Suvi Liimatainen 6.7.2025

Määrittelydokumentti - Audiosignaalin prosessointi

Opinto-ohjelma

Tietojenkäsittelytieteen kandidaatti (TKT)

Dokumentaation kieli

Koodi ja sen kommentit ovat englanniksi, muu kurssiin liittyvä vaadittava dokumentaatio on suomeksi.

Ratkaistava ongelma

Projektin aikana toteutan Audiosignaalin prosessointia varten ohjelmiston, joka sisältää alipäästö- ja ylipäästösuodattimet. Näiden suodattimien avulla voi suodattaa musiikista esiin vain joko matalat- tai korkeat säveltaajuudet, kuten elektronista musiikkia soittava DJ saattaa tehdä, kun hän yhdistelee eri kappaleita toisiinsa.

Projektin aikana tekemäni ohjelmisto sisältää pienen valikon, josta voi lisätä ohjelmaan audiotiedoston, johon ohjelma asettaa halutun alipäästö- tai ylipäästösuodattimen. Analysoitavan kappaleen tiedoston käsittelyn jälkeen ohjelmisto ilmoittaa olevansa valmis, jolloin käyttäjä voi kuunnella suodatetun äänen painamalla “play”. Mikäli minulla on projektin aikana mahdollisuus, haluaisin myös tehdä visualisoinnin siitä, miltä lopputulos näyttää. Käyttäjä voisi esimerkiksi nähdä, että jäljelle jäävä musiikki koostuu vain korkeista sävelistä ja matalat jäävät pois.

Projektissa apuna käytän NumPya, Matplotlibyä ja PyAudiota.

Testauksessa käytetään Pytestiä ja riippuvuuksien hallintaan käytetään Poetryä.

Projektin ydin

Projektin ydin on kirjoittaa oma versioni nopeasta Fourier-muunnos –algoritmista (FFT) ja käyttää sitä näiden suodattimien valmistamiseen. Aion perehtyä Cooley-Tukey-algoritmin toimintaan, joka on yksi tapa toteuttaa FFT.

Tavoitteena olevat aika- ja tilavaativuudet

Lähestyn tavoitteena olevia aika- ja tilavaatimuksia siitä näkökulmasta, että Diskreetti Fourier-muunnos, joka on idealtaan yksinkertaisempi, on aikavaativuudeltaan $O(N^2)$, sillä se koostuu pohjimmiltaan kaikkien signaalinäytteiden sinien ja kosineiden summista.

Cooley-Tukey algoritmi taas perustuu DFT:hen, mutta pilkkoo alkuperäisen signaalin rekursiivisesti pienempiin osiin ja yhdistää tulokset tehokkaasti hyödyntämällä näytteiden indeksien jakoa parillisiin ja parittomiin ja sitten hyödyntämällä kompleksieksponenttien molemminpuolista symmetriaa. Tämä on tehokkaampi tapa tehdä tarvittavat laskutoimitukset, joten aikavaatimus on (N log N).

Cooley-Tukeyn algoritmin tilavaatimuksena voidaan pitää O(N), sillä vaikka kaikki käsiteltävät näytteet ja näiden tulokset on pidettävä muistissa, toimii algoritmi rekursiivisesti eikä niitä tarvitse pitää muistissa yhtä aikaa. Kukin rekursiokerros käsittelee aina myös vain puolta edellisestä datasta

Käytettävät ohjelmointikielet

Koodaan projektin pääosin Python-ohjelmointikielellä. Saatan tehdä visuaalista puolta myös Javascriptillä, mutta tämä ei tule sisältämään monimutkaista logiikkaa.

Oma koodausosaaminen painottuu tällä hetkellä fronttipuoleen: Voin arvioida pythonin lisäksi valitettavasti lähinnä Javascriptiä ja Typescriptiä.

Tarvittavat tietorakenteet

Luultavasti Python-kielessä jo olemassa olevat tietorakenteet (kuten jono, sanakirja, pino, joukko jne. Ovat riittäviä. Tässä vaiheessa en usko, että joudun kirjoittamaan uusia tietorakenteita.

Lähteet

Tässä joitakin lähteitä, joita käytän apuna projektissa:

[1] James Cooley ja John Tukey. “An algorithm for the Machine Calculation of Complex Fourier Series”. *Mathematics of Computation* 19.90 (1965), s.297-301.

[2] Udo Zoelzer. Dafx: Digital Audio Effects. USA: John Wiley & Sons, Inc., 2002. ISBN: 0471490784.

[3] Yliluoma, Joel. *“Nopea Fourier-muunnos – Teoria ja toteutus modernilla C++:lla”.* 1/2024. Pro gradu. Helsingin yliopisto. URL:

http://hdl.handle.net/10138/573505

Videolähteet:

Xu, Simon. “Discrete Fourier Transform - Simple Step by Step”. Vierailtu 12.07.2025.
URL:*https://www.youtube.com/watch?v=mkGsMWi_j4Q&list=PLvjz0R0HYFYfJHlNKoD_AWR-CtAK89wQ7&index=1*

Xu, Simon. “The FFT Algorithm - Simple Step by Step”. Vierailtu 12.07.2025.
URL:
https://www.youtube.com/watch?v=htCj9exbGo0&list=PLvjz0R0HYFYfJHlNKoD_AWR-CtAK89wQ7&index=4