1.

|  |
| --- |
| 입력된 영상의 대부분의 화소를 이동하면서 마스크와 곱하여 수행하는 것을 회선이라고 한다. |

2.

|  |
| --- |
| 블러링 : 영상에서 화소값이 급격하게 변하는 부분을 감소시켜 영상을 부드럽게 만드는 기술  샤프닝 : 화소끼리 차이를 크게 되도록 만들어 날카로운 느낌이 나도록 만드는 기술 |

3.

|  |
| --- |
| 로버츠에지검출, 프리윗 에지, 소밸 에지검출,  라플라시안 에지 검출, LoG, DoG에지 검출, 캐니 에지검출 |

4.

|  |
| --- |
| 로버츠 마스크  장점 : 다른 1차 미분 마스크들보다 계산이 단순하다  단점: 잡음에 매우 민감하다.  프리윗 마스크  장점 : 수직,수평 에지 동등하게 찾는데 효과적이다.  단점 : 변화에 대한 비중이 적어 에지추출시 에지가 덜 부각된다.  소벨 마스크  장점 : 대각선 방향의 에지도 추출이 잘된다.  단점 : 마스크의 크기가 커지면 오히려 에지 검출이 잘 안될 수 있다. |

5.

|  |
| --- |
| 1. 라플라시안  잡음 성분에 민감하여 실제보다 더 많은 에지를 검출하게 된다.  2.LoG  복잡한 공식에 의해 마스크를 생성하기 때문에 수행시간이 오래 걸린다.  3.DoG  LoG와 기능은 유사하지만 수행시간이 오래 걸린다는 단점을 보안한 알고리즘이다.  4. 캐니  네 단계의 알고리즘으로 구성되어 OpenCV에서 제공하는 함수의 성능과 비슷하다. |

6.

|  |
| --- |
| 1. 가우시안 블러링을 통한 노이즈 제거  2. 소벨마스크를 이용해 화소기울기의 강도와 방향 검출  3. 비최대치 억제  4. 이력 임계값으로 에지 결정 |

7.

|  |
| --- |
| 마스크로 씌워진 부분의 화소의 최소, 최댓값을 출력화소로 지정하는 방법이다 |

8.

|  |
| --- |
| 평균값 필터링 : 마스크로 씌워진 부분의 평균을 출력화소로 지정하는 방법  미디안 필터링 : 마스크로 씌워진 부분의 중간값을 출력화소로 지정하는 방법이다. |

9.

|  |
| --- |
| 열림연산 : 침식연산을 먼저 수행하고 팽창연산으로 객체의 크기 축소를 방지한다.  닫힘연산 : 팽창연산을 먼저 수행하고 침식연산으로 객체의 크기 확장을 방지한다. |

10.1(bluring)

|  |
| --- |
| import numpy as np, cv2  def filter(image, mask):  rows, cols = image.shape[:2]  dst = np.zeros((rows, cols), np.float32)   for i in range(1, rows -1):  for j in range(1, cols - 1):  y1, y2 = i - 1, i + 2  x1, x2 = j - 1, j + 2  roi = image[y1:y2, x1:x2].astype('float32')  tmp = cv2.multiply(roi, mask)  dst[i, j] = cv2.sumElems(tmp)[0]  return dst  image = cv2.imread("C:/computervision/chap07/image/img1.jpg", cv2.IMREAD\_COLOR) if image is None : raise Exception("영상파일 읽기 오류") data = [1/9, 1/9, 1/9,  1/9, 1/9, 1/9,  1/9, 1/9, 1/9]  b, g, r = cv2.split(image) mask = np.array(data, np.float32).reshape(3, 3)  blur\_b = filter(b, mask) blur\_b = blur\_b.astype('uint8') blur\_g = filter(g, mask) blur\_g = blur\_g.astype('uint8') blur\_r = filter(r, mask) blur\_r = blur\_r.astype('uint8')  dst = cv2.merge((blur\_b, blur\_g, blur\_r))  opencv = cv2.filter2D(image, -1, mask) cv2.imshow("blur\_user", dst) cv2.imshow("orginal", image) cv2.imshow("blur\_opencv", opencv) cv2.waitKey()1 |

10.2(sharpening)

|  |
| --- |
| import numpy as np, cv2  def filter(image, mask):  rows, cols = image.shape[:2]  dst = np.zeros((rows, cols), np.float32)   for i in range(1, rows -1):  for j in range(1, cols - 1):  y1, y2 = i - 1, i + 2  x1, x2 = j - 1, j + 2  roi = image[y1:y2, x1:x2].astype('float32')  tmp = cv2.multiply(roi, mask)  dst[i, j] = cv2.sumElems(tmp)[0]  return dst  image = cv2.imread("C:/computervision/chap07/image/img3.jpg", cv2.IMREAD\_COLOR) if image is None : raise Exception("영상파일 읽기 오류") data = [ 0, -1, 0,  -1, 5, -1,  0, -1, 0]  b, g, r = cv2.split(image) mask = np.array(data, np.float32).reshape(3, 3)  blur\_b = filter(b, mask) blur\_b = blur\_b.astype('uint8') blur\_g = filter(g, mask) blur\_g = blur\_g.astype('uint8') blur\_r = filter(r, mask) blur\_r = blur\_r.astype('uint8')  dst = cv2.merge((blur\_b, blur\_g, blur\_r))  opencv = cv2.filter2D(image, -1, mask) cv2.imshow("sharpening\_user", dst) cv2.imshow("orginal", image) cv2.imshow("sharpening\_opencv", opencv) cv2.waitKey() |

11.1(prewitt)

|  |
| --- |
| import numpy as np, cv2  def filter(image, mask):  rows, cols = image.shape[:2]  dst = np.zeros((rows, cols), np.float32)   for i in range(1, rows -1):  for j in range(1, cols - 1):  y1, y2 = i - 1, i + 2  x1, x2 = j - 1, j + 2  roi = image[y1:y2, x1:x2].astype('float32')  tmp = cv2.multiply(roi, mask)  dst[i, j] = cv2.sumElems(tmp)[0]  return dst  image = cv2.imread("C:/computervision/chap07/image/img1.jpg", cv2.IMREAD\_GRAYSCALE) if image is None : raise Exception("영상파일 읽기 오류") data1 = [-1, 0, 1,  -1, 0, 1,  -1, 0, 1] data2 = [-1, -1, -1,  0, 0, 0,  1, 1, 1]  mask1 = np.array(data1, np.float32).reshape(3, 3) mask2 = np.array(data2, np.float32).reshape(3, 3)  dst1 = filter(image, mask1) dst2 = filter(image, mask2) dst1, dst2 = np.abs(dst1), np.abs(dst2) dst = cv2.magnitude(dst1, dst2)  dst = np.clip(dst, 0, 255).astype('uint8')  cv2.imshow("image", image) cv2.imshow("prewitt", dst) cv2.waitKey() |

11.2(robers)

|  |
| --- |
| import numpy as np, cv2  def filter(image, mask):  rows, cols = image.shape[:2]  dst = np.zeros((rows, cols), np.float32)   for i in range(1, rows -1):  for j in range(1, cols - 1):  y1, y2 = i - 1, i + 2  x1, x2 = j - 1, j + 2  roi = image[y1:y2, x1:x2].astype('float32')  tmp = cv2.multiply(roi, mask)  dst[i, j] = cv2.sumElems(tmp)[0]  return dst  image = cv2.imread("C:/computervision/chap07/image/img1.jpg", cv2.IMREAD\_GRAYSCALE) if image is None : raise Exception("영상파일 읽기 오류") data1 = [-1, 0, 0,  0, 1, 0,  0, 0, 0] data2 = [0, 0, -1,  0, 1, 0,  0, 0, 0]  mask1 = np.array(data1, np.float32).reshape(3, 3) mask2 = np.array(data2, np.float32).reshape(3, 3)  dst1 = filter(image, mask1) dst2 = filter(image, mask2) dst1, dst2 = np.abs(dst1), np.abs(dst2) dst = cv2.magnitude(dst1, dst2)  dst = np.clip(dst, 0, 255).astype('uint8')  cv2.imshow("image", image) cv2.imshow("roberts", dst)  cv2.waitKey() |

11.3(sobel)

|  |
| --- |
| import numpy as np, cv2  def filter(image, mask):  rows, cols = image.shape[:2]  dst = np.zeros((rows, cols), np.float32)   for i in range(1, rows -1):  for j in range(1, cols - 1):  y1, y2 = i - 1, i + 2  x1, x2 = j - 1, j + 2  roi = image[y1:y2, x1:x2].astype('float32')  tmp = cv2.multiply(roi, mask)  dst[i, j] = cv2.sumElems(tmp)[0]  return dst  image = cv2.imread("C:/computervision/chap07/image/img1.jpg", cv2.IMREAD\_GRAYSCALE) if image is None : raise Exception("영상파일 읽기 오류") data1 = [-1, 0, 1,  -2, 0, 2,  -1, 0, 1] data2 = [-1, -2, -1,  0, 0, 0,  1, 2, 1]  mask1 = np.array(data1, np.float32).reshape(3, 3) mask2 = np.array(data2, np.float32).reshape(3, 3)  dst1 = filter(image, mask1) dst2 = filter(image, mask2) dst1, dst2 = np.abs(dst1), np.abs(dst2) dst = cv2.magnitude(dst1, dst2)  dst = np.clip(dst, 0, 255).astype('uint8')  cv2.imshow("image", image) cv2.imshow("sobel", dst) cv2.waitKey() |

12.

|  |
| --- |
| import numpy as np, cv2  image = cv2.imread("C:/computervision/chap07/image/img1.jpg", cv2.IMREAD\_GRAYSCALE) if image is None : raise Exception("영상파일 읽기 오류")  lap = cv2.Laplacian(image, cv2.CV\_16S, 1)  gaus1 = cv2.GaussianBlur(image, (3, 3), 0) gaus2 = cv2.GaussianBlur(image, (9, 9), 0) DoG = gaus1 - gaus2  canny = cv2.Canny(image, 100, 150)  cv2.imshow("image", image) cv2.imshow("Laplacian", lap.astype('uint8')) cv2.imshow("DoG", DoG.astype('uint8')) cv2.imshow("Canny", canny.astype('uint8'))  cv2.waitKey() |

13.

|  |
| --- |
| import numpy as np, cv2  image = cv2.imread("C:/computervision/chap07/image/img1.jpg", cv2.IMREAD\_GRAYSCALE) if image is None : raise Exception("이미지 읽기 오류")  kernel = np.ones((5,5), np.float32)/25 convolution = cv2.filter2D(image, -1, kernel)  averaging = cv2.blur(image, (5,5))  gaussian = cv2.GaussianBlur(image, (5,5), 0)  median = cv2.medianBlur(image, 5)  bilateral = cv2.bilateralFilter(image, 9, 75, 75)  cv2.imshow("convolution", convolution) cv2.imshow("averaging", averaging) cv2.imshow("gaussian", gaussian) cv2.imshow("median", median) cv2.imshow("bilateral", bilateral)  cv2.waitKey() |

14.

|  |
| --- |
| import numpy as np, cv2  image=cv2.imread('C:/computervision/chap07/image/img1.jpg', cv2.IMREAD\_GRAYSCALE) if image is None : raise Exception("이미지읽기 오류") cv2.namedWindow('dst')  def threshold(value):  pass  cv2.createTrackbar('th1', 'dst', 0, 255, threshold) cv2.createTrackbar('th2', 'dst', 0, 255, threshold)  while True:  if cv2.waitKey(30) >= 0: break   th1 = cv2.getTrackbarPos('th1', 'dst')  th2 = cv2.getTrackbarPos('th2', 'dst')   dst = cv2.Canny(image, th1, th2)  cv2.imshow('dst', dst) |

15.

|  |
| --- |
| import numpy as np, cv2  def median\_filter(image, size):  rows, cols = image.shape[:2]  dst = np.zeros((rows, cols), np.uint8)   for i in range(1, rows -1):  for j in range(1, cols - 1):  y1, y2 = i - 1, i + 2  x1, x2 = j - 1, j + 2  mask = image[y1:y2, x1:x2].flatten()   sort\_mask = cv2.sort(mask, cv2.SORT\_EVERY\_COLUMN)  dst[i, j] = sort\_mask[sort\_mask.size//2]   return dst  def salt\_pepper\_noise(img, n):  h, w = img.shape[:2]  x, y = np.random.randint(0, w, n), np.random.randint(0, h, n)  noise = img.copy()  for(x, y) in zip(x,y):  noise[y, x] = 0 if np.random.rand() < 0.5 else 255  return noise  image = cv2.imread("c:/computervision/chap07/image/1.jpg", cv2.IMREAD\_GRAYSCALE) if image is None : raise Exception("영상파일 읽기 오류")  noise = salt\_pepper\_noise(image, 500) med\_img = median\_filter(noise, 5)  cv2.imshow("image", image) cv2.imshow("noise", noise) cv2.imshow("median", med\_img) cv2.waitKey() |

16.

|  |
| --- |
| import numpy as np, cv2  def mopholgy(image, mask=None):  dst = np.zeros(image.shape, np.uint8)  if mask is None : mask = np.ones((3, 3), np.uint8)   rows, cols = image.shape[:2]  mcnt = cv2.countNonZero(mask)  for i in range(1, rows -1):  for j in range(1, cols - 1):  y1, y2 = i - 1, i + 2  x1, x2 = j - 1, j + 2   roi = image[y1:y2, x1:x2]  temp = cv2.bitwise\_and(roi, mask)  cnt = cv2.countNonZero(temp)  dst[i, j] = 255 if (cnt == mcnt) else 0   for i in range(1, rows -1):  for j in range(1, cols - 1):  y1, y2 = i - 1, i + 2  x1, x2 = j - 1, j + 2   roi = image[y1:y2, x1:x2]  temp = cv2.bitwise\_and(roi, mask)  cnt = cv2.countNonZero(temp)  dst[i, j] = 0 if (cnt == 0) else 255  return dst  def salt\_pepper\_noise(img, n):  h, w = img.shape[:2]  x, y = np.random.randint(0, w, n), np.random.randint(0, h, n)  noise = img.copy()  for(x, y) in zip(x,y):  noise[y, x] = 0 if np.random.rand() < 0.5 else 255  return noise  image = cv2.imread("c:/computervision/chap07/image/1.jpg", cv2.IMREAD\_GRAYSCALE) if image is None : raise Exception("영상파일 읽기 오류") noise = salt\_pepper\_noise(image, 10000)  data = [0, 1, 0,  1, 1, 1,  0, 1, 0] mask = np.array(data, np.uint8).reshape(3, 3) th\_img = cv2.threshold(noise, 128, 255, cv2.THRESH\_BINARY)[1]  dst = mopholgy(th\_img, mask)  cv2.imshow('image', noise) cv2.imshow('binary', th\_img) cv2.imshow('mopholgy', dst) cv2.waitKey() |

17.

|  |
| --- |
| import numpy as np, cv2  c = 1 while True:  key = cv2.waitKeyEx(30)  if key == 0x280000:  c-=1  elif key == 0x260000:  c+=1  elif key == 0x1B:  break   fname = "C:/computervision/chap07/image/{0:01d}.jpg".format(c)  image = cv2.imread( fname, cv2.IMREAD\_COLOR)  if image is None : print(str(c) + "번 영상이 없습니다."); continue   mask = np.ones((5, 17), np.uint8)  gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  gray = cv2.blur(gray, (5,5))  gray = cv2.Sobel(gray, cv2.CV\_8U, 1, 0, 5)   th\_img = cv2.threshold(gray, 120, 255, cv2.THRESH\_BINARY)[1]  morph = cv2.morphologyEx(th\_img, cv2.MORPH\_CLOSE, mask, iterations=3)   cv2.imshow('image', image)  cv2.imshow('binary', th\_img)  cv2.imshow('opening', morph) |