영상 처리

OpenCV_intro

서울,동부, 전남,ICT Y-A, KIM

Overview

1. Introduction
2. Modules
3. Basic structures
4. Basic routines

1. Introduction

What is OpenCV?

- **OpenCV: Open Source Computer Vision & Machine Learning software library**
  - Created in 1999 by Intel
  - Supported from 2008 by WillowGarage (a SME dedicated to hardware & open source software for personal robotics applications)
- **Cross-platform**
  Windows | Linux | Android | Mac OS | iOS …
- **Free under BSD License**
  Commercial & non-commercial applications

What is OpenCV?

- **Written in C / C++ , Js, java, Python**
  - Stable source code — **opencv.org/releases**
  - Developments
    - > Source code — **github.com/opencv**
    - > Wiki — **github.com/opencv/opencv/wiki**
- **APIs available for a variety of programming languages**

- **Current stable version is 4.8.0**

What is OpenCV?

- **Online documentation** docs.opencv.org
  Reference | Tutorials | QuickStart | Examples
- **Online resources** opencv.org > Resources
  Books | Publications | Useful links
- **Q&A Forum** answers.opencv.org

Installing OpenCV

- **Download page**

  OpenCV: Introduction to OpenCV

  - Precompiled distributions available for standard OS

2. Modules

Main packages

- **core** ▶ Base data structures & core routines
- **imgproc** ▶ Image processing routines
  - Linear / nonlinear image filtering
  - Geometric image transforms
  - Shape descriptors
  - Basic image operators
  - Histograms
  - Basic feature detection
- **video** ▶ Video analysis routines
  - Motion estimation
  - Motion segmentation
  - Background subtraction
  - Object tracking

Main packages

- **calib3D** ▶ Basic multiple-view geometry algorithms
  - Single / stereo camera calibration
  - Object pose estimation
  - Stereo correspondence
  - 3D reconstruction
- **features2D** ▶ 2D image features routines
  - Feature detectors
  - Descriptor extractors
  - Descriptor matchers
  - Object categorization
- **objdetect** ▶ Object detection routines
  - Detection of objects and instances of predefined classes
    e.g. faces | eyes | mugs | people | cars | ...

Main packages

- **ml** ▶ Shallow Machine Learning
  - K-Nearest Neighbors
  - Decision Trees
  - Boosting
  - Expectation - Maximization
  - Support Vector Machines
  - Random Forests
  - Multi-Layer Perceptron
  - Logistic regression
- **dnn** ▶ Deep Neural Networks
  - Standard neural layers & Layer API
  - DNN model APIs
    classification | object detection | pose estimation | segmentation | …
  - Standard framework importers
    Caffe2 | Darknet | ONNX | TensorFlow | torch | …

Main packages

- **highgui** ▶ High-level GUI
  - Simple GUI capabilities
- **imgcodecs** ▶ I/O for image files
  - Standard image codecs
- **videoio** ▶ I/O for video files | image sequences | cameras
  - Video capturing & codecs
    incl. OpenNI-compatible depth sensors (Kinect | XtionPRO | ...)
- **gapi** ▶ Graph-based execution model
  - CPU & GPU backends

| 

| 

| ...

Main packages

- **Other helper modules**
  - **FLANN** ▶ Fast Library for Approximate Nearest Neighbors
    Clustering & search in multidimensional spaces
  - **cv2** ▶ OpenCV-Python bindings
  - **photo** ▶ Computational photography
  - **stitching** ▶ Image stitching
  - **superres** ▶ Image super resolution
  - **viz** ▶ 3D visualizer
  - …

3.Basic structures

Image IOs

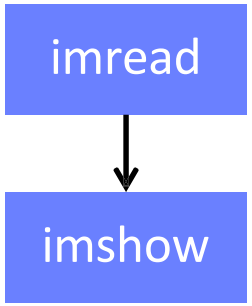
- **Loading an image:** imread()
- **Writing an image:** imwrite()
- **Displaying an image:** imshow()

Basic Example a_img_show.py

```
import cv2  as cv

img_file = "../img/apple.jpg"
img = cv.imread(img_file)

if img is not None:
  cv.imshow('MyImg', img)
  cv.waitKey()
  cv.destroyAllWindows()
else:
    print('No image file.')
```



Window routines

- Creation: **namedWindow()**

> namedWindow(string &winName, int flags)

  - flags
    - WINDOW_NORMAL — free resize by user
    - WINDOW_AUTORESIZE — fit to content [default]
    - WINDOW_OPENGL — OpenGL support

- Destruction: **destroyWindow()** **destroyAllWindows()**

> destroyWindow(string &winName)

> destroyAllWindows()

Window routines

- Move: **moveWindow ()**

> moveWindow(string &winName, int x, int y)

- Resize: **resizeWindow()**

> resizeWindow(string &winName, int width, int height)

- Update content: **updateWindow()**

> updateWindow(string &winName)

Window event handling

- Set mouse events handler: **setMouseCallback()**

```
> setMouseCallback(string &winName,
                   MouseCallback onMouse,
                   void *userdata)
```

  - onMouse mouse events handling routine with prototype

```
onMouse(int event, int x, int y, int flags, void *userdata)
```

- Wait for & get pressed key: **waitKey ()**

```
> int waitKey (int delay = 0)
```

Graphic User Interface

- **Slider routines**
  - Create a slider widget: **createTrackbar()**

```
> createTrackbar(string &trackbarName,
                 string &parentWindowName,
                 int *value, int maxValue,
                 TrackbarCallback onChange,
                 void *userdata)
```

  - onChange slider change handling routine with prototype onChange(int value, void *userdata)

Slider minimum value is always 0

Graphic User Interface

- **Slider routines**
  - Get slider value: **getTrackbarPos()**

> getTrackbarPos(string &trackbarName, string &winName)

  - Set slider value: **setTrackbarPos()**

> setTrackbarPos(string &trackbarName, string &winName, int value)

class Mat

- 요소당 할당된 비트 수(이미지의 픽셀)
- 요소 값을 나타내는 데 사용되는 데이터 형식

- bit_depth 8 | 16 | 32 | 64
- data_type U | S | F
  unsigned char | signed short integer | float
- nb_channels 1 | 2 | 3 | … | 512

class Mat = Numpy

- **Examples**

(2x3) single channel array with 8-bit unsigned char data

```
np.zeros((2, 3), dtype=np.uint8)
```

(10x10) 3-channel array with 16-bit signed short integer data

```
np.zeros((10, 10, 3), dtype=np.int16)
```

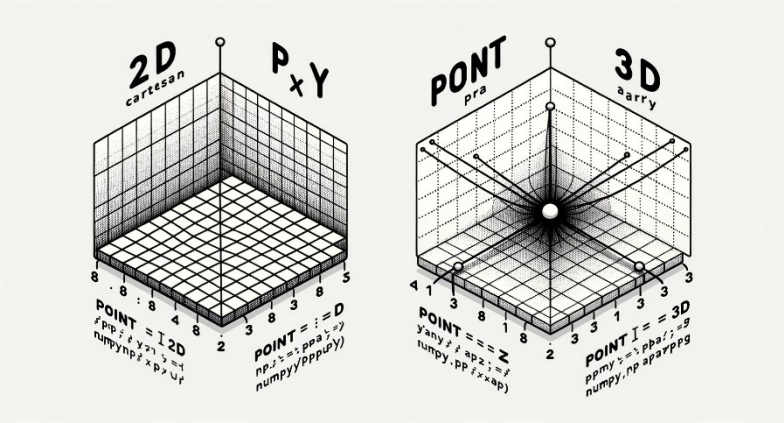
(67x53) 15-channel array with 64-bit float data

```
np.zeros((67, 53, 15), dtype=np.float64)
```

2D / 3D point = tuple , numpy array

point2D = (x, y) or point2D = np.array([x, y])

point3D = (x, y, z) or point3D = np.array([x, y, z])



Video IOs

- **Video from file / device** — **Class VideoCapture**
  - Open file / device — **open()**
  > bool open(string &fileName)
  > bool open(int device)
  - Check for initialization — **isOpened()**
  > bool isOpened()
  - Get / Set device property — **get() / set()**
  > double get(int propertyID)
  > bool set(int propertyID, double value)
  - Close file / device — **release()**
  > void release()

Video IOs

- **Video from file / device** — **Class VideoCapture**
  - Grab next frame — **grab()**

  > bool grab()

  - Decode grabbed frame — **retrieve()**

  > bool retrieve(Mat &image, int channel = 0)

  - Grab & decode next frame — **read()** | *operator* **>>**

  > bool read(Mat &image)

  > *operator* >> (Mat &image)

Video IOs

- **Save video** — **Class VideoWriter**
  - (Re-)initialize video writer — **open()**

> bool open(string &fileName, int fourcc, double fps,
> Size framesize, bool isColor = true)

  - fourcc — codec 4-character code > see **fourcc.org**
  - fps — frame rate

  - Check for initialization — **isOpened()**

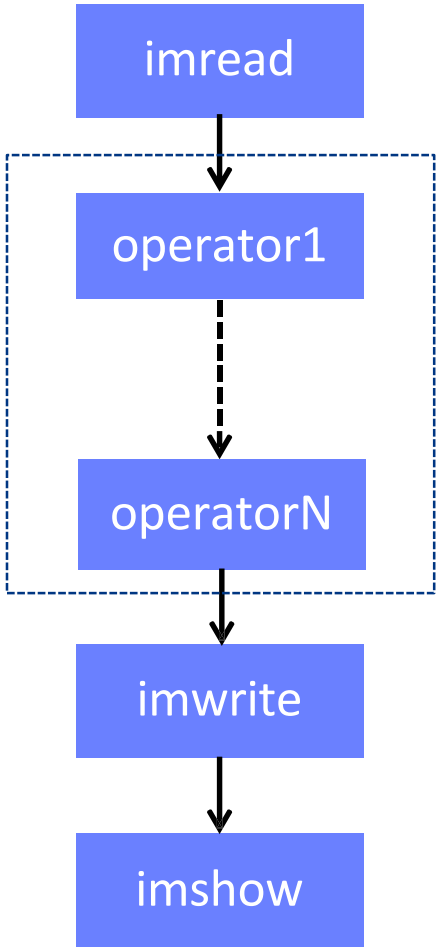
> bool isOpened()

  - Write next frame — **write()** | *operator* **<<**

> void write(Mat &image)
> *operator* << (Mat &image)

3. Basic routines

processing pipeline



- IO
- Processing
- IO
- Visualization

Image processing

- **Image linear filtering**
  - 2D image convolution: **filter2D()**

  > void filter2D(InputArray src, OutputArray dst, *parameters*)

Image processing

- **Image denoising**
  - Average filtering: **blur () / boxFilter()**

> void blur(InputArray src, OutputArray dst, *parameters*)

> void boxFilter(InputArray src, OutputArray dst, *parameters*)

  - Gaussian filtering: **gaussianBlur()**

> void gaussianBlur(InputArray src, OutputArray dst, *parameters*)

  - Bilateral filtering: **bilateral Filter()**

> void bilateralFilter(InputArray src, OutputArray dst, *parameters*)

  - Median filtering: **medianBlur()**

> void medianBlur(InputArray src, OutputArray dst, *parameters*)

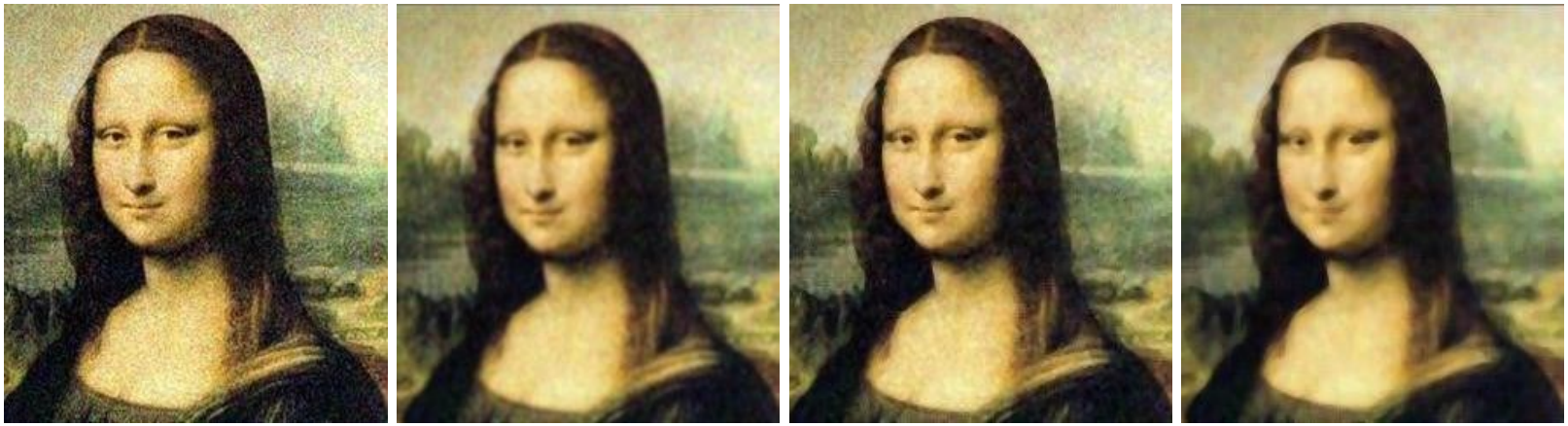
Image processing

Image denoising

cv2.GaussianBlur(src, ksize, sigmaX, sigmaY=None, borderType=cv2.BORDER_DEFAULT)

cv2.bilateralFilter(src, d, sigmaColor, sigmaSpace, dst=None, borderType=cv2.BORDER_DEFAULT)

cv2.medianBlur(src, ksize)



original Gaussian bilateral median

Image processing

Mathematical morphology

- Erosion / Dilation

cv2.erode(src, kernel, dst=None, anchor=(-1, -1), iterations=1, borderType=cv2.BORDER_CONSTANT, borderValue=cv2.morphologyDefaultBorderValue())

cv2.dilate(src, kernel, dst=None, anchor=(-1, -1), iterations=1, borderType=cv2.BORDER_CONSTANT, borderValue=cv2.morphologyDefaultBorderValue())

- Higher-order operators **morphologyEx()**

cv2.morphologyEx(src, op, kernel,

dst=None, anchor=(-1, -1),iterations=1,

borderType=cv2.BORDER_CONSTANT, borderValue=cv2.morphologyDefaultBorderValue())

`src`: Input image.

`op`: Type of a morphological operation. It could be one of the following:

- `cv2.MORPH_OPEN`: Opening
- `cv2.MORPH_CLOSE`: Closing
- `cv2.MORPH_GRADIENT`: Morphological gradient
- `cv2.MORPH_TOPHAT`: Top hat
- `cv2.MORPH_BLACKHAT`: Black hat
- `cv2.MORPH_HITMISS`: Hit or miss

`kernel`: Structuring element.

Image processing

Mathematical morphology

- Erosion / Dilation **erode () / dilate()**

cv2.erode(src, kernel, dst=None, anchor=(-1, -1), iterations=1, borderTyp
e=cv2.BORDER_CONSTANT, borderValue=cv2.morphologyDefaultBorderV
alue())

cv2.dilate(src, kernel, dst=None, anchor=(-1, -1), iterations=1, borderTyp
e=cv2.BORDER_CONSTANT, borderValue=cv2.morphologyDefaultBorderV
alue())

original erosion dilation

Image processing

- **Edge detection**
  - Sobel 2D gradient filter: **sobel()**

> sobel(InputArray src, OutputArray dst, *parameters*)

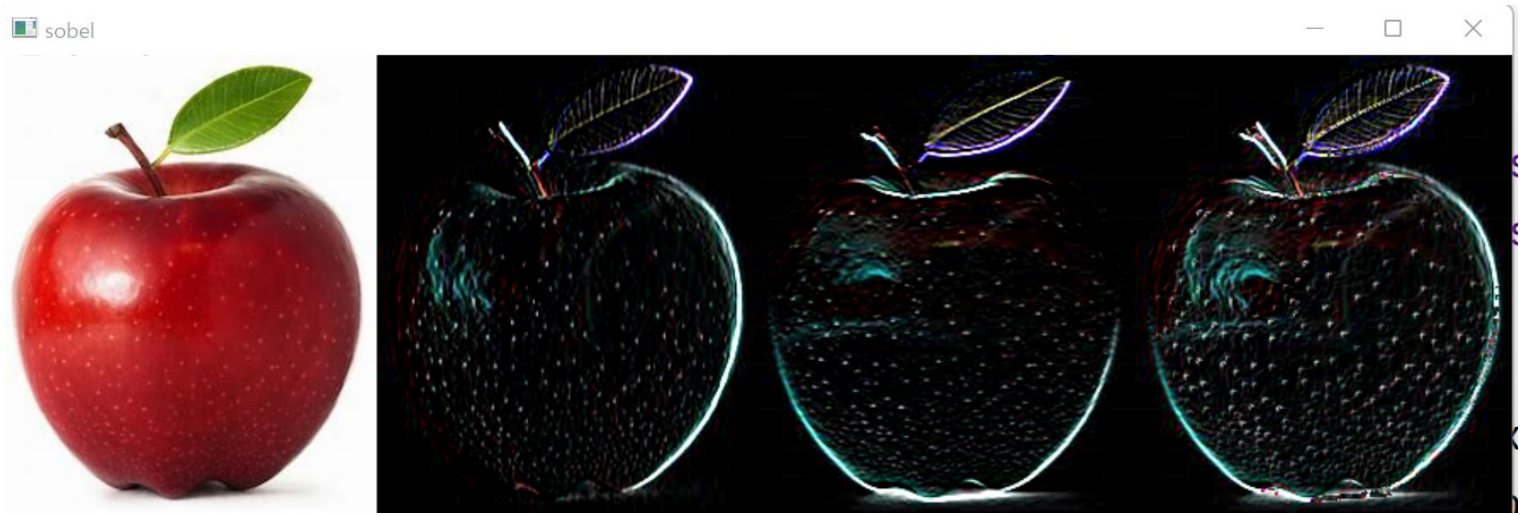


Image processing

- **Edge detection**
  - 2D Laplacian filter: **laplacian()**

> laplacian(InputArray src, OutputArray dst, *parameters*)

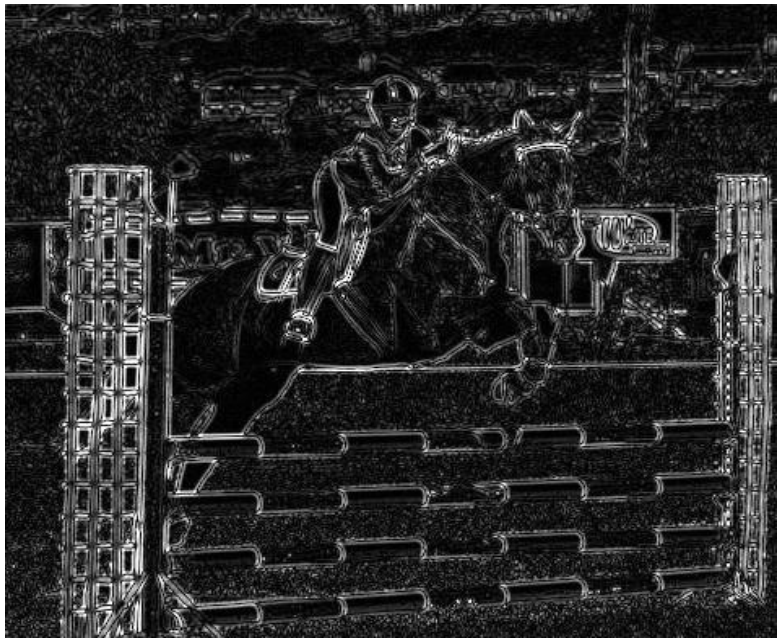


Image processing

- **Edge detection**
  - Canny 2D edge detector: **Canny()**
    Canny gradient filter + hysteresis threshold

Image processing

- **Edge detection**
  - Chain & organize edge points into contour hierarchies: **findContours()**

Canny → findContours → drawContours

Image processing

Thresholding

- Fixed-level threshold: **threshold()**

  임계값을 초과하는 모든 픽셀 값은 지정된 값(이진 임계값의 경우 일반적으로 255)으로 설정되고 임계값 아래의 값은 0으로 설정

- Adaptive threshold: **adaptiveThreshold()**

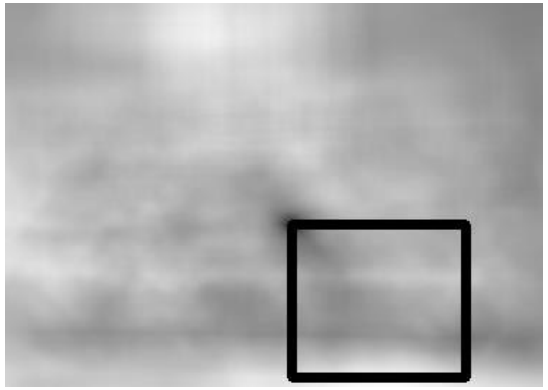
  이미지의 작은 영역에 대한 임계값을 계산하여 영역마다 다른 임계값을 허용 , 조명 조건이 다양한 이미지에 적용

Image processing

- **Histograms**
  - Histogram calculation: **calcHist()**
  - Histogram similarity: **compareHist()**
  - Histogram equalization: **equalizeHist()**

Image processing

- **Object detection**
  - Template matching: **matchTemplate()**




template

matchTemplate → minMaxLoc

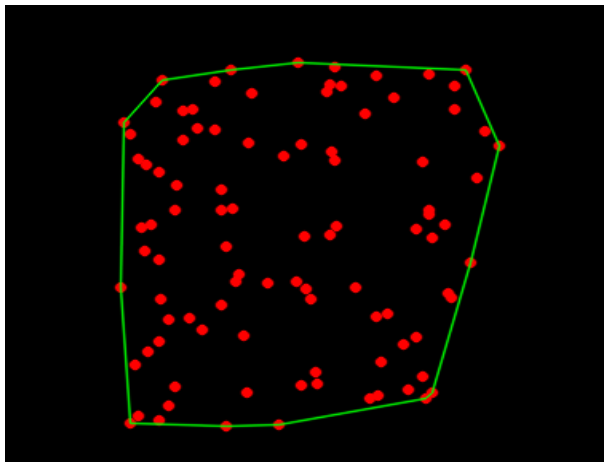
Image processing

- **Structural analysis**
  - Bounding rectangle: **boundingRect()**

  > boundingRect(InputArray pts)

  - Convex hull: **convexHull()**

  > convexHull(InputArray pts, OutputArray dst, *parameters*)



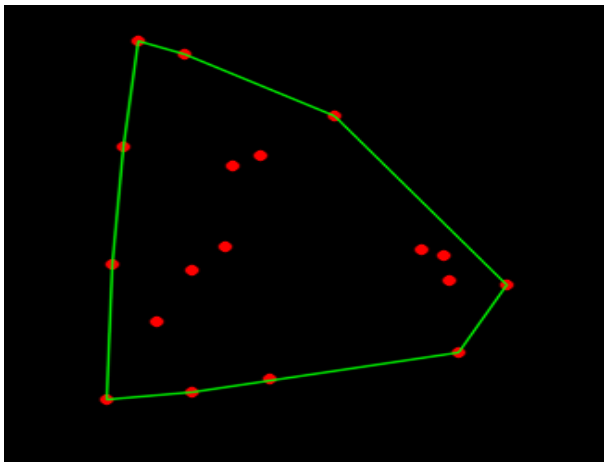


Image processing

Corner detection

- Shi-Tomasi corner map:

  cv2.goodFeaturesToTrack(image, maxCorners, qualityLevel, minDistance)

  Harris 방법을 개선한 방식으로 더욱 정확하고 견고한 코너를 검출한다.

- Harris corner map: cv2.cornerHarris

  창의 이미지 전체를 이동시킬 때의 변화량을 측정하여 코너를 찾는 방법

Image processing

- **Corner detection**
  - Harris / Shi-Tomasi detector: **goodFeaturesToTrack()**

Harris

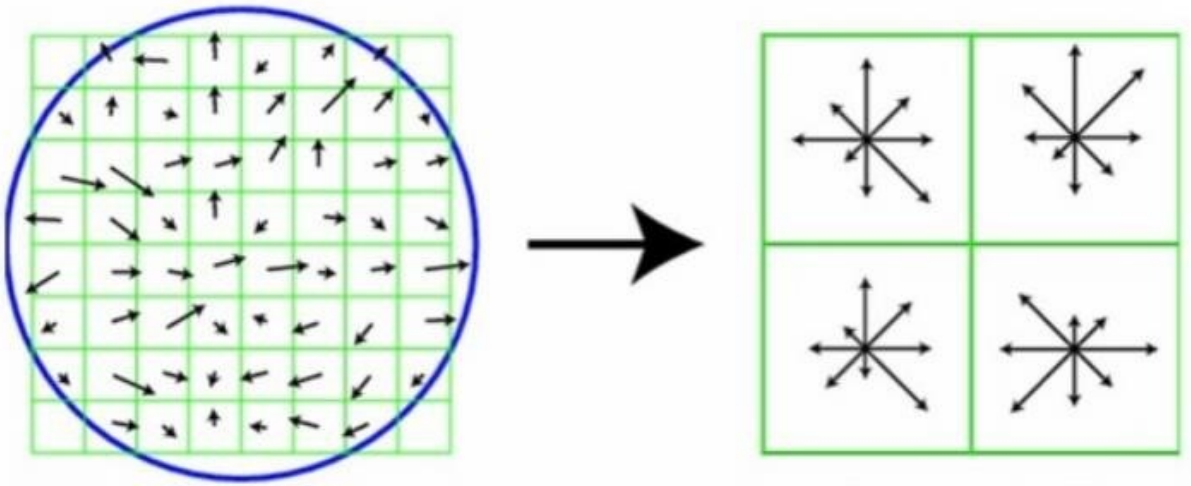
Shi-Tomasi

Feature2D

- **Feature point descriptors**

  Compute a set of parameters characterizing salient image points based their neighborhood, with various (*e.g.* scale, rotation, contrast) invariance properties

  Classes: **ORB | BRISK | FREAK | FAST | SURF | SIFT**

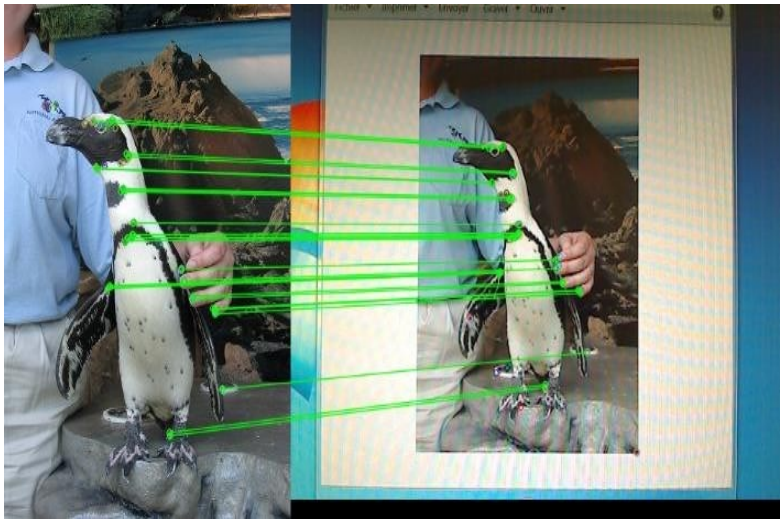


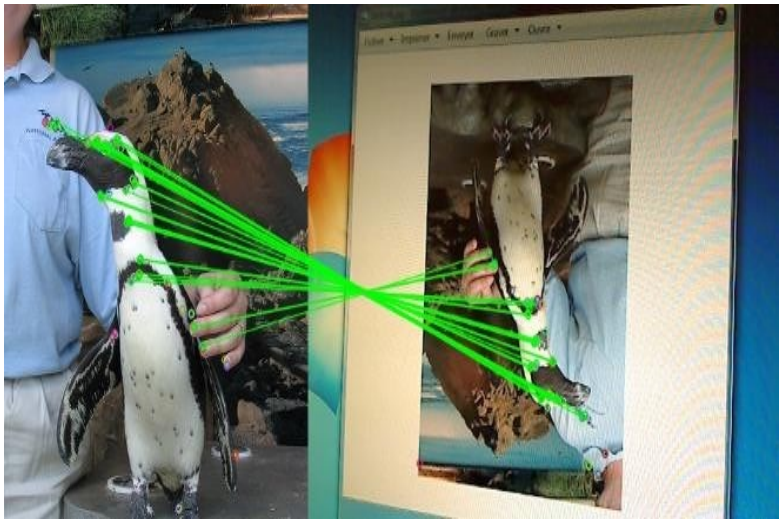
Feature2D

- **Matching descriptors**

  Match keypoint descriptor sets. Matched descriptors are represented as nD vectors comprising pairwise descriptors and best match indices

  Classes **DescriptorMatcher::(match | knnMatch | radiusMatch)**
  BFMatcher | FlannBasedMatcher





Calibration & 3D reconstruction

- **Matching 2D point sets**
  - Perspective transform estimation: **findHomography()**

  > findHomography(InputArray srcPts, InputArray dstPts, int method, *parameters*)

  – method CV_RANSAC RANSAC
  CV_LMEDS Least-Median

findHomography

↓

warpPerpective