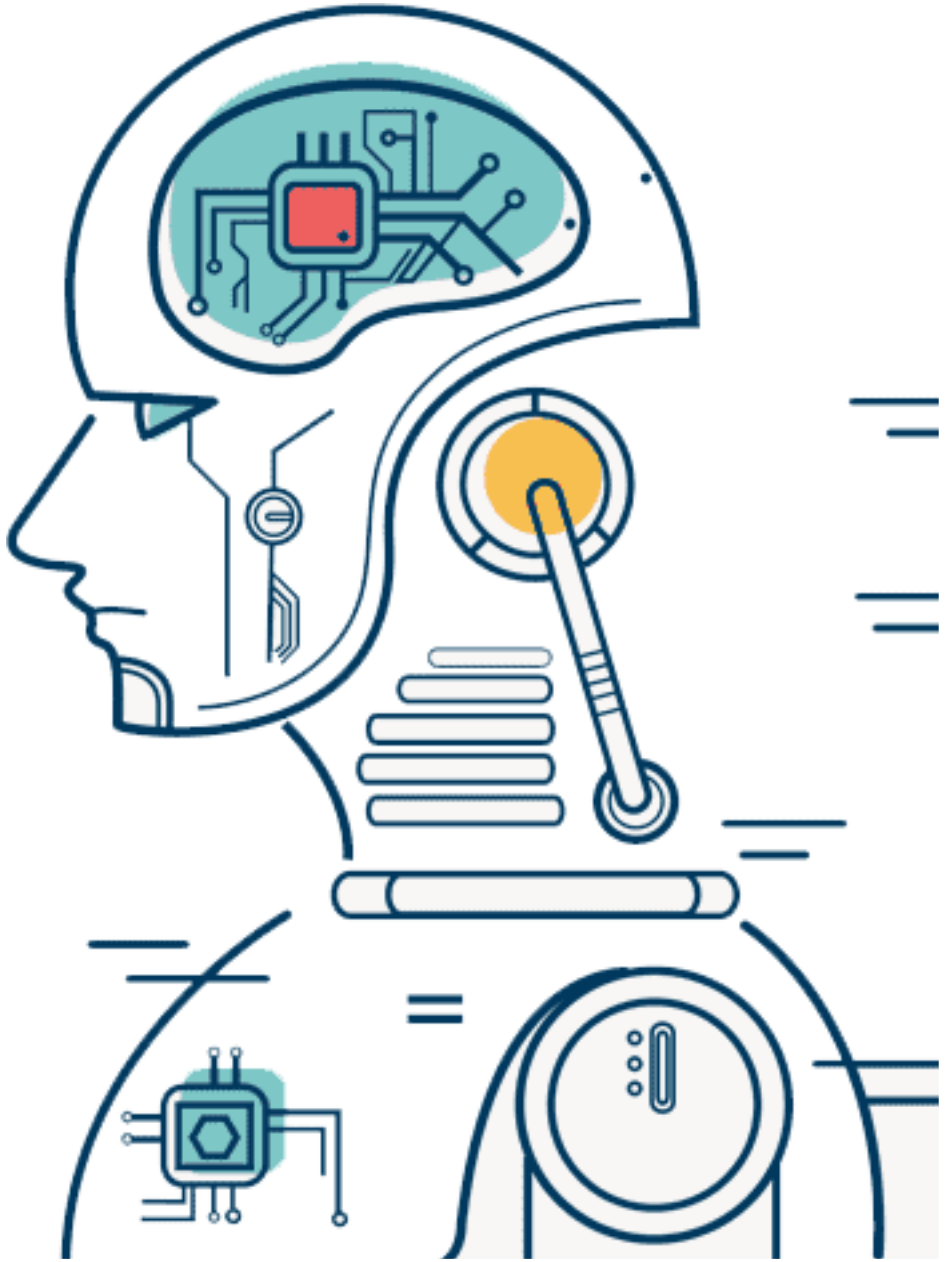
Smart Media
스마트미디어인재개발원

Deep Learning

Chapter 4 OpenCV

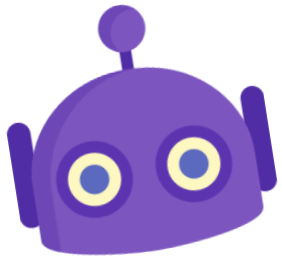


학습목표

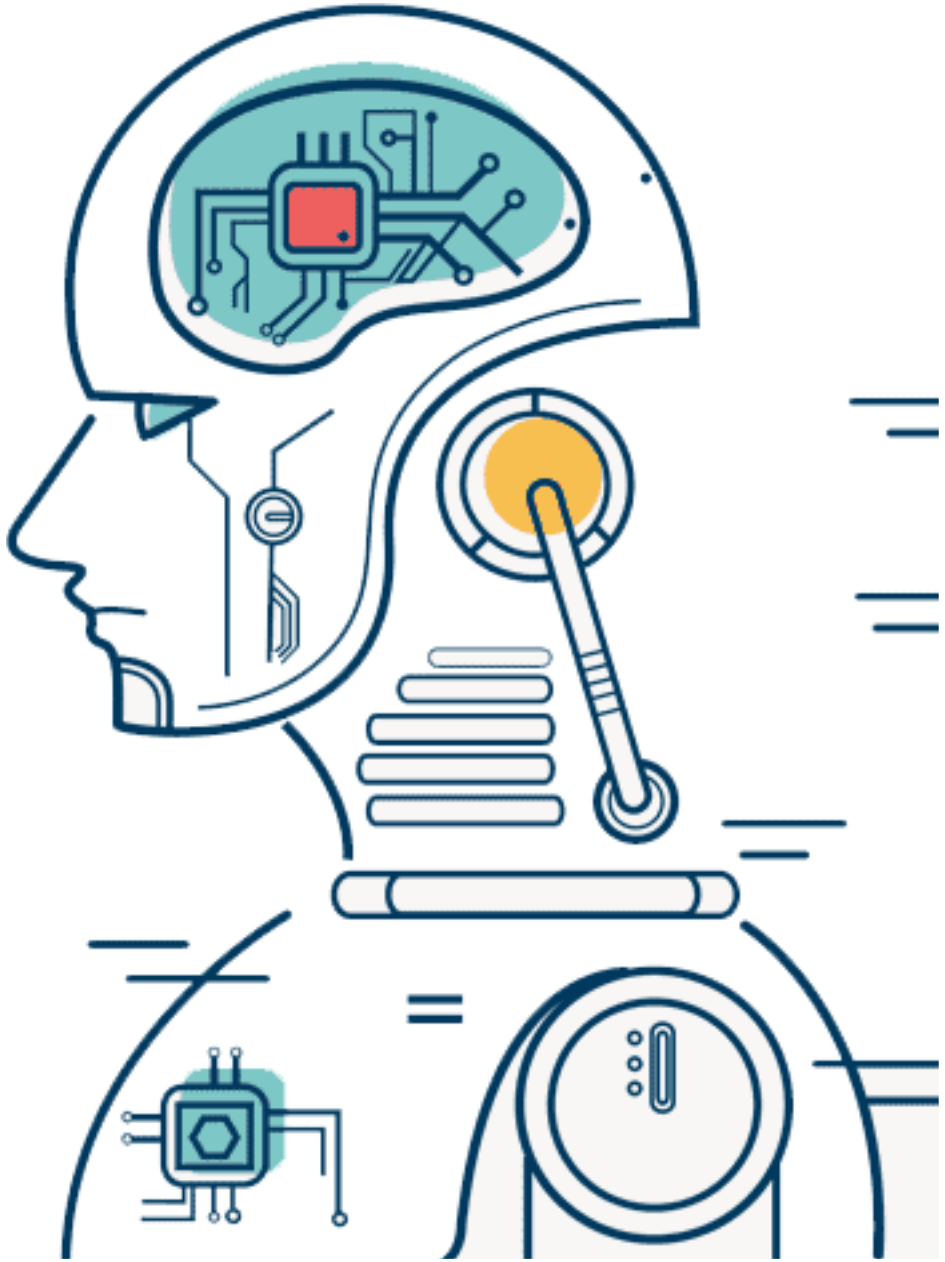
Smart Media
스마트미디어인재개발원

- OpenCV 사용법에 대해 학습한다.
- OpenCV를 활용하여 이미지 처리 방법을 학습한다

Smart Media
스마트미디어인재개발원



OpenCV와 Python 연동



OpenCV + Python 개요

Smart Media
스마트미디어인재개발원

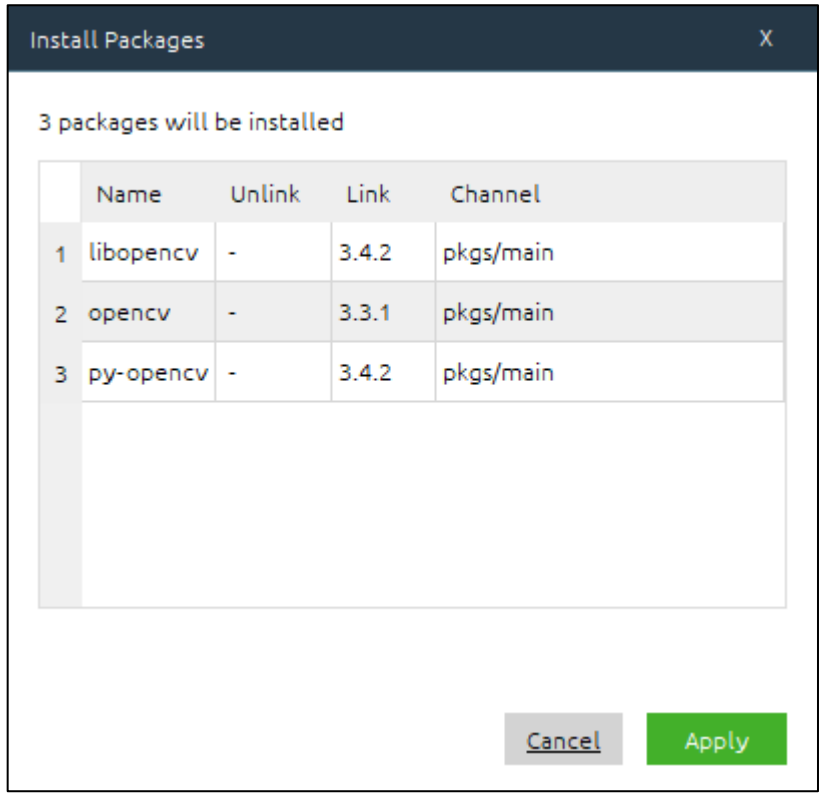
- **OpenCV** : Gray Bradsky에 의해 1996년 인텔에서 시작된 프로제토로 컴퓨터 비전, 머신러닝과 관련된 다양한 알고리즘을 지원하고 있으며 C++, Python, Java 등의 언어를 지원하고 CUDA(Compute Unified Device Architecture), OpenCL (Open Computing Language)에 기반한 인터페이스를 지원
- **OpenCV + Python** : OpenCV의 파이썬 API로 C/C++로 된 OpenCV 라이브러리들을 파이썬 래퍼로 감싼 후에 파이썬 모듈을 추가시킨 것
  - 속도문제를 해결하기 위해 속도가 문제되는 코드는 C/C++로 코딩하고 파이썬에서 불러 사용할 수 **있는 파이썬 래퍼를 제공**
  - OpenCV 배열은 Numpy 배열로 변환되어 내부 처리를 수행 (Numpy로 가능한 모든 연산 가능)
  - Numpy를 활용한 SciPy, Matplotlib 라이브러리와도 호환

https://opencv-python-tutroals.readthedocs.io/en/latest/index.html

OpenCV + Python 설치

Smart Media
스마트미디어인재개발원

- Python 3.6 이상 권장
- Anaconda Navigator에서 libopencv, opencv, py-opencv를 설치한다

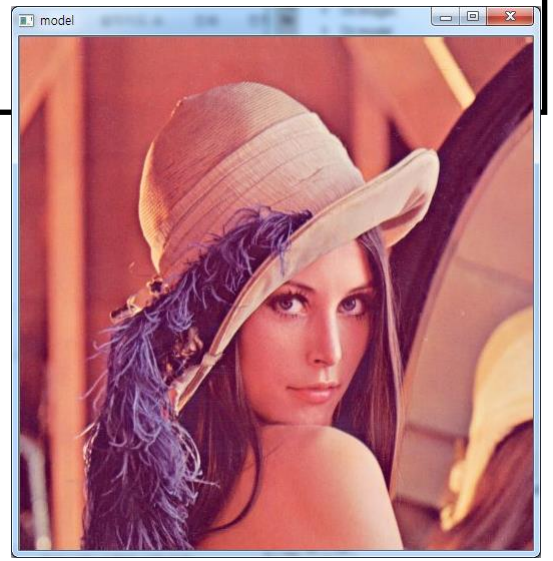


OpenCV + Python를 이용한 영상 처리

Smart Media
스마트미디어인재개발원

- 이미지 로딩 및 출력 (**http://arome.hosting.paran.com/data/deeplearning/lenna.png**)

```
1  import cv2
2
3  img = cv2.imread("images/lenna.png", cv2.IMREAD_COLOR)
4
5  cv2.imshow("model", img)
6  cv2.waitKey(0)
7  cv2.destroyAllWindows()
```



- **imread(파일명, 이미지 형식)** : 이미지 파일을 읽기 위한 객체를 리턴
- 칼라 (cv2.IMREAD_COLOR), 흑백(cv2.IMREAD_GRAYSCALE) 등
- **imshow(제목, 이미지 객체)** : 이미지 출력
- **waitKey(0)** : 키보드 입력을 기다리는 시간 설정 (0 : 계속 기다림)
- **destroyAllWindows()** : 생성한 윈도우를 모두 닫는다

- Matplotlib로 출력하기

```
1  import cv2
2  import matplotlib.pyplot as plt
3
4  img = cv2.imread("images/lenna.png", cv2.IMREAD_GRAYSCALE)
5
6  plt.imshow(img, cmap='gray', interpolation='bicubic')
7  plt.xticks([])
8  plt.yticks([])
9  plt.title('model')
10 plt.show()
```

- **interploation** : 보간법 설정 (none, nearest, bicubic 등)
- **title()** : 타이틀 표시
- **xticks(), yticks()** : 눈금 값 표시

- 비디오 파일 캡처하기

```
1   import cv2
2
3   try:
4       print('카메라를 구동합니다')
5       cap = cv2.VideoCapture(0)
6   except:
7       print("카메라 구동 실패")
8
9   cap.set(3, 480)
10  cap.set(4, 320)
11
12  while True:
13      ret, frame = cap.read()
14
15      if not ret:
16          print("비디오 읽기 오류")
17          break
18      color = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
19      cv2.imshow('video', color)
```

- 비디오 파일 캡처하기

```
20
21      k = cv2.waitKey(30)
22
23      if k == 49:
24          cap.release()
25          cv2.destroyAllWindows()
26          break
```

- 비디오 파일 캡처하기

```
5    cap = cv2.VideoCapture(0)
```

- **VideoCapture(0)** : 0 번째 카메라로부터 비디오를 캡처 (카메라가 여러 개인 경우 번호로 설정)
- **VideoCapture(파일명)** : 동영상 파일로부터 비디오를 캡처
- **http://arome.hosting.paran.com/data/deeplearning/video.mp4**

- 비디오 파일 캡처하기

```
12  while True:
13      ret, frame = cap.read()
14
15      if not ret:
16          print("비디오 읽기 오류")
17          break
18      color = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
19      cv2.imshow('video', color)
```

- **read()** : 비디오의 한 프레임을 읽음 (ret : 읽기 성공 여부, frame : 읽은 프레임)
- **cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)** : 프레임을 칼라로 변환 (흑백 : COLOR_BGR2GRAY)

- 비디오 파일 캡처하기

```
20
21      k = cv2.waitKey(30)
22
23      if k == 49:
24          cap.release()
25          cv2.destroyAllWindows()
26          break
```

- **cv2.waitKey(30)** : 0.03초간 키 입력을 대기
- **if k == 49** : 숫자 1을 눌렀다면
- **cap.release()** : 비디오 재생을 해제한다

- 비디오 녹화하기

```
1   import cv2
2
3   try:
4       print('카메라를 구동합니다')
5       cap = cv2.VideoCapture(0)
6   except:
7       print("카메라 구동 실패")
8
9   fps = 30.0
10  width = int(cap.get(3))
11  height = int(cap.get(4))
12  fcc = cv2.VideoWriter_fourcc('D', 'I', 'V', 'X')
13
14  out = cv2.VideoWriter('video01.avi', fcc, fps, (width, height))
15  print("녹화를 시작합니다")
16
17  while True:
18      ret, frame = cap.read()
19
```

- 비디오 녹화하기

```
20      if not ret:
21          print("비디오 읽기 오류")
22          break
23
24      cv2.imshow('video', frame)
25      out.write(frame)
26
27      k = cv2.waitKey(30)
28
29      if k == 49:
30          print("녹화를 종료합니다")
31          cap.release()
32          out.release()
33          cv2.destroyAllWindows()
34          break
```

- 비디오의 한 프레임을 이미지로 저장하기

```
1   import cv2
2
3   try:
4       print('카메라를 구동합니다')
5       cap = cv2.VideoCapture(0)
6   except:
7       print("카메라 구동 실패")
8
9   while True:
10      ret, frame = cap.read()
11
12      k = cv2.waitKey(30)
13      if k == 50:
14          cv2.imwrite('test.png', frame, params=[cv2.IMWRITE_PNG_COMPRESSION, 0])
15
16      cv2.imshow("frame", frame)
17      if k == 49:
18          cap.release()
19          cv2.destroyAllWindows()
20          break
```

OpenCV + Python를 이용한 영상 처리

- 비디오의 한 프레임을 이미지로 저장하기
  - 소스가 있는 폴더에 test.png 파일이 생성되었는지 확인

- 비디오의 한 프레임을 이미지로 저장하기

```
13      if k == 50:
14          cv2.imwrite('test.png', frame, params=[cv2.IMWRITE_PNG_COMPRESSION, 0])
```

- [13] 숫자 2를 누르면
- **imwrite()** : 이미지 파일을 저장
- **params=[cv2.IMWRITE_PNG_COMPRESSION, 0])** : 압축 하지 않는 PNG, 세 번째 파라미터는 압축률로 0~9까지 설정 가능 (0은 압축 없음)

- 이미지 픽셀 값 출력하기

```
1  import cv2
2
3  img = cv2.imread("images/lenna.png")
4  px = img[200, 100]
5  print(px)
```

```
[ 75  61 173]
```

- Blue (75), Green (61), Red (173) 순으로 출력

- 이미지 픽셀 값 변경하기

```
1  import cv2
2
3  img = cv2.imread("images/lenna.png")
4  img[200, 100] = [0, 0, 0]
5
6  cv2.imshow("model", img)
7  cv2.waitKey(0)
8  cv2.destroyAllWindows()
```

OpenCV + Python를 이용한 영상 처리

Smart Media
스마트미디어인재개발원

- 이미지 픽셀 값 변경하기 (RGB 채널 값을 각각 변경)

```
1  import cv2
2
3  img = cv2.imread("images/lenna.png")
4
5  img.itemset((200, 100, 1), 255)
6
7  cv2.imshow("model", img)
8  cv2.waitKey(0)
9  cv2.destroyAllWindows()
```

- itemset((픽셀y위치, 픽셀x위치, 채널), 색상레벨)
- 채널 (0 : Blue, 1 : Green, 2 : Red)

- 이미지 속성 출력 (크기 (색상 형식), 픽셀 수, 데이터 타입)

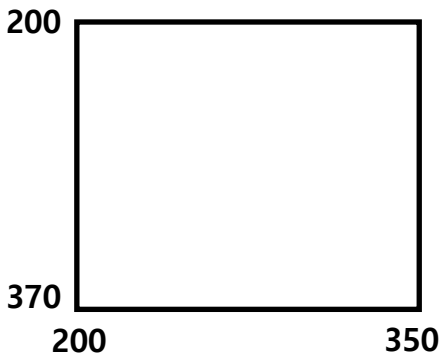
```
1  import cv2
2
3  img = cv2.imread("images/lenna.png")
4
5  print(img.shape)
6  print(img.size)
7  print(img.dtype)
```

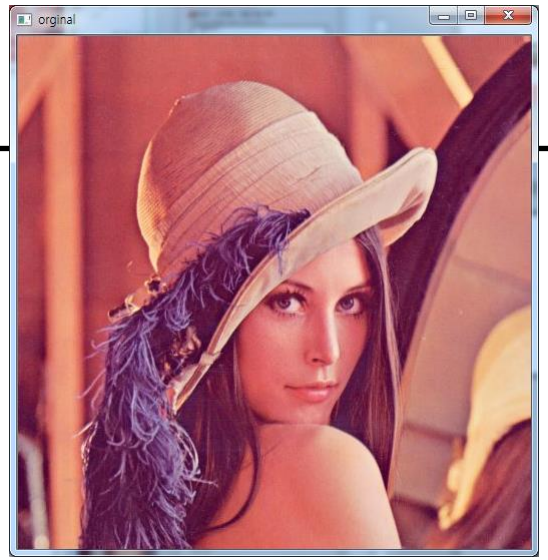
```
(512, 512, 3)
786432
uint8
```

Smart Media
스마트미디어인재개발원

- 이미지 ROI (Region of Image) 설정

```
1   import cv2
2
3   img = cv2.imread("images/lenna.png")
4   cv2.imshow("orginal", img)
5
6   subimg = img[200:370, 200:350]
7   cv2.imshow("cutting", subimg)
8
9   cv2.waitKey(0)
10  cv2.destroyAllWindows()
```



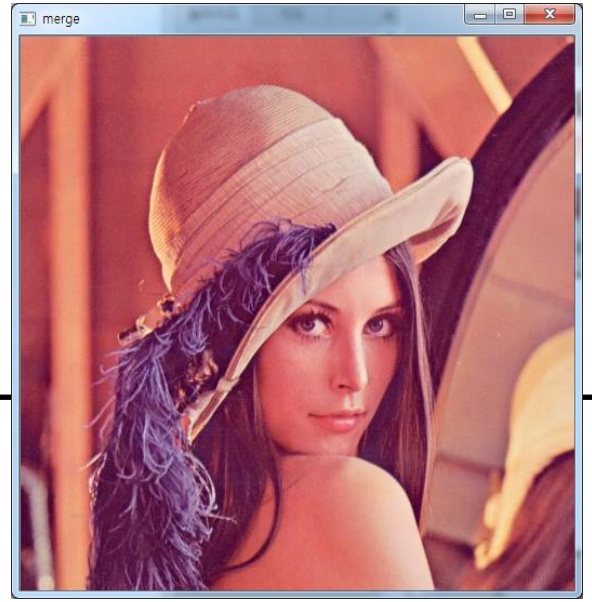


- RGB 채널을 따로 출력하기

```
1   import cv2
2
3   img = cv2.imread("images/lenna.png")
4
5   b = img[:, :, 0]
6   g = img[:, :, 1]
7   r = img[:, :, 2]
8
9   cv2.imshow("blue channel", b)
10  cv2.imshow("green channel", g)
11  cv2.imshow("red channel", r)
12
13  cv2.waitKey(0)
14  cv2.destroyAllWindows()
```

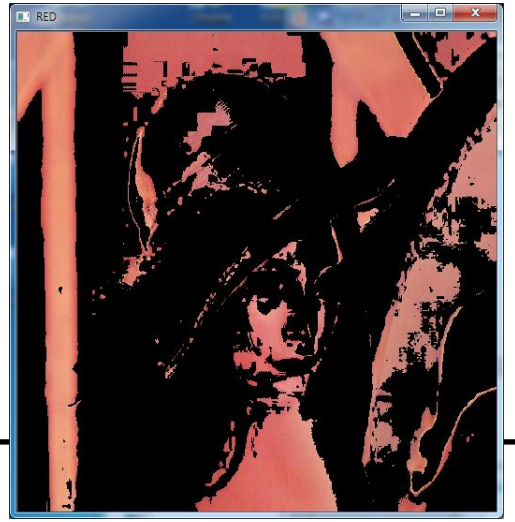
- RGB 채널 합치기

```
1   import cv2
2
3   img = cv2.imread("images/lenna.png")
4
5   b = img[:, :, 0]
6   g = img[:, :, 1]
7   r = img[:, :, 2]
8
9   merge_img = cv2.merge((b, g, r))
10
11  cv2.imshow("merge", merge_img)
12
13  cv2.waitKey(0)
14  cv2.destroyAllWindows()
```



- 색상 추적하기

```
1  import cv2
2  import numpy as np
3
4  img = cv2.imread("images/lenna.png")
5
6  hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
7
8  lower_red = np.array([-10, 100, 100])
9  upper_red = np.array([10, 255, 255])
10
11 mask_red = cv2.inRange(hsv, lower_red, upper_red)
12 red = cv2.bitwise_and(img, img, mask=mask_red)
13
14 cv2.imshow('RED', red)
15
16 k = cv2.waitKey(0)
17 cv2.destroyAllWindows()
```



- 색상 추적하기

```
6   hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
...
11  mask_red = cv2.inRange(hsv, lower_red, upper_red)
12  red = cv2.bitwise_and(img, img, mask=mask_red)
```

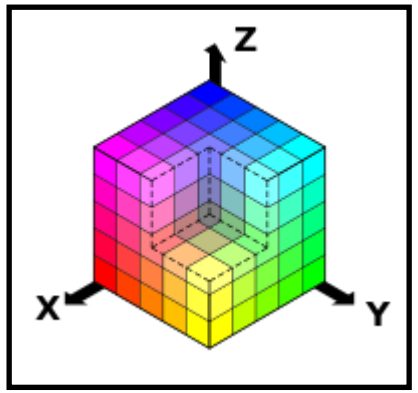
- **cv2.cvtColor(img, cv2.COLOR_BGR2HSV)** : HSV 색상 공간으로 변경
- **cv2.inRange(hsv, lower_red, upper_red)** : 범위에 해당하는 값이 아니면 0으로 채움
- **cv2.bitwise_and(img, img, mask=mask_red)** : mask 값이 0이 아닌 부분만 AND 연산

- blue, green의 경우

```
lower_blue = np.array([110, 100, 100])
upper_blue = np.array([130, 255, 255])

lower_green = np.array([50, 100, 100])
upper_green = np.array([70, 255, 255])
```

- 색공간

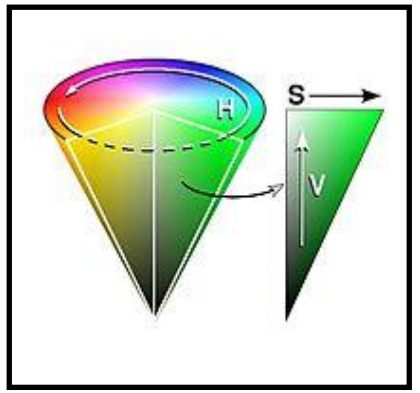


RGB

Red, Green, Blue

CMYK

Cyan, Magenta, Yellow, Black



HSV

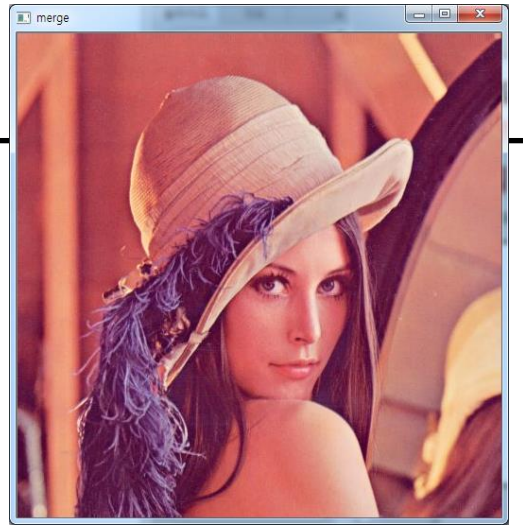
Hue(색상), Saturation(채도), Value(명도)

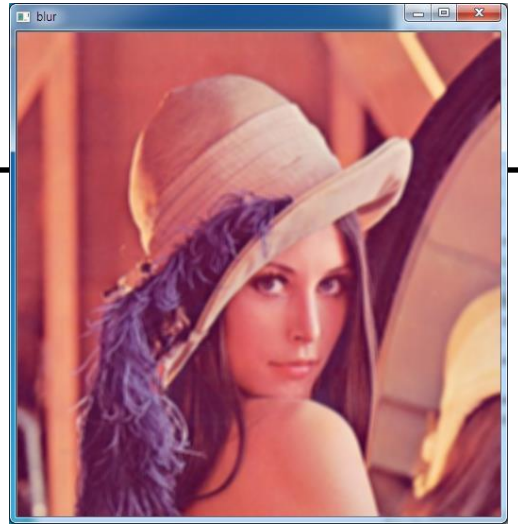
OpenCV + Python를 이용한 영상 처리

Smart Media
스마트미디어인재개발원

- 이미지 필터링 – blur

```
1   import cv2
2   import numpy as np
3
4   img = cv2.imread("images/lenna.png")
5
6   kernel = np.ones((5, 5), np.float32) / 25
7   blur = cv2.filter2D(img, -1, kernel)
8
9   cv2.imshow('blur', blur)
10
11  k = cv2.waitKey(0)
12  cv2.destroyAllWindows()
```

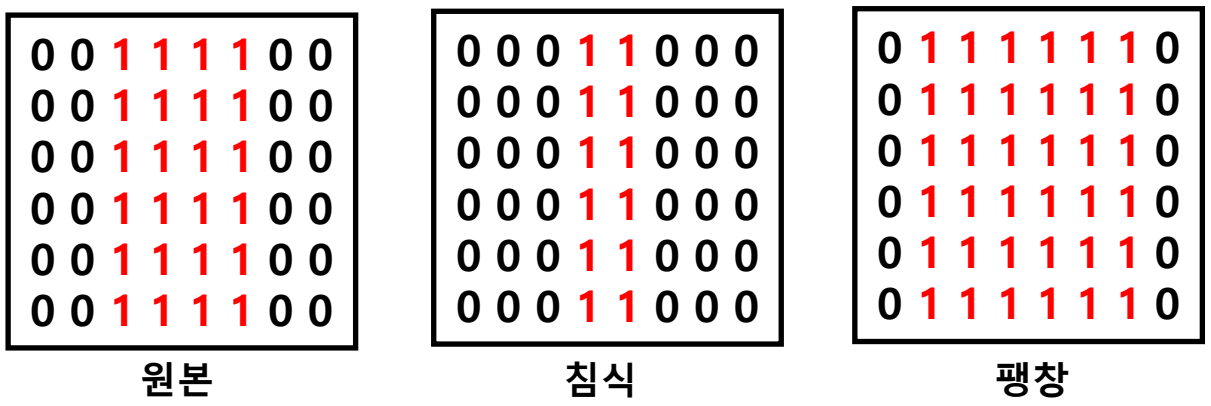




- **ones(5,5)** : 5행 5열의 원소를 1로 채움
- **cv2.filter2D(img, -1, kernel)**
  : 2번째 파라미터는 색의 깊이
  (몇 비트)로 -1이면 원본 이미지와 동일

$$K = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

- 침식과 팽창
  - **Erosion (침식)** : 이미지의 경계부분을 배경 픽셀로 변경하는 작업
  - **Dilation (팽창)** : 이미지의 배경부분을 전경 픽셀로 변경하는 작업

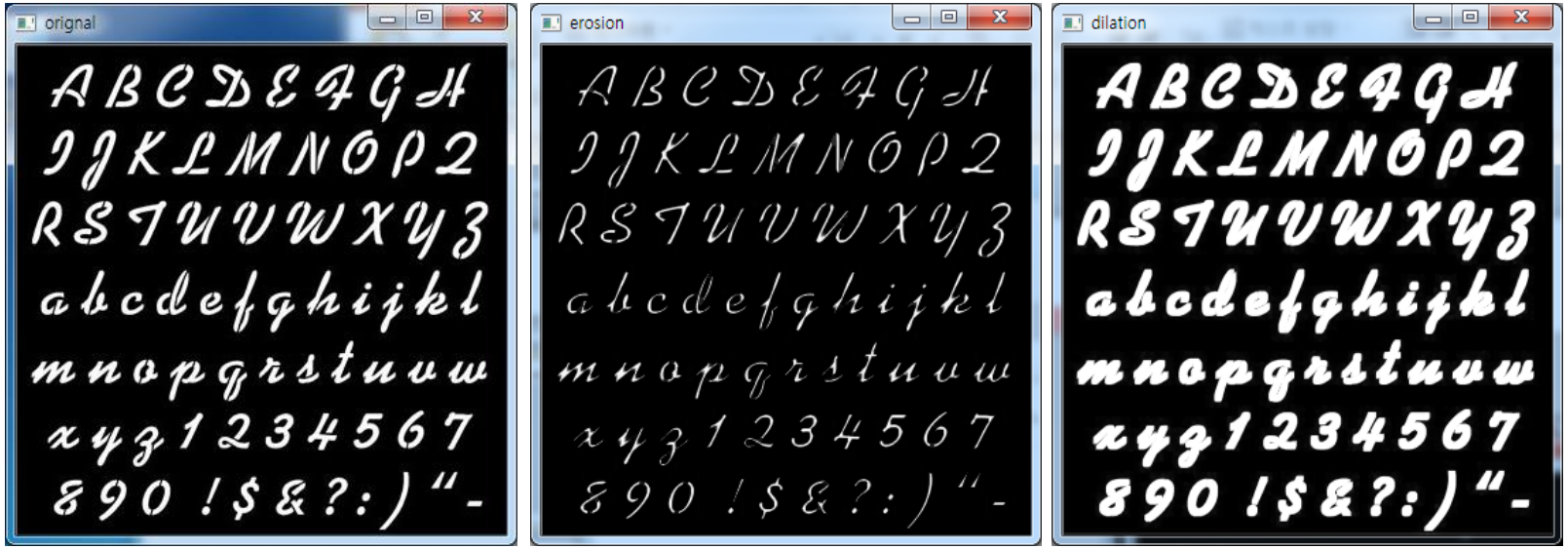


- 침식과 팽창 (http://arome.hosting.paran.com/data/deeplearning/number.png)

```
1  import cv2
2  import numpy as np
3
4  img = cv2.imread("images/number.png", cv2.IMREAD_GRAYSCALE)
5
6  kernel = np.ones((3, 3), np.uint8)
7
8  erosion = cv2.erode(img, kernel, iterations=1)
9  dilation = cv2.dilate(img, kernel, iterations=1)
10
11 cv2.imshow('orignal', img)
12 cv2.imshow('erosion', erosion)
13 cv2.imshow('dilation', dilation)
14
15 k = cv2.waitKey(0)
16 cv2.destroyAllWindows()
```

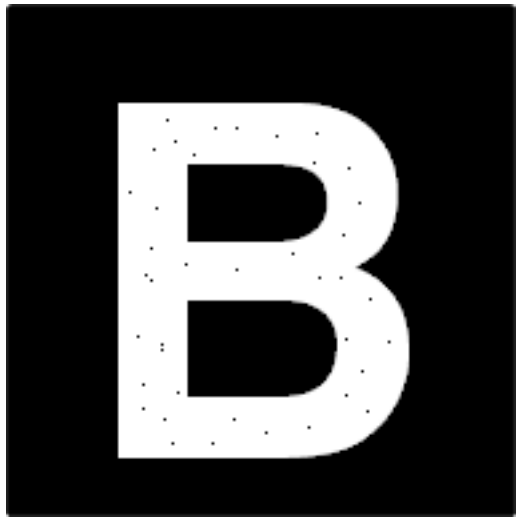
- **iterations = 1** : 반복 적용 회수 (회수가 많아지면 침식과 팽창의 효과가 커짐)

- 침식과 팽창



원본 침식 팽창

- Opening과 Closing
  - Opening : dilate 수행 후에 바로 원래 이미지 크기로 돌려 놓는 것
  - Closing : erosion 수행 후에 바로 원래 이미지 크기로 돌려 놓는 것



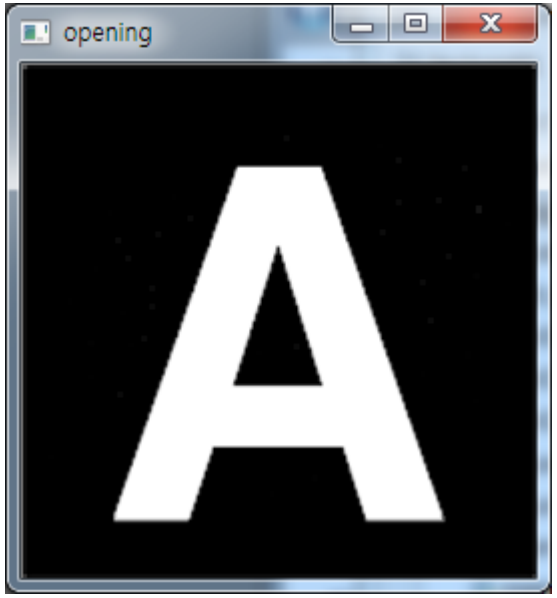
http://arome.hosting.paran.com/data/deeplearning/noise_A.png

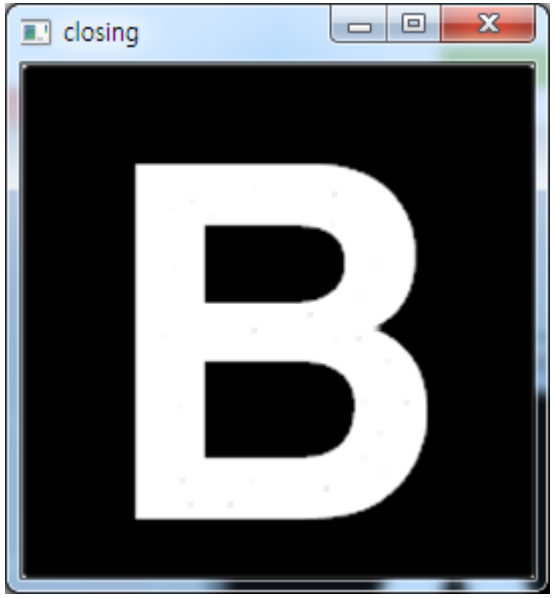
http://arome.hosting.paran.com/data/deeplearning/noise_B.png

- Opening과 Closing

```
1   import cv2
2   import numpy as np
3
4   img1 = cv2.imread("images/noise_A.png", cv2.IMREAD_GRAYSCALE)
5   img2 = cv2.imread("images/noise_B.png", cv2.IMREAD_GRAYSCALE)
6
7   kernel = np.ones((3, 3), np.uint8)
8
9   opening = cv2.morphologyEx(img1, cv2.MORPH_OPEN, kernel)
10  closing = cv2.morphologyEx(img2, cv2.MORPH_CLOSE, kernel)
11
12  cv2.imshow('opening', opening)
13  cv2.imshow('closing', closing)
14
15  k = cv2.waitKey(0)
16  cv2.destroyAllWindows()
```

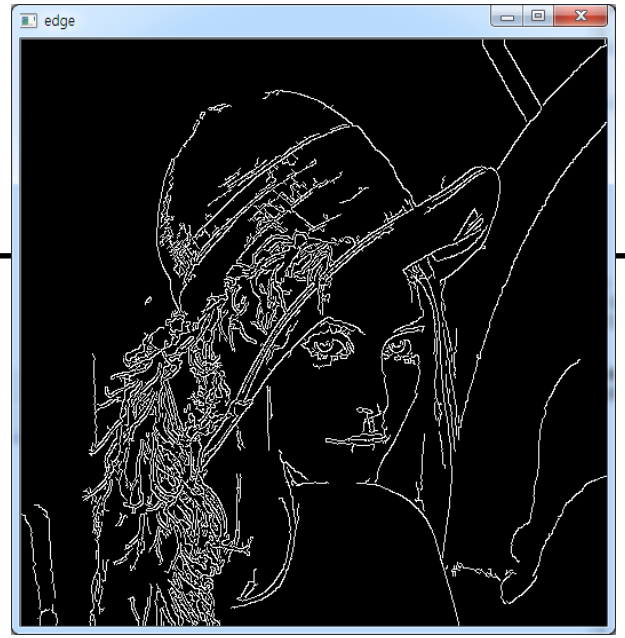
- Opening과 Closing





- 에지 추출 – Canny 에지 추출기

```
1  import cv2
2
3  img = cv2.imread("images/lenna.png", cv2.IMREAD_GRAYSCALE)
4
5  edge = cv2.Canny(img, 50, 200)
6
7  cv2.imshow("edge", edge)
8
9  k = cv2.waitKey(0)
10 cv2.destroyAllWindows()
```



- cv2.Canny(이미지, 최소값, 최대값)

- 외곽선 추출

```
1   import cv2
2
3   img = cv2.imread("images/earth.png")
4   gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
5
6   ret, thr = cv2.threshold(gray, 127, 255, 0)
7   contours, _ = cv2.findContours (thr, cv2.RETR_TREE , cv2.CHAIN_APPROX_SIMPLE )
8   cv2.drawContours(img, contours, -1, (0, 0, 255), 1)
9
10  cv2.imshow("contour", img)
11
12  k = cv2.waitKey(0)
13  cv2.destroyAllWindows()
```

- 외곽선 추출

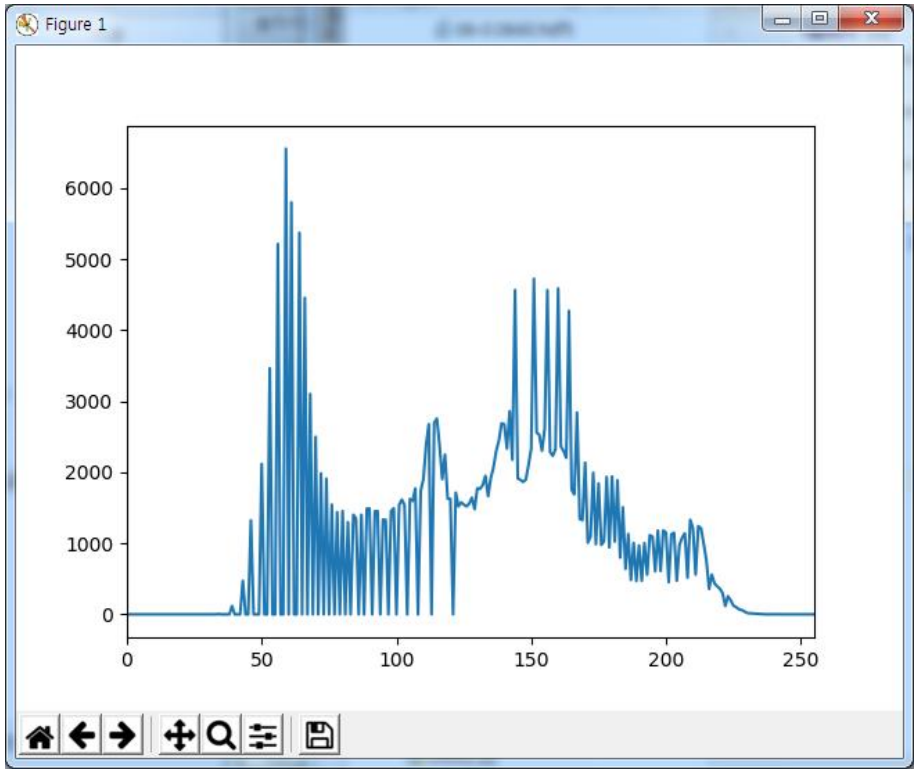
```
6  ret, thr = cv2.threshold(gray, 127, 255, 0)
7  contours, _ = cv2.findContours(thr, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
8
9  cv2.drawContours(img, contours, -1, (0, 0, 255), 1)
```

- **cv2.threshold(이미지, 문턱값, 픽셀 최대값, 문턱값 적용 방법)** : 이진 이미지를 생성 (0인 경우 문턱값보다 크면 픽셀 최대값을 할당하고, 작으면 0을 할당)
- **cv2.findContours(이미지, 추출 모드, 근사 방법)** : RETR_TREE (모든 외곽선을 추출하고 외곽선 간의 상관관계 추출), CHAIN_APPROX_SIMPLE (수평, 수직, 대각선인 경우 중간의 점들은 버리고 끝점 들만 남김)
- **cv2.drawContours(원본이미지, 외곽선 값, 그릴 외곽선 인덱스, 외곽선 색상, 선의 두께)** : 인덱스가 -1이면 모든 외곽선을 그림

- **히스토그램 (Histogram)** : 이미지의 색상 별로 픽셀의 개수를 그래프로 표시한 것
- Numpy, Matplotlib 라이브러리에서 함수를 제공하지만 OpenCV에서 제공하는 calcHist() 함수가 가장 성능이 좋음
  - **calcHist([이미지], [채널], 특정 부분 마스크, [색상 개수], [픽셀값의 범위])**
  - 채널은 흑백영상인 경우는 0, 칼라영상인 경우는 B(0), G(1), R(2)

```
1  import cv2
2  import matplotlib.pyplot as plt
3
4  img = cv2.imread("images/lenna.png", cv2.IMREAD_GRAYSCALE)
5
6  hist = cv2.calcHist([img], [0], None, [256], [0, 256])
7
8  plt.plot(hist)
9  plt.xlim([0, 255])
10 plt.show()
```

- 히스토그램 (Histogram)



- **푸리에 변환 (Fourier Transform)** : 이미지를 주파수 영역으로 표현하는 함수
- 이미지에서 주파수 : 픽셀의 변화량

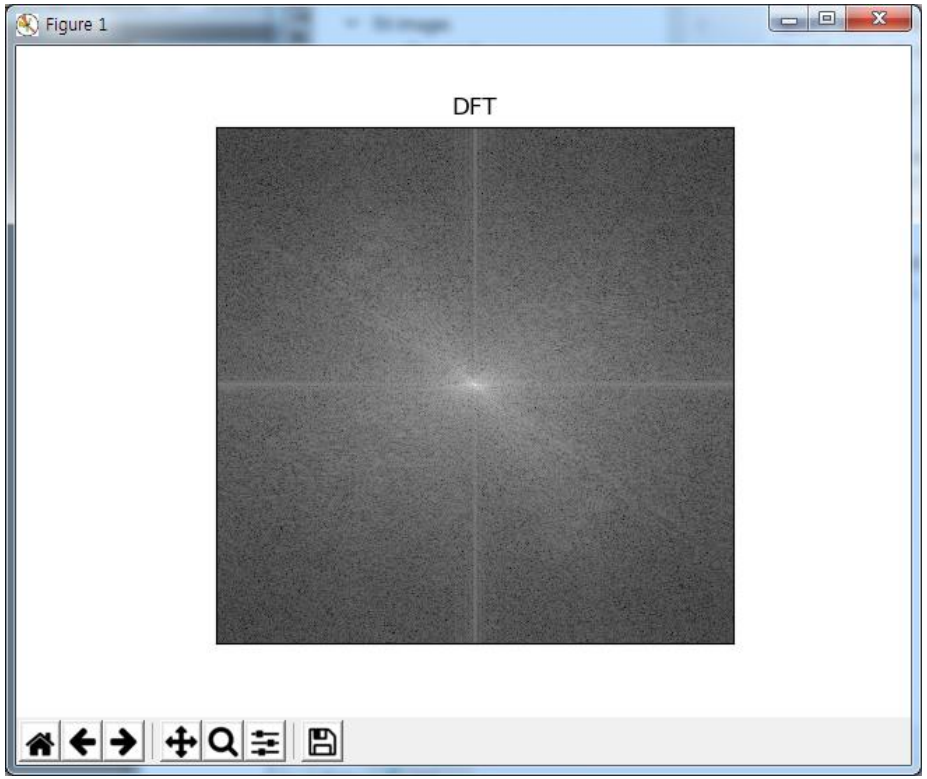
일반적인 이미지는 저주파와 고주파 중 어느 성분이 많을까요 ?

- 푸리에 변환 (Fourier Transform)

```
1   import cv2
2   import matplotlib.pyplot as plt
3   import numpy as np
4
5   img = cv2.imread("images/lenna.png", cv2.IMREAD_GRAYSCALE)
6
7   f = np.fft.fft2(img)
8   fshift = np.fft.fftshift(f)
9   m_spectrum = 20 * np.log(np.abs(fshift))
10
11  plt.imshow(m_spectrum, cmap="gray")
12  plt.title("DFT"), plt.xticks([]), plt.yticks([])
13  plt.show()
```

- [7] 푸리에 변환을 수행
- [8] 주파수가 0인 컴포넌트를 각 모서리에서 중앙으로 이동하고 재배열시킴
- [9] 진폭 스펙트럼을 계산

- 푸리에 변환 (Fourier Transform)



- 푸리에 변환 (Fourier Transform)의 중앙부분만 사용하기

```
1  import cv2
2  import matplotlib.pyplot as plt
3  import numpy as np
4
5  img = cv2.imread("images/lenna.png", cv2.IMREAD_GRAYSCALE)
6
7  f = np.fft.fft2(img)
8  fshift = np.fft.fftshift(f)
9
10 rows, cols = img.shape
11 crow, ccol = int(rows/2), int(cols/2)
12
13 fshift[crow-30:crow+30, ccol-30:ccol+30] = 0
14 f_ishift = np.fft.ifftshift(fshift)
15 img_back = np.fft.ifft2(f_ishift)
16 img_back = np.abs(img_back)
17
18 plt.imshow(img_back, cmap="gray")
19 plt.title("DFT"), plt.xticks([]), plt.yticks([])
20 plt.show()
```

- 푸리에 변환 (Fourier Transform)의 중앙부분만 사용하기

- 푸리에 변환 (Fourier Transform)의 중앙부분만 사용하기

```
10  rows, cols = img.shape
11  crow, ccol = int(rows/2), int(cols/2)
12
13  fshift[crow-30:crow+30, ccol-30:ccol+30] = 0
14  f_ishift = np.fft.ifftshift(fshift)
15  img_back = np.fft.ifft2(f_ishift)
16  img_back = np.abs(img_back)
```

- [10] DFT 변환된 이미지의 크기를 반환
- [11] DFT 변환된 이미지의 중앙점을 반환
- [12] DFT 변환된 이미지의 중앙에서 (-30, -30)~(30, 30) 크기의 사각형 영역을 0으로 채움
- [14] 재배열된 주파수 값들의 위치를 원래대로 되돌림
- [15] 역 DFT를 수행하여 원래 이지미 영역으로 전환
- [16] 모든 값을 양수로 변경

Smart Media
스마트미디어인재개발원

- 템플릿 매칭 : 어떤 이미지에서 부분 이미지를 검색하는 방법

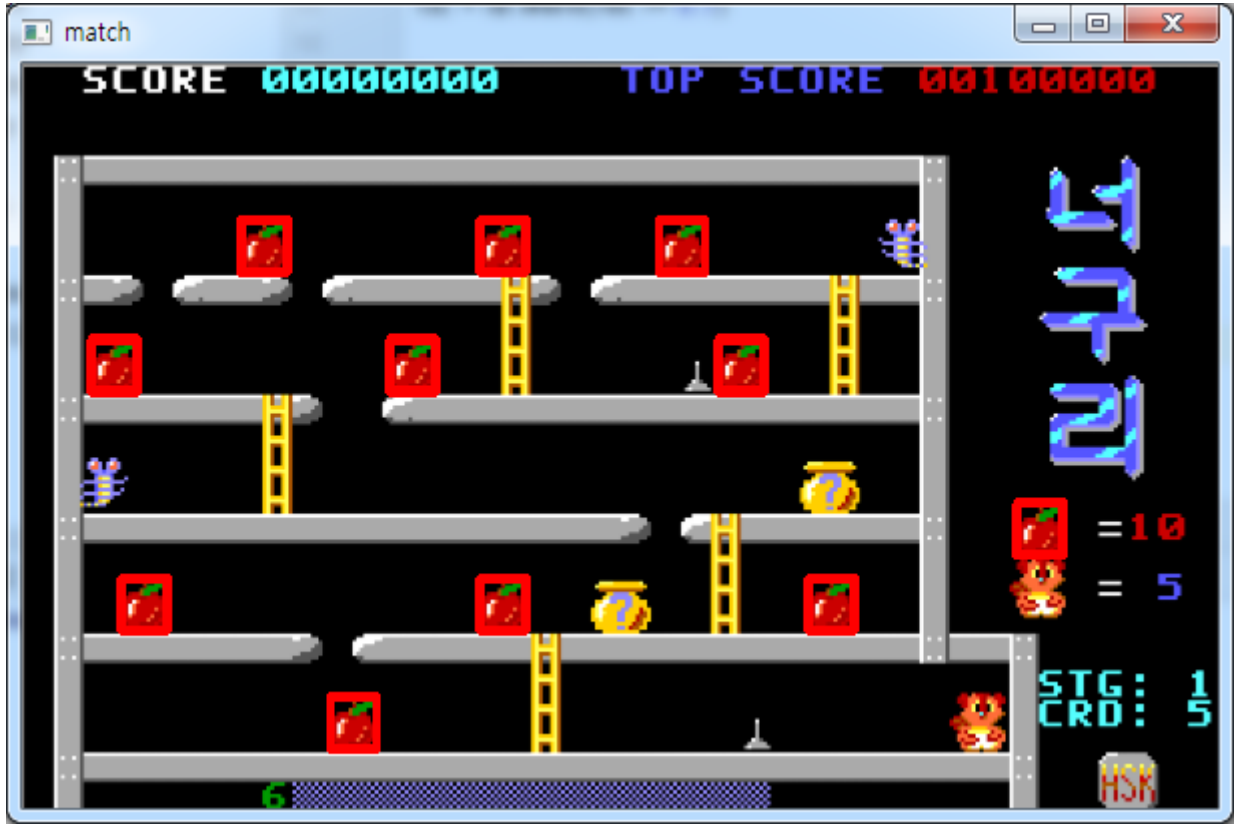
http://arome.hosting.paran.com/data/deeplearning/game.png

http://arome.hosting.paran.com/data/deeplearning/game_cut.png

- 템플릿 매칭

```
1  import cv2
2  import numpy as np
3
4  img = cv2.imread("images/game.png")
5  imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
6  temp = cv2.imread("images/game_cut.png", cv2.IMREAD_GRAYSCALE)
7  w, h = temp.shape[::-1]
8
9  res = cv2.matchTemplate(imgray, temp, cv2.TM_CCOEFF_NORMED)
10
11 loc = np.where(res >= 0.7)
12
13 for pt in zip(*loc[::-1]):
14     cv2.rectangle(img, pt, (pt[0]+w, pt[1]+h), (0, 0, 255), 2)
15
16 cv2.imshow('match', img)
17
18 k = cv2.waitKey(0)
19 cv2.destroyAllWindows()
```

- 템플릿 매칭


match
SCORE 00000000
TOP SCORE 00100000
너구리
=10
= 5
STG: 1
CRD: 5
HSK
6

- 템플릿 매칭

```
7   w, h = temp.shape[::-1]
8
9   res = cv2.matchTemplate(imgray, temp, cv2.TM_CCOEFF_NORMED)
10
11  loc = np.where(res >= 0.7)
12
13  for pt in zip(*loc[::-1]):
14      cv2.rectangle(img, pt, (pt[0]+w, pt[1]+h), (0, 0, 255), 2)
```

- [7] 이미지 크기를 반환
- [9] **cv2.matchTemplate(원본이미지, 부분이미지, 매칭 방법)**
- [11] res가 0.7보다 큰 값들의 위치를 튜플로 반환
- [13] 동일한 개수를 가진 리스트나 튜플을 같은 위치의 멤버들끼리 묶어서 튜플로 만든 다음 이를 멤버로 하는 리스트를 만듬
- *loc[::-1] : loc의 순서를 거꾸로 하여 튜플로 묶고 리스트로 만듬 (x, y 좌표의 순서가 바뀌므로)
- **cv2.rectangle(이미지, 시작위치, 종료위치, 색상, 두께)** : 사각형을 그리는 함수

- where() 함수 테스트

```
1  import numpy as np
2
3  a = np.arange(10)
4  loc = np.where(a > 5)
```

```
(array([6, 7, 8, 9], dtype=int64),)
```

- zip() 함수 테스트

```
1  a = zip([0, 1, 2, 3], [4, 5, 6, 7])
2
3  for pt in a:
4      print(pt)
```

```
(0, 4)
(1, 5)
(2, 6)
(3, 7)
```

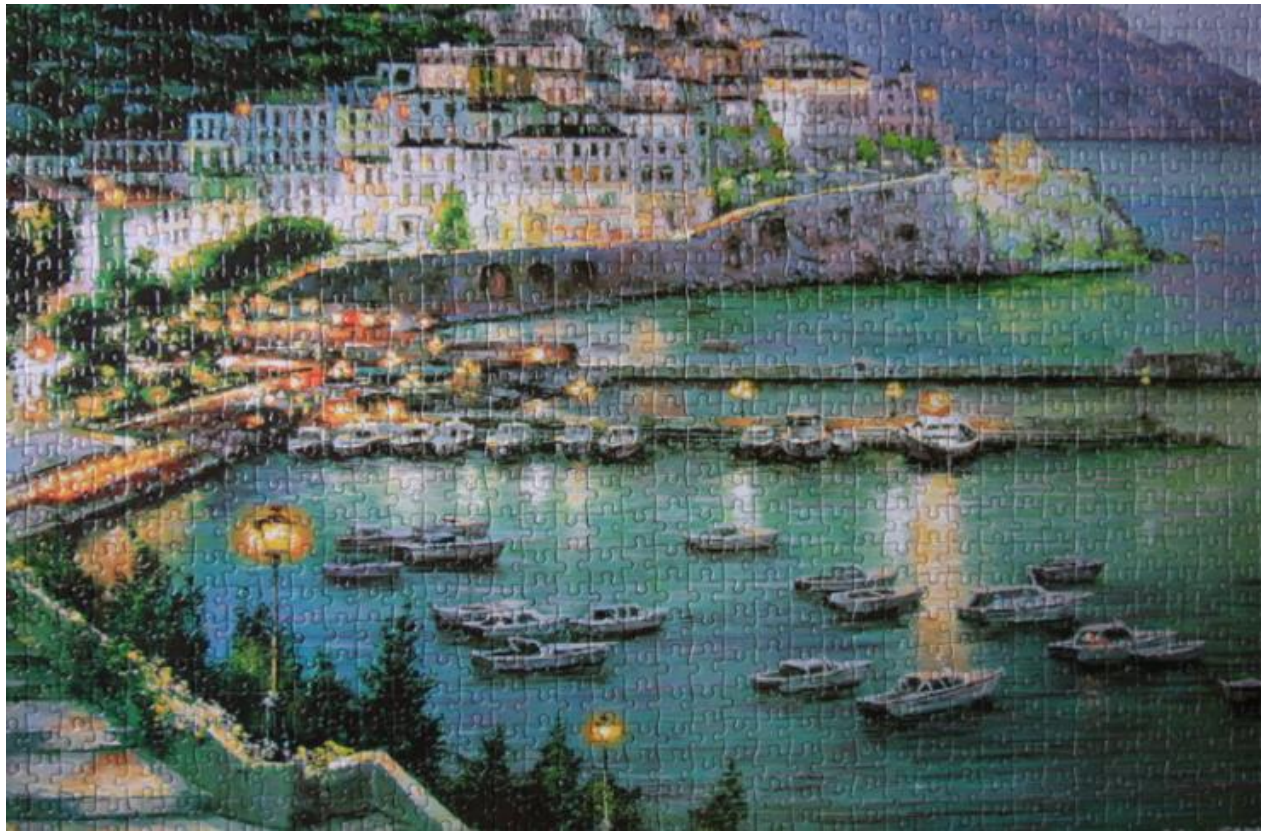
- zip(*loc[::-1]) 테스트

```
1  import numpy as np
2
3  loc = (np.array([0, 1, 2, 3]), np.array([4, 5, 6, 7]))
4
5  for pt in zip(*loc[::-1]):
6      print(pt)
```

```
(4, 0)
(5, 1)
(6, 2)
(7, 3)
```

- 이미지 특성 이해

여러분은 이런 퍼즐을 어떻게 맞추나요 ?

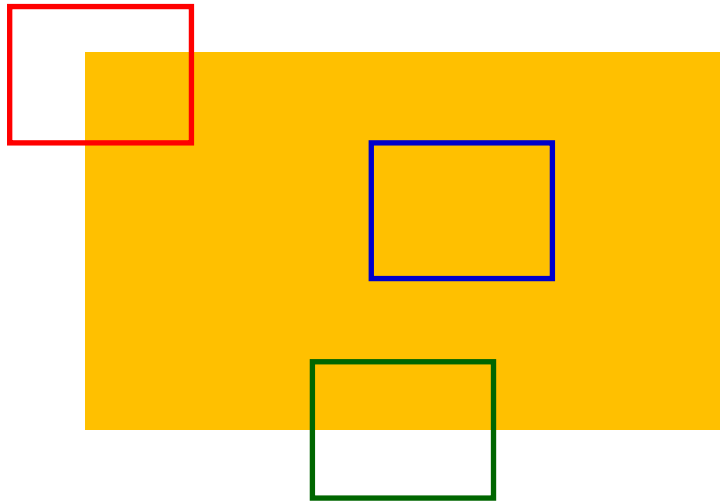


- 이미지 특성 이해

그럼 컴퓨터는 퍼즐을 어떻게 맞출까요 ?

- A, B : 전체 이미지에서 어디인지 정확히 알 수 없음
- C, D : 건물 지붕인 건 알겠는데 정확한 위치는 알 수 없음
- E, F : 전체 이미지에서 어디인지 정확히 알 수 있음

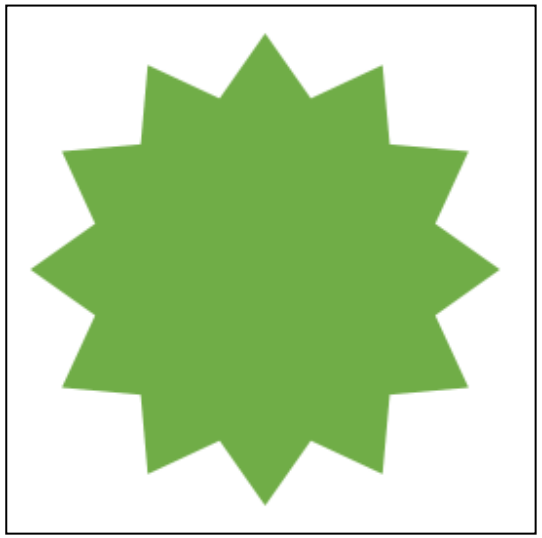
- 이미지 특성 이해



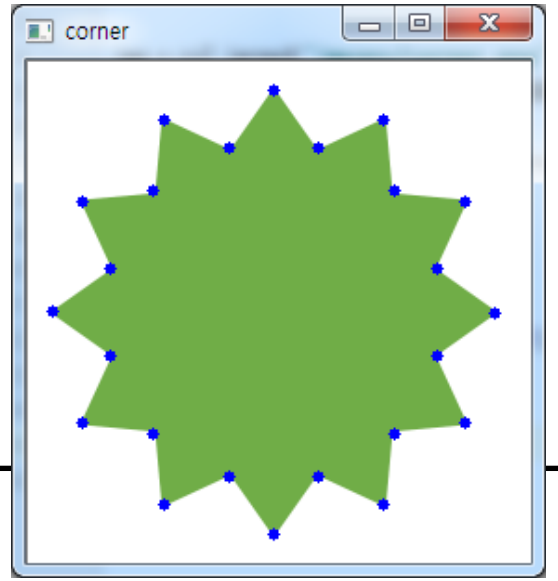
- 적색 박스 : 약간만 움직여도 내부의 변화를 알 수 있음
- 청색 박스 : 움직여도 내부의 특성이 전혀 변하지 않음
- 녹색 박스 : 상하 이동의 경우만 내부의 변화를 알 수 있음

→ 이미지의 코너를 찾아서 이미지의 특성을 검출하는 방법이 유용

- 코너 검출 (**http://arome.hosting.paran.com/data/deeplearning/corner.png**)



```
1   import numpy as np
2   import cv2
3
4   img = cv2.imread("images/corner.png")
5   imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
6
7   corners = cv2.goodFeaturesToTrack(imgray, 25, 0.01, 10)
8   corners = np.int0(corners)
9
10  for i in corners:
11      x, y = i.ravel()
12      cv2.circle(img, (x, y), 3, (255, 0, 0), -1)
13
14  cv2.imshow('corner', img)
15  cv2.waitKey(0)
16  cv2.destroyAllWindows()
```



```
7   corners = cv2.goodFeaturesToTrack(imgray, 25, 0.01, 10)
8   corners = np.int64(corners)
9
10  for i in corners:
11      x, y = i.ravel()
12      cv2.circle(img, (x, y), 3, (255, 0, 0), -1)
```

- [7] Shi-Tomasi 코너 검출 알고리즘 함수 **(이미지, 코너 검출 개수, 문턱값, 코너 간 최소 거리)**
- [8] 정수로 변환 (int0로도 쓸 수 있음)
- [11] **ravel()** : 다차원 배열을 1차원 배열로 변환 (**flatten()**은 배열만 사용 가능, **reshape()** 함수와 반대되는 기능)
- [12] 이미지(img) 상의 해당 위치(x, y)에 반지름이 3이고 두께가 -1(원을 채움)인 파란색(255, 0, 0)의 원을 그린다

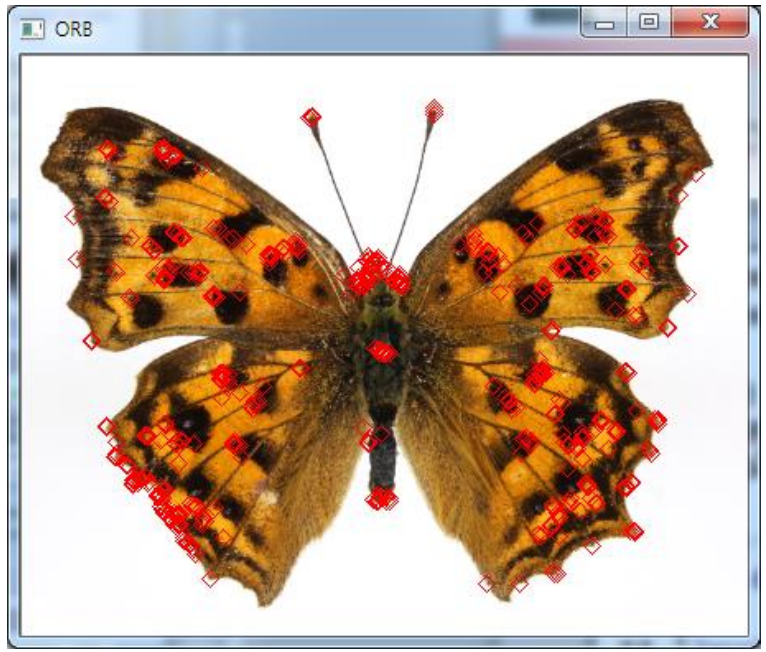
- **ORB (Oriented FAST and Rotated BRIEF) 알고리즘** : SIFT, SURF 대신에 자유롭게 사용할 수 있는 OpenCV의 이미지 특성 검출 알고리즘

- **http://arome.hosting.paran.com/data/deeplearning/butterfly.png**

- ORB (Oriented FAST and Rotated BRIEF) 알고리즘

```
1   import cv2
2   import numpy as np
3
4   img = cv2.imread("images/butterfly.png")
5   imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
6   img2 = None
7
8   orb = cv2.ORB_create()
9   kp, des = orb.detectAndCompute(img, None)
10
11  img2 = img.copy()
12  for marker in kp:
13      img2 = cv2.drawMarker(img2, tuple(int(i) for i in marker.pt),
14          markerType=3, markerSize=10, thickness=1, color=(0, 0, 255))
15
16  cv2.imshow('ORB', img2)
17  cv2.waitKey(0)
18  cv2.destroyAllWindows()
```

- ORB (Oriented FAST and Rotated BRIEF) 알고리즘



- ORB (Oriented FAST and Rotated BRIEF) 알고리즘

```
 8  orb = cv2.ORB_create()
 9  kp, des = orb.detectAndCompute(img, None)
10
11  img2 = img.copy()
12  for marker in kp:
13      img2 = cv2.drawMarker(img2, tuple(int(i) for i in marker.pt),
14          markerType=3, markerSize=10, thickness=1, color=(0, 0, 255))
```

- [8] ORB 객체 생성
- [9] img의 키포인트(영상 특징점)들과 디스크립터(키포인터 주변 영역의 특성을 표현하는 영상 기술자)를 계산
- [12]-[14] 키포인트를 그린다
  - markerType=3 : 마커 형태
  - markerSize=10 : 마커 크기
  - thickness=1 : 마커 선 두께
  - color=(0, 0, 255) : 마커 색상

OpenCV + Python를 이용한 영상 처리

Smart Media
스마트미디어인재개발원

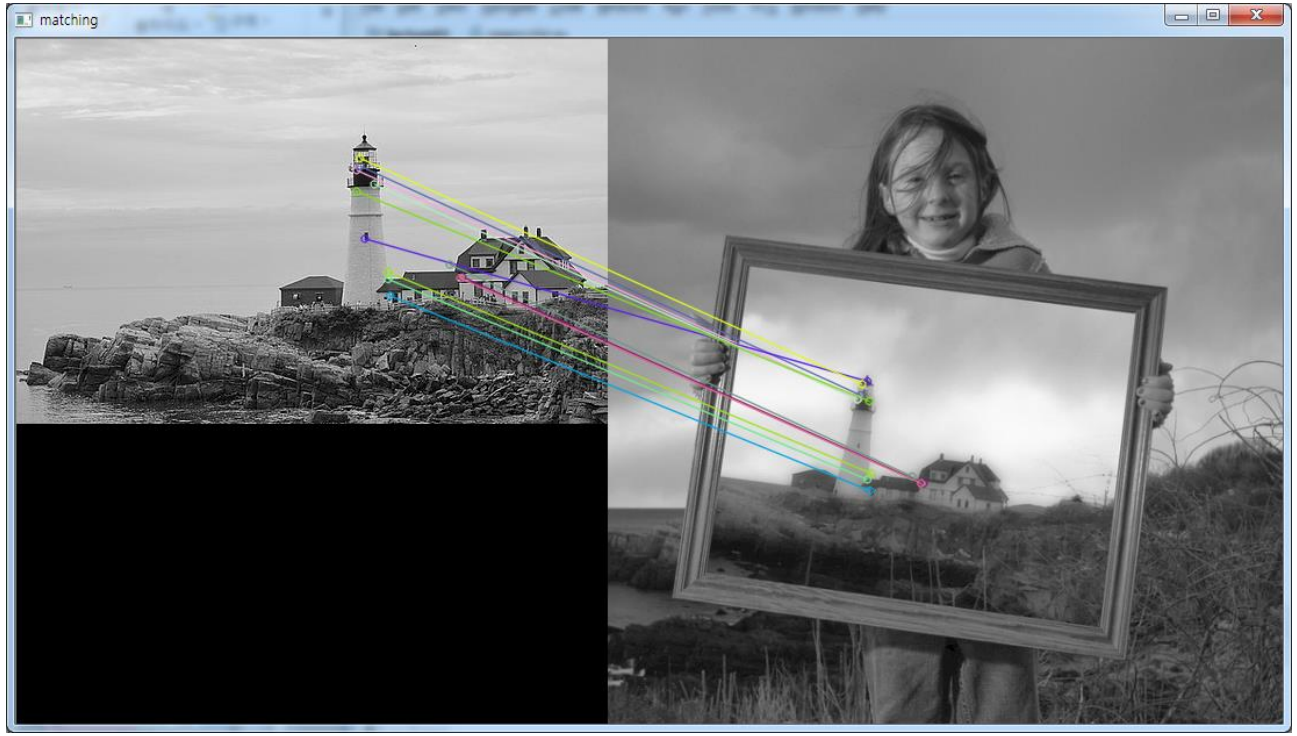
- ORB 기반의 이미지 특성 매칭

- **http://arome.hosting.paran.com/data/deeplearning/girl_pic.png**
- **http://arome.hosting.paran.com/data/deeplearning/pic.png**

- ORB 기반의 이미지 특성 매칭

```
1  import cv2
2  import numpy as np
3
4  img1 = cv2.imread("images/pic.png", cv2.IMREAD_GRAYSCALE)
5  img2 = cv2.imread("images/girl_pic.png", cv2.IMREAD_GRAYSCALE)
6  res = None
7
8  orb = cv2.ORB_create()
9  kp1, des1 = orb.detectAndCompute(img1, None)
10 kp2, des2 = orb.detectAndCompute(img2, None)
11
12 bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
13 matches = bf.match(des1, des2)
14
15 matches = sorted(matches, key=lambda x:x.distance)
16 res = cv2.drawMatches(img1, kp1, img2, kp2, matches[:12], res, flags=0)
17
18 cv2.imshow('matching', res)
19 cv2.waitKey(0)
20 cv2.destroyAllWindows()
```

- ORB 기반의 이미지 특성 매칭



- ORB 기반의 이미지 특성 매칭

```
 8  orb = cv2.ORB_create()
 9  kp1, des1 = orb.detectAndCompute(img1, None)
10  kp2, des2 = orb.detectAndCompute(img2, None)
11
12  bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
13  matches = bf.match(des1, des2)
14
15  matches = sorted(matches, key=lambda x:x.distance)
16  res = cv2.drawMatches(img1, kp1, img2, kp2, matches[:12], res, flags=0)
```

- [8] ORB 객체 생성
- [9-10] img1과 img2의 키포인트(영상 특징점)들과 디스크립터(키포인터 주변 영역의 특성을 표현하는 영상 기술자)를 계산
- [12-13] 매칭 실행
- [15] 두 이미지의 특성 포인트들이 가장 일치하는 순으로 정렬
- [16] **matches[:12]** : 가장 유사한 12개 쌍, **flags = 0** (매칭과 상관없이 모든 특성 포인트 표시, 2 (매칭된 특성포인트만 표시))

- key=lambda의 의미

```
1  mylist = [3,6,3,2,4,8,23]
2
3  sort = sorted(mylist, key=lambda x: x%2==0)
4
5  print(sort)
```

```
[3, 3, 23, 6, 2, 4, 8]
```

[3, 6, 3, 2, 4, 8, 23] → [0, 1, 0, 1, 1, 1, 0] → [3, 3, 23, 6, 2, 4, 8]

OpenCV + Python를 이용한 영상 처리

- 얼굴 검출 (Harr cascade classifier 활용) - 이미지

- **http://arome.hosting.paran.com/data/deeplearning/face.png**
- **http://arome.hosting.paran.com/data/deeplearning/harrcascade.zip**
  - 압축을 풀고 프로젝트에 복사한다

- 얼굴 검출 (Harr cascade classifier 활용) - 이미지

```
1   import cv2
2
3   eye_detect = False
4   face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
5   eye_cascade = cv2.CascadeClassifier("haarcascade_eye.xml")
6
7   img = cv2.imread("images/face.png")
8   gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
9
10  faces = face_cascade.detectMultiScale(gray, 1.3, 5)
11
12  for (x, y, w, h) in faces:
13      cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
14
15      if eye_detect:
16          roi_gray = gray[y:y+h, x:x+w]
17          roi_color = img[y:y+h, x:x+w]
18          eyes = eye_cascade.detectMultiScale(roi_gray)
19
```

- 얼굴 검출 (Harr cascade classifier 활용) - 이미지

```
20          for (ex, ey, ew, eh) in eyes:
21              cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)
22
23  cv2.imshow("frame", img)
24  k = cv2.waitKey(0)
25
26  if k == 49:
27      cv2.destroyAllWindows()
```

- 얼굴 검출 (Harr cascade classifier 활용) – 이미지

```
4   face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
5   eye_cascade = cv2.CascadeClassifier("haarcascade_eye.xml")
…   …
10  faces = face_cascade.detectMultiScale(gray, 1.3, 5)
…   …
12  for (x, y, w, h) in faces:
13      cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2))
```

- [4-5] 얼굴과 눈 검출을 위한 Harr-Cascade 트레이닝 데이터를 가져와 CascadeClassifier를 생성
- [10] **face_cascade.detectMultiScale(이미지, 스케일 값, 최소 이웃값)** : 해당 설정에 따라 얼굴을 검출하고 얼굴 위치를 리스트로 반환 (좌상단 위치 (x, y), 가로세로 크기 (w, h)의 튜플)
- [13] 각 얼굴 위치 리스트를 읽어 img에 두께가 2인 파란색의 사각형을 그린다

- 얼굴 검출 (Harr cascade classifier 활용) – 이미지

```
15      if eye_detect:
16          roi_gray = gray[y:y+h, x:x+w]
17          roi_color = img[y:y+h, x:x+w]
18
19          for (ex, ey, ew, eh) in eyes:
21              cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)
```

- 눈 영역이 추출되면 선 두께가 2인 녹색 사각을 그린다

- 얼굴 검출 (Harr cascade classifier 활용) - 동영상

```
1   import cv2
2
3   eye_detect = False
4   face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
5   eye_cascade = cv2.CascadeClassifier("haarcascade_eye.xml")
6
7   try:
8       print('카메라를 구동합니다')
9       cap = cv2.VideoCapture(0)
10  except:
11      print("카메라 구동 실패")
12
13  while True:
14      ret, frame = cap.read()
15
16      gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
17      faces = face_cascade.detectMultiScale(gray, 1.3, 5)
18
```

- 얼굴 검출 (Harr cascade classifier 활용) - 동영상

```
19      for (x, y, w, h) in faces:
20          cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
21
22          if eye_detect:
23              roi_gray = gray[y:y+h, x:x+w]
24              roi_color = frame[y:y+h, x:x+w]
25              eyes = eye_cascade.detectMultiScale(roi_gray)
26
27              for (ex, ey, ew, eh) in eyes:
28                  cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)
29
30      cv2.imshow("frame", frame)
31      k = cv2.waitKey(30)
32
33      if k == 49:
34          cap.release()
35          cv2.destroyAllWindows()
36          break
```