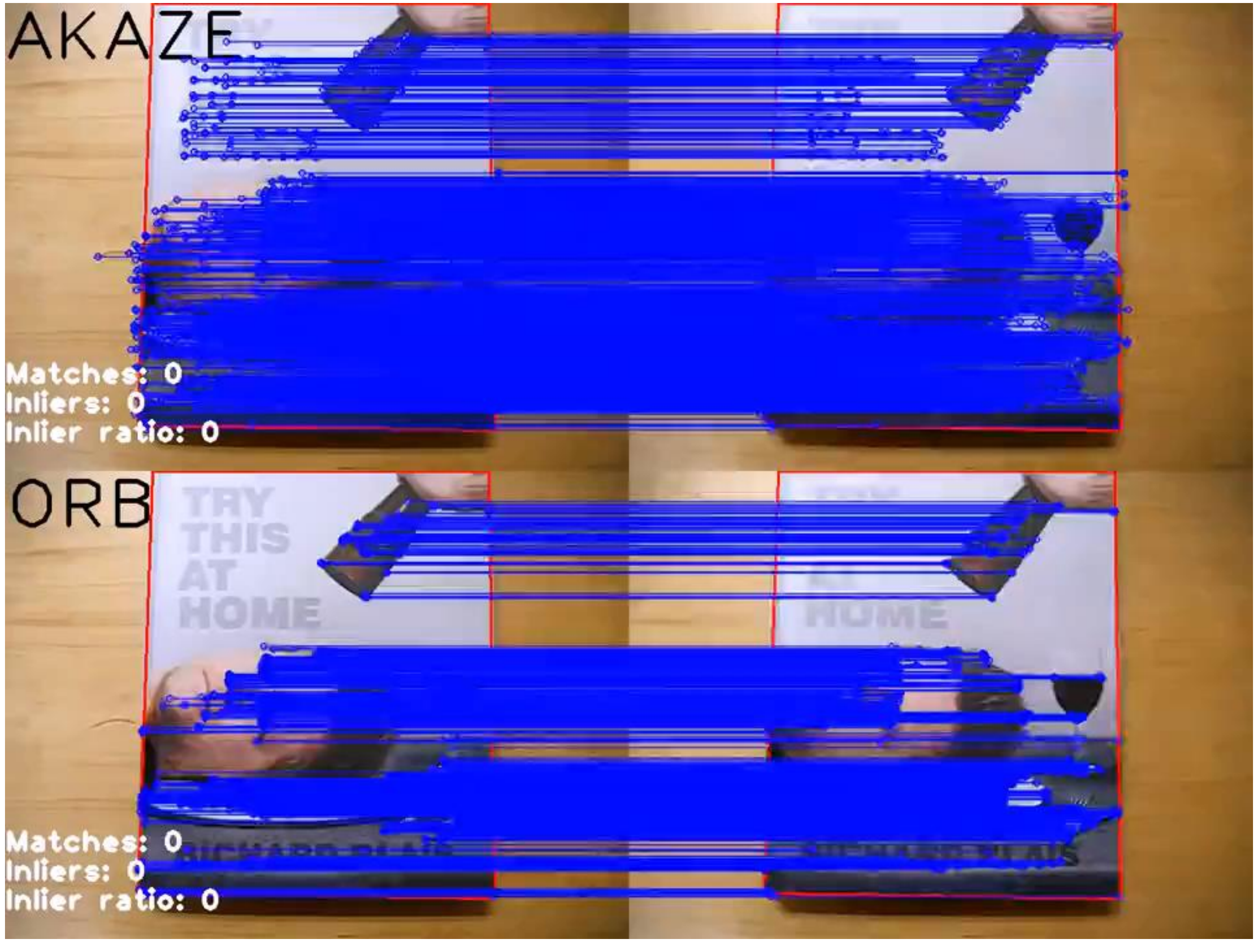
OpenCV 3.x

Example Video

04

A-Kaze vs ORB Feature

Accelerated-KAZE



- Matches 626
- Inlier ratio 0.58
- Keypoints 1117

- Matches 504
- Inlier ratio 0.56
- Keypoints 1112
- faster

Q & A