5. naloga: Hitra Fourierova transformacija in korelacijske funkcije

Gregor Žunič

Diskretno Fourierovo transformacijo smo definirali kot

$$H_k = \sum_{n=0}^{N-1} h_n \exp(2\pi \mathrm{i} kn/N), \qquad k = -\tfrac{N}{2}, \ldots, \tfrac{N}{2},$$

oziroma

$$H_k = \sum_{n=0}^{N-1} W_N^{nk} h_n, \qquad W_N = \exp(2\pi \mathrm{i}/N).$$

Ta postopek ima očitno časovno zahtevnost N^2. Račun pa je mogoče izvesti tudi z bistveno manj operacijami. Osnovni premislek je razcep

$$H_k = H_k^{\mathrm{sod}} + W_N^k H_k^{\mathrm{lih}} \,,$$

kjer smo transformiranko H izrazili s transformacijama njenih sodih in lihih členov, pri čemer je vsota vsake od transformacij zdaj dolžine N/2. Gornjo relacijo lahko uporabljamo rekurzivno: če je N enak potenci števila 2, lahko rekurzijo razdrobimo do nizov, ki imajo samo še en člen. Zanj je transformacija identiteta. Za obrat pri eni vrednosti frekvence (pri danem m) je potrebno na vsakem koraku rekurzije le eno množenje s potenco W, korakov pa je $\log_2 N$. Skupna časovna zahtevnost je torej le še $N \log_2 N$.

Da ne iščemo pripadnikov niza po vsej tabeli, si podatke preuredimo. Lahko je pokazati, da je v prvotni tabeli treba med seboj zamenjati podatke, katerih vrstna števila v binarnem zapisu so obrnjena: v novem redu jemljemo člene kar po vrsti. Tudi potenc W ne izražamo vedno znova s sinusi in kosinusi, pač pa jih računamo z rekurzijo. Tak ali podoben postopek je osnova vseh algoritmov hitre Fourierove transformacije (FFT).

Z neko transformacijo iz družine FFT bomo izračunali korelacijsko funkcijo dveh signalov. Korelacija periodičnih funkcij $g(t)$ in $h(t)$ s periodo T je definirana kot:

$$\phi_{gh}(\tau) = \frac{1}{T} \int_0^T g(t+\tau)\, h(t)\, \mathrm{d}t \,,$$

oziroma diskretno

$$\phi_{gh}(n) = \frac{1}{N} \sum_{k=0}^{N-1} g_{k+n}\, h_k \,.$$

Računamo torej skalarni produkt funkcij, ki sta časovno premaknjeni za τ oziroma n. Če je za določeno vrednost premika ta funkcija višja kot v okolici, potem to pomeni, da sta si funkciji podobni, le da ju je treba premakniti, da se to vidi.

V primeru, da sta funkciji (signala), ki ju primerjamo, enaki, računamo njuno *avtokorelacijsko funkcijo*: ta je mera za to, ali signal ostaja s pretekanjem časa sam sebi podoben. Če je signal slabo koreliran (sam s sabo), korelacija $\phi_{hh}(n)$ relaksira h kvadratu povprečnega signala $\langle h \rangle^2$, kjer je

$$\langle h \rangle = \frac{1}{N} \sum_{k=0}^{N-1} h_k \,.$$

Iz lokalnih maksimov v avtokorelacijski funkciji sklepamo na periodičnosti, bodisi popolne ali približne. Pri periodičnih signalih je tudi avtokorelacijska funkcija striktno periodična, za stohastične procese pa je značilna eksponentna avtokorelacijska funkcija. še bolj nas zanima, kako *hitro* se korelacija izgublja: računamo rajši reskalirano obliko avtokorelacije

$$\widetilde{\phi}_{hh}(n) = \frac{\phi_{hh}(n) - \langle h \rangle^2}{\phi_{hh}(0) - \langle h \rangle^2} \, ,$$

kjer je imenovalec nekakšno merilo za varianco signala,

$$\sigma^2 = \phi_{hh}(0) - \langle h \rangle^2 = \frac{1}{N} \sum_{k=0}^{N-1} (h_k - \langle h \rangle)^2 \, .$$

Pri zgornjih enačbah moramo še "peš" poskrbeti za periodično zaključenost signala pri $n = N$, torej da je perioda enaka velikosti vzorca. Če tega ne moremo narediti, je bolj pravilna definicija avtokorelacije

$$\phi_{hh}(n) = \frac{1}{N-n} \sum_{k=0}^{N-n-1} h_{k+n} \, h_k \, .$$

Praktičen račun po zgornji formuli lahko postane za velike vzorce prezamuden. Avtokorelacijo rajši računamo s FFT (DFT) $\mathcal{F}$, saj je korelacija obratna Fourierova transformacija $\mathcal{F}^{-1}$ produkta Fourierovih transformacij $\mathcal{F}$, torej z $G = \mathcal{F}g$ in $H = \mathcal{F}h$ dobimo

$$\phi_{gh}(n) = \frac{1}{N-n} \mathcal{F}^{-1} \left[G \cdot (H)^* \right]$$

oziroma

$$\phi_{hh}(n) = \frac{1}{N-n} \mathcal{F}^{-1} \left[\, |H|^2 \, \right] \, .$$

Za račun s FTT signale dolžine N najprej prepišemo v dvakrat daljše, periodično zaključene podatkovne nize, $\widetilde{h}_n = h_n$, $\widetilde{h}_{n+N} = 0$ za $n = 0, \ldots, N-1$ in $\widetilde{h}_{n+2N} = \widetilde{h}_n$. Tedaj se avtokorelacija zapiše v obliki

$$\phi_{hh}(n) = \frac{1}{N-n} \sum_{k=0}^{2N-1} \widetilde{h}_{k+n} \, \widetilde{h}_k \, ,$$

kar lahko izračunamo s FFT.

Naloga: Na spletni strani MF praktikuma najdeš posnetke oglašanja velike uharice, naše največje sove. Posneti sta dve sovi z minimalnim ozadjem (`bubomono` in `bubo2mono`) in nekaj mešanih signalov, ki zakrivajo njuno oglašanje (`mix`, `mix1`, `mix2` in `mix22`). V signalih `mix2` in `mix22` je oglašanje sove komaj še zaznavno. Izračunaj avtokorelacijsko funkcijo vseh signalov in poskusi ugotoviti, za katero sovo gre pri teh najbolj zašumljenih signalih!

Priporočamo uporabo rutine `four1` iz Numerical Recipes ali knjižnice `fftw3`, ki je še dosti hitrejša. V okolju Python so te rutine vključene v 'fft' paket. (Pri tako velikih vzorcih je skorajda nujno uporabiti FFT namesto počasne navadne DFT.)

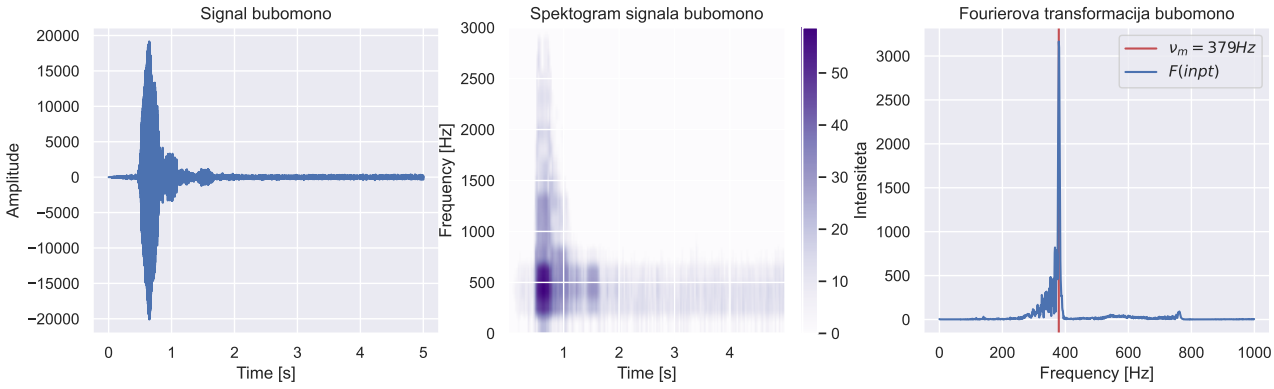
Dodatna naloga: Izračunaj še avtokorelacijsko funkcijo za kak signal, ki ga posnameš sam ali za kak proces, za katerega sam poiščeš ustrezne podatke.

1 Reševanje

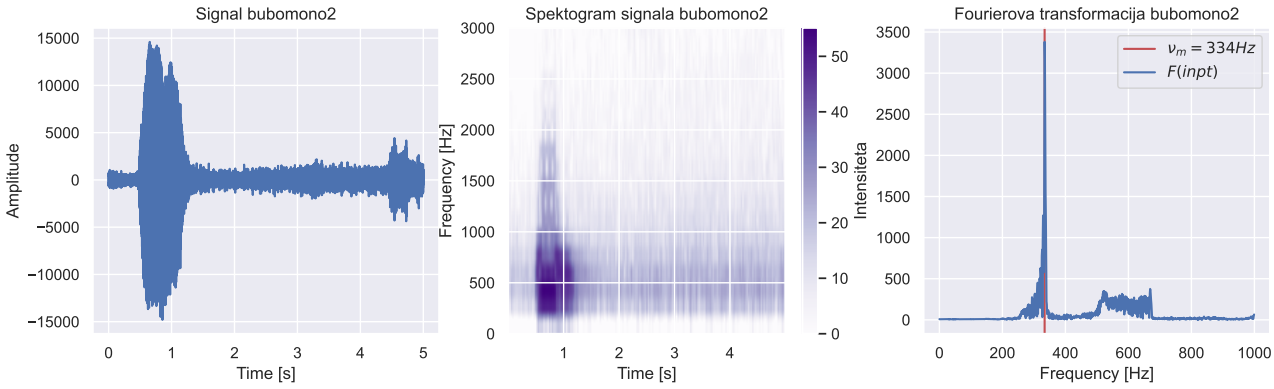
Ugotoviti moramo kateri posnetek pripada kateri sovi. Najlažji način, da rešimo ta problem je, da bi poslušali posnetke (metoda ostrega poslušanja), vendar to mogoče ni najbolj primerno za nivo 3. letnika fakultete. Šala na stran, najlažje se je naloge zares lotiti malce bolj primitivno. Kar lahko naredimo je, da za vsak vhodni signal narišemo Fourierovo transformacijo in pogledamo, kje je maksimalna frekvenca. To deluje zelo dobro, ker je frekvenca sove konstantna in na koncu prevlada. Torej lahko z zelo malo dela opazimo, katera sova je katera.

Ko vsako `mp3` datoteko posnetek pretvorimo v seznam in iz nje naredimo Fourierovo transformacijo, lahko zgolj primerjamo med referenčnima posnetkoma sove in mixi katera sova se pojavi v katerem mixu.

Najprej je smiselno narisati graf posnetka obeh sob, spektrogram in Fourierovo transformacijo.



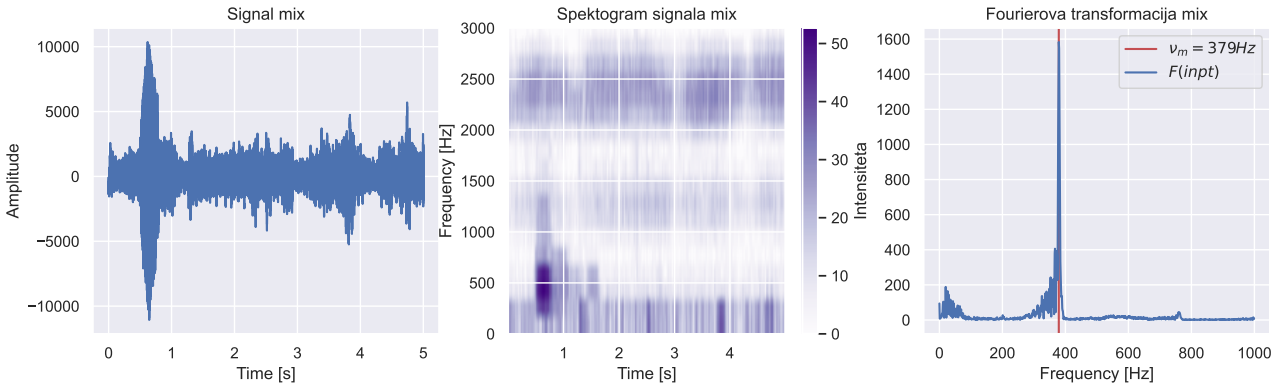
Slika 1: Posnetek in obdelave refenčnega posnetka Bubomono.



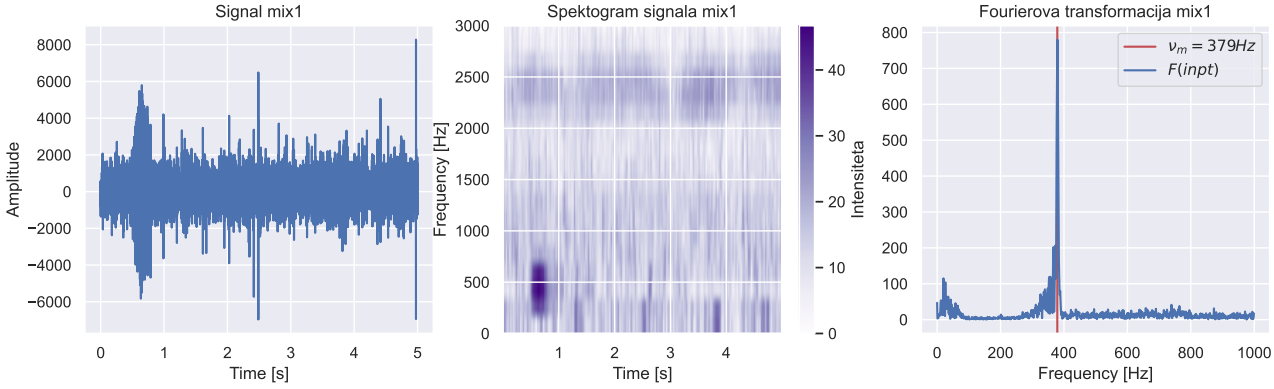
Slika 2: Posnetek in obdelave refenčnega posnetka Bubomono 2.

Iz grafov je vidno, da sta si oglaševanje sov precej podobne. Spektograma imata pege, kjer se sovi oglašata. Sova bubomono se oglaša pri malo višji frekvenci.

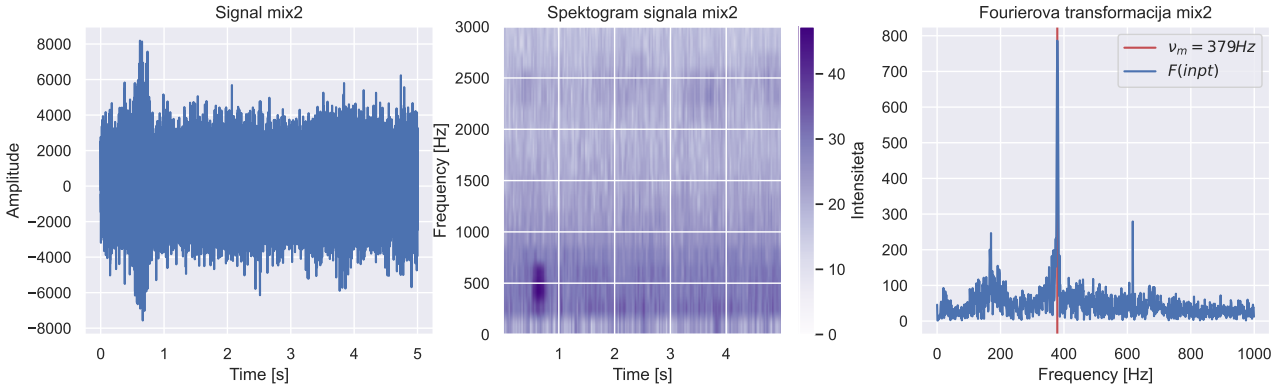
Sedaj lahko narišem enake grafe še za vse tri posnetke.



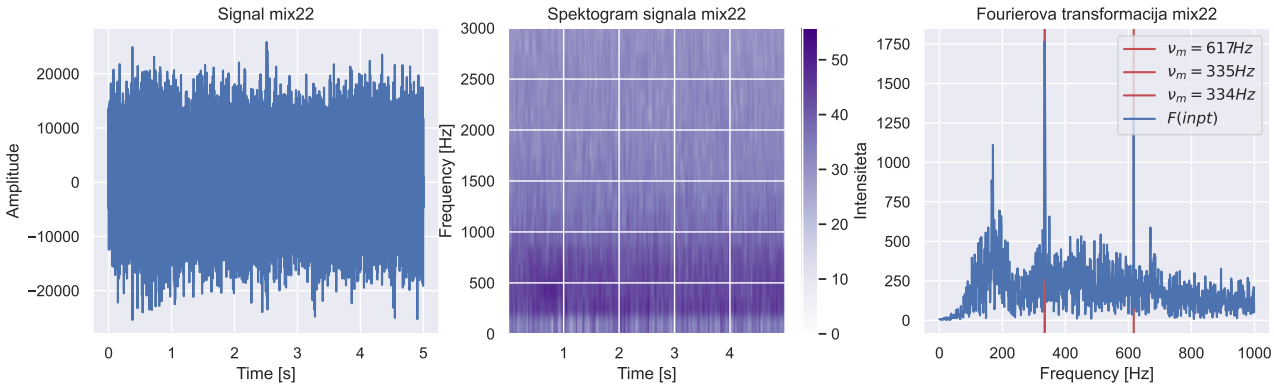
Slika 3: Posnetek in posnetka mix.



Slika 4: Posnetek in posnetka mix1.



Slika 5: Posnetek in posnetka mix2.



Slika 6: Posnetek in posnetka mix22.

Katera sova je katera je torej očitno iz maksimalne frekvence pri Fourierovi transformaciji. Na spektorgramih se tudi opazi, da je prisotna sova. To se vidi, ker nastane pega pri malo manj kot eni sekundi, raztega pa se na majhnem območju nekje med 250 in 750 Hz. Na grafih so prisotne tudi nekatere frekvence z visoko amplitudo, ki pa so seveda frekvence čričkov. Iz grafov lahko tudi opazimo, da je profesor z miksanjem posnetkov `mix2` in`mix22` uporabljal enake posnetke čričkov in vode ter samo povečal intenziteto motenj.

Torej so miksi

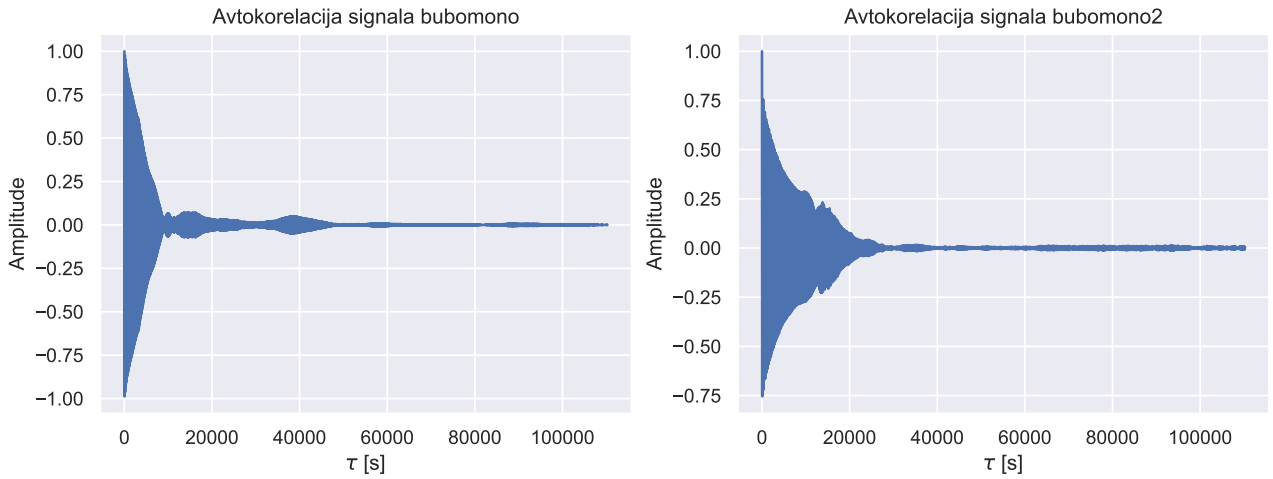
posnetek	sova	frekvenca
mix	bubomono	379 Hz
mix1	bubomono	379 Hz
mix2	bubomono	379 Hz
mix22	bubomono2	334 Hz

Sedaj lahko nalogo, ki je vbistvu že rešena naredimo tudi na način, da med posnetki izvedemo avtokorelacijo.

1.1 Avtokorelacija

Najprej je smiselno narisati avtokorelacijo obeh referenčnih signalov sov, da vidimo kako izgleda graf. Kar pričakujemo je, da bo vrednost blizu ničle zelo velika, ker sta funkciji kar enaki pri $\tau = 0$, potem pa bo graf padel na vrednost zelo blizu ničli. Če grafa, ki ju poskušamo korelirati nista korelirana, bomo dobili grafe, katero obnašanje očitno ne prevladuje pri neki vrednosti. To pomeni, da funkciji nista korelirani.

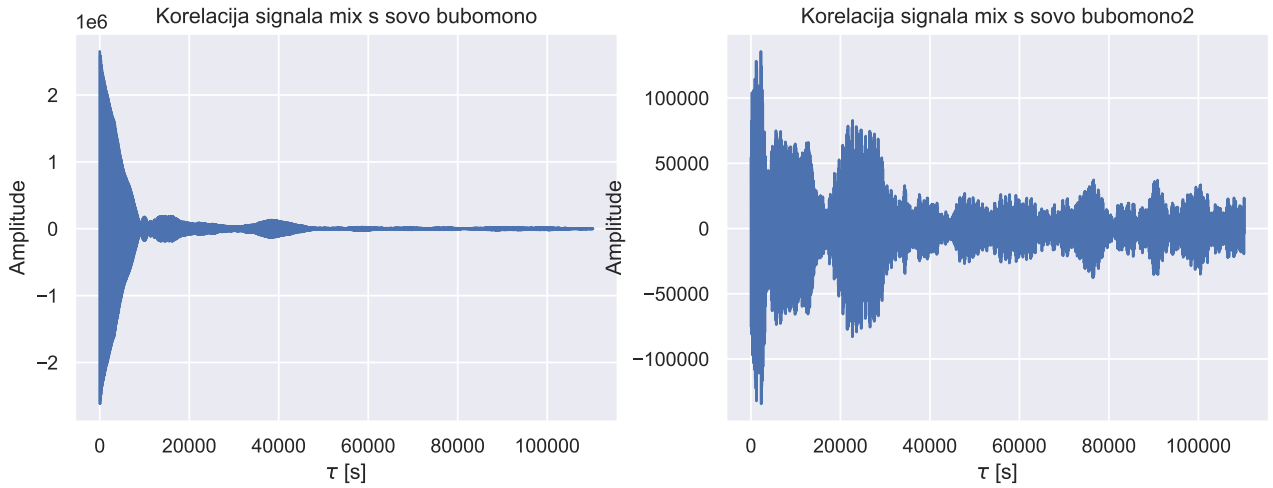
Torej najprej poglejmo kako izgledajo avtokorelacije signalov sov.



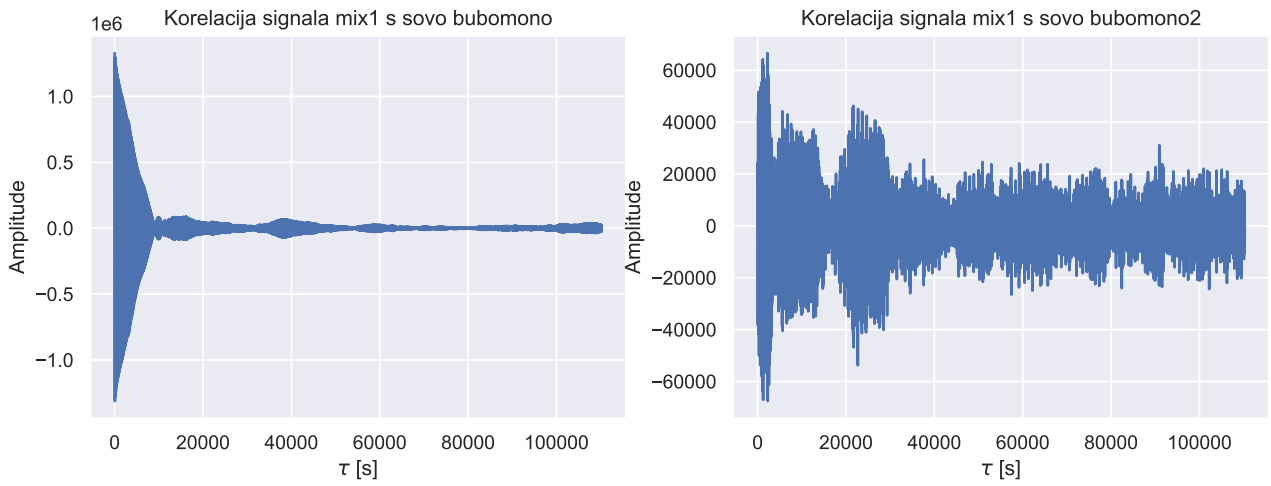
Slika 7: Avtokorelaciji obeh posnetkov sov. Opazimo, da je očitno korelacija močna blizu vrednosti 0, kar pomeni, da smo periodo kar precej dobro zadeli.

Iz grafa 7 vidimo, da smo signal žadeli"na periodo natančno. Iz tega sledi, da lahko uporabimo formulo za eno periodo.

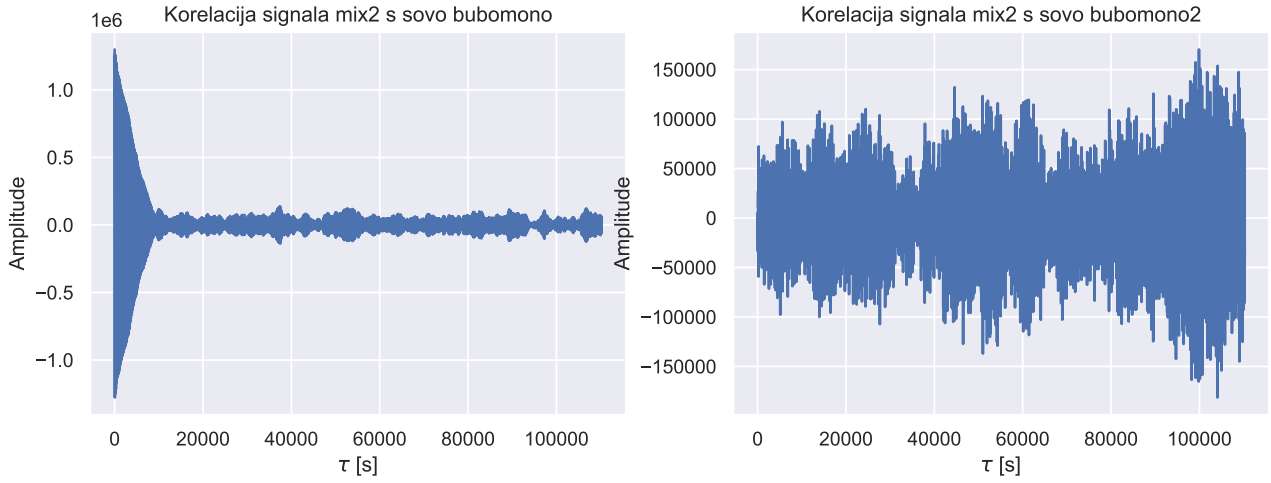
Narišimo zdaj vse korelacije.



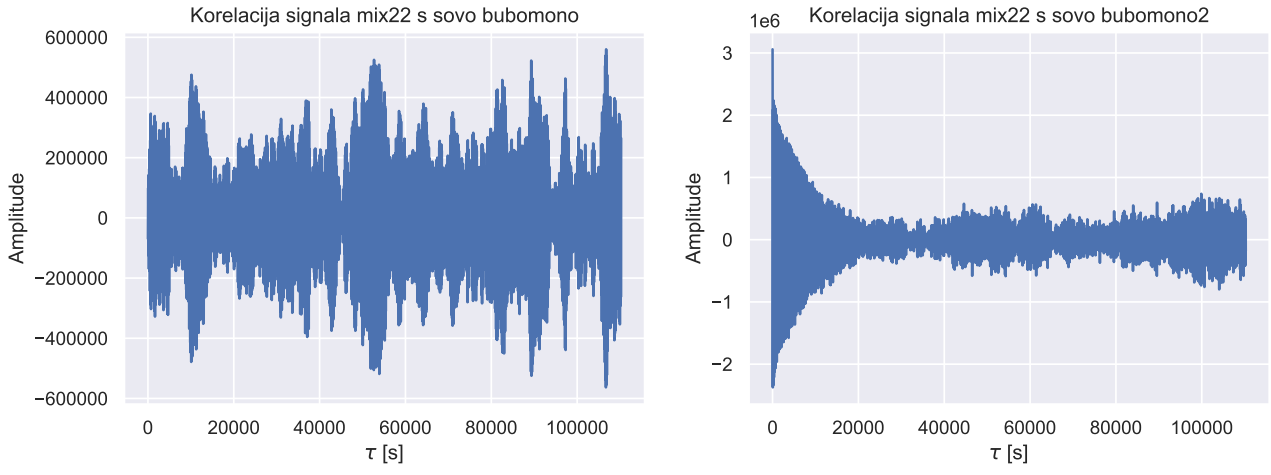
Slika 8: Korelacija z obema sovama posnetka mix.



Slika 9: Korelacija z obema sovama posnetka mix1.



Slika 10: Korelacija z obema sovama posnetka mix2.



Slika 11: Korelacija z obema sovama posnetka mix22.

Tudi iz korelacij med funkcijami je očitno, da naša tabela 1 drži. Iz grafov lahko opazimo, da je

funkcija kar očitno korelirana ali pa ne.

Časovna zahtevnost je pri tej nalogi zgolj teoretična, ker jo je kar težko zmeriti, namreč vsak izračun traja manj kot 1 sekundo (ker so vzorci relativno majhni).

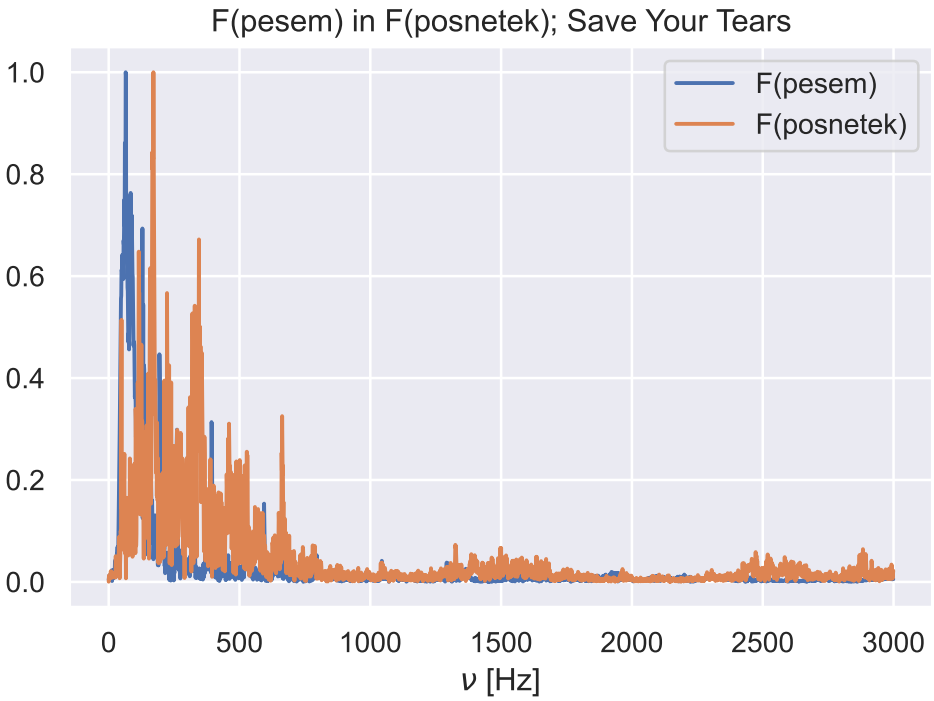
2 Dodatna naloga - Shazam

Shazam je popularna aplikacija za identifikacijo pesmi, ki jih lahko poslušamo s telefonom. Na server strani mora narediti fourierovo transformacijo vseh pesmi, ki jih imajo v databazi in zgenerirati nek fingerprint. Ta algoritem je malo bolj zakompliciran, kot se bom tega lotil jaz, vendar v pricipu se ga da zapakirati tako, da lahko tudi z majhnimi deli oblikujemo fingerprint za celotno pesem. To je zares zelo koristno za identifikacijo, ker moramo z zelo kratkimi odseki pesmi zaznati za katero pesem gre. Avtokorelacija pri njih ne pride v poštev, ker imajo v databazi miljarde pesmi, kar seveda ni smiselno računati za vsako pesem posebej. Torej, kar bom naredil jaz je, da bom za vsako pesem v databazi (imel jih bom 100), zgeneriral fingerprint pesmi z Fourierovo transformacijo na celotni pesmi (vse bodo samplane s 441000 Hz) ter ga s skalarnim produktom primerjal s Fourierovo transformacijo posnetka na telefonu. Tista primerjava, ki bo imela največji skalarni produkt, bo seveda match. Če bi se hotel igrati tudi z outlierji, bi lahko pogledal kje zares je spodnja meja vredonosti skalarnega produkta, da še dobimo nekaj smiselnega. Kar je lahko naslednji korak in kar zelo verjetno tudi počnemo na Shazamu je, da so natrenirali neko nevronsko mrežo, ki je zelo dobra pri odstranjanju "general"šuma iz ozadja ter tudi za idenfikacijo če sploh gre za pesem (kar zelo zmanjša čas procesiranja, ker ne rabimo pošiljat samplov na server, če ne gre za pesem). Vendar to je seveda malo bolj kompliciranо. Moj pristop bo zgolj, da bom vzel največjo vrednost skalarnega produkt frekvenc.

Torej najprej sem iz Youtuba legalno prenesel 10 pesmi ter jih kot celoto transformiral z Fourierovo transformacijo, nato pa vrednosti shranil v seznam. Nato sem posnel posnetek pesmi `(Save Your Tears, The Weeknd)` in `(Mockingbird, Eminem)` s telefonom, zraven pa sem dodal precej hrupa (čez zvok sem govoril, in udarjal po mikrofonu, tako da se komaj kaj sliši dejanske pesmi. Verjetno ne pove prav veliko, ker zvok težko prikažem.

Lahko sedaj kar zaženem moj algoritem na posameznem posnetku in ugotovim, če pravilno napove katera pesem je.

Najprej bom uporabil pesem `(Save Your Tears, The Weeknd)`. Zanimivo je, da čeprav je sample posnetka precej precej krajši (10 sekund) je graf Fourierove transformacije kar podobne oblike, kar kaže da metoda ni tako zgrešena, in tudi kar dobro deluje (čeprav so outlierji).

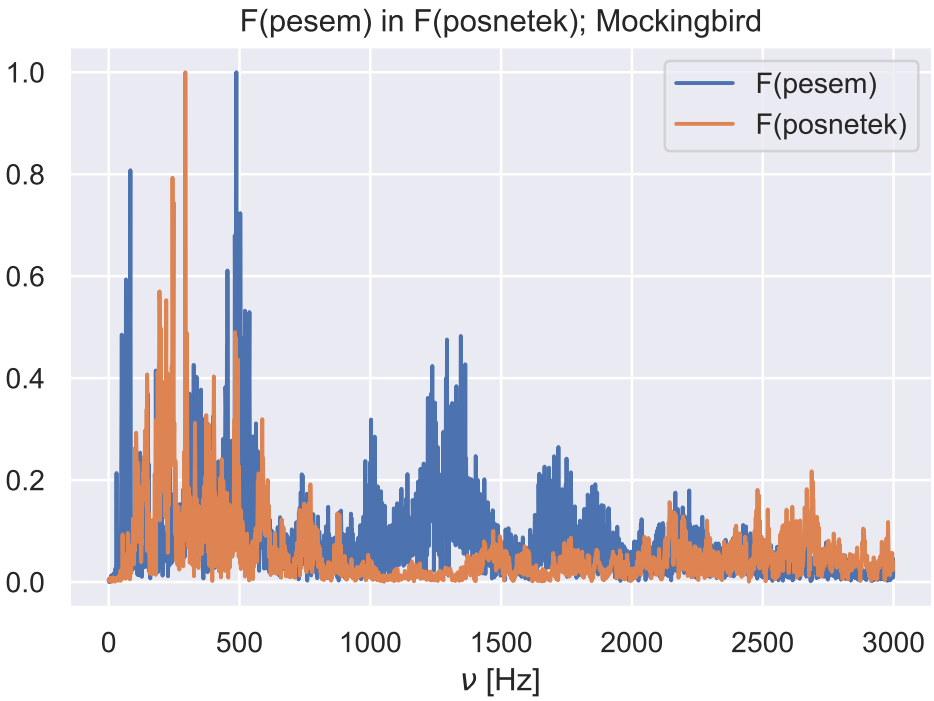


Slika 12: Fourier posnetka in pesmi.

song	value
Avicii-TheNights-UtF6Jej8yb4-192k-163340136690...	40483.781299
TheWeeknd-SaveYourTears-OfficialMusicVideo–XX...	31287.951054
Eminem-Mockingbird-OfficialMusicVideo–S9bCLPw...	27159.299280
TheWeeknd-TakeMyBreath-OfficialMusicVideo–rhT...	23206.531579
TheKidLAROI,JustinBieber-STAY-OfficialVideo–k...	12264.994628
GlassAnimals-HeatWaves-OfficialVideo–mRD0-Gxq...	6242.968198
MikePosner-ITookAPillInIbiza-SeebRemix–Explic...	NaN

Kar dobimo je kar precej dobra predikcija, če odšetejemo, da program vedno da eno pesem `The nights, Avicii` na prvo mesto. Razlog za to je, da je precej bolj ravna funkcija pri nižjih frekvencah. Iz tega sledi, da praktično delamo korelacijo med funkcijo z viskimi vrednostmi in jo primerjamo z korelacijo z nizkimi vrednostmi. Zato bo seveda zmagala vedno večja vrednost, ne glede na obliko.

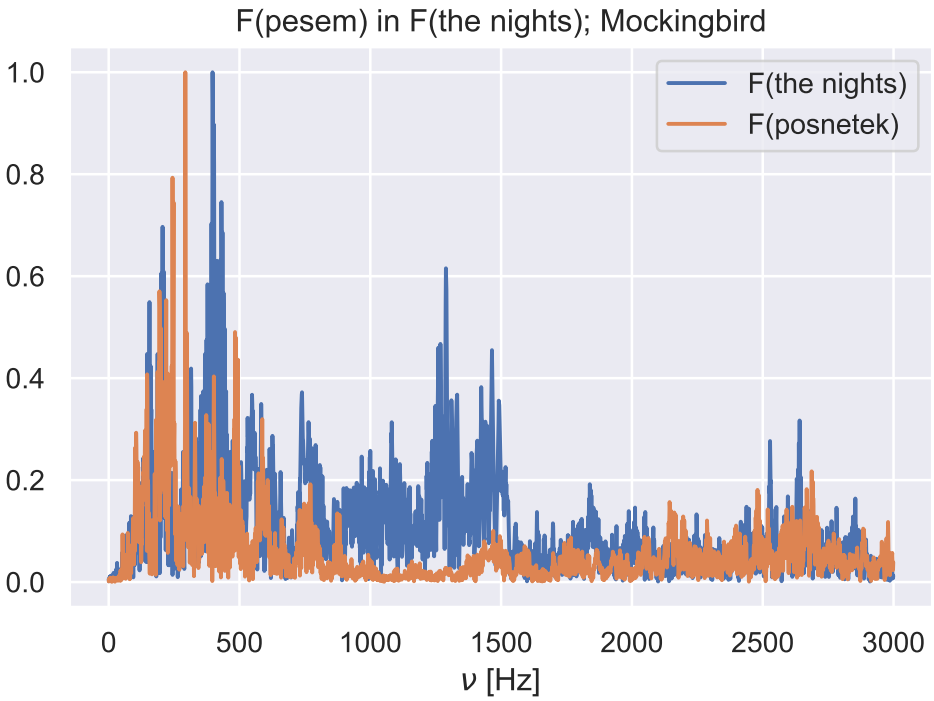
Enako lahko naredimo tudi za pesem `Mockingbird, Eminem`.



Slika 13: Fourier posnetka in pesmi.

song	value
Avicii-TheNights-UtF6Jej8yb4-192k-163340136690...	5868.829027
Eminem-Mockingbird-OfficialMusicVideo–S9bCLPw...	2562.026471
TheKidLAROI,JustinBieber-STAY-OfficialVideo–k...	2240.357660
TheWeeknd-SaveYourTears-OfficialMusicVideo–XX...	1578.344091
TheWeeknd-TakeMyBreath-OfficialMusicVideo–rhT...	797.569793
GlassAnimals-HeatWaves-OfficialVideo–mRD0-Gxq...	581.505771
MikePosner-ITookAPillInIbiza-SeebRemix–Explic...	NaN

Zgolj iz vizualnih razlogov, lahko še pokažem, obliko funkcije pesmi, ki vedno prevlada.



Slika 14: Fourier posnetka in pesmi.

Funkcije je precej bolj ravna in nima zelo distinktinih tonov, zato prevlada.

Iz vsega tega sledi, da ta postopek ni najboljši. Shazam očitno ne vzame Fourierove transformacije celotne pesmi, vendar je treba ugotoviti nekaj boljšega, ker si ne morejo privoščiti, da vedno predlaga eno pesem, outlierjov pa je verjetno preveč, da bi lahko vse izbrisali na roko ali kaj podobnega.

2.1 Dodatek dodatka - kako dejansko deluje Shazam

Po tem, ko sem napisal mojo kodo in ugotovil, da postopek z Fourierovo analizo celotnega posnetka ne deluje najbolje (čeprav zazna pesmi, vendar je razlika skalarnih produktor precej majhna), sem preučil kako Shazam dejansko deluje (oziroma je v prvotnih fazah podjetja), kar opisano v članku, ki so ga objavili kmalu po ustanovitvi firme [1]. Najbolj pomemben je algotirem fingerprinta, ker ko ga enkrat imamo je stvar primerjave oziroma ugotovitve pravilne pesmi iz databaze lahek proces (primerjava vektorjev z neko mero), je pa tudi velik problem če je zelo veliko hrupa, kar je vbistvu kar Shazam razlikuje od ostalih.

Najprej zares naredijo spektrogram pesmi (torej Fouriera ob vsakem $n\Delta t$). Kar naredimo potem je, da pri vsakem Δt_n izberejo frekvenco z maksimalno amplitudo. To namesto 2D matrike poda zgolj 1D vektor dolžine njihovega frekvenčnega območja. To je kar naš fingerprint. Potem pa z ne naredimo direktno skalarnega produkta, vendar gledamo oddaljenost v 2D prostoru od posameznih točk pri primerljivih časih, kar je seveda smiselno zato, ker je lahko posnetek iz poljubnega odseka v pesmi.

Literatura

[1] Li-Chun Wang, A. An industrial-strength audio search algorithm - columbia university.