1.

|  |
| --- |
| 변환영역: 직교변환에 의해 얻어진 영상 데이터의 다른 표현 (화소 값이 직접 표현되지 않고 변환계수로 표현된다.)  **주파수 변환영역의 처리과정**  1. 영상이 입력되면 주파수 영역으로 변환.  2. 주파수 변환으로 얻어진 계수의 특정 주파수 영역에 원하는 영상처리를 적용  3. 영상처리가 적용된 데이터를 주파수 역변환을 통해 공간영역의 영상으로 변환해서 출력영상 생성  (p.413) |

2.

|  |
| --- |
| 영상에서 주파수는 화소 밝기의 변화율이다. 따라서 화소 밝기의 변화율이 큰 부분은 고주파 공간 영역이고 작은 부분은 저주파 공간 영역이다.  저주파 공간 영역은 보통 영상의 배경부분이나 객체의 내부에 많이 있으며 고주파 공간 영역은 경계 부분이나 객체의 모서리 부분에 많이 있다.  (p.412) |

3.

|  |
| --- |
| 1. 1차원 이산 푸리에 변환을 가로, 세로 방향으로 연속해서 두 번 적용한다. (전치 이용)  2. 푸리에 변환으로 생성된 주파수 스펙트럼 영상에서 계수 값에 로그함수를 적용한 후 정규화한다.  3. 시프트 연산을 통해서 영상의 중심이 저주파, 바깥쪽 부분이 고주파 영역이 되도록 만든다.  (p.418~422) |

4.

|  |
| --- |
| 시프트(shift) : 영상을 사분면으로 나누어 원본행렬의 관심 영역을 반환행렬의 관심 영역으로 복사하여 사분면의 맞교환을 수행한다.    **DFT에서 시프트 연산이 필요한 이유**  DFT를 수행하고 나면 저주파 영역이 모서리 부분에 위치하고 고주파 영역이 중심부에 있게 된다. 이런 상태에서는 주파수 영역의 어떤 처리가 불편하기 때문에 시프트 연산을 사용한다.  (p.421) |

5.

|  |
| --- |
| 01 dst = cv2.magnitude(complex[:, :, 0], complex[:, :, 1])  02 dst = cv2.log(dst + 1)  03 cv2.normalize(dst, dst, 0, 255, cv2.NORM\_MINMAX)  04 return cv2.convertScaleAbs(dst)  푸리에 변환을 수행하면 복소수의 행렬이 결과로 생성된다. 이것을 영상으로 확인하기 위해서는 복소수의 실수부와 허수부를 벡터로 간주하여 벡터의 크기를 구하면 된다.  1. 01번 줄에서 magnitude 함수를 이용하여 실수부와 허수부의 크기를 따로 구한다.  2. 02번 줄에서 log함수를 적용한다.  3. 03번 줄에서 정규화한다.  4. 04번 줄에서 실수를 정수로 바꾸어 반환한다.  (p.420) |

6.

|  |
| --- |
| 삼각함수의 주기성을 이용해 반복계산을 줄여 속도를 향상시킨다.  원본신호를 짝수신호와 홀수신호로 연속적으로 분리하면 최종적으로 입력신호를 2개 원소 씩 묶을 수 있다.  (p.426~427) |

7.

|  |
| --- |
| 영 삽입 : 원본 영상의 가로와 세로 크기를 2의 자승이 되게 키우고, 빈공간을 검은색(0)으로 채우는 것  **FFT에서 영 삽입을 하는 이유**  원본 신호를 2개 원소를 가질 때까지 연속적인 분리를 하기 때문에 원본 신호의 길이는 2의 지수승이 되어야 한다. 하지만 영상의 크기는 항상 2의 자승이 아니라는 문제가 있어 영삽입을 통해 이 문제를 해결한다.  (p.427~428) |

8.

|  |
| --- |
| 버터플라이 : 이웃한 두 원소에 대해서 이산 푸리에 변환을 수행하는 것  버터플라이 과정은 원본 신호길이가 2개 원소를 가질 때까지 분리하며, 분리횟수만큼 연속적으로 반복한다. |

11.

|  |
| --- |
| import numpy as np, cv2  def calc\_spectrum(complex):  if complex.ndim==2:  dst = abs(complex)  else:  dst = cv2.magnitude(complex[:, :, 0], complex[:, :, 1])  dst = cv2.log(dst + 1)  cv2.normalize(dst, dst, 0, 255, cv2.NORM\_MINMAX)  return cv2.convertScaleAbs(dst)  def fftshift(img):  dst = np.zeros(img.shape, img.dtype)  h, w = dst.shape[:2]  cy, cx = h//2, w//2  dst[h-cy:, w-cx:] = np.copy(img[0:cy, 0:cx])  dst[0:cy, 0:cx] = np.copy(img[h-cy:, w-cx:])  dst[0:cy, w-cx:] = np.copy(img[h-cy:, 0:cx])  dst[h-cy:, 0:cx] = np.copy(img[0:cy, w-cx:])  return dst  image = cv2.imread("c:/computervision/chap09/image/1.jpg", cv2.IMREAD\_GRAYSCALE) if image is None : raise Exception("이미지 읽기 오류")  dft = cv2.dft(np.float32(image), flags=cv2.DFT\_COMPLEX\_OUTPUT) spectrum = calc\_spectrum(fftshift(dft)) idft = cv2.idft(dft, flags=cv2.DFT\_SCALE)[:,:,0]  cv2.imshow('spectrum', spectrum) cv2.waitKey() |

12.

|  |
| --- |
| import numpy as np, cv2, math  def zeropadding(img):  h, w = img.shape[:2]  m = 1 << int(np.ceil(np.log2(h)))  n = 1 << int(np.ceil(np.log2(w)))  dst = np.zeros((m ,n), img.dtype)  dst[0:h, 0:w] = img[:]  return dst  def calc\_spectrum(complex):  if complex.ndim==2:  dst = abs(complex)  else:  dst = cv2.magnitude(complex[:, :, 0], complex[:, :, 1])  dst = cv2.log(dst + 1)  cv2.normalize(dst, dst, 0, 255, cv2.NORM\_MINMAX)  return cv2.convertScaleAbs(dst)  def fftshift(img):  dst = np.zeros(img.shape, img.dtype)  h, w = dst.shape[:2]  cy, cx = h//2, w//2  dst[h-cy:, w-cx:] = np.copy(img[0:cy, 0:cx])  dst[0:cy, 0:cx] = np.copy(img[h-cy:, w-cx:])  dst[0:cy, w-cx:] = np.copy(img[h-cy:, 0:cx])  dst[h-cy:, 0:cx] = np.copy(img[0:cy, w-cx:])  return dst  def dft(g):  N = len(g)  dst = [sum(g[n] \* math.exp(k\*n/N) for n in range(N)) for k in range(N)]  return np.array(dst)  def idft(g):  N = len(g)  dst = [sum(g[n] \* math.exp(-k\*n/N) for n in range(N)) for k in range(N)]  return np.array(dst) / N  def butterfly(pair, L, N, dir):  for k in range(L):  Geven, Godd = pair[k], pair[k+L]  pair[k] = Geven + Godd \* math.exp(dir \* k /N)  pair[k + L] = Geven - Godd \* math.exp(dir \* k /N)  def parring(g, N, dir, start=0, stride=1):  if N==1: return [g[start]]  L = N // 2  sd = stride \* 2  part1 = parring(g, L, dir, start, sd)  part2 = parring(g, L, dir, start + stride, sd)  pair = part1 + part2  butterfly(pair, L, N, dir)  return pair  def fft(g):  return parring(g, len(g), 1)  def ifft(g):  fft = parring(g, len(g), -1)  return [v / len(g) for v in fft]  def fft2(image):  pad\_img = zeropadding(image)  tmp = [fft(row) for row in pad\_img]  dst = [fft(row) for row in np.transpose(tmp)]  return np.transpose(dst)  def ifft2(image):  tmp = [ifft(row) for row in image]  dst = [ifft(row) for row in np.transpose(tmp)]  return np.transpose(dst)  image = cv2.imread("c:/computervision/chap09/image/1.jpg", cv2.IMREAD\_GRAYSCALE) if image is None : raise Exception("이미지 읽기 오류") image = image[200:400, 200:400]  dft = fft2(image) idft = ifft2(dft).real  h, w = image.shape[:2] dst = idft[:h, :w]  cv2.imshow('1', cv2.convertScaleAbs(idft)) cv2.imshow('2', cv2.convertScaleAbs(dst)) cv2.waitKey() |