# Detekcija ritma v glasbi

David Rubin (david.rubin@student.um.si)

22. avgust 2019

## 1 Uvod

S pomočjo diskretne valčne transformacije boste izdelali postopek za detekcijo ritma basov v glasbi. Postopek detekcije ritma se bo izvajal paketno (ang. ofline), po pritisku na poseben gumb, saj je postopek obdelave za procesno enoto dokaj zahteven.

Postopek detekcije ritma je povzet po opisu v članku [1], v 3. poglavju (Feature Extraction), podpoglavje C. Sestoji iz naslednjih korakov:

1. **Odstranjevanje drugega kanala v primeru stereo zapisa.**

2. **Izračun diskretne valčne transformacije nad signalom.**

   S pomočjo poljubne knjižnice izračunajte diskretno valčno transformacijo (DVT) signala. Uporabite Daubechies-ev valček 4. stopnje (*db4*). Število nivojev dekompozicije določite sami, pazite le, da boste z izbranim nivojem dobro zaznali base v glasbi. Dobro preverite, kako so v izhodu DVT-ja zapisani izhodni signali (podrobnosti in približek vhodnega signala). Izhodne signale prevzorčite nazaj na osnovno vzorčevalno frekvenco. Izberite EN SAM izhodni signal (podrobnosti ali približek na določeni stopnji) - tisti, ki najbolje kodira ritem v glasbi. Ne pozabite na stopnjo podvzorčenja izhodnega signala DVT (v primerjavi z osnovno vzorčevalno frekvenco)!

3. **Izračun enostavne ovojnice** Za izbran izhodni signal izračunajte enostavno ovojnico (implementirajte funkcijo $y = ovojnica(signal)$), ki sestoji iz naslednjih korakov:

   3.1. **Izračun absolutne vrednosti signala**. Vsem vzorcem v signalu pripišemo pozitivni predznak: $z[n] = abs(y[n])$.

   3.2. **Filtriranje ovojnice z nizkoprepustnim enopolnim filtrom** $w[n] = (1 - a) \cdot z[n] + a \cdot w[n - 1]$, kjer je $a = 0.99$. S tem zgladimo ovojnico.

   3.3. **Podvzorčenje ovojnice** $w[n] = w[Factor \cdot n]$, kjer je navadno $Factor = 16$. S tem pohitrimo kasnješe korake algoritma.

   3.4. **Odstranitev srednje vrednosti** Od signala odštejemo njegovo srednjo vrednost ($mean$).

4. **Izračun ritma** Z namenom, da bi zajeli dinamiko ritma, bomo korake od 4.1 do 4.3 izvedli na pet sekund dolgih odsekih ovojnice. V ta namen bomo najprej obdelali prvih 5 sekund ovojnice (t.j. izvedli bomo vse korake 4.1-4.3 na odseku od 0. do 5. sekunde ovojnice), nato pa se bomo pomaknili za eno sekundo naprej in obdelali naslednji pet-sekundni interval (t.j. izvedli bomo vse korake od 4.1 do 4.3 nad intervalom od 1. do 6. sekunde). Ta postopek bomo ponavljali do konca glasbenega posnetka. $w$ = *izbran odsek ovojnice.*

   4.1. **Izračun avtokorelacije ovojnice** S pomočjo avtokorelacije odseka ovojnice w bomo dobili trenutke udarcev (*beats*) v obravnavanem zvočnem signalu (udarci so predstavljeni z vrhovi v avtokorelaciji odseka ovojnice)

   4.2. **Iskanje vrhov** Maksimumi v avtokoreliranem signalu $w_{corr}$ predstavljajo trenutke "udarcev"ritma. Pri iskanju maksimumov je priporočljivo določiti tudi minimalno razdaljo med sosednjima maksimumoma, ki predstavlja zgornjo dopustno mejo ritma. Ta naj bo v našem primeru okoli 5 udarcev na sekundo ($5Hz$).

   4.3. **Izračun ritma celega signala s pomočjo povprečenja časovnih razlik med sosednjimi vrhovi v** $w_{corr}$ V zadnjem koraku s pomočjo časovni razlik med sosednjimi udarci (maksimumi) v $w_{corr}$ izračunamo povprečen ritem za izbrani odsek ovojnice. V izhodno polje hranimo tako trenutni ritem, kot tudi časovno pozicijo odseka ovojnice (v sekundah). Ne pozabite kompenzirati podvzorčenje iz koraka 3.3!

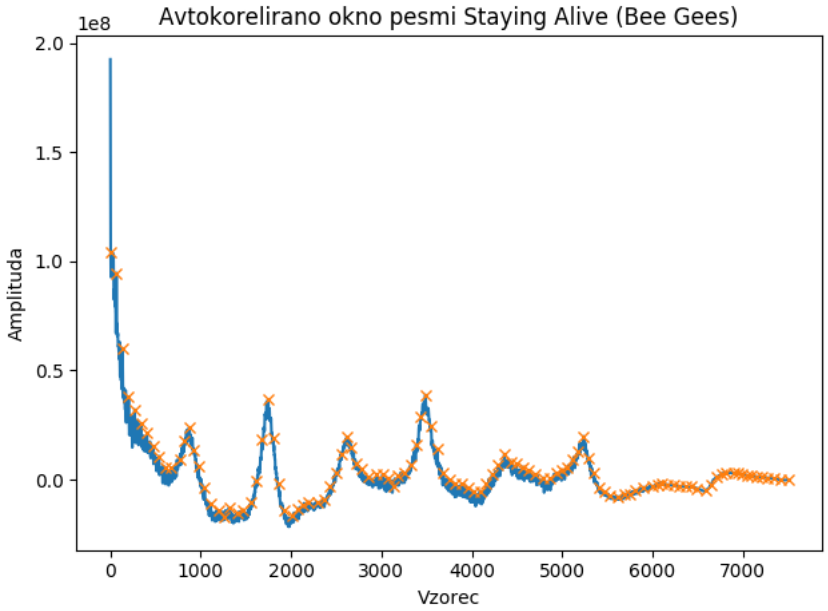
Rezultat postopka naj bo izrisan graf, ki prikazuje ritem glasbe/posnetka v odvisnosti od časa (na celem zvočnem signalu!). Ne pozabite na oznake osi (x os "čas (s)", y os "udarci na minuto")!

# 2 Rezultati

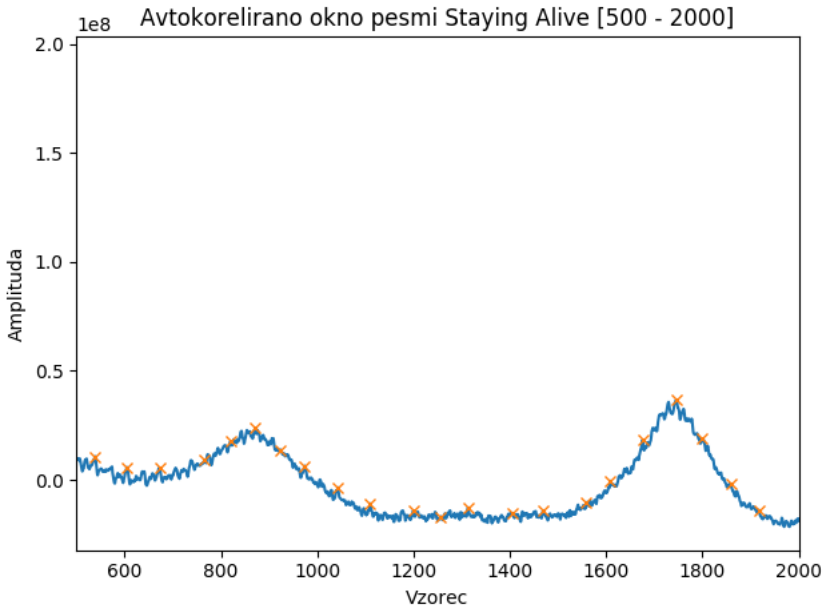
Nalogo sem implementiral kot *Python* program, ki zahteva knjižnice navedene v *requirements.txt*. Pri implementaciji naloge sem imel nekaj težav s sledenjem navodil, tako da sem nekaj korakov priredil po svoje v želji po doseganju boljših rezultatov. Slednje je opisano v naslednjih podpoglavjih.

## 2.1 Algoritem z več vrhovi

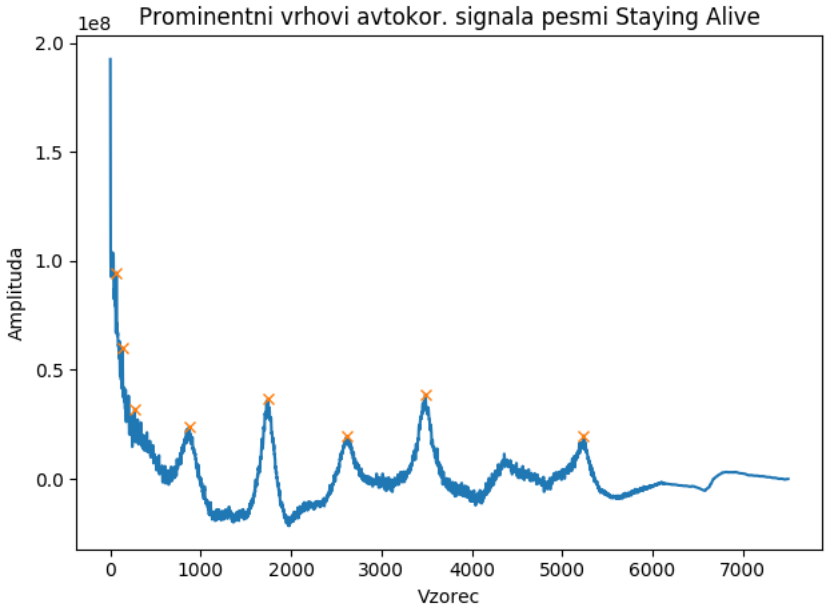
Prvotno sem (kot je bilo navedeno v navodilih) iskal vrhove in za izračun povprečnega ritma v določenem oknu vzel razdalje med temi vrhovi. Moja detekcija vrhov je temeljila na funkciji *find_peaks* iz knjižnice *scipy*, kateri sem dodal razdaljo med vrhovi 50 vzorcev. Rezultat te operacije je prikazan na grafu 1. Na približanem odseku med 500. in 1000. vzorcem na sliki 2 vidimo, da je signal realitnvo šumen in *find_peaks* zazna veliko lokalnih maksimumov. Z nekaj poizkušanja z različnimi parametri, sem nato preko prominence vrhov in izbiro $n$-najbolj prominentnih prišel do izida na sliki 3.




Slika 1: Avtokoreliran signal prvih 5 sekund pesmi Staying Alive (skupine Bee Gees) v modrem in vrhovi najdeni preko *scipy.signal.find_peaks* z razdaljo 50 kot oranžni križci.




Slika 2: Približan avtokoreliran signal slike 1.



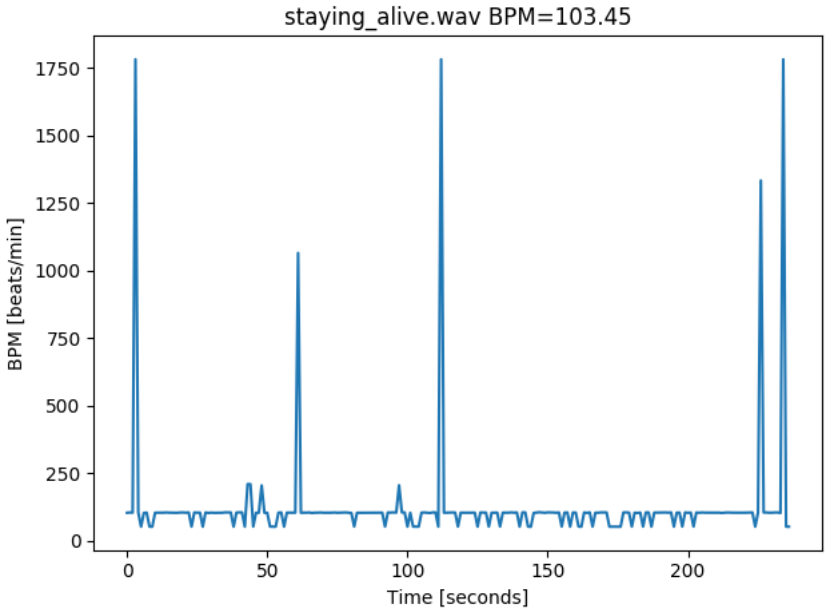

Slika 3: Vrhovi avtokoreliranega signala pesmi Staying Alive (skupina Bee Gees), najdeni s *scipy.signal.find_peaks*, pri čemer je $distance = 50$, nato pa so še sortirani preko prominence (*scipy.signal.peak_prominences*, pri čemer je $wlen = 300$. Izbranih je 8 (uporabniški parameter programa) vrhov.

V nadaljevanju iskanja rešitve sem se opiral na naslednjo enačbo:

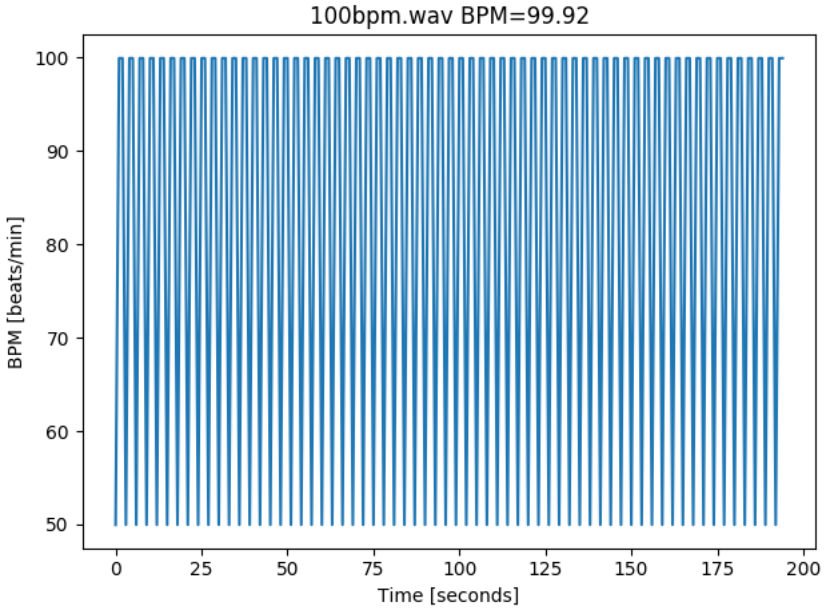
$$BPM = \frac{60 \cdot F_s}{i_{sample} \cdot dec} \tag{1}$$

Pri čemer je $F_s$ frekvenca vzorčenja, $i_{sample}$ je indeks vzorca, $dec$ pa predstavlja faktor podvzorčenja. S to enačbo sem pretvoril vsak vrh v ustrezen BPM, nato pa še poračunal povprečje (za boljše rezultate se je izkazala mediana) za posamezno okno. Pri uporabi 3 vrhov (kot so navedli v članku [1]) in prej omenjenega postopka dobimo za pesem Staying Alive skupine Bee Gees BPM prikazan na sliki 4. Rezultat postopka je 103.45 BPM, kar je dokaj skladno z rezultati drugih algoritmov na spletu kot tudi z namigom v navodilih. Na grafu pa so vidne tudi prve težave mojega prvotnega algoritma - nenadni poskoki, za kar je po mojem mnjenju krivo iskanje vrhov katere uporabim za preračunavanje BPM. Težave prvotnega algoritma potrdi tudi uporaba na drugih zvočnih posnetkih. Za še en kontrolni zvočni signal sem na spletu našel zvok s konstantnim ritmom 100 udarcev na minuto[1]. Z uporabo enakih parametrov kot pri pesmi Staying Alive je tukaj preračunan ritem 49 BPM, z uporabo drugih parametrov (slika 5) pa sicer dobimo boljši rezultat vendar še vedno opazimo precej veliko nihanje.

[1]Najdeno na Youtube, 100 BPM - Metronome: `https://youtu.be/6oz0ivczNSY`




Slika 4: Graf udarcev na minuto pesmi Staying Alive z uporabo prvotnega algoritma, pri čemer uporabimo 3 vrhove, $distance = 50$, in prominenco katere $wlen = 300$.




Slika 5: Graf udarcev na minuto zvočnega posnetka s konstantnim ritmom 100 BPM (težava prvotnega algoritma). Za izračun BPM so uporabljeni 3 vrhovi na okno, $distance = 50$, in prominenco katere $wlen = 50$.

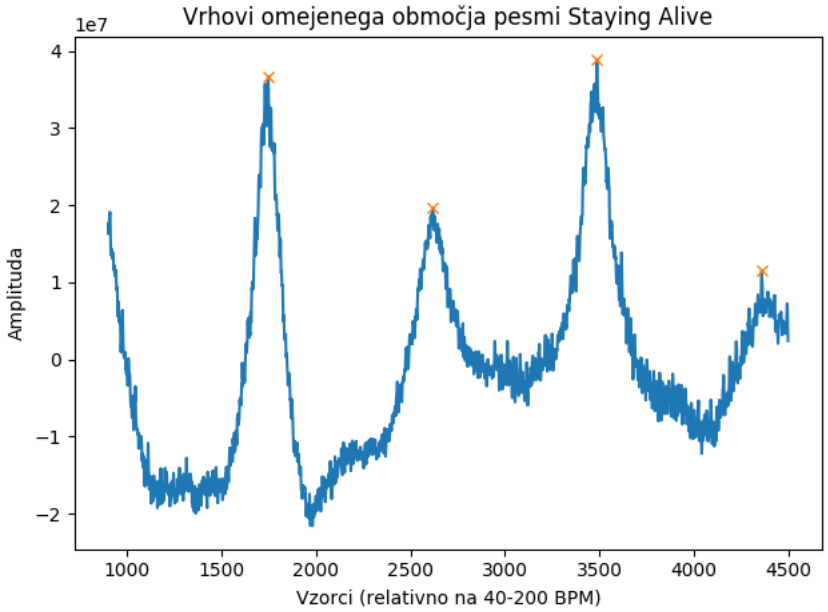
## 2.2 Algoritem omejenega območja

Težavo algoritma opisanega v prejšnjem poglavju sem rešil tako, da sem omejil območje iskanja vrhov. V članku omenjajo, da si ustvarijo histogram, v katerega nato uvrščajo 3 vrhove in iz tega izračunajo povprečen BPM. Slednjega sicer pridobijo z izboljšano avtokorelacijo (skaliranje in seštevanje avtokorelacije signala pri različnih faktorjih), sam pa za izračun še vedno uporabljam enačbo 1. Iz enačbe ven lahko izpeljemo najnižji in najvišji indeks vzorca glede na območje udarcev na minuto (kot v članku sem vzel od 40 do 200 udarcev na minuto):

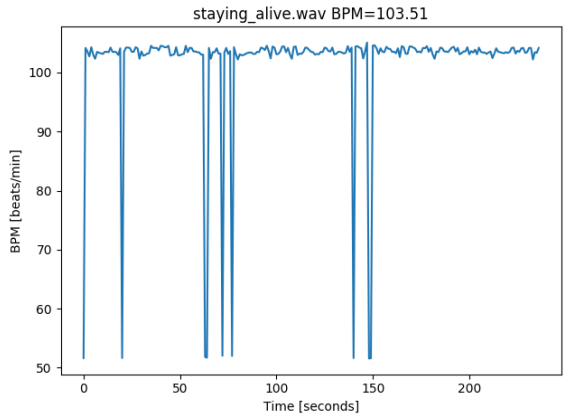
$$min_i = \frac{60 \cdot F_s}{40 \cdot dec} \tag{2}$$

$$max_i = \frac{60 \cdot F_s}{200 \cdot dec} \tag{3}$$

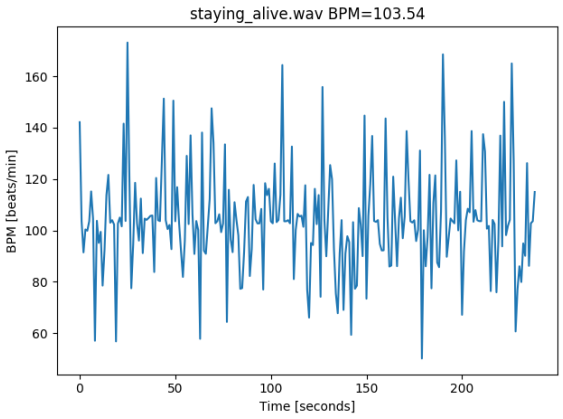
S slednjima mejama pa lahko uvedemo meje v avtokoreliran signal in nato iščemo vrhove (slika 6). Z omejitvijo območja se izognemo skoraj vedno deketiranem vrhu v bližini 0 (avtokorelacija) in pa tudi ponovitvam (ni potrebno računati razlik med vrhovi). Slabost pa je, da lahko detektiramo le na območju določenim z enačbama 2 in 3. Poskusi kažejo, da opisani postopek deluje najbolje v primeru, ko nad omejenim območjem vzamemo le 1 vrh in sicer tistega z najvišjo vrednostjo. V kolikor vpeljemo več vrhov je potrebno dodatno spremeniti še ostale parametre (recimo dolžino okna), da dobimo podoben rezultat (glej sliki 7a in 7b).




Slika 6: 4 najbolj prominentni vrhovi na omejenem območju (900 - 4500 vzorec) prvih 5 sekund pesmi Staying Alive, razdalja med vrhovi (*find_peaks distance*) je 5.




(a) BPM pesmi Staying Alive skozi čas, pri čemer uporabimo 1 vrh in dolžino okna 5 sekund. Uporabljen je izboljšan algoritem, kjer smo omejili območje iskanja BPM.




(b) BPM pesmi Staying Alive skozi čas, pri čemer uporabimo 10 vrhov in dolžino okna 3 sekunde. Uporabljen je izboljšan algoritem, kjer smo omejili območje iskanja BPM.

Slika 7: Primerjava enega vrhova in uporabo večih. Slednje zahteva spremembo še ostalih parametrov za primerljive rezultate.

### 2.3 Testiranje na zvočnih posnetkih

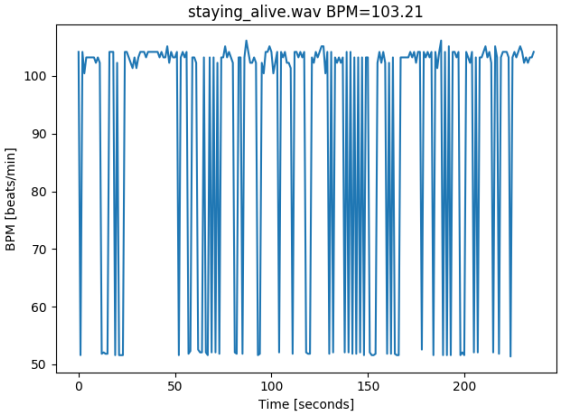
Poleg prej omenjenih parametrov sem preveril tudi uporabo povprečja in mediane za računanje BPM. Izkazalo se je, da mediana daje v mojem primeru boljše rezultate (razni poskoki na grafih se manj upoštevajo), prav tako pa sem preveril tudi ali dobimo boljše rezultate s sestavljanjem dekompoziranih signalov (in ustrezno podvzorčenje na vsaki stopnji). Povprečen rezultat udarcev na minuto se ni spremenil, po mojem mnenju pa se je polepšal prikaz BPM na grafu. Na sliki 8 in ustreznih podslikah vidimo primerjavo uporabe podvzorčenja in samo aproksimacije ali vseh stopenj DVT. Medtem ko je na končni rezultat vpliv minimalen, se iz vidika grafa lepše opazi konstanost ritma Staying Alive v primerih, ko uporabimo sestavljen dekompoziran signal.

Poleg pesmi Staying Alive in konstantnega ritma 100 BPM sem s svojim algoritmom testiral tudi druge pesmi. Rezultati[2] so prikazani v tabeli 1.

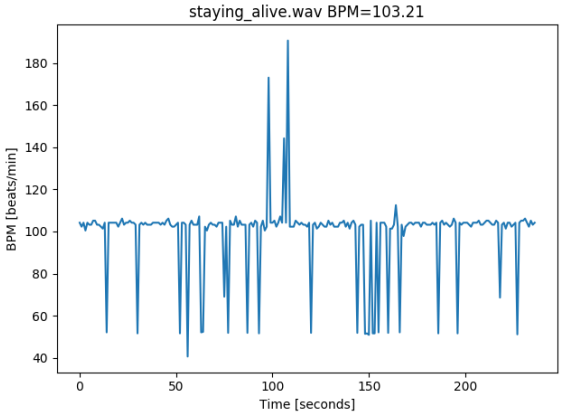
| Pesem | BPM | bpm.py | graf |
|---|---|---|---|
| Staying Alive | 104 | 103.21 | 8b |
| Bad Guy | 135 | 135.54 | 9 |
| Miracle | 100 | 101.35 | 10 |
| Bohemian Rhapsody | 143 | 144.23 | 11 |
| Lutka | 160 | 127.12 | 12 |
| Kažu | 170 | 160.71 | 13 |

Tabela 1: Primerjava točnosti BPM za različne pesmi.

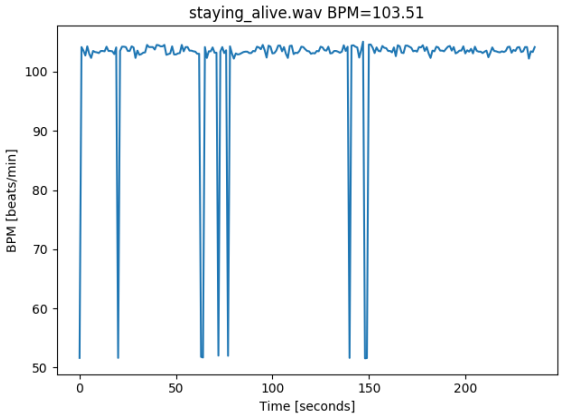
[2]Vir za vrednosti v stolpcu BPM v tabeli 1 je spletna stran `https://songbpm.com/`




(a) Kot podatke o pesmi vzamemo le aproksimacijo signala na 4. stopnji DVT. Signal tudi podvzorčimo s stopnjo 16.

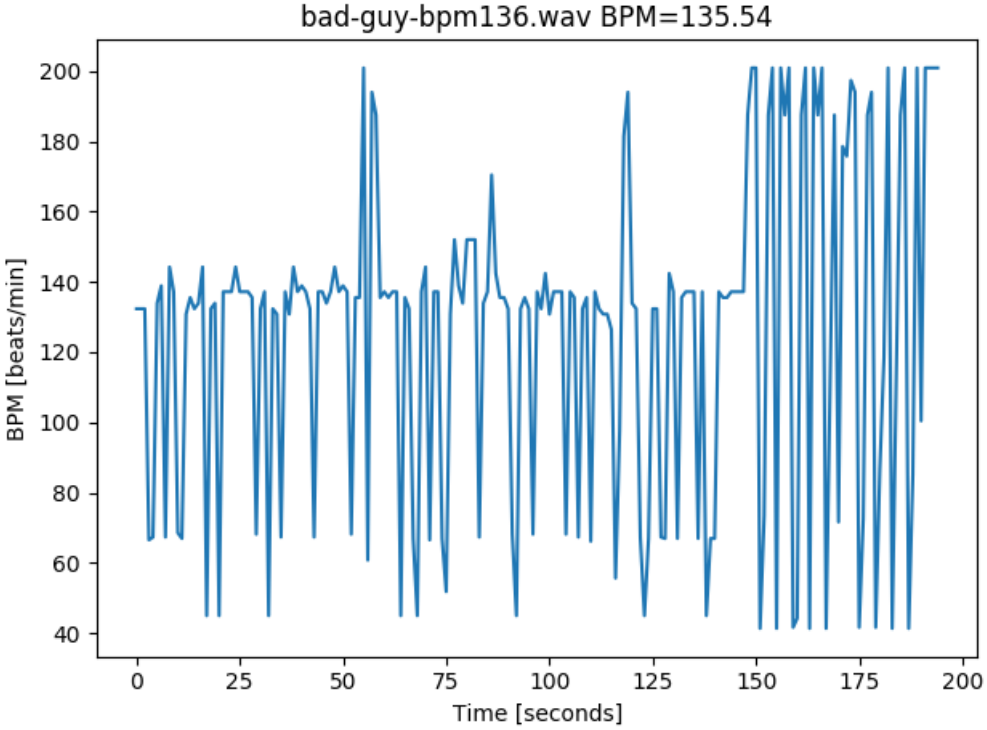



(b) Ovojnice nad posameznimi stopnjami DVT se (ustrezno) podvzorčene združijo in podvzorčijo s faktorjem 16.

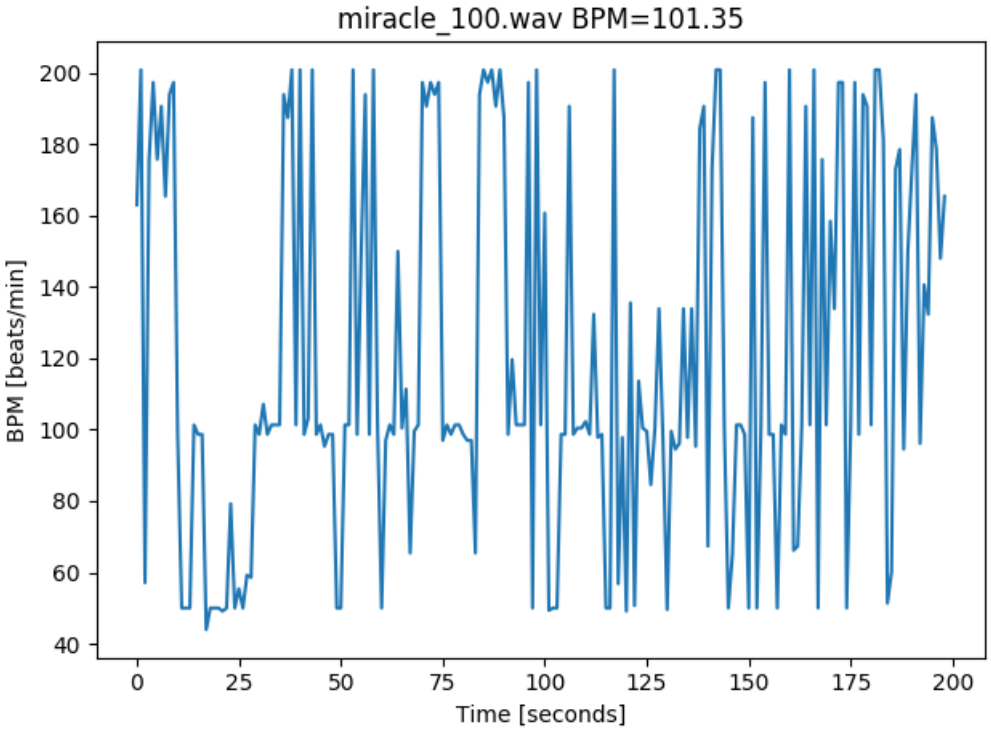



(c) Ovojnice nad posameznimi stopnjami le združimo (podvzorčimo glede na vsako stopnjo) brez dodatnega podvzorčenja na najnižjem nivoju.

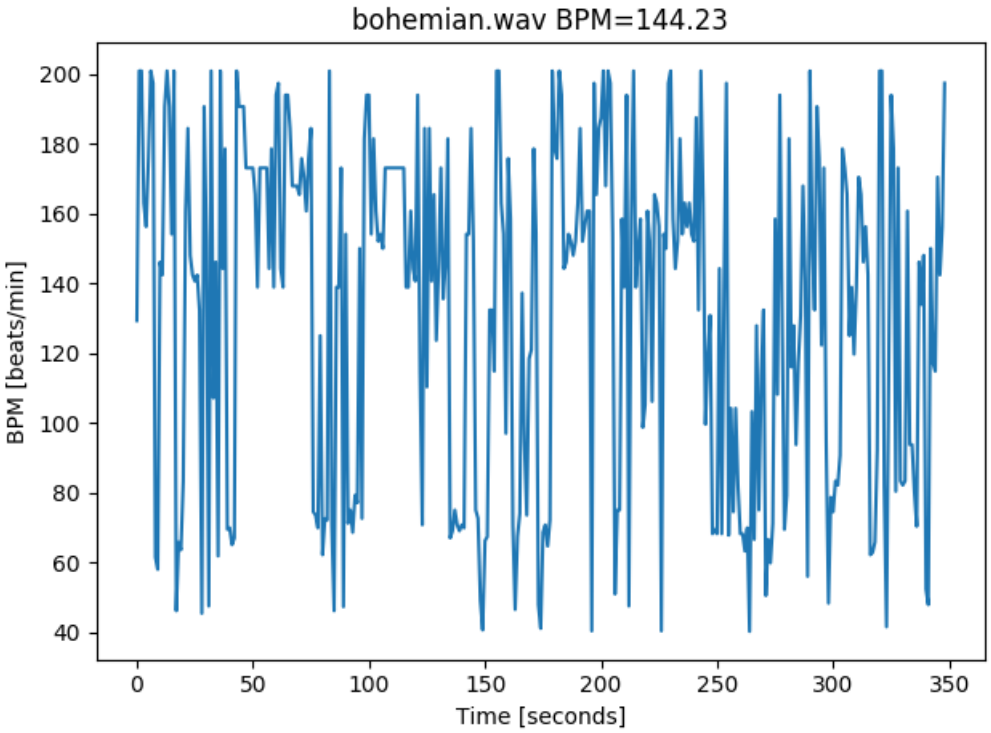
Slika 8: Primerjava uporabe samo aproksimacije DVT (8a), združevanje posameznih stopenj in vpeljava podvzorčenja s faktorjem (8b) in združevanje stopenj DVT brez podvzorčenja na najnižjem nivoju (8c). Uporabljena pesem je Staying Alive, pri čemer se ostali parametri razen omenjenih niso spreminjali.



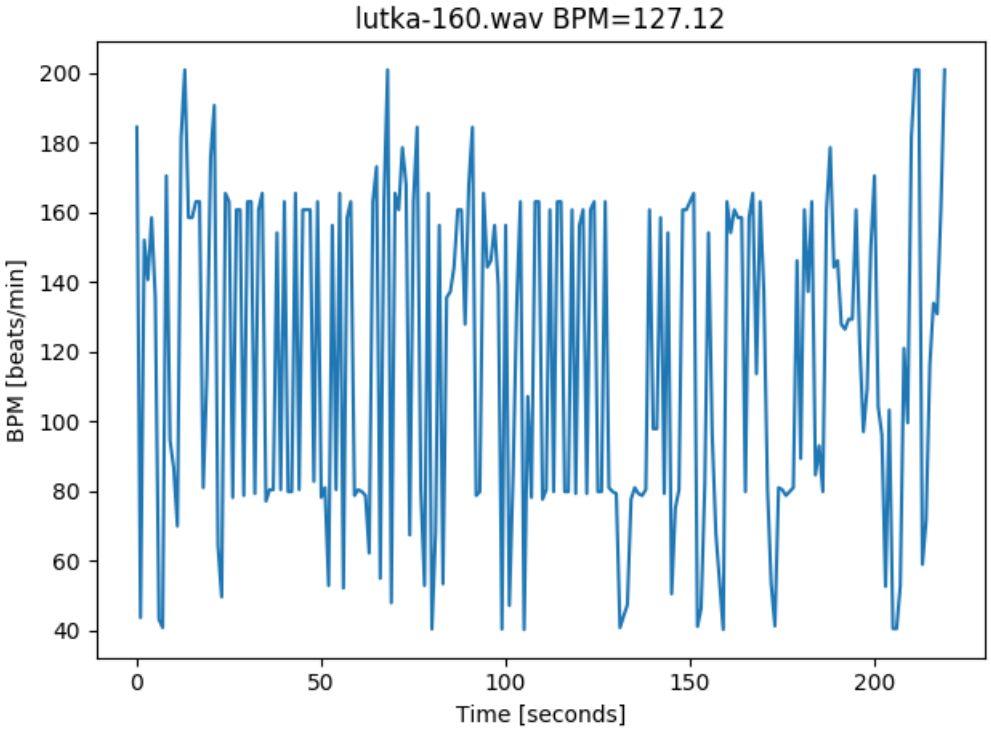

Slika 9: BPM Graf pesmi Billie Eillish - Bad guy (`https://youtu.be/Jqi4GBxmwgg`).



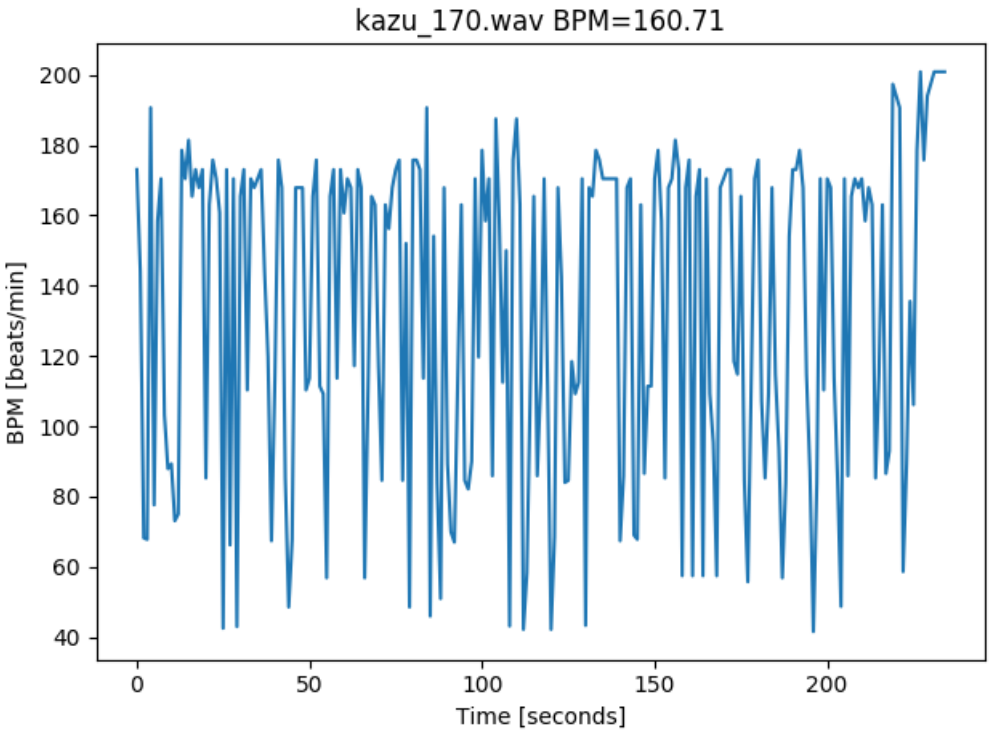

Slika 10: BPM Graf pesmi Caravan Palace - Miracle (`https://youtu.be/QdabIfmcqSQ`).




Slika 11: BPM Graf pesmi Queen - Bohemian Rhapsody (`https://youtu.be/V8QqZ0Qb3U8`).




Slika 12: BPM Graf pesmi S.A.R.S. - Lutka (`https://youtu.be/E8Ms56gX-Tc`).




Slika 13: BPM Graf pesmi Dubioza Kolektiv - Kažu (`https://youtu.be/FZZJeMKJV3M`).

## Literatura

[1] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.