OPENCV_03

음성 신호 인식 학습 및 변환영역 처리

ICT , DE&DS : Y-A, KIM

음성 신호 인식 학습 및 변환영역 처리 수행 단계

데이터 수집 및 전 처리

- 잡음 제거, 신호 강화 등의 전처리 수행

특징 추출

- 음성 특징 추출
- 변환 영역 처리

데이터셋 분리:

- 훈련, 검증, 테스트 세트로 데이터를 분리

모델 설계 및 학습:

- CNN, RNN, LSTM, GRU 등의 모델을 설계

평가 및 최적화:

- 검증 및 테스트 데이터셋으로 모델의 성능 평가
- 하이퍼파라미터 튜닝, 모델 구조 변경 등을 통한 최적화 수행

실시간 처리 및 응용

- 실시간 음성 인식이 필요한 경우, 슬라이딩 윈도우나 실시간 데이터 스트리밍 방법 적용
- 다양한 응용 분야에 모델 배포 (예: 음성 비서, 자동 번역 등)

지속적인 모니터링 및 업데이트

- 새로운 데이터나 변경된 환경에 따라 모델을 주기적으로 업데이트하거나 재학습

음성 신호 인식 및 변환 영역 처리 특징추출

음성 특징 추출 방법

- MFCC (Mel-Frequency Cepstral Coefficients), Spectrogram, Chromagram 등

변환 영역 처리 방법

- FFT (Fast Fourier Transform) , Wavelet 변환 활용 후 시간 도메인의 신호를 주파수 도메인으로 변환

특징추출

MFCC (Mel-Frequency Cepstral Coefficients)

- 음성 인식에서 널리 사용되는 되며 사람의 청각 시스템이 모든 주파수를 동일하게 인식하지 않는 것을 기반한다.
- Mel 스케일은 사람의 청각 반응을 모방하는 주파수 스케일입니다.
- MFCC는 사운드의 전반적인 모양을 포착하며, 특히 사람의 음성을 구별하는 데 매우 효과적입니다.

Spectrogram

- 시간에 따른 신호의 주파수 컨텐츠를 시각적으로 표현한 것입니다.
- X축은 시간, Y축은 주파수, 각 포인트의 색상 또는 밝기는 해당 시간 및 주파수에서의 신호의 강도를 나타냅니다.
- FFT를 사용하여 일정한 간격으로 시간 도메인 신호를 주파수 도메인으로 변환하여 생성됩니다.

Chromagram

- 음악 신호에서 12개의 다른 음절 (C, C#, D, D#,..., B)의 에너지나 활동 레벨을 나타내는 시각적 표현입니다.
- 음악 인식, 특히 코드 및 코드 진행의 인식에 유용합니다.

FFT (Fast Fourier Transform)

- 시간 도메인의 신호를 주파수 도메인으로 변환하는 알고리즘입니다.
- 신호의 구성 성분을 분석하는 데 사용되며, 여러 응용 분야에서 주파수 분석이 필요할 때 사용됩니다.

Wavelet Transform

- 신호를 서로 다른 주파수 구성 요소로 분해하는 시간-주파수 변환 방법입니다.
- FFT와는 달리, Wavelet 변환은 신호의 고주파 및 저주파 성분에 대한 로컬 정보를 제공하므로 시간 및 주파수의 변화를 동시에 캡처할 수 있습니다.

음성 신호를 활용한 모듈 생성

1. 기초 모듈	**Waveform 그리기**: 오디오 파일에서 waveform을 그리는 기본적인 예제
	Spectrogram 생성: FFT(Fast Fourier Transform)나 STFT(Short-Time Fourier Transform)를 사용하여 spectrogram을 그리기
2. 특성 추출 모듈	**MFCC 계산**: Mel-frequency cepstral coefficients를 계산하고 그래프로 표시
	Chroma features 계산: Chroma 특성을 추출하여 그래프로 표시
3. 음악 정보 검색	**템포 추정**: 오디오에서 템포(beat)를 추정
	음고 분석: 기본 주파수(F0)나 피치를 추출
4. 분류 문제 모듈	**장르 분류**: 여러 음악 장르에 대해 MFCC나 다른 특성을 사용하여 분류 모델 생성
	음성/음악 구분: 음성과 음악을 구분하는 분류 모델 생성
5. 고급 분석 모듈	**음성 인식**: 오디오에서 특정 단어나 구절을 인식
	감정 분석: 음성에서 감정을 분석 (ex: 행복, 슬픔, 분노 등)
6. 시계열 데이터 분석	**Onset Detection**: 음악이나 음성에서 onset(시작 부분)을 감지
	Beat Tracking: 비트를 추적하고 그 위치를 표시
7. 응용 프로그램	**실시간 오디오 처리**: 실시간으로 오디오 스트림을 받아 특성을 추출하거나 분석
	음악 추천 시스템: 사용자의 음악 취향을 분석하여 추천

음악 정보 검색 (Music Information Retrieval, MIR) 온라인 리소스

1. PyDub	다양한 오디오 형식을 지원하며, 오디오 파일의 편집이나 변환에 유용 PyDub GitHub Repository
2. Music21	음악 이론, 음악 분석, 그리고 악보 생성에 사용 Music21 Website
3. FluidSynth	MIDI 이벤트를 실시간 오디오로 렌더링 FluidSynth Website
4. pretty_midi	MIDI 파일을 읽고 쓰고, 분석 사용 pretty_midi GitHub Repository
5. Essentia	매우 광범위한 오디오 분석 기능을 제공 Essentia Website
6. Madmom	음악 정보 검색과 오디오 신호 처리에 특화된 라이브러리, 리듬 분석에 강점 Madmom GitHub Repository
7. TensorFlow Audio	TensorFlow를 활용하여 복잡한 오디오 분석과 모델링을 수행 TensorFlow Audio GitHub Repository
8. scipy.io.wavfile	SciPy 라이브러리 내에 있는 wav 파일 입출력 모듈 SciPy IO Documentation

Librosa - Python으로 음성 처리 , 음악 분석

- librosa의 환경 구축
  - pip install librosa

- Anaconda에 설치
  - conda install -c conda-forge librosa

Librosa 모듈

librosa는 음악과 오디오를 처리하고 분석하는 Python 패키지

- **librosa 모듈 목록**
  - librosa : ibrosa의 핵심
  - librosa.beat : 템포, 비트를 검출하는 기능
  - librosa.decompose : 고조파 타악기음 분리 및 스펙트로그램의 분해에 관한 기능
  - librosa.display: 음성 데이터의 시각화하는 기능와 같은 음성 처리 기능
  - librosa.feature : 음성의 특징 추출 기능
  - librosa.filters : 필터 뱅크를 생성하는 기능
  - librosa.onset : 온셋 검출과 온셋 강도 계산에 관한 기능
  - librosa.segment : 재귀 행렬 구축, 순차 제약 클러스터링
  - 문서: https://librosa.org/doc/latest/index.html

Librosa_ 주요메소드

librosa.load() 오디오 파일을 읽어와서 오디오 시간 계열 데이터로 변환 , 입력 오디오 파일의 경로 ,

출력은 오디오 데이터 샘플링 레이트

librosa.display.waveshow() 오디오 신호를 표시

librosa.stft() 주어진 오디오에 대해 단기 푸리에 변환 (STFT)을 계산

librosa.amplitude_to_db() 아날로그 신호의 진폭을 dB로 변환

librosa.feature.mfcc() 오디오의 MFCC (Mel-Frequency Cepstral Coefficients)를 계산

librosa.feature.chroma_stft() 오디오의 크로마 단기 푸리에 변환을 계산

librosa.feature.spectral_contrast() 오디오의 스펙트럼 대비 계산

librosa.feature.melspectrogram() 오디오에 대한 멜 스펙트로그램 계산

librosa.effects.pitch_shift() 오디오의 피치를 변경

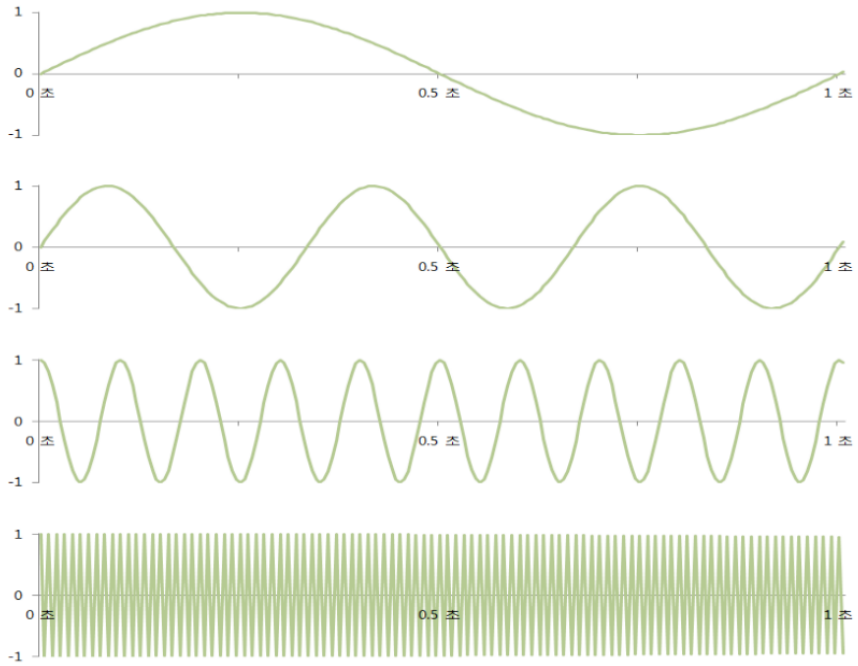
librosa.effects.time_stretch() 오디오의 재생 속도를 변경하지 않고 길이를 조절

librosa.onset.onset_detect() **오디오** 디오에서 발견되는 충격 웨이브 또는 공격 포인트를 감지

librosa.beat.beat_track() 오디오에서 리듬 또는 비트를 추적

샘플링 — 표본빈 필수도 =

- 헤르츠(Hz)
  - 주파수를 표현하는 단위, 1초 동안에 진동하는 횟수
  - 전파라는 신호에 국한된 표현



1초에 한번 진동 = 1Hz

1초에 세번 진동 = 3Hz

1초에 열번 진동 = 10Hz

1초에 백번 진동 = 100Hz

샘플 레이트 (Sample Rate)와 비트 뎁스 (Bit Depth)

샘플 레이트 (Sample Rate)

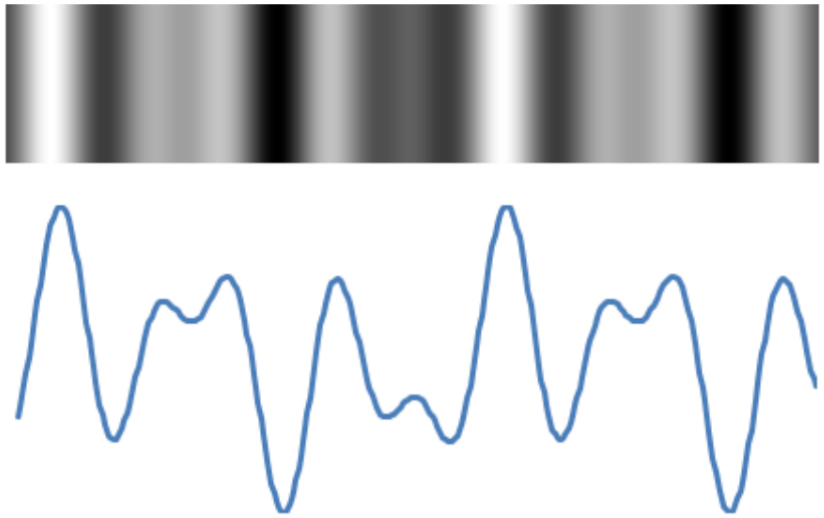
- 샘플 레이트는 초당 샘플링된 횟수
- 디지털 오디오에서는 소리의 아날로그 신호를 디지털로 변환하기 위해 얼마나 자주 샘플을 추출하는지를 나타냄
- 단위는 초당 샘플 수 (samples per second)로, 헤르츠(Hz)로 표현(44.1kHz는 초당 44,100번 샘플링 한다는 것을 의미)
- 일반적으로 CD 품질의 오디오는 44.1kHz의 샘플 레이트를 사용하며, 전문가 수준의 오디오는 48kHz, 96kHz 또는 그 이상의 높은 샘플 레이트를 사용함

비트 뎁스 (Bit Depth)

- 각 샘플의 정보량을 나타내며, 오디오의 동적 범위(가장 작은 소리와 가장 큰 소리 사이의 차이)를 결정합니다.
- 각 샘플에 할당된 비트 수로 표현되며 16비트 오디오는 각 샘플마다 16비트의 데이터를 사용하는 것을 의미
- 일반적으로 CD 품질의 오디오는 16비트 비트 뎁스를 사용, 전문가 수준의 오디오는 24비트 또는 그 이상의 비트 뎁스를 사용함

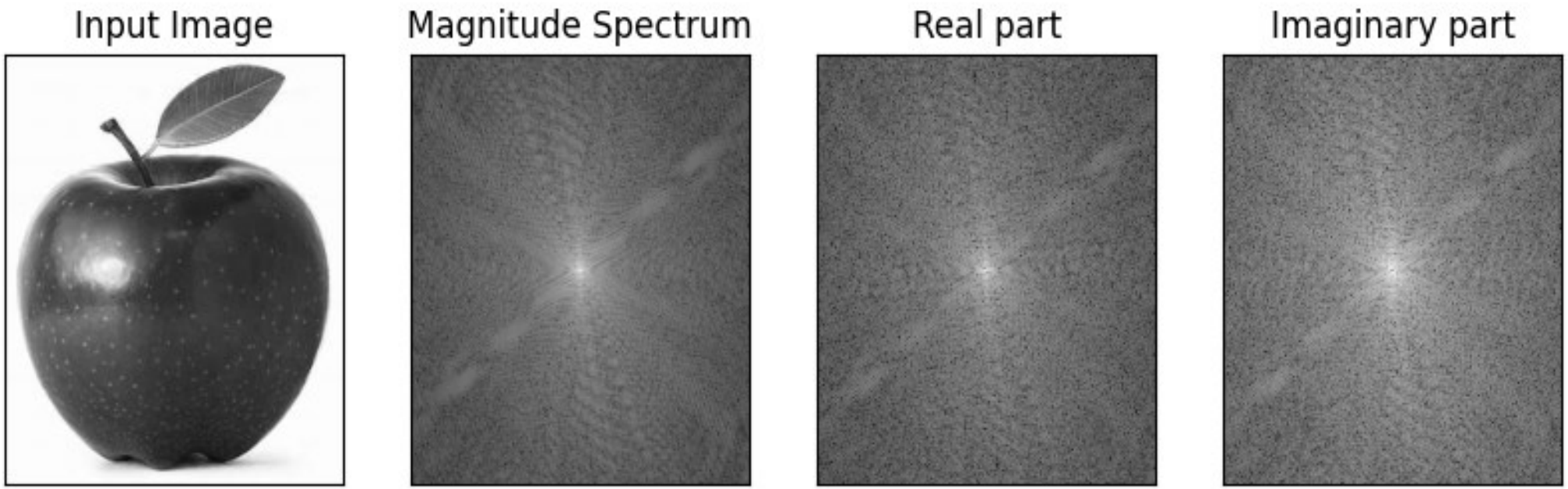
- 영상처리에서는 공간 주파수(spatial frequency) 개념 사용

- 확장된 의미에서 주파수
  - 이벤트가 주기적으로 재 발생하는 빈도
    - 예) 화소 밝기의 변화도



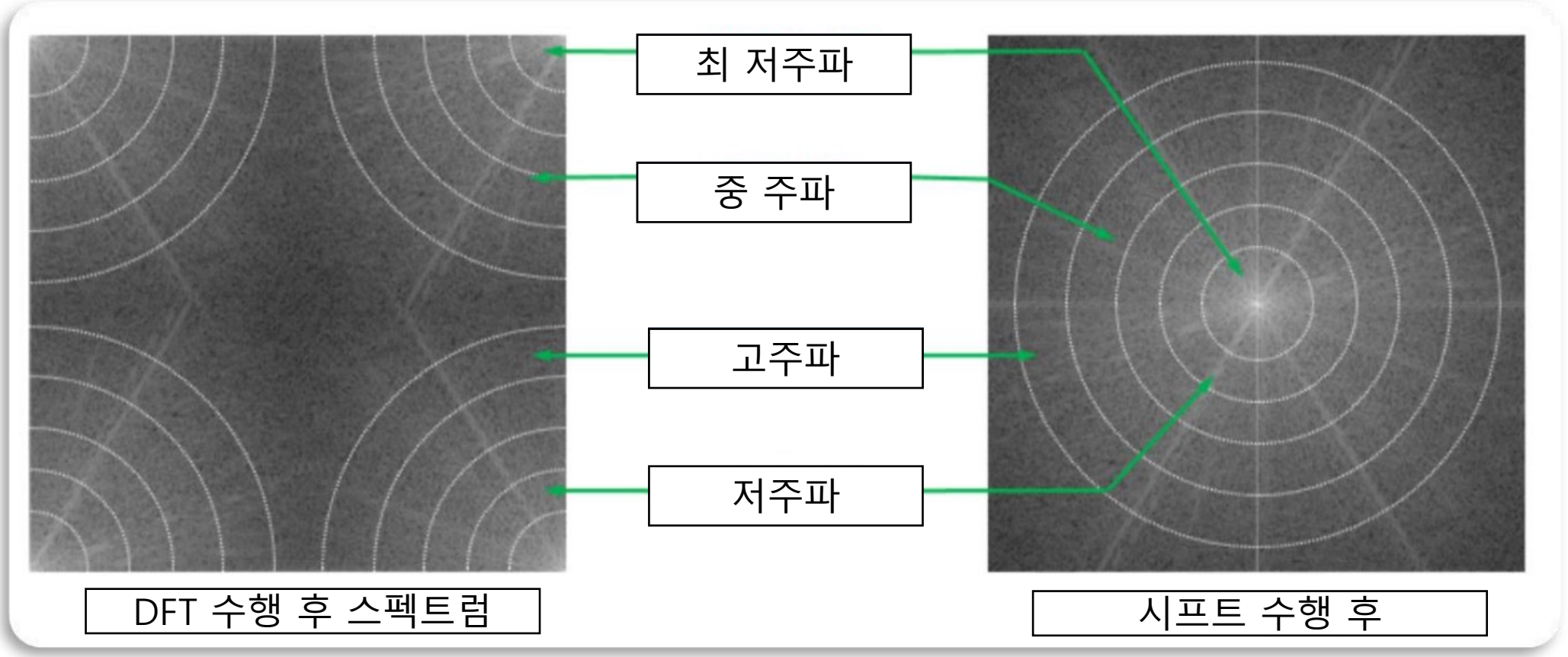
- 공간 주파수 - 밝기의 변화 정도에 따라서 고주파 영역/ 저주파 영역으로 분류
  - 저주파 공간 영역
    - 화소 밝기가 거의 변화가 없거나 점진적으로 변화하는 것
    - 영상에서 배경 부분이나 객체의 내부에 많이 있음
  - 고주파 공간 영역
    - 화소 밝기가 급변하는 것
    - 경계부분이나 객체의 모서리 부분

- 주파수 스펙트럼 영상
  - 저주파 영역의 계수값이 고주파 영역에 비해 상대적으로 너무 큼
  - 계수값을 정규화해서 영상으로 표현하면 최저주파 영역만 흰색으로 나타나고 나머지 영역은 거의 검은색으로 나타남 → 계수값에 로그함수 적용



[로그 함수 후 정규화 / 주파수 분리]

- 주파수 스펙트럼 영상
  - 저주파 영역이 모서리 부분에 위치, 고주파 부분이 중심부에 위치
  - 사각형의 각 모서리를 중심으로 원형의 밴드를 형성하여 주파수 영역 분포
    - 해당 주파수 영역 처리시 불편함 → 모양 변경 필요



고속 푸리에 변환

- 이산 푸리에 변환
  - 입력 신호의 한 원소에 곱해지는 기저 함수의 원소들을 원소 길이만큼 반복적으로 곱해짐
    - 때문에 신호가 커질수록 계산 속도는 기하급수적으로 증가

- 고속 푸리에 변환
  - 삼각함수의 주기성 이용해 작은 단위 분리
  - 반복적 수행하고 합해져서 효율성 높이는 방법

고속 푸리에 변환

- 푸리에 변환 수식 – 짝수부 / 홀수부 분리 가능

$$G(k)=\sum_{n=0}^{L-1} g[2n]\cdot e^{-j2\pi k\frac{2n}{2L}}+\sum_{n=0}^{L-1} g[2n+1]\cdot e^{-j2\pi k\frac{2n+1}{2L}}$$

$$G(k)=\left\{\sum_{n=0}^{L-1} g[2n]\cdot e^{-j2\pi k\frac{n}{L}}\right\}+\left\{\sum_{n=0}^{L-1} g[2n+1]\cdot e^{-j2\pi k\frac{n}{L}}\right\}\cdot e^{-j2\pi k\frac{1}{2L}}$$

  - 짝수부 변환 수식

$$G_{even}(k)=\sum_{n=0}^{L-1} g[2n]\cdot e^{-j2\pi k\frac{n}{L}}$$

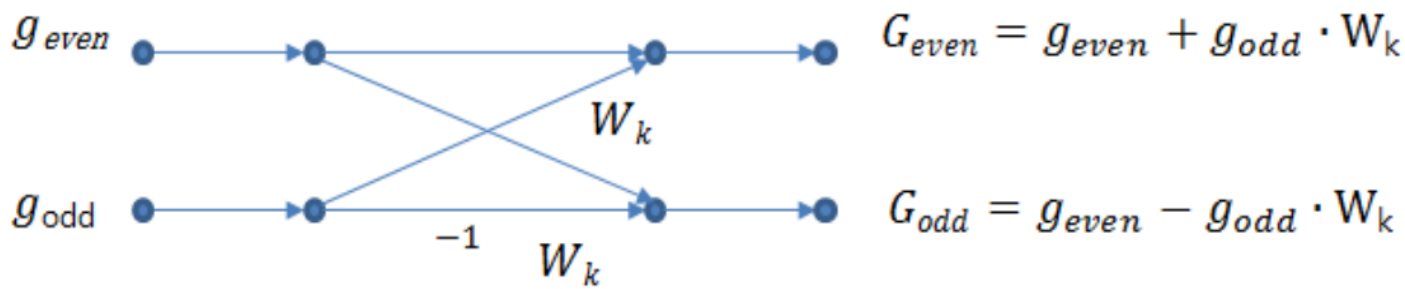
  - 홀수부 변환 수식

$$G_{odd}(k)=\sum_{n=0}^{L-1} g[2n+1]\cdot e^{-j2\pi k\frac{n}{L}}$$

- 수식의 정리

$$G(k)=G_{even}(k)+G_{odd}(k)\cdot e^{-j2\pi k\frac{1}{2L}}$$

- 버터플라이(butterfly) 과정
  - 스크램블 결과 원소에서 이웃한 두 원소에 대해서 이산 푸리에 변환 수행
    - W_k : 푸리에 변환의 기저함수



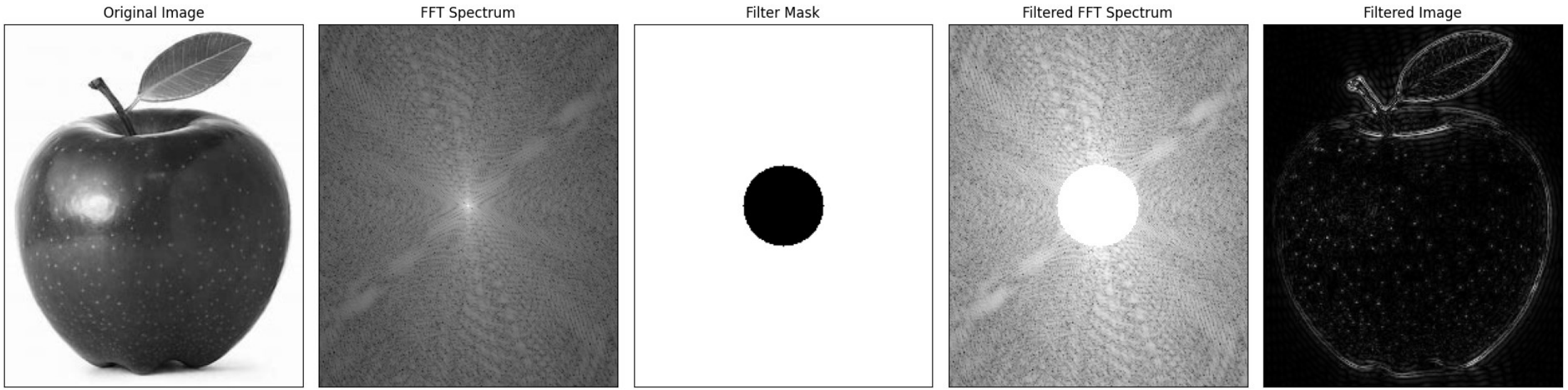
- 원본 신호를 연속적으로 짝수부와 홀수부로 분리 → 원본 신호의 원소개수는 2^n의 자승

FFT를 이용한 주파수 영역 필터링

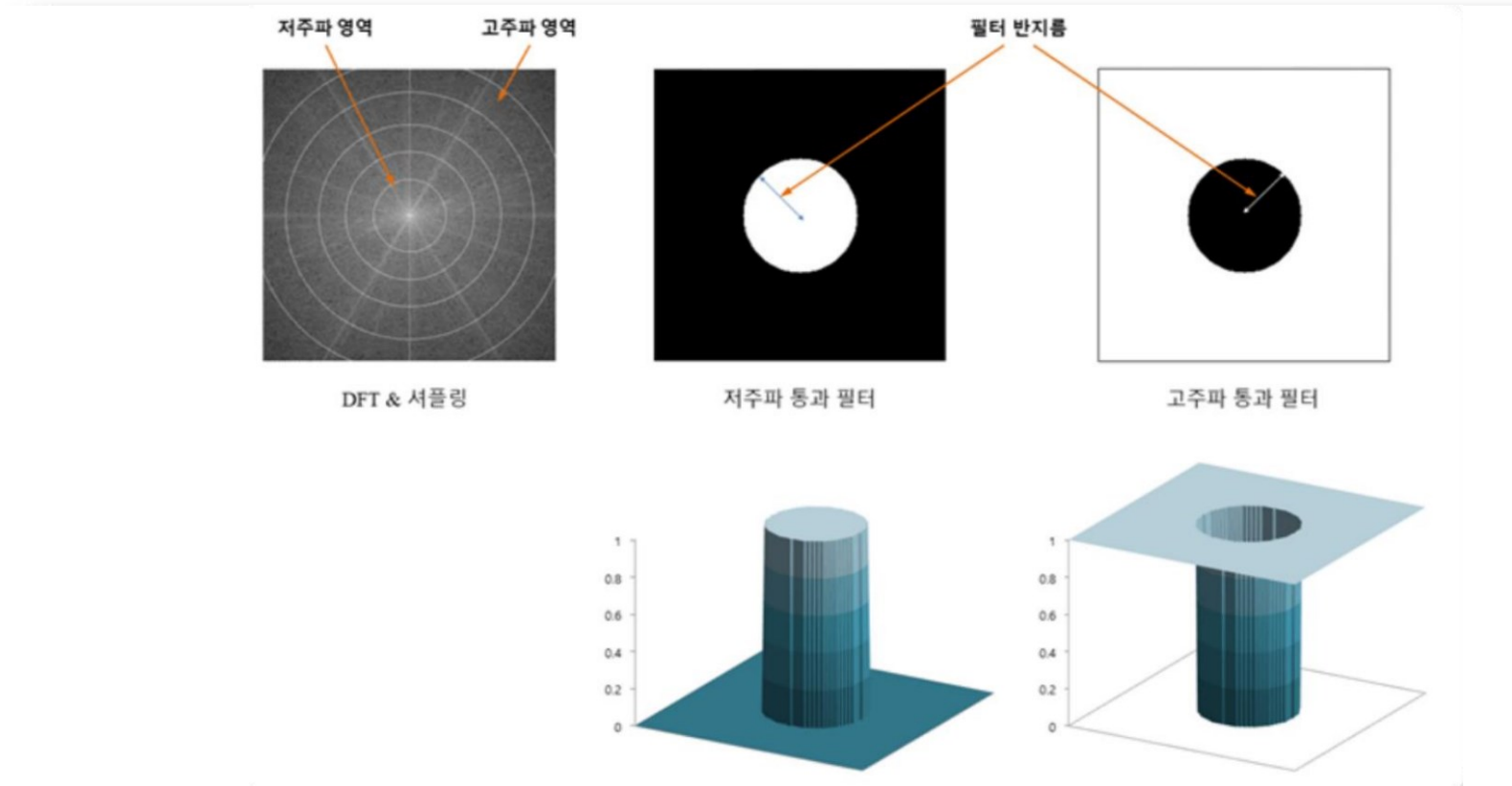
- 주파수 영역 필터링의 과정
- 저주파 및 고주파 통과 필터링
- 버터워스, 가우시안 필터링

주파수 영역 필터링

- 영상의 주파수 영역 변환
  - 저주파 영역과 고주파 영역 분리 → 특정 주파수 영역 강화, 약화, 제거 가능
- 주파수 영역 필터링 과정



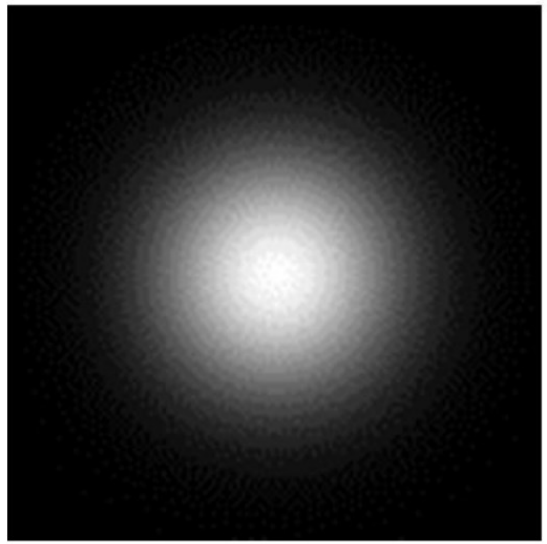
- 저주파 통과 필터링
  - DFT 변환 영역에서 저주파 영역의 계수들은 통과, 고주파 영역의 계수 차단
- 고주파 통과 필터링
  - 고주파 영역의 계수들을 통과시키고 저주파 영역의 계수들 차단



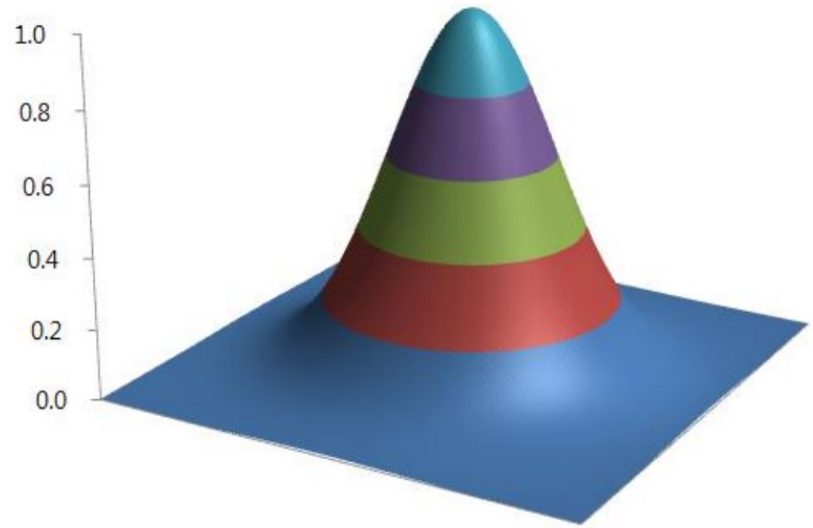
- 대역 통과 필터
  - 특정한 대역에서 급격하게 값을 제거하기 때문에 결과 영상의 화질 저하
  - 객체의 경계부분 주위로 잔물결 같은 무늬(ringing pattern) 나타남
- 해결방법
  - 필터 원소값을 차단 주파수에서 급격하게 0으로하지 않고 완만한 경사 이루도록 구성
  - 버터워즈 필터(Butterworth filter)나 가우시안 필터(Gaussian filter)

- 가우시안 필터
  - 필터 원소의 구성을 가우시안 함수의 수식 분포를 갖게 함으로써 차단 주파수 부분을 점진적으로 구성한 것

$$f(x, y) = exp\left(-\frac{dx^2 + dy^2}{2R^2}\right), \qquad \begin{matrix} dx = x - center.x \\ dy = y - center.y \\ R : \text{주파수 차단 반지름} \end{matrix}$$



필터 계수를 밝기로 표현



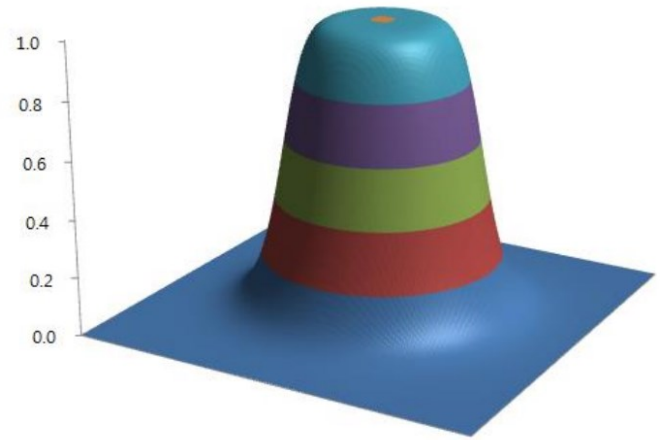
필터 계수를 3차원 표현

- 버터워즈 필터
  - 차단 주파수 반지름 위치(R)와 지수의 승수인 n 값에 따라서 차단 필터의 반지름과 포물선의 곡률 결정

$$f(x, y) = -\frac{1}{1+\left(\frac{\sqrt{dx^2+dy^2}}{R}\right)^{2n}},$$

$$dx = x - center.x$$
$$dy = y - center.y$$
R : 주파수 차단 반지름

필터 계수를 밝기로 표현



필터 계수를 3차원 표현