CH06 형태학적 영상처리

Joonki Paik

Image Processing and Intelligent Systems Laboratory
Chung-Ang University
paikj@cau.ac.kr

C2-002 영상처리기초

Table of Contents

Table of Contents

주요 참고자료 및 학습내용

주요 참고자료

- 디지털 영상처리 이론 참고문헌 : [1]
- 본 강의 주교재 : [2]
- 본 강의에서 사용되는 Python 코드 : [3]

형태학적 영상처리

- 영상의 특징 및 모양과 관련된 비선형 연산의 집합, 이진/명암도 영상에 적용
- 형태학적 영상처리의 도구 : 구조요소(형태소, structural element ; SE)를 템플릿으로 사용해서 입력 영상의 구조를 분석
- 형태학적 영상처리의 종류 : 팽창(dilation), 침식(erosion), 열림(opening), 닫힘(closing), 세선화(thining), 골격화(skeletonizing), 형태학적 에지 검출기(morphological edge detectors), 적중/실패 필터(hit or miss filter), 순위 필터(rank filter), 메디안 필터(median filter), 다수결 필터(majority filter)

학습내용

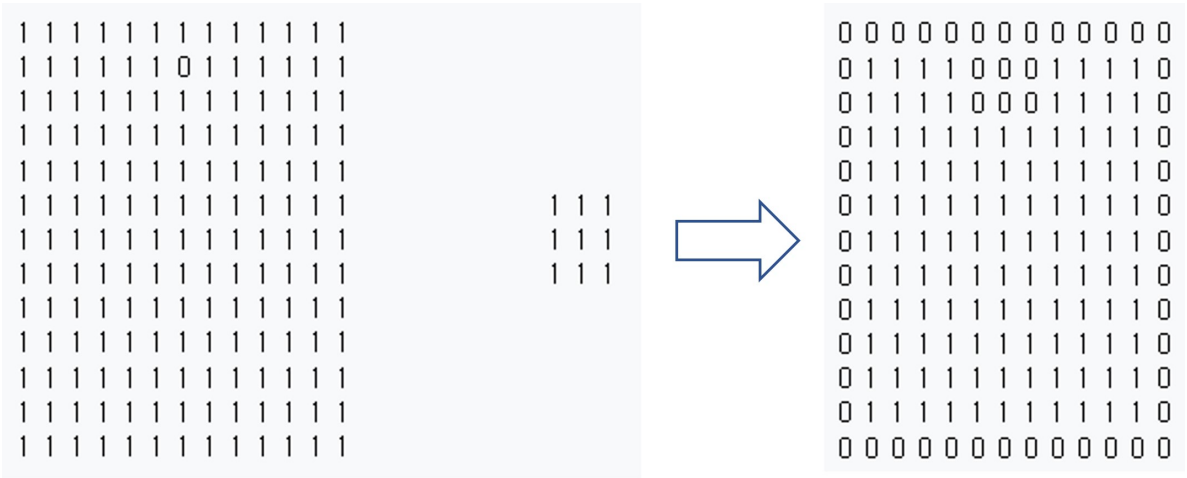
1. scikit-image morphology 모듈을 사용한 형태학적 영상처리
2. scikit-image filter.rank 모듈을 사용한 형태학적 영상처리
3. scipy.ndimage.morphology 모듈을 사용한 형태학적 영상처리

Table of Contents

침식(Erosion) I

- 개념 : 입력 영상의 객체 크기를 줄이고, 경계를 부드럽게 하고, 반도(peninsular) 등 작은 형태를 제거
- 입력 영상 A와 형태소(SE) B가 주어질 때, 형태학적 침식된 영상은 $f_E = A \ominus B$로 표현
- 입력 영상 A의 (i, j) 번째 화소를 (i, j)를 중심으로 하는 이웃 영역(B에 의해서 결정)에 속한 화소값들 중 최소값으로 설정.
- $b_{(i,j)}$를 B가 (i, j) 만큼 이동한 영역이라 할 때,

$$f_E(i, j) = \min_{b_{(i,j)}} A$$



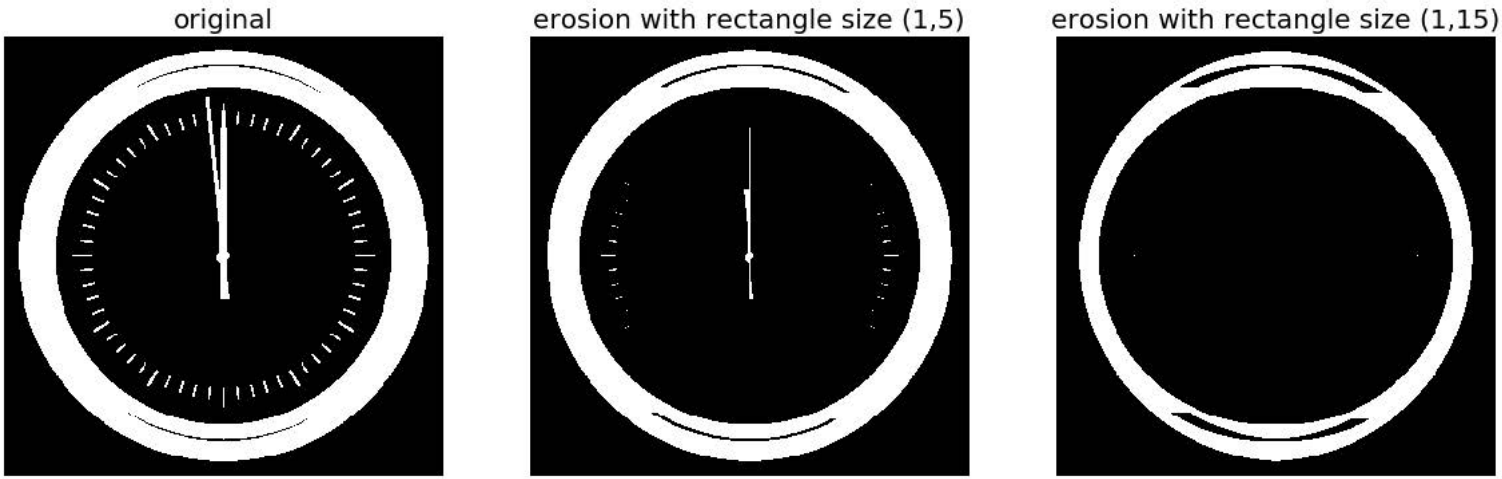
A B $f_E = A \ominus B$

침식(Erosion) II

Listing – binary_erosion()을 사용한 침식

```
1  %matplotlib inline
2  from skimage.io import imread
3  from skimage.color import rgb2gray
4  import matplotlib.pylab as pylab
5  from skimage.morphology import binary_erosion, rectangle
6
7  def plot_image(image, title=''):
8      pylab.title(title, size=20), pylab.imshow(image)
9      pylab.axis('off') # comment this line if you want axis ticks
10
11 im = rgb2gray(imread('../images/clock2.jpg'))
12 im[im ≤ 0.5] = 0 # create binary image with fixed threshold 0.5
13 im[im > 0.5] = 1
14 pylab.gray()
15 pylab.figure(figsize=(20,10))
16 pylab.subplot(1,3,1), plot_image(im, 'original')
17 im1 = binary_erosion(im, rectangle(1,5))
18 pylab.subplot(1,3,2), plot_image(im1, 'erosion with rectangle size (1,5)')
19 im1 = binary_erosion(im, rectangle(1,15))
20 pylab.subplot(1,3,3), plot_image(im1, 'erosion with rectangle size (1,15)')
21 pylab.show()
```

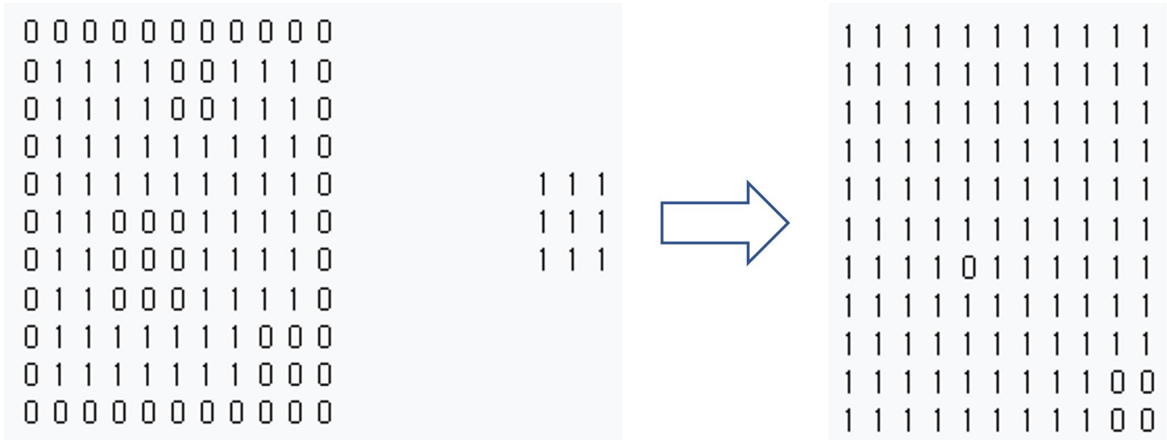
침식(Erosion) III



팽창(Dilation) I

- 개념 : 입력 영상의 객체 크기를 확장하고, 경계를 부드럽게 하고, 구멍을 막고 틈의 사이를 채우는 형태학적 연산
- 입력 영상 A와 형태소(SE) B가 주어질 때, 형태학적 팽창된 영상은 $f_D = A \oplus B$로 표현
- 입력 영상 A의 (i, j) 번째 화소를 (i, j)를 중심으로 하는 이웃 영역(B에 의해서 결정)에 속한 화소값들 중 최대값으로 설정.
- $b_{(i,j)}$를 B가 (i, j) 만큼 이동한 영역이라 할 때,

$$f_D(i, j) = \max_{b_{(i,j)}} A$$



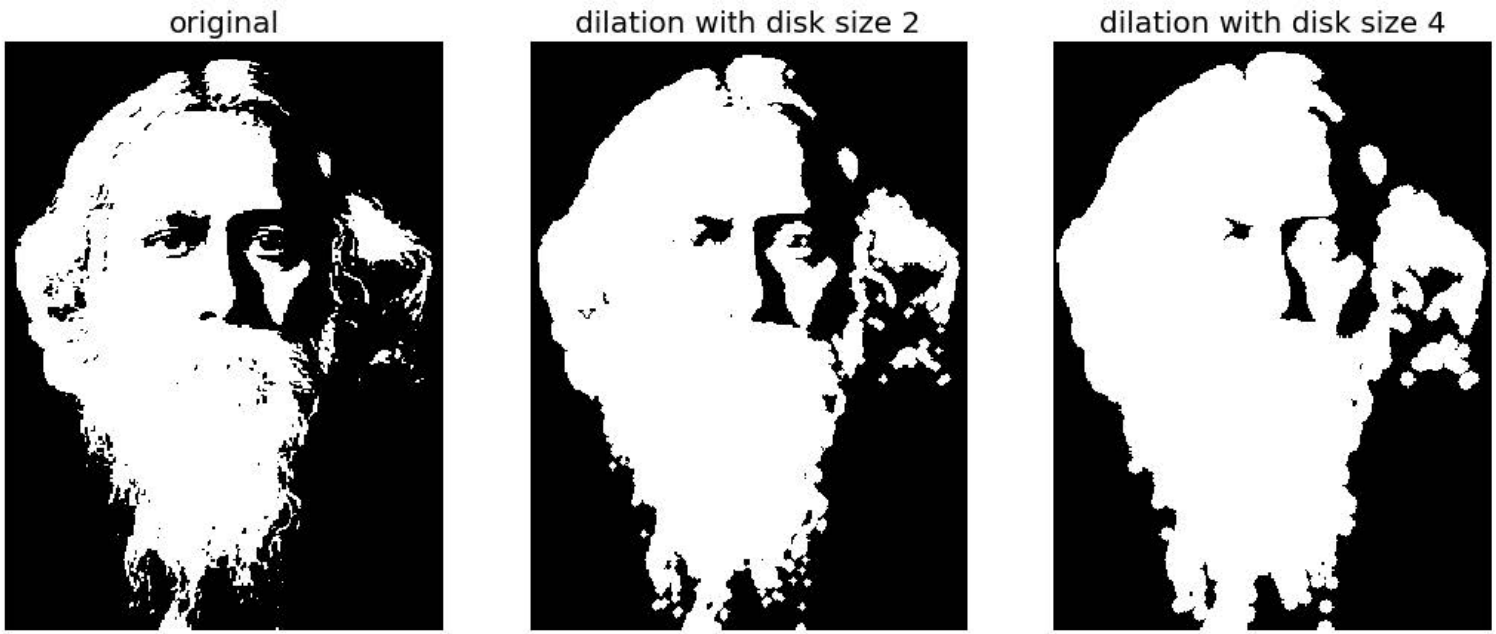
A B $f_D = A \oplus B$

팽창(Dilation) II

Listing – binary_dilation()을 사용한 팽창

```
1  from skimage.morphology import binary_dilation, disk
2  from skimage import img_as_float
3  im = img_as_float(imread('../images/tagore.png'))
4  im = 1 - im[...,3]
5  im[im ≤ 0.5] = 0
6  im[im > 0.5] = 1
7  pylab.gray()
8  pylab.figure(figsize=(18,9))
9  pylab.subplot(131)
10 pylab.imshow(im)
11 pylab.title('original', size=20)
12 pylab.axis('off')
13 for d in range(1,3):
14 pylab.subplot(1,3,d+1)
15 im1 = binary_dilation(im, disk(2*d))
16 pylab.imshow(im1)
17 pylab.title('dilation with disk size ' + str(2*d), size=20)
18 pylab.axis('off')
19 pylab.show()
```

팽창(Dilation) III



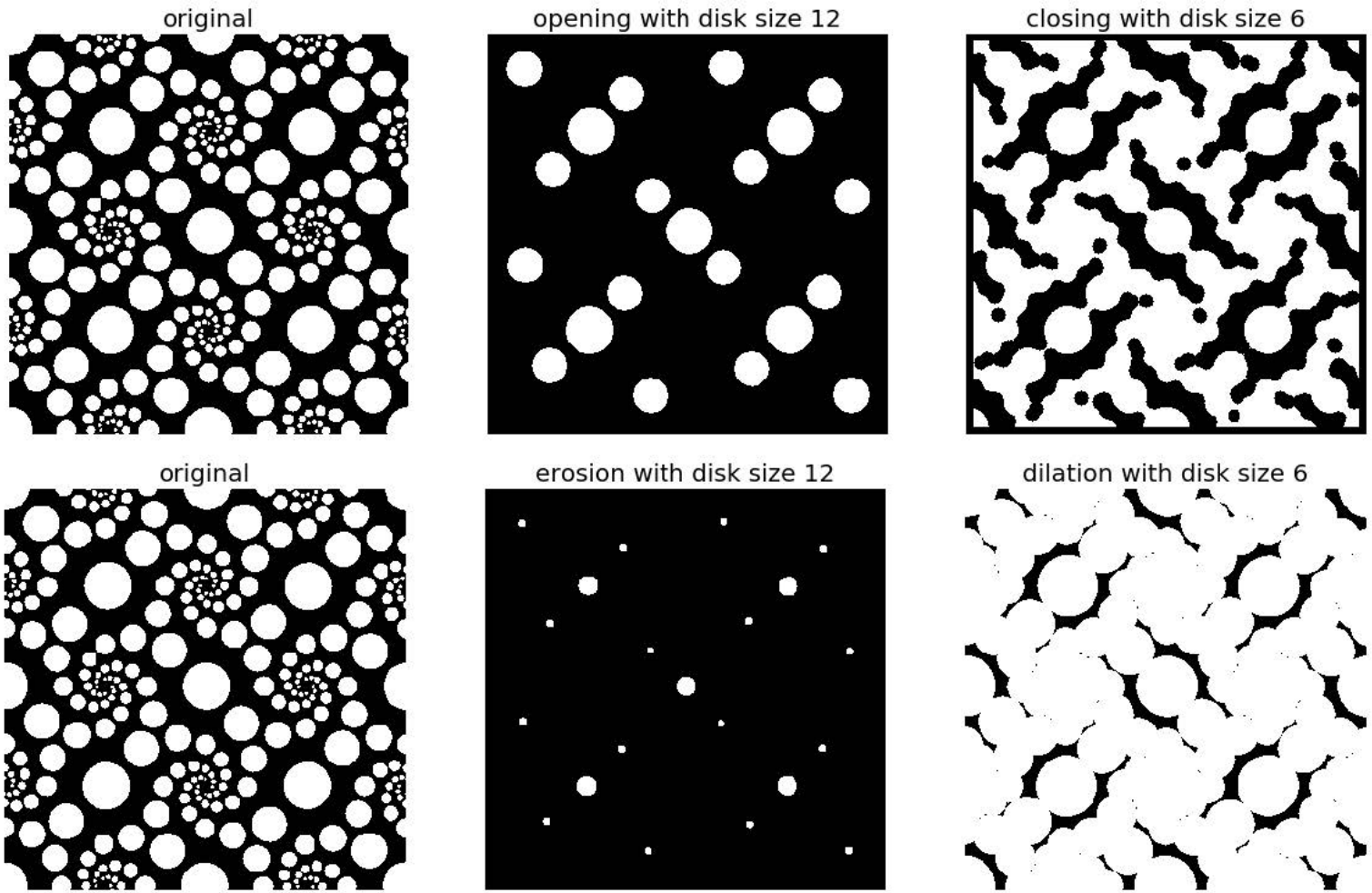
열림(opening)과 닫힘(closing) I

- 열림 : $f_O = (A \ominus B) \oplus B \rightarrow$ 작은 객체를 제거하고 큰 객체만 유지
- 닫힘 : $f_C = (A \oplus B) \ominus B \rightarrow$ 작은 구멍을 채움

Listing – scikit-image morphology 모듈의 함수를 사용한 opening과 closing

```
1  from skimage.morphology import binary_opening, binary_closing, ...
       binary_erosion, binary_dilation, disk
2  im = rgb2gray(imread('../images/circles.jpg'))
3  im[im ≤ 0.5] = 0
4  im[im > 0.5] = 1
5  pylab.gray()
6  pylab.figure(figsize=(20,10))
7  pylab.subplot(1,3,1), plot_image(im, 'original')
8  im1 = binary_opening(im, disk(12))
9  pylab.subplot(1,3,2), plot_image(im1, 'opening with disk size ' + str(12))
10 im1 = binary_closing(im, disk(6))
11 pylab.subplot(1,3,3), plot_image(im1, 'closing with disk size ' + str(6))
12 pylab.show()
```

열림(opening)과 닫힘(closing) II



골격화(skeletoning) I

- 개념 : 형태학적 세선화를 수행해서 단일 화소 너비 골격으로 축소

Listing – 이진 영상의 골격화

```
1  def plot_images_horizontally(original, filtered, filter_name, sz=(18,7)):
2      pylab.gray()
3      pylab.figure(figsize = sz)
4      pylab.subplot(1,2,1), plot_image(original, 'original')
5      pylab.subplot(1,2,2), plot_image(filtered, filter_name)
6      pylab.show()
7
8  from skimage.morphology import skeletonize
9  im = img_as_float(imread('../images/dynasaur.png')[...,3])
10 threshold = 0.5
11 im[im ≤ threshold] = 0
12 im[im > threshold] = 1
13 skeleton = skeletonize(im)
14 plot_images_horizontally(im, skeleton, 'skeleton',sz=(18,9))
```

골격화(skeletoning) II

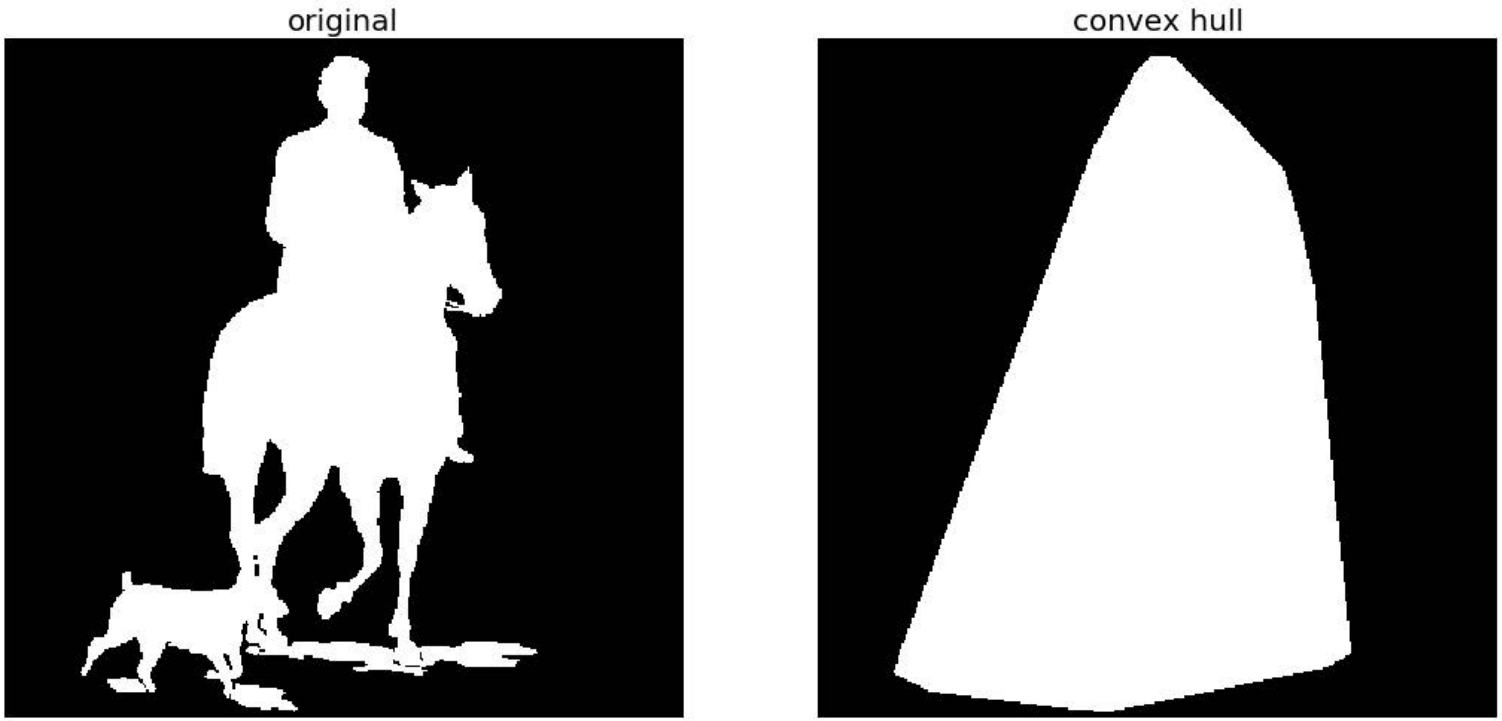
볼록 선체(convex hull) 계산 I

- 개념 : 입력 영상의 모든 전경(흰색 혹은 1인 화소)을 둘러싸는 가장 작은 볼록 다각형을 구하는 형태학적 연산

Listing – 볼록 선체 연산

```
1  from skimage.morphology import convex_hull_image
2  im = rgb2gray(imread('../images/horse-dog.jpg'))
3  threshold = 0.5
4  im[im < threshold] = 0 # convert to binary image
5  im[im ≥ threshold] = 1
6  chull = convex_hull_image(im)
7  plot_images_horizontally(im, chull, 'convex hull', sz=(18,9))
```

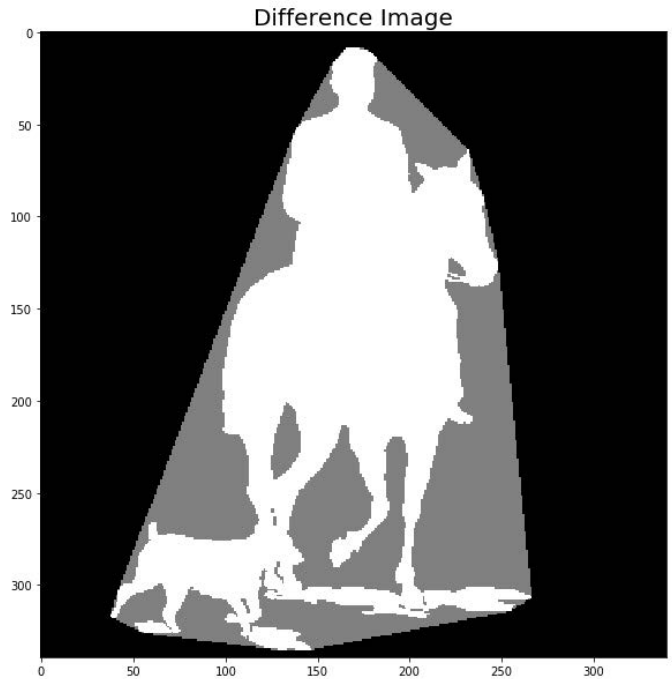
볼록 선체(convex hull) 계산 II



볼록 선체(convex hull) 계산 III

Listing – 입력 영상과 볼록 선체의 차분 영상

```
1  import numpy as np
2
3  im = im.astype(np.bool)
4  chull_diff = ...
       img_as_float(chull.copy())
5  chull_diff[im] = 2
6  pylab.figure(figsize=(20,10))
7  pylab.imshow(chull_diff, ...
       cmap=pylab.cm.gray, ...
       interpolation='nearest')
8  pylab.title('Difference Image', ...
       size=20)
9  pylab.show()
```



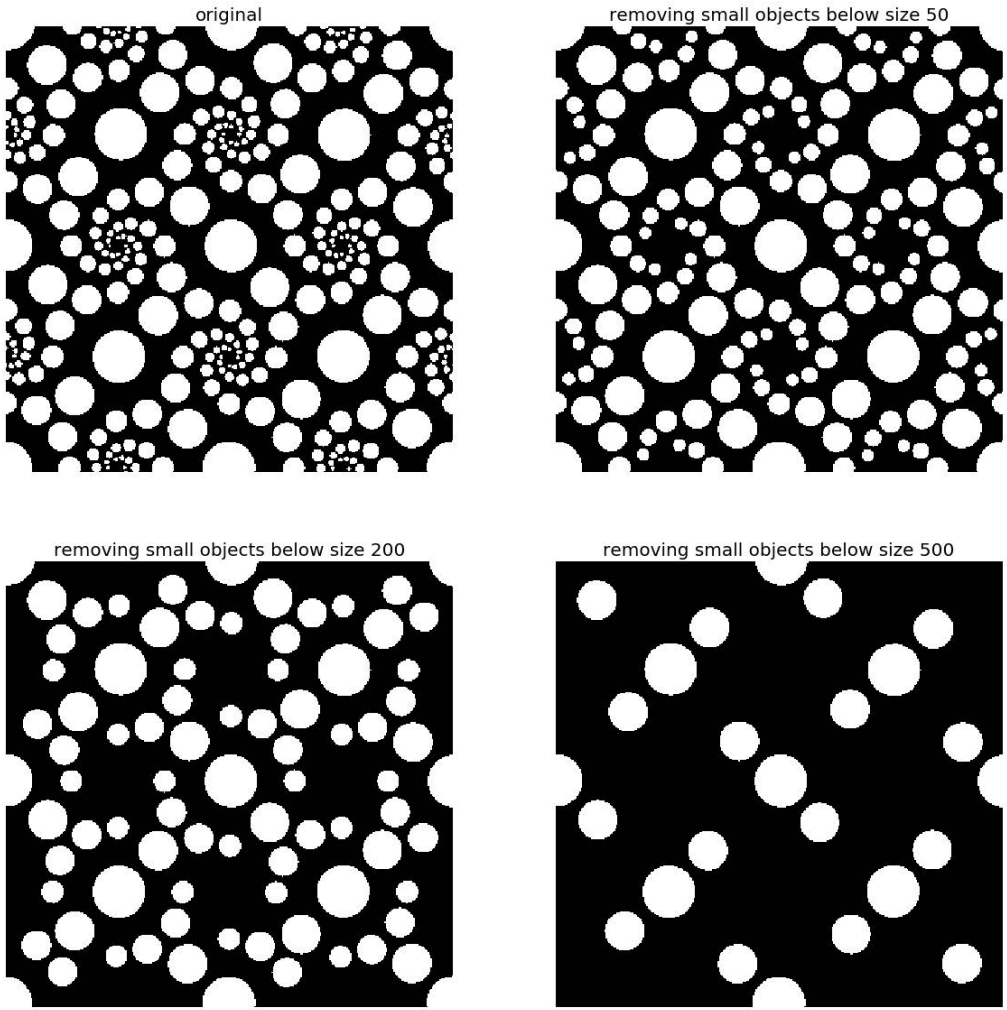
작은 객체 제거 I

- 개념 : `remove_small_objects()` 함수를 사용하여, 지정된 임계치보다 작은 객체를 모두 제거

Listing – 작은 객체 제거

```
1  from skimage.morphology import remove_small_objects
2  im = rgb2gray(imread('../images/circles.jpg'))
3  im[im > 0.5] = 1 # create binary image by thresholding with fixed threshold
4  0.5
5  im[im ≤ 0.5] = 0
6  im = im.astype(np.bool)
7  pylab.figure(figsize=(20,20))
8  pylab.subplot(2,2,1), plot_image(im, 'original')
9  i = 2
10 for osz in [50, 200, 500]:
11     im1 = remove_small_objects(im, osz, connectivity=1)
12     pylab.subplot(2,2,i), plot_image(im1, 'removing small objects below size ...
           ' + str(osz))
13     i += 1
```

작은 객체 제거 II


original
removing small objects below size 50
removing small objects below size 200
removing small objects below size 500

흰색과 검은색 탑햇(top-hats) I

- 흰색 탑햇 : 형태소보다 작은 밝은 점들을 계산, 입력영상과 그의 형태학적 열림 영상의 차분
- 검은색 탑햇 : 형태소보다 작은 어두운 점들을 계산, 입력영상과 그의 형태학적 닫힘 영상의 차분

Listing – scikit-image morphology 모듈 함수를 사용한 흰색과 검은색 탑햇 추출

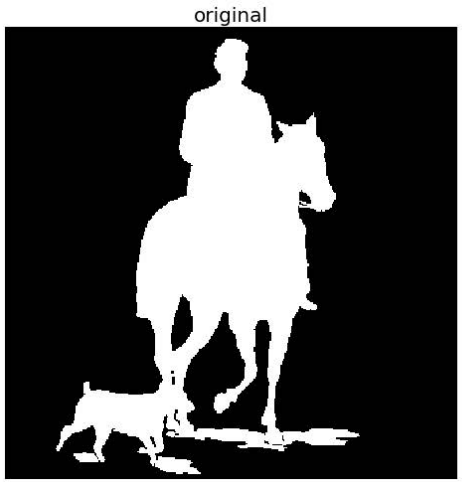
```
1  from skimage.morphology import white_tophat, black_tophat, square
2  im = imread('../images/tagore.png')[...,3]
3  im[im ≤ 0.5] = 0
4  im[im > 0.5] = 1
5  im1 = white_tophat(im, square(5))
6  im2 = black_tophat(im, square(5))
7  pylab.figure(figsize=(20,15))
8  pylab.subplot(1,2,1), plot_image(im1, 'white tophat')
9  pylab.subplot(1,2,2), plot_image(im2, 'black tophat')
10 pylab.show()
```

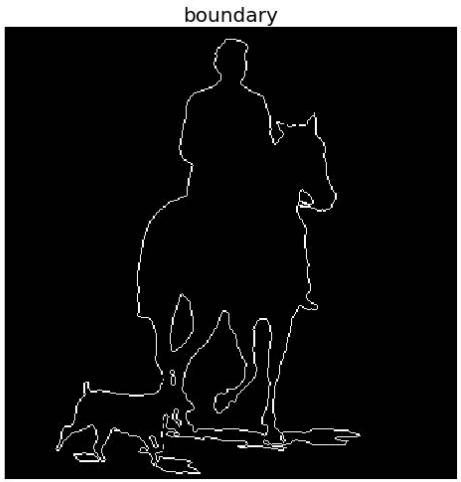
경계 추출 I

- 경계를 추출하기 위해서 입력 영상과 그의 침식 영상의 차분을 계산

Listing – 경계 추출

```
1  from skimage.morphology import binary_erosion
2  im = rgb2gray(imread('../images/horse-dog.jpg'))
3  threshold = 0.5
4  im[im < threshold] = 0
5  im[im ≥ threshold] = 1
6  boundary = im - binary_erosion(im)
7  plot_images_horizontally(im, boundary, 'boundary',sz=(18,9))
```





열림과 닫힘을 사용한 지문 개선 I

- 열림과 닫힘을 순차적으로 사용하여 입력 영상의 잡음(작은 전경 객체)을 제거

Listing – 경계 추출

```
1  im = rgb2gray(imread('../images/fingerprint.png'))
2  im[im ≤ 0.5] = 0 # binarize
3  im[im > 0.5] = 1
4  im_o = binary_opening(im, square(2))
5  im_c = binary_closing(im, square(2))
6  im_oc = binary_closing(binary_opening(im, square(2)), square(2))
7  pylab.figure(figsize=(20,20))
8  pylab.subplot(221), plot_image(im, 'original')
9  pylab.subplot(222), plot_image(im_o, 'opening')
10 pylab.subplot(223), plot_image(im_c, 'closing')
11 pylab.subplot(224), plot_image(im_oc, 'opening + closing')
12 pylab.show()
```

열림과 닫힘을 사용한 지문 개선 II

명암도 연산 I

Listing – 명암도 연산의 침식

```
1  from skimage.morphology import dilation, erosion, closing, opening, square
2  im = imread('../images/zebras.jpg')
3  im = rgb2gray(im)
4  struct_elem = square(5)
5  eroded = erosion(im, struct_elem)
6  plot_images_horizontally(im, eroded, 'erosion')
```

명암도 연산 II

Listing – 명암도 연산의 팽창

```
1  dilated = dilation(im, struct_elem)
2  plot_images_horizontally(im, dilated, 'dilation')
```

명암도 연산 III

Listing – 명암도 연산의 열림

```
1  opened = opening(im, struct_elem)
2  plot_images_horizontally(im, opened, 'opening')
```

명암도 연산 IV

Listing – 명암도 연산의 닫힘

```
1  closed = closing(im, struct_elem)
2  plot_images_horizontally(im, closed, 'closing')
```

Table of Contents

형태학적 콘트라스트 향상 I

scikit-image의 filter.rank 모듈의 형태학적 콘트라스트 강조 필터는 형태소에 의해서 정해진 이웃 화소만을 고려. 원 화소의 값에 따라 중심 화소를 지역 최소 혹은 지역 최대값으로 변경

Listing – 형태학적 콘트라스트 향상과 적응형 히스토그램 평활화 방법 비교

```
1  from skimage.filters.rank import enhance_contrast
2  from skimage import exposure
3  def plot_gray_image(ax, image, title):
4  ax.imshow(image, cmap=pylab.cm.gray),
5  ax.set_title(title), ax.axis('off')
6  ax.set_adjustable('box')
7
8  image = rgb2gray(imread('../images/squirrel.jpg'))
9  sigma = 0.05
10 noisy_image = np.clip(image + sigma * ...
       np.random.standard_normal(image.shape), 0, 1)
11 enhanced_image = enhance_contrast(noisy_image, disk(5))
12 equalized_image = exposure.equalize_adapthist(noisy_image)
13
14 fig, axes = pylab.subplots(1, 3, figsize=[18, 7], sharex='row',sharey='row')
15 axes1, axes2, axes3 = axes.ravel()
16 plot_gray_image(axes1, noisy_image, 'Original')
17 plot_gray_image(axes2, enhanced_image, 'Local morphological contrast ...
       enhancement')
18 plot_gray_image(axes3, equalized_image, 'Adaptive Histogram equalization')
```

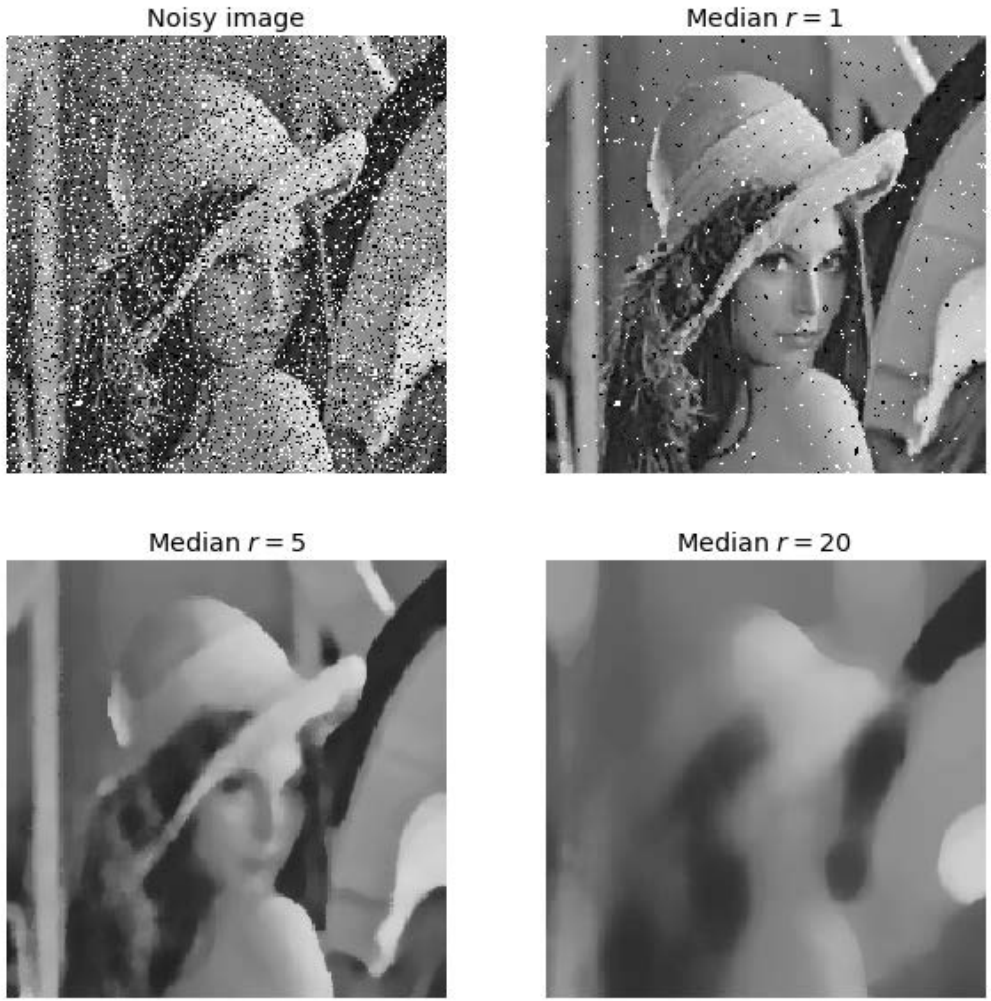
형태학적 콘트라스트 향상 II

메디안 필터를 사용한 잡음 제거 I

scikit-image filters.rank 모듈의 형태학적 메디안 필터를 사용한 잡음 제거

```
1  rom skimage.filters.rank import median
2  from skimage.morphology import disk
3  noisy_image = (rgb2gray(imread('../images/lena.jpg'))*255).astype(np.uint8)
4  noise = np.random.random(noisy_image.shape)
5  noisy_image[noise > 0.9] = 255
6  noisy_image[noise < 0.1] = 0
7  fig, axes = pylab.subplots(2, 2, figsize=(10, 10), sharex=True, sharey=True)
8  axes1, axes2, axes3, axes4 = axes.ravel()
9  plot_gray_image(axes1, noisy_image, 'Noisy image')
10 plot_gray_image(axes2, median(noisy_image, disk(1)), 'Median $r=1$')
11 plot_gray_image(axes3, median(noisy_image, disk(5)), 'Median $r=5$')
12 plot_gray_image(axes4, median(noisy_image, disk(20)), 'Median $r=20$')
```

메디안 필터를 사용한 잡음 제거 II



로컬 엔트로피 계산 I

엔트로피(entropy) : 영상의 불확실성 혹은 임의성의 척도 :

$$H = -\sum_{i=0}^{255} p_i \log_2 p_i$$

p_i : 명암도 영상에서 밝기값 i를 갖는 화소의 확률(빈도)

Listing – 로컬 엔트로피 계산

```
1  from skimage.morphology import disk
2  from skimage.filters.rank import entropy
3  image = rgb2gray(imread('../images/birds.png'))
4  fig, (axes1, axes2) = pylab.subplots(1, 2, figsize=(18, 10), sharex=True, ...
       sharey=True)
5  fig.colorbar(axes1.imshow(image, cmap=pylab.cm.gray), ax=axes1)
6  axes1.axis('off'), axes1.set_title('Image', size=20), ...
       axes1.set_adjustable('box')
7  fig.colorbar(axes2.imshow(entropy(image, disk(5)), cmap=pylab.cm.inferno), ...
       ax=axes2)
8  axes2.axis('off'), axes2.set_title('Entropy', size=20), ...
       axes2.set_adjustable('box')
9  pylab.show()
```

로컬 엔트로피 계산 II

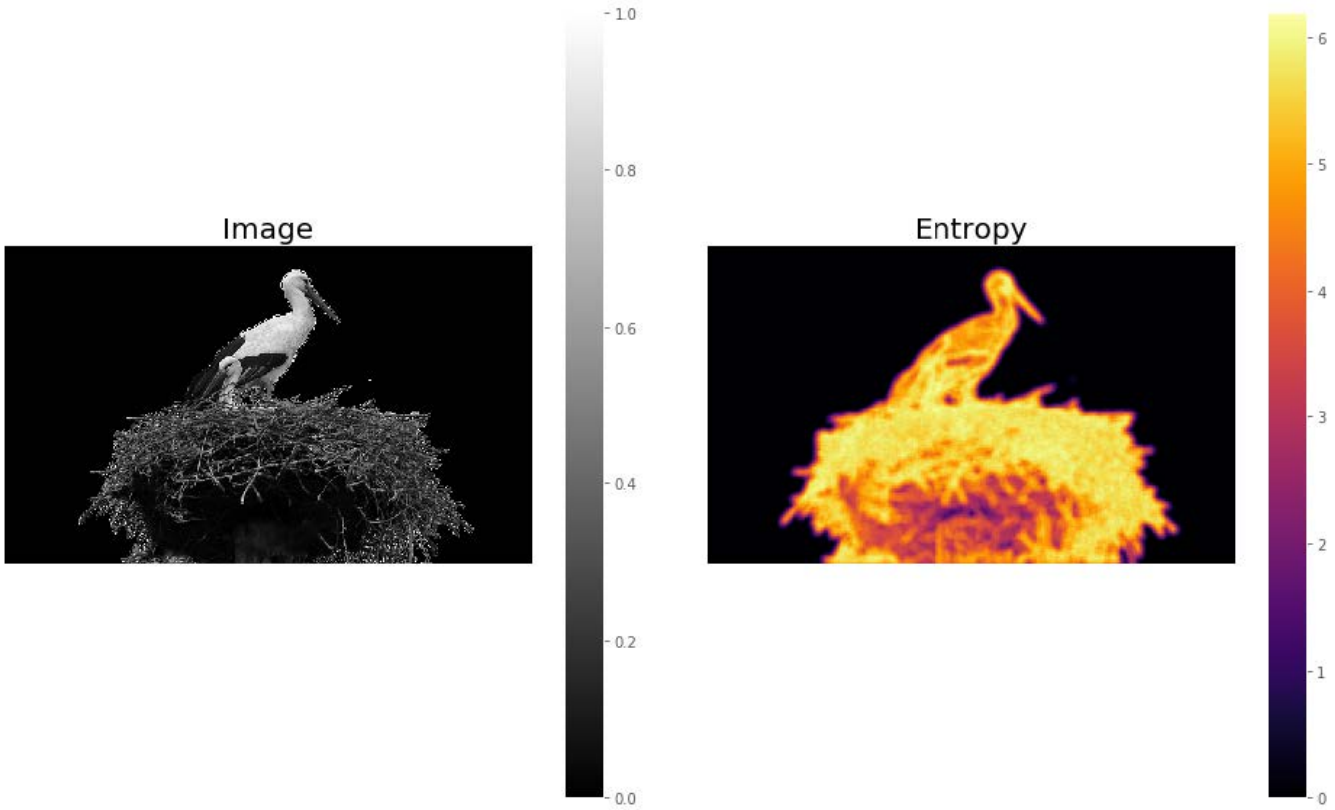


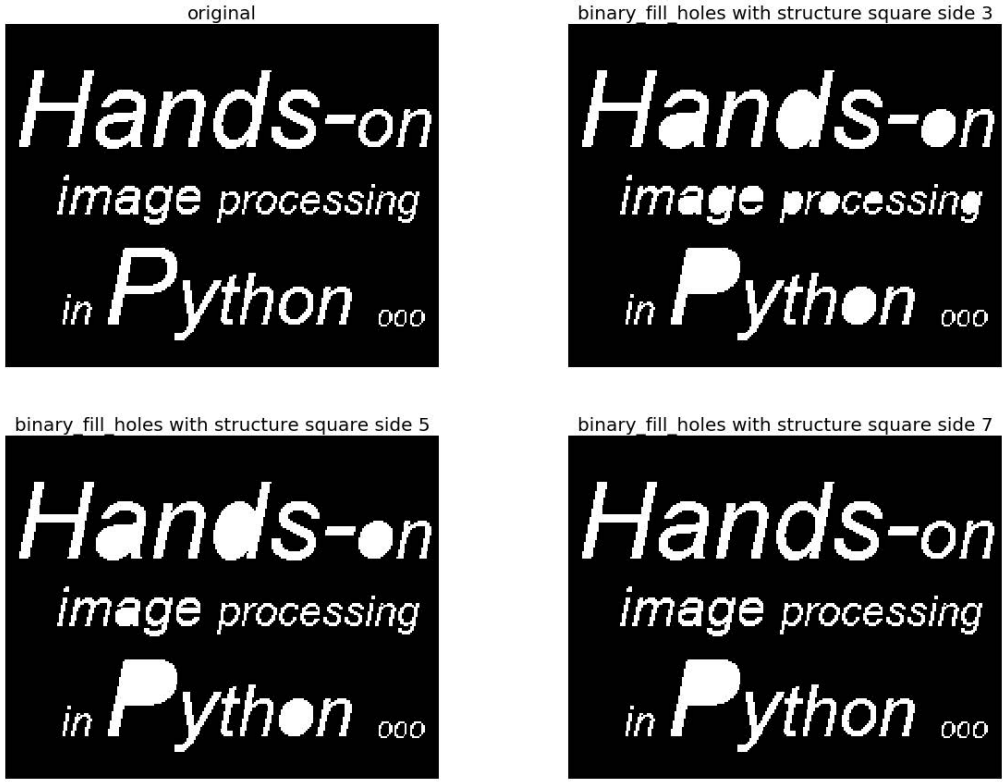
Table of Contents

이진 객체의 구멍 채우기 I

scipy ndimage.morphology 모듈 함수를 사용

```
1  from scipy.ndimage.morphology import binary_fill_holes
2  im = rgb2gray(imread('../images/text1.png'))
3  im[im ≤ 0.5] = 0
4  im[im > 0.5] = 1
5  pylab.figure(figsize=(20,15))
6  pylab.subplot(221), pylab.imshow(im), pylab.title('original', ...
       size=20),pylab.axis('off')
7  i = 2
8  for n in [3,5,7]:
9      pylab.subplot(2, 2, i)
10     im1 = binary_fill_holes(im, structure=np.ones((n,n)))
11     pylab.imshow(im1), pylab.title('binary_fill_holes with structure square ...
           side ' + str(n), size=20)
12     pylab.axis('off')
13     i += 1
14 pylab.show()
```

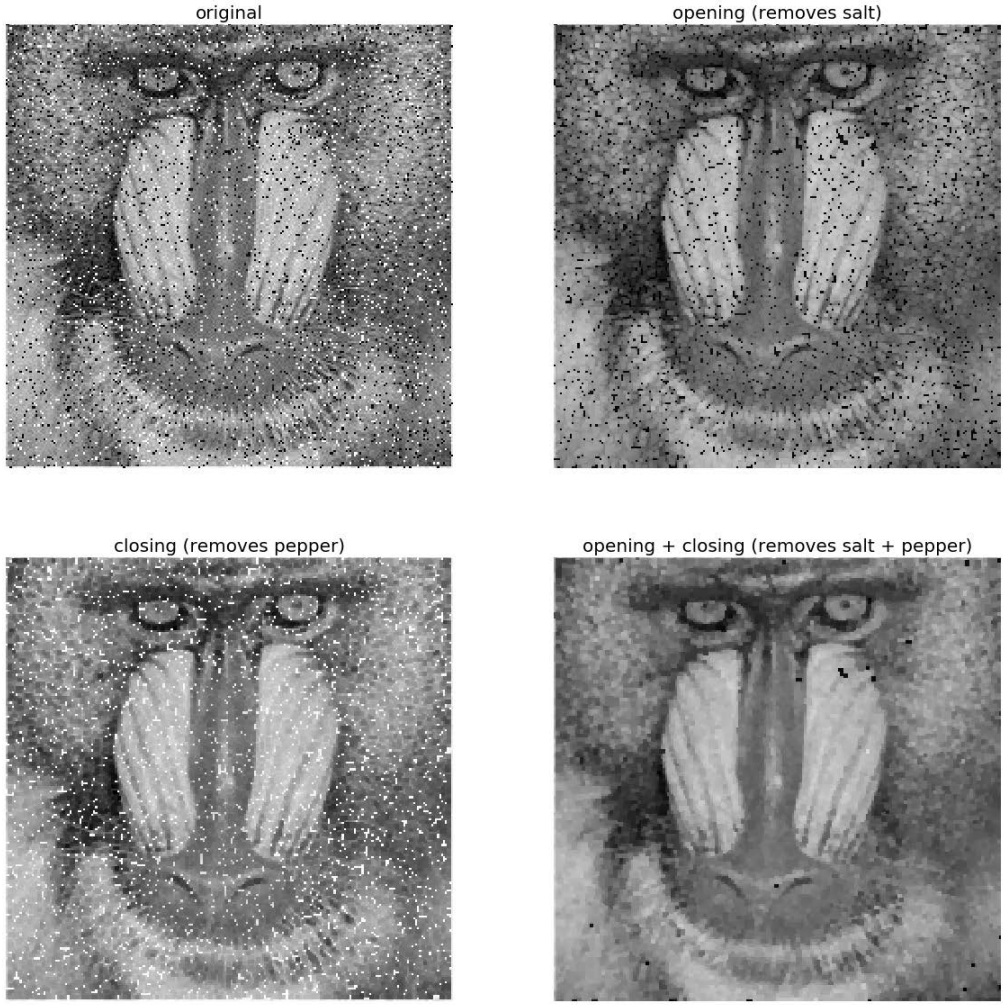
이진 객체의 구멍 채우기 II



열림과 닫힘을 사용한 잡음 제거 I

```
1  from scipy import ndimage
2  im = rgb2gray(imread('../images/mandrill_spnoise_0.1.jpg'))
3  im_o = ndimage.grey_opening(im, size=(2,2))
4  im_c = ndimage.grey_closing(im, size=(2,2))
5  im_oc = ndimage.grey_closing(ndimage.grey_opening(im, size=(2,2)), size=(2,2))
6  pylab.figure(figsize=(20,20))
7  pylab.subplot(221), pylab.imshow(im), pylab.title('original', size=20), ...
       pylab.axis('off')
8  pylab.subplot(222), pylab.imshow(im_o), pylab.title('opening (removes ...
       salt)', size=20), pylab.axis('off')
9  pylab.subplot(223), pylab.imshow(im_c), pylab.title('closing (removes ...
       pepper)', size=20),pylab.axis('off')
10 pylab.subplot(224), pylab.imshow(im_oc), pylab.title('opening + closing ...
       (removes salt + pepper)', size=20)
11 pylab.axis('off')
12 pylab.show()
```

열림과 닫힘을 사용한 잡음 제거 II

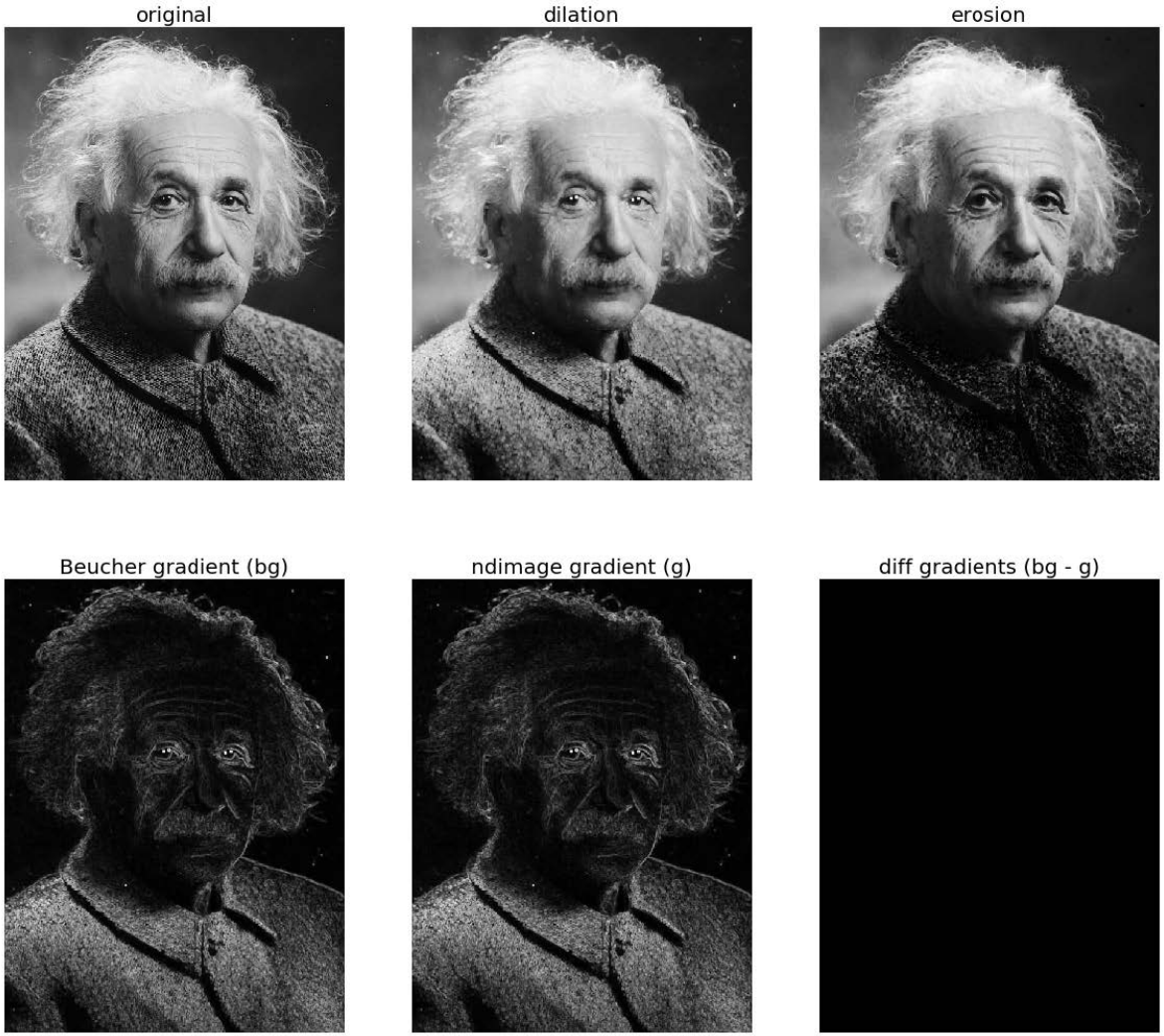


형태학적 베커(Beucher) 그레디언트 계산 I

형태학적 베커 그레디언트는 명암도 영상의 팽창된 버전과 침식된 버전의 차분으로 계산

```
1  from scipy import ndimage
2  im = rgb2gray(imread('../images/einstein.jpg'))
3  im_d = ndimage.grey_dilation(im, size=(3,3))
4  im_e = ndimage.grey_erosion(im, size=(3,3))
5  im_bg = im_d - im_e
6  im_g = ndimage.morphological_gradient(im, size=(3,3))
7  pylab.gray()
8  pylab.figure(figsize=(20,18))
9  pylab.subplot(231), pylab.imshow(im), pylab.title('original', size=20),
10 pylab.axis('off')
11 pylab.subplot(232), pylab.imshow(im_d), pylab.title('dilation', size=20),
12 pylab.axis('off')
13 pylab.subplot(233), pylab.imshow(im_e), pylab.title('erosion', size=20),
14 pylab.axis('off')
15 pylab.subplot(234), pylab.imshow(im_bg), pylab.title('Beucher gradient ...
       (bg)', size=20), pylab.axis('off')
16 pylab.subplot(235), pylab.imshow(im_g), pylab.title('ndimage gradient (g)', ...
       size=20), pylab.axis('off')
17 pylab.subplot(236), pylab.title('diff gradients (bg - g)', size=20), ...
       pylab.imshow(im_bg - im_g)
18 pylab.axis('off')
19 pylab.show()
```

형태학적 베커(Beucher) 그레디언트 계산 II



형태학적 라플라스 계산

```
1  im = imread('../images/tagore.png')[...,3]
2  im_g = ndimage.morphological_gradient(im, size=(3,3))
3  im_l = ndimage.morphological_laplace(im, size=(5,5))
4  pylab.figure(figsize=(15,10))
5  pylab.subplot(121), pylab.title('ndimage morphological laplace', size=20), ...
       pylab.imshow(im_l)
6  pylab.axis('off')
7  pylab.subplot(122), pylab.title('ndimage morphological gradient', size=20),
8  pylab.imshow(im_g)
9  pylab.axis('off')
10 pylab.show()
```

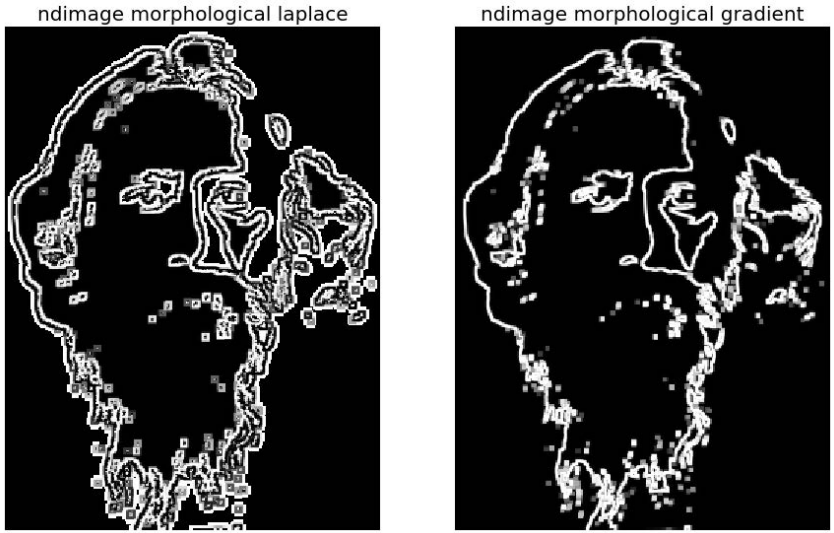


Table of Contents

Homework 06

1. CH06.ipynb 파일의 셀들을 실행해보고, 모든 셀 들의 내용을 요약 설명하고, 해당 결과를 출력해서 제출하시오.

Table of Contents

References

[1] R. Gonzales and R. Woods, *Digital image processing 4th edition*.
Pearson, 2018.

[2] S. Dey, *Hands-On Image Processing with Python : Expert techniques for advanced image analysis and effective interpretation of image data*.
Packt Publishing Ltd, 2018.

[3] "Hands-on-image-processing-with-python." https://github.com/PacktPublishing/Hands-On-Image-Processing-with-Python.
Accessed : 2022-07-16.