UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Marko Kuder

Iskanje prevladujoče melodije v glasbenih posnetkih

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Matija Marolt

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01581/2009
Datum: 01.09.2009

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MARKO KUDER**

Naslov: **ISKANJE PREVLADUJOČE MELODIJE V GLASBENIH POSNETKIH**

EXTRACTION OF PREDOMINANT MELODY FROM AUDIO RECORDINGS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V okviru diplomske naloge raziščite postopke za iskanje prevladujoče melodije v glasbenih posnetkih. Analizirajte in implementirajte algoritem PreFEst in ga dopolnite z lastnimi postopki za gradnjo melodičnih sledi in ugotavljanje prisotnosti melodije. Izboljšave preizkusite na javno dostopnih testnih množicah.

Mentor:

doc. dr. Matija Marolt

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Franc Solina

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Namesto te strani **vstavite** original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo!

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani/-a Marko Kuder,

z vpisno številko 63040083,

sem avtor/-ica diplomskega dela z naslovom:

Iskanje prevladujoče melodije v glasbenih posnetkih

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom
  doc. dr. Matije Marolta
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 8.3.2010 Podpis avtorja/-ice:

Zahvala

Zahvaljujem se mentorju doc. dr. Matiji Maroltu za pomoč in potrpežljivost med nastajanjem diplomskega dela. Prav tako se zahvaljujem vsem, ki so me v tem času podpirali in spodbujali.

Kazalo

Seznam uporabljenih kratic in simbolov

Acc - točnost algoritma
$c_{0i}^{(t)}(h|F,m)$ - začetna utež harmonične komponente
$c_{\mathrm{ML}}^{(t)}(h|F,m)$ - najverjetnejša utež harmonične komponente (iterativni približek)
d' - sposobnost ločevanja med razredoma prisotnosti melodije (ang. discriminability, d-prime)
EM - maksimizacija pričakovane vrednosti (ang. expectation-maximization)
f_{cent} - frekvenca v centih (logaritmično merilo)
f_{Hz} - frekvenca v hertzih
f_s - frekvenca vzorčenja
F - osnovna frekvenca
F0 - prevladujoča frekvenca
FA - delež referenčnih vrednosti z neprisotno melodijo, ki jih je algoritem označil napačno (ang. false alarms)
Fl - spodnja mejna frekvenca pasovnoprepustnega filtra
Fh - zgornja mejna frekvenca pasovnoprepustnega filtra
$g_{m,h}$ - faktor sodih harmoničnih komponent
$G(h;1,U_i)$ - Gaussova distribucija spremenljivke h okoli povprečja 1 s standardnim odklonom U_i
L_{tr} - dolžina sledi
m - tip modela
MAP - maksimalna aposteriorna verjetnost (ang. maximum a posteriori probability)
N_{tr} - število sledi
o - povprečna ocena sledi
$p(x|F,m,\mu'^{(t)}(F,m))$ - model harmoničnih komponent z osnovno frekvenco F, tipom modela m in obliko $\mu'^{(t)}(F,m)$ ob času t

$p_{\psi}^{(t)}(x)$ - opazovana funkcija verjetnostne gostote spektra (ang. probability density function - PDF)
$P(t)$ - vsota Fourierove transformacije frekvenčnega spektra v časovnem okviru ob času t
$\overline{P(T)}$ - povprečna vsota Fouriereve transformacije frekvenčnih spektrov v časovnih okvirih sledi T
STFT - kratkočasovna Fourierova transformacija (ang. short-time Fourier transform)
t - čas
T - sled melodije
U_i - standardni odklon
$v(T)$ - prisotnost melodije v času sledi T
vxRec - delež referenčnih vrednosti s prisotno melodijo, ki jih je kot takšne ocenil tudi algoritem (ang. voice recall)
$\overline{w^{(t)}(F,m)}$ - utež modela m z osnovno frekvenco F ob času t
$\overline{w_{\mathrm{ML}}^{(t)}(F,m)}$ - najverjetnejša utež modela (iterativni približek)
$w'^{(t)}$ - utež modela v predhodni iteraciji
x - frekvenca v centih (logaritmično merilo)
$\alpha_{i,m}$ - normalizacijski faktor modela
$\mu^{(t)}(F,m)$ - oblika modela

Oznake dodatkov k osnovnemu algoritmu (podrobnejši opis v tabeli 3.2):

- ZS - združevanje spektrov
- FL - razširitev množice filtrov
- TR - sledenje vrhov v spektrogramu
- OO - odpravljanje odstopajočih vrednosti
- NN - izravnava hipoteze z najboljšimi sosedi
- PP - prepoznavanje prehodov med sosednimi vrednostmi
- AD - izračun točnih frekvenc med diskretnimi vrednostmi košev
- VX - ugotavljanje prisotnosti melodije

Povzetek

Iskanje prevladujoče melodije v glasbenih posnetkih je problem, ki na vsakoletnih tekmovanjih MIREX še vedno dokazuje, da ga ni enostavno rešiti. Algoritmi, razviti v ta namen, poskušajo določiti potek melodije (višino prevladujočega tona ob različnih časih) v posnetkih ter ugotoviti, kje je melodija prisotna in kje ne. Rezultati tekmovanj kažejo, da nobeden od teh dveh problemov še ni popolnoma rešljiv, saj algoritmi delajo napake tudi pri človeku lahko razumljivih skladbah.

V okviru te diplomske naloge sem implementiral algoritem PreFEst, ki ga je v letih 1999-2004 razvijal Masataka Goto in je v osnovi obetaven pristop, konkurenčen tedanjim rešitvam, a ga avtor pozneje ni več izboljševal. V tem diplomskem delu sem poskusil algoritem dodelati in ob lastni implementaciji algoritma (brez Gotove različice sledenja) preizkusil več možnih izboljšav - ugotavljanje prisotnosti melodije, spremembo izračuna spektrograma z dodatno stopnjo v množici filtrov in/ali združevanjem spektrogramov z različnimi okni, iterativno gradnjo sledi, metodo odpravljanja vrednosti, odstopajočih od povprečja, izravnavanje hipoteze s pomočjo ocenjevanja najboljših naslednikov, prepoznavanje prehodov s Houghovo transformacijo in prilagajanje hipoteze vmesnim frekvencam med diskretnimi koši frekvenčne lestvice.

Svojo razširitev Gotovega algoritma sem preizkusil na testni množici tekmovanja ISMIR 2004 in učni množici MIREX 2005. Rezultate na testni množici sem primerjal z rezultati algoritmov z dosedanjih tekmovanj v iskanju melodije. Preveril sem vpliv različnih izboljšav in ugotovil možne slabosti in prednosti algoritma z analizo večih hipotez, ki jih je izračunal na testnih datotekah.

Ključne besede:

melodija, glasba, spektralna analiza, algoritem PreFEst, tekmovanje MIREX

Abstract

Audio melody extraction is a problem that still presents itself as not easily soluble on each annual MIREX competition. Algorithms developed for this purpose try to establish a track of melody (frequency of predominant tone at each moment) in songs and determine, whether the melody is even present. The results of competitions show that none of these two problems is completely soluble, since the algorithms make errors even on songs easily understandable by humans.

In this thesis I describe my implementation of the PreFEst algorithm, developed by Masataka Goto from 1999 to 2004. It is based on a promising approach that was very competitive at the time, but hasn't been developed further by the author. In this paper I propose my own implementation of the algorithm (without Goto's version of tracking) with several possible improvements - voicing detection, alternate spectrogram calculation with an additional level in the multi-rate filter bank and an optional combination of multiple window sizes, iterative tracking of peaks, outlier elimination, hypothesis balancing with the use of best successor evaluation, transition recognition using the Hough transform and adaptation of hypothesis to inter-frequency-bin values.

I have tested my expanded version of Goto's algorithm on the ISMIR 2004 competition database and MIREX 2005 learning set. I have compared my results with other algorithms from previous competitions in audio melody extraction. I have established the effect of using different improvements and determined possible weaknesses and strengths of this algorithm by analysing several hypotheses calculated on test data.

Key words:

melody, music, spectral analysis, PreFEst algorithm, MIREX competition

Poglavje 1

Uvod

Glasba v vseh razen v najbolj enostavnih oblikah vsebuje veliko informacij. Lahko je sestavljena iz zvočnih zapisov več kot 100 inštrumentov in ob tem vsebuje še človeški glas, kar tvori zapleteno strukturo zvoka, ki pa jo človeški možgani relativno lahko razumejo. Človek z razmeroma dobrim posluhom lahko brez težav sledi posameznim inštrumentom v skladbi in njihovi melodiji, si to melodijo zapomni ter jo ponovi s svojim glasom ali lastnim naučenim inštrumentom. Prav tako lahko med vsemi komponentami skladbe prepozna glavno melodijo, ki jo večina ljudi enako določi, kljub temu, da nimajo enakega glasbenega okusa ali posluha.

Pri poskusu računalniške obdelave se ta naloga izkaže za precej težjo, predvsem zato, ker so vse komponente skladbe (vse melodije inštrumentov in glasov) združene v eno zvočno sled - eno spreminjajoče nihanje. Obstajajo sicer tudi stereo oz. večglasni zapisi, a človeški možgani vse oblike, tudi mono zapis, približno enako dobro razločijo in precej enostavno določijo glavno melodijo, kar je dober argument za trditev, da bi bilo možno te informacije prepoznavati tudi z računalnikom, če bi lahko glasbo dovolj dobro razčlenili in ugotovili, na katere zvoke smo ljudje najbolj pozorni.

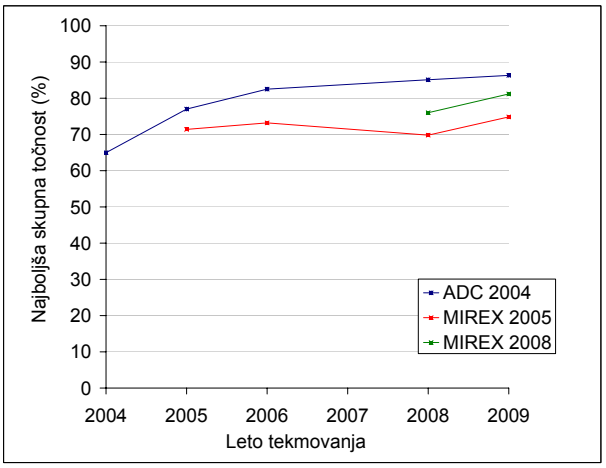
Začetki raziskav na tem področju segajo že v 70-ta leta prejšnjega stoletja, še posebej poskusi avtomatskega zapisa glasbe (npr. Askenfelt, Moorer in Rabiner [1, 10, 13]), resnejši razvoj pa se je začel z razmahom informatike sredi 90-ih let prejšnjega stoletja (Goto [8]). Eden vidnejših predstavnikov konec 90-ih je bil Masataka Goto, ki je leta 1999 razvil algoritem PreFEst (Predominant-F0 Estimation [5, 6, 7, 9]). Algoritem deluje na principu združevanja harmonskih komponent posameznih frekvenc in ocenjevanja relativne prevlade teh frekvenc v posameznih časovnih okvirih. Pomembna lastnost tega algoritma je, da deluje neodvisno od števila virov in ne izloča najdenih poudarjenih

frekvenc, ampak išče najbolj verjetno ravnotežje med njimi. Po pridobitvi najbolj izstopajočih frekvenc se oceni še njihova časovna stabilnost in poišče najbolj verjetne kandidate za glavno melodijo z uporabo agentov za sledenje.

Leta 2004 so na univerzi Pompeu Fabra v Barceloni v okviru konference ISMIR (International Society of Music Information Retrieval) organizirali prvo mednarodno tekmovanje v opisovanju zvoka ADC 2004 (Audio Description Contest), katerega del je bilo tudi tekmovanje v iskanju melodije (poleg tekmovanj v prepoznavanju žanra, izvajalca, tempa in ritma). Namen tega je bila predvsem primerjava najboljših algoritmov in trenutne tehnologije na tem področju, konferenca pa sicer obsega precej širše področje informatike v povezavi z glasbo. Ker je bil odziv na tekmovanje zelo pozitiven, so naslednje leto v laboratoriju IMIRSEL na ameriški Univerzi v Illinoisu organizirali samostojno mednarodno tekmovanje MIREX 2005 (Music Information Retrieval Evaluation eXchange), v katerega so dodali še več novih kategorij, razširili testne baze podatkov in od takrat ga prirejajo vsako leto. Tekmovanje je že od začetka odprtega tipa, kar pomeni, da lahko vsak prispeva svoj algoritem, ki ga po vnaprej določenem postopku pripravi, pošlje in organizatorji ga preizkusijo na svojih podatkovnih bazah. Število kategorij je naraslo s pet v letu 2004 na sedemnajst v letu 2009, od tega je bilo iskanje melodije del tekmovanja vsakič razen leta 2007. Prav tako je naraslo število algoritmov na tekmovanju s štirih na dvanajst, torej se očitno za to raziskovalno področje zanima vedno več ljudi. Več o zgodovini tekmovanja je v svojem članku [2] napisal eden od članov laboratorija IMIRSEL, J. Stephen Downie.

Od prvotne množice testnih podatkov leta 2004 (20 glasbenih posnetkov dolžine okoli 20s) je bilo predlaganih že več novih. Od leta 2005 naprej se (poleg različnih drugih) vedno uporablja množica 25 posnetkov iz različnih žanrov, ki pa žal ni prosto dostopna. To je razumljivo, saj je možno ob dostopnih podatkih algoritem prilagoditi in naučiti dobrega prepoznavanja, kar je tudi opaziti v rezultatih tekmovanj. Uspešnost algoritmov na prosto dostopni množici iz leta 2004 namreč iz leta v leto narašča, medtem ko uspešnost na skriti množici MIREX 2005 ostaja na približno enakem nivoju (slika 1.1).

Goto je izmed vseh tekmovanj MIREX sodeloval le v letu 2005, kjer je bil v določanju absolutne in kromatične višine zelo uspešen [12], a po skupni oceni slab predvsem zaradi tega, ker njegov algoritem ne prepoznava mest, kjer melodija ni prisotna. Goto algoritma žal ni vidno razvijal naprej, kljub temu, da bi zaključni del sledenja z agenti lahko nadomestili tudi z drugimi (morda naprednejšimi) metodami. V tej diplomski nalogi so preizkušene možne izboljšave algoritma PreFEst. Izboljšave temeljijo predvsem na zamenjavi sistema agentov z dodatno obdelavo izstopajočih vrhov v frekvenčnem prostoru,



Slika 1.1: Skupne točnosti najboljših algoritmov na testnih množicah preteklih tekmovanj. Uspešnost na prosto dostopni množici ADC 2004 vidno narašča, vsaj deloma najbrž zaradi prilagojenosti algoritmov. Leta 2005 se na tekmovanju množica ADC 2004 sicer ni uporabila, vrednost je približna uspešnost takratnega algoritma Karin Dressler (ki redno zelo uspešno sodeluje na tekmovanjih), prevzeta iz njenega članka [3].

njihovim združevanjem preko sosednih časovnih okvirov in s tem gradnjo sledi, ter iskanju najboljše kombinacije sestavljenih sledi po sistemu ocenjevanja večih kriterijev. To vključuje iskanje najboljših naslednikov sledi ter izravnavo hipoteze glede na povezanost sledi z nasledniki, izločanje odstopajočih vrednosti, prepoznavanje in izračun prehodov, določanje frekvenčnih vrednosti, natančnejših od frekvenčne ločjivosti spektra, poskusil pa sem odpraviti tudi glavno slabost Gotovega algoritma na tekmovanju MIREX 2005 - nezmožnost ugotavljanja prisotnosti melodije.

Poglavje 2

Algoritem

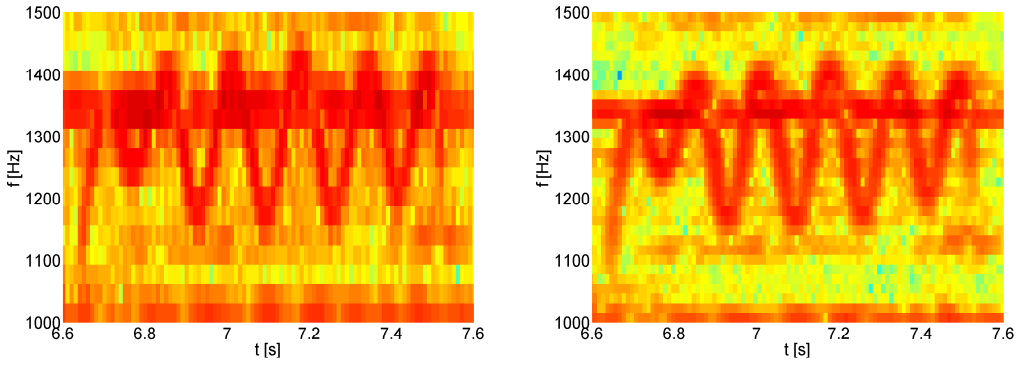
Algoritem, ki sem ga razvil in preizkusil v okviru tega diplomskega dela, za osnovo uporablja prvi in drugi del Gotovega algoritma PreFEst, brez implementacije agentov za sledenje vrhov. Namesto tega po začetni eliminaciji neperspektivnih vrhov v vhodni množici začne zanko, v kateri vrhove postopoma združuje v sledi. Po zadostnem številu iteracij se množica sledi dodatno obdela in v njej poišče najprimernejše, iz katerih se sestavi končni ugotovljeni zapis melodije v vhodnem posnetku.

2.1 Algoritem PreFEst

Masataka Goto je razvil algoritem PreFEst (Predominant-F0 Estimation) [9], ki iz vhodnega glasbenega posnetka razbere glavno melodijo in/ali basovsko linijo. Glavna ideja algoritma je predpostavka, da je posnetek sestavljen iz vseh možnih harmoničnih struktur pri vseh možnih osnovnih frekvencah, a z neznanimi utežmi. Tako je v osnovi obdelava neodvisna od števila zvočnih virov (inštrumentov ali glasu). Algoritem poskuša ugotoviti uteži in obliko različnih harmoničnih struktur z oceno MAP (maksimalna aposteriorna verjetnost). Harmonična struktura z največjo utežjo se predpostavi kot glavni vir in prevladujoča frekvenca F0, ugotovljenim frekvencam v posameznih časovnih okvirih pa se nato ugotovi še stabilnost z uporabo agentov, ki vrhovom sledijo v zaporednih časovnih okvirih. Natančnejši opis korakov algoritma sledi v naslednjih podpoglavjih.

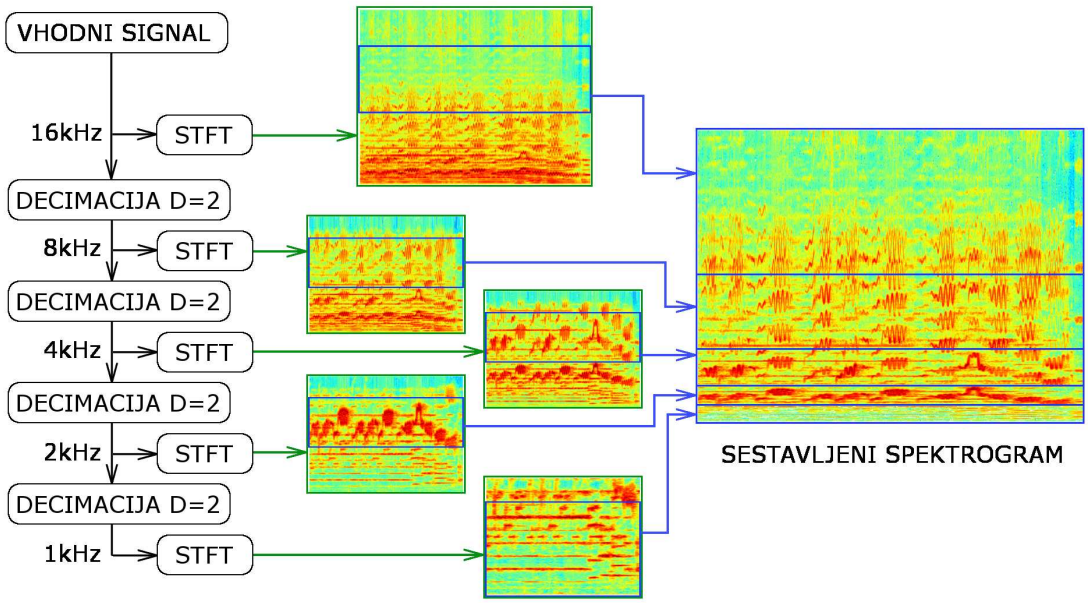
2.1.1 Predobdelava

Algoritem najprej pridobi magnitude frekvenčnih komponent v več časovnih odsekih (z ločljivostjo 10ms) vhodnega signala z uporabo kratkočasovne Fourierove transformacije (ang. Short-time Fourier Transform - STFT). Obdelava je razdeljena na večstopenjsko množico filtrov, kjer se za določanje magnitud nižjih frekvenc uporabi vhodni signal z nižjo frekvenco vzorčenja f_s (dobimo ga z uporabo decimacije). To je koristno, ker s signalom na originalni frekvenci vzorčenja dobimo razmeroma slabo ločljivost frekvenčnih košev, ki še posebej škoduje pri nižjih frekvencah (slika 2.1), saj signal pozneje obdelujemo na logaritemski lestvici. Boljšo frekvenčno ločljivost sicer dobimo tudi z uporabo večjega okna pri STFT, a to prinese slabšo natančnost na časovni skali. Zato se iz vsakega izhoda Fouriereve transformacije uporabi le zgornja polovica frekvenčne lestvice (ki jo dobimo le pri dovolj visoki frekvenci vzorčenja), za določanje magnitud nižjih frekvenc pa se postopoma uporabijo transformacije signalov z nižjimi frekvencami vzorčenja (slika 2.2). Natančneje, vzame se območje od 40% do 80% frekvenčne lestvice na posamezni stopnji ([0.4*f_s/2 , 0.8*f_s/2]), saj je vrh spektrograma blizu $f_s/2$ neprimeren za obdelavo. Goto je v svojem članku na vsaki stopnji sicer uporabil področje od 45% do 90%, a sem opazil, da so pri tako določenem rezu meje med posamez-



(a) 16000Hz (b) 8000Hz

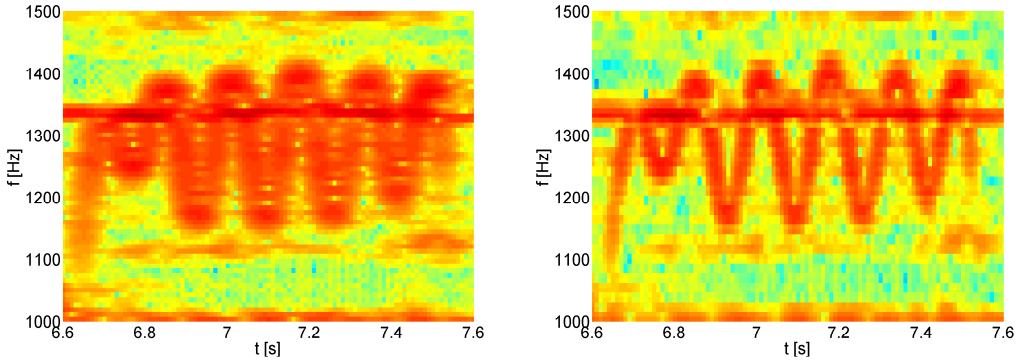
Slika 2.1: Primerjava spektrogramov pri različnih frekvencah vzorčenja vhodnega signala. Obe sliki pripadata istemu delu spektra testne datoteke opera_fem4.wav. Oba spektra imata isto časovno ločljivost, spekter (b) ima zaradi dvakrat nižje frekvence vzorčenja dvakrat večjo frekvenčno ločljivost, a do nižje maksimalne frekvence.



Slika 2.2: Sestava spektrograma iz večstopenjske množice filtrov. Na vsaki stopnji filtrov se vzame območje od 40% do 80% frekvenčne lestvice spektrograma, razen spektrograma v zadnji stopnji pri frekvenci vzorčenja 1kHz, kjer se vzame celotno območje, nižje od 80%. Posamezni deli se sestavijo v končni spektrogram, ki se poda na izhodu za nadaljnjo obdelavo.

nimi deli v izhodnem spektrogramu zelo opazne. Do tega pride zaradi nizkoprepustnega (lowpass) filtra, ki se uporablja pri decimaciji. Algoritem sem implementiral v Matlabu, kjer je privzet filter v vgrajeni funkciji za decimacijo Čebišev filter tipa 1 z mejno frekvenco 0,8*f_s/2. Če bi določili območje po Gotovem predlogu kot [0.45*f_s/2 , 0.90*f_s/2], bi to vsebovalo že del prehodnega območja v filtru in bi zato lahko privedlo do slabših rezultatov, manjši premik območja pa je zanesljivejši in manj zahteven kot ponovna implementacija decimacije.

Ob pregledovanju končnih sestavljenih spektrov pri različnih vhodnih datotekah sem opazil, da pri melodijah z zelo strmimi prehodi prihaja do slabše razpoznavnosti teh prehodov. Pri močnem vibratu opernih pevcev se na primer zaradi širine okna pri STFT vrednosti spektra po časovni osi razmažejo. Sosednji prehodi se lahko celo združijo in tako dobimo celotno območje visokih vrednosti, v katerem je pravo sled težje prepoznati (slika 2.3 a). Nasprotno od teh



(a) osnovni spekter (b) združena spektra z okni različnih širin

Slika 2.3: Primerjava osnovnega sestavljenega spektra po Gotovi metodi in spektra pri združevanju spektrov, pridobljenih s STFT širine 512 in 256 vzorcev. Obe sliki pripadata istemu delu spektra testne datoteke opera_fem4.wav. Spekter a) ima sicer dobro frekvenčno ločljivost, a je točnost zaradi razmazanih vrednosti slaba, pri spektru b) pa je sled veliko lažje razbrati. Primerjava s sliko 2.1 b) kaže, da je združeni spekter ostrejši in v tem pogledu pridobi s povečano frekvenčno ločljivostjo.

nejasnih visokih vrednosti pa so pri posameznih prehodih med toni (torej ne vibratu) lahko vrednosti v spektru precej šibkejše kot pri vzdrževanih tonih in jih nadaljnja obdelava lahko zanemari kot nezanimive vrhove, še posebej zaradi tonov, med katerimi je prehod, ker se njihove sledi zaradi razmazanih vrednosti pogosto prekrivajo. Če uporabimo ožje okno pri STFT, prehodi postanejo ostrejši, a izgubimo na frekvenčni ločljivosti. Zaradi te dvojnosti sem se odločil, da poskusim ostrejši spekter in spekter z večjo frekvenčno ločljivostjo združiti in preveriti, če bodo dobljeni rezultati boljši od siceršnjih. To sem izvedel tako, da sem v spektru z oknom 256 vzorcev (in ostrejšimi prehodi) vsako vrstico normaliziral na vsoto 1, vsak sosednji par vrstic interpoliral in tako dobil navidezno večjo frekvenčno ločljivost spektra, zatem pa sem te normalizirane vrstice množil z osnovnim spektrom, pridobljenim z oknom 512 vzorcev. Takšno združevanje sicer res lahko privede do rahlih lokalnih nepravilnosti na mestih, kjer se različne sledi križajo (kot je opazno na sliki 2.3, kjer vodoravna sled seka sled vibrata), a v splošnem je končni rezultat ostrejši in navidezno združuje prednosti obeh spektrov. Vpliv tega združevanja je prikazan v primerjavi končnih rezultatov.

Poleg združevanja spektrov sem preizkusil tudi razširitev množice filtrov

z dodatno decimacijo in filtrom pri frekvenci vzorčenja 500Hz. Frekvenčni razpon melodije, v katerem pregledujemo, je od 130,8 Hz navzgor, sicer najnižji filter pa analizira frekvenčni spekter do 400Hz, kar pomeni, da bi morda lahko pridobili pri analizi nižjih frekvenc z dodatno stopnjo, ki bi določala spekter pod 200Hz. Preizkusil sem delovanje algoritma z in brez dodatne stopnje, vpliv te spremembe prav tako komentiram v poglavju namenjenem končnim rezultatom.

Ker iščemo glavno melodijo posnetka in je ta običajno v srednjem frekvenčnem prostoru, v tem koraku tudi omejimo frekvenčni pas s pasovnoprepustnim (ang. bandpass) filtrom. Za ta filter sem določil (kot v Gotovem članku [9]) spodnjo mejo 3600 in zgornjo 8400 centov. Ti meji ustrezata frekvencama 130,8 Hz in 2093 Hz, pretvorjenimi na logaritemsko lestvico, v kateri je razlika med poltonoma vedno 100 centov. Takšna oblika je bližje človeškemu sluhu, katerega občutljivost za frekvence prav tako narašča po logaritemski lestvici, hkrati pa bolj ustreza tudi standardnim zapisom glasbe v tonih, ob čemer bi bila možna dodatna obdelava vhodnega signala, če predpostavimo, da so glasbeni inštrumenti uglašeni in ima vokal dovolj dober posluh (kar pa še posebej pri amaterskih posnetkih ne moremo vedno zagotavljati). Pretvorba iz frekvence v Hz v frekvenco v centih poteka po enačbi 2.1.

$$f_{\text{cent}} = 1200 \cdot \left(\log_2 \left(\frac{f_{\text{Hz}}}{13{,}75 \text{ Hz}} \right) - 0{,}25 \right) \tag{2.1}$$

2.1.2 Jedro - Ocenitev funkcije verjetnostne gostote F0

V tem delu algoritma se za vsako možno osnovno frekvenco melodije F0 poskuša poiskati primeren model harmoničnih komponent in njegovo utež, ki naj bi predstavljala poudarjenost te F0 v določenem časovnem okviru. V Gotovem algoritmu je vsak model podan s tremi parametri - F, m in $\mu^{(t)}(F, m)$. F predstavlja osnovno frekvenco modela, m tip modela (če uporabljamo več začetnih oblik), $\mu^{(t)}(F, m)$ pa obliko modela, ki je pravzaprav množica uteži vseh harmoničnih komponent v modelu. Goto je prvotno preizkušal dva začetna tipa (slika 2.4). Oba sta imela izračunano obliko, tj. uteži vseh harmoničnih komponent, po enačbi:

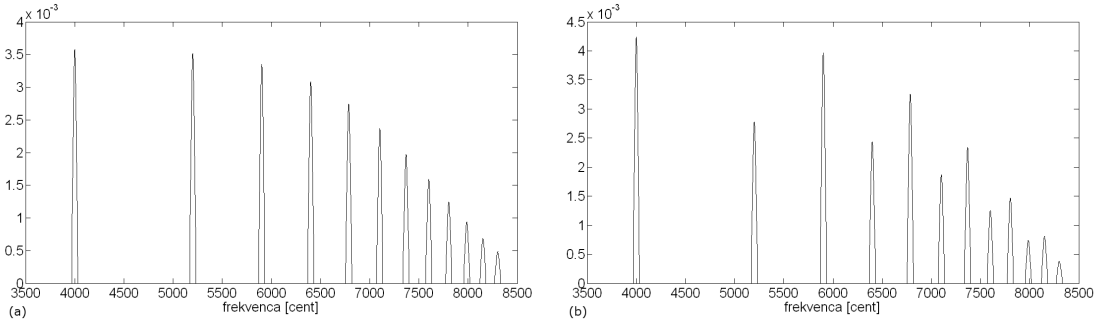
$$c_{0i}^{(t)}(h|F, m) = \alpha_{i,m} g_{m,h} G(h; 1, U_i), \tag{2.2}$$

Pri tem je $\alpha_{i,m}$ normalizacijski faktor, da je vsota vseh harmoničnih komponent enaka 1, $g_{m,h}$ dodaten faktor, po katerem se razlikujeta Gotova tipa, $G(h; 1, U_i)$ pa določa, da višje harmonične komponente postopoma padajo po

Gaussovi krivulji s sigmo U_i (ki je v Gotovem primeru enaka 5,5):

$$G(h;1,U_i) = \frac{1}{\sqrt{2\pi {U_i}^2}} e^{-\frac{(h-1)^2}{2U_i^2}} \tag{2.3}$$

Faktor $g_{m,h}$ je pri prvem modelu vedno enak 1, pri drugem pa je za vsako drugo komponento enak 2/3. V celotnem posameznem modelu, ki ga določajo frekvenca, tip in oblika, imajo vsi vrhovi, ki predstavljajo harmonične komponente, obliko Gaussove krivulje s standardnim odklonom 17 na frekvenčni lestvici v centih in povprečjem pri frekvenci, ki ustreza posamezni harmonični komponenti.



Slika 2.4: Začetni obliki tonskih modelov z osnovnima frekvencama 4000 Hz. Višje harmonične komponente padajo po Gaussovi krivulji, pri modelu (b) je vsaka druga pomnožena še s faktorjem 2/3.

V jedru se nato predpostavi, da je frekvenčni spekter v posameznem časovnem okviru nastal kot utežena vsota modelov vseh možnih osnovnih frekvenc. Na tem mestu je možno uporabiti tudi apriorno distribucijo najbolj verjetnih osnovnih frekvenc za nastavitev začetnih uteži posameznim modelom. Program nato poskuša prilagoditi uteži ter oblike modelov do najbolj verjetne kombinacije z oceno maksimalne aposteriorne verjetnosti MAP in uporabo algoritma EM (Expectation-Maximization), s katerim preko izpeljave (natančneje v Gotovem članku [9]) pridemo do končnih enačb za pridobitev novih uteži in oblik modelov (enačbi 2.4 in 2.5).

$$\overline{w_{\mathrm{ML}}^{(t)}(F,m)} = \int_{-\infty}^{\infty} p_{\psi}^{(t)}(x) \frac{w'^{(t)}(F,m)p(x|F,m,\mu'^{(t)}(F,m))}{\int_{\mathrm{Fl}}^{\mathrm{Fh}} \sum_{\nu=1}^{\mathrm{M_i}} w'^{(t)}(\eta,\nu)p(x|\eta,\nu,\mu'^{(t)}(\eta,\nu))d\eta}\, dx \tag{2.4}$$

$$\overline{c_{\mathrm{ML}}^{(t)}(h|F,m)} = \frac{1}{\overline{w_{\mathrm{ML}}^{(t)}(F,m)}} \int_{-\infty}^{\infty} p_{\psi}^{(t)}(x) \frac{w'^{(t)}(F,m)p(x,h|F,m,\mu'^{(t)}(F,m))}{\int_{\mathrm{Fl}}^{\mathrm{Fh}} \sum_{\nu=1}^{\mathrm{M_i}} w'^{(t)}(\eta,\nu)p(x|\eta,\nu,\mu'^{(t)}(\eta,\nu))d\eta}\, dx \tag{2.5}$$

Ti dve enačbi iterativno izvajamo in tako pridobivamo vedno natančnejša ravnotežja med modeli. Ob preizkušanju algoritma sem ugotovil, da je za dobro ravnotežje dovolj že 5 iteracij. Gotov članek sicer vključuje uporabo še dveh enačb, ki pa sta potrebni le, če imamo apriorne podatke o distribuciji uteži, teh pa ne moremo določiti, če hočemo obdržati splošno uporabnost algoritma.

2.2 Predobdelava in eliminacija vrhov

Množica vrhov, ki jo dobimo iz algoritma PreFEst, je običajno zelo obsežna, zato jo je v namen doseganja krajših časov izvajanja dobro omejiti. V tej stopnji algoritma se tako v vsakem časovnem okviru množico najprej omeji na lokalne maksimume, nato pa v ob zbiranju največjih vrhov izločimo vse tiste, ki so manjši od 10% uteži največjega vrha, ali manjši od 5% magnitudne vrednosti v spektrogramu, ki jo ima vrh z največjo utežjo. Tako se izloči veliko vrhov, ki najverjetneje ne predstavljajo zelo izstopajočega zvoka in jih v gradnji hipoteze tako ne potrebujemo.

2.3 Gradnja sledi

Melodija (tok frekvence F0) je večinoma zvezna po času. Nezveznosti so le ob prehodih med toni, a tudi na teh mestih je v spektrogramih pogostokrat (še posebej pri vokalnih melodijah) opaziti ozke pasove, ki v resnici zvezno povezujejo posamezne tone, a jih je kot izstopajoče vrhove precej težje detektirati.

Kot izhod prejšnje stopnje dobimo množico vrhov, ki zaenkrat še niso povezani, so pa (v vsakem časovnem okviru posebej) urejeni po velikosti. Za učinkovito odločanje in iskanje najboljših kandidatov za končno hipotezo je dobro te vrhove združevati po času v sledi, ker nam to omogoči, da melodijo sestavljamo iz večjih enot in tako dobimo enakomernejšo končno hipotezo. Z združevanjem ocen posameznih vrhov izravnamo tudi razlike med vrhovi, ki pripadajo isti sledi, a morda zaradi motenj v posnetku (kot "motnje" tu lahko obravnavamo tudi slučajno izstopajoče zvoke drugih inštrumentov, ki pa ne pripadajo melodiji) v določenih časovnih okvirih dobijo precej različne ocene.

Vsako sled predstavlja objekt, v katerem so shranjeni začetni časovni okvir, končni časovni okvir, povprečna ocena sledi in seznam vrednosti v posameznih časovnih okvirih. Delovanje te stopnje – združevanje vrhov v sledi – poteka iterativno po korakih, opisanih v naslednjih podpoglavjih.

2.3.1 Obdelava novih vrhov

Na začetku vsake iteracije se obdelajo vrhovi, ki so po velikosti na istem mestu kot je številka trenutne iteracije (v tretji iteraciji se tako npr. obdelajo vrhovi, ki so bili v posameznih časovnih okvirih tretji največji – v nadaljevanju je za to uporabljen izraz velikostni red vrha) in sicer tako, da se iz prvega v časovnem redu ustvari nova sled, naslednji pa se dodajajo k tej sledi, dokler se od zadnjega na sledi še razlikujejo za manj kot 100 centov. To je relativno velika razlika (v primerjavi z 25 centi pri ocenjevanju točnosti), a se izkaže, da so vrhovi, ki so blizu po velikostnem redu in hkrati tudi blizu po frekvenci, a ne sodijo na isto sled, zelo redki, zato sem dopustil večje razlike z namenom lažjega povezovanja prehodov. Začetna ocena sledi je enaka velikostnemu redu vrha, ki pa se pozneje lahko spreminja ob združevanju z drugimi sledmi. Iz tega je razvidno, da je nižja ocena boljša.

Če se trenutni vrh od zadnjega vrha v trenutni sledi razlikuje za več kot 100 centov, se z njim ustvari nova sled, katero nadalje uporabimo za dodajanje sledečih vrhov. Z zaključkom prejšnje sledi se preveri njena stabilnost – če je sled daljša od 10 časovnih okvirov, se shrani v seznam stabilnih sledi, sicer v seznam nestabilnih. Kot bo razvidno iz naslednjih korakov v iteraciji, se seznama različno uporabljata.

2.3.2 Iskanje ujemanj nestabilnih sledi s stabilnimi sledmi

V tem koraku se pregleda celoten seznam nestabilnih sledi in primerja frekvenčne vrednosti teh sledi z vrednostmi stabilnih sledi, ki vsebujejo iste časovne okvire. Stabilne sledi namreč lahko vsebujejo dopolnjujoče vrednosti, ki ob rasti posamezne sledi niso pripadale konkretnim vrhovom, ampak so bile npr. izračunane za zapolnitev prehoda med dvema stabilnima sledema, ki sta se združili. S poznejšimi iteracijami se lahko tako ustvarijo nestabilne sledi s temi manjkajočimi vrednostmi, ki jih je dobro združiti z dopolnjujočimi.

Če se frekvenčne vrednosti v vseh okvirjih nestabilne sledi od ujemajočih vrednosti določene stabilne sledi razlikujejo za manj kot 25 centov, se nestabilna sled prepozna kot ujemajoča in se stabilno sled prilagodi k njenim vred-

nostim, tako da se izračuna povprečne frekvenčne vrednosti med obema sledema, ki se nato zapišejo v stabilno sled namesto prejšnjih, nestabilna sled pa se izbriše.

Ob združevanju sledi se popravi tudi ocena stabilne sledi in sicer se izračuna kot uteženo povprečje obeh sledi glede na delež popravljenih vrednosti. Če je bila npr. spremenjena tretjina vrednosti, se za izračun povprečja uporabi utež 1/3 za oceno nestabilne in 2/3 za oceno stabilne sledi.

2.3.3 Dodajanje nestabilnih sledi k stabilnim sledem

Glavna naloga iteracij je, da spajajo sledi, ki spadajo skupaj. Ujemanje je najlažje ugotoviti glede na položaj v času – kjer se ena sled konča, se naslednja sled (če ta obstaja) začne, vendar se ti meji običajno ne ujemata popolnoma. Spektrogram se namreč računa s prekrivajočimi kratkočasovnimi Fourierovimi transformacijami in zaradi prekrivanja oken pride tudi do razmazanih vrednosti glede na časovno os. Najprej se poskuša dodati nestabilne sledi na konce stabilnih sledi. Pri tem se primeren par preveri glede na sledeče kriterije:

- razdaljo med koncem ene in začetkom druge – maksimum te se veča iz iteracije v iteracijo, tako da se najprej spojijo najbolj ujemajoče sledi, odvisen pa je tudi od dolžine posamezne stabilne sledi
- razliko med povprečnima ocenama sledi
- razliko med mejnima frekvenčnima vrednostima

Zadnja dva kriterija sta pravzaprav združena v enem pogoju, ki dopušča večje razlike med frekvencami, če sta povprečni oceni bližje:

$$(|f_1 - f_2| + 1) \cdot (|o_1 - o_2| + 1) < 100 \tag{2.6}$$

2.3.4 Združevanje preostalih nestabilnih sledi

V tem koraku se, podobno kot v prejšnjem, med seboj združujejo nestabilne sledi, le da so tu kriteriji strožji - desna stran enačbe 2.6 je zmanjšana na 50, sledi se ne smeta prekrivati, največja razdalja v času med njima pa je za pol manjša, s tem da tu za razliko od prejšnjega koraka ni pomembna dolžina samih sledi.

2.3.5 Združevanje stabilnih sledi

Po združevanju nestabilnih sledi se za ujemanje preveri še pare stabilnih sledi, s tem da se tu desna stran enačbe 2.6 zmanjša na 1.5, razlika v frekvencah na levi pa se koreni. S tem se združevanje stabilnih sledi omeji le na tiste pare, ki so si po oceni zelo blizu. Za končno sestavljanje hipotez je namreč precej slabše, če pri tem uporabljamo napačno združene stabilne sledi, kot pa če moramo obdelati večje število sledi. V prvem primeru se je težje izogniti napačni hipotezi, medtem ko v drugem primeru le povečamo čas procesiranja, ki pa nam tu ni tako pomemben, dokler je v razumnih mejah.

2.4 Iskanje najboljših naslednikov

Pridobljene sledi v prejšnjih korakih so zvezne po času in načeloma tudi v frekvenčnem prostoru, razen manjših skokov, ki jih lahko dovolj gotovo vzamemo kot del sledi. Vendar pa na koncu kot izhod algoritma hočemo podati eno sled, ki bo predstavljala melodijo v celotnem glasbenem izseku (razen morebitnih presledkov brez melodije). To pomeni, da je potrebno v naslednjem koraku sledi iz prejšnjega koraka združiti v najbolj verjetno kombinacijo, ki pa najverjetneje ni več zvezna.

Iskanje najprimernejše kombinacije bi lahko implementirali kot iskanje kritične poti, ki poišče zaporedje sledi z najboljšo povprečno oceno. Ker pa je možnih sledi lahko tudi več tisoč, po času pa si ne sledijo tesno druga za drugo, (začetek naslednje je lahko celo koncem prejšnje), dobimo časovno zelo zahteven problem, ki se povečuje eksponentno z velikostjo problema. Za pohitritev problema je tako dobro množico naslednikov vsake sledi urediti in čim bolj omejiti, najbolje pa je, če najdemo enega samega dovolj gotovega naslednika, s katerim lahko sled tudi združimo in tako zmanjšamo velikost problema. Iskanje naslednikov sem zasnoval tako, da se za vsako sled v primerno veliki okolici poišče vse možne naslednike in se jih oceni glede na več kriterijev. Te ocene se v obliki kazni za neskladnost med sledmi sešteje in izbere le naslednike z dovolj nizko oceno, ki se jih nato shrani za nadaljnjo obdelavo. Kriteriji, po katerih se kaznuje možne naslednike, so:

- razdalja v časovnih okvirih med koncem osnovne sledi in začetkom naslednika
- dolžina naslednika - če je naslednik tako kratek, da je njegov konec oddaljen manj kot 10 časovnih okvirov od konca osnovne sledi, se ga dodatno kaznuje

- razlika v povprečni oceni sledi
- razlika v frekvenci - če sta mejni frekvenci oddaljeni za manj kot 50 centov, se razdaljo kaznuje enostavno linearno, sicer pa se kaznuje po dveh kriterijih:
  - razliki v tonih - razdalji med frekvencama, zaokroženi na 100 centov
  - nekonsistentnosti v tonih - kazen glede na to, koliko se razlika med frekvencama razlikuje od najbližjega večkratnika 100 centov (kazen, če prehod "ni po posluhu")

Ocene po posameznih kriterijih se uteži in sešteje v eno oceno, ki ne sme presegati določenega praga, da se naslednik sprejme. Uteži vseh ocen, velikost preiskovalnega prostora za naslednike in zgornji prag skupne ocene sem določil eksperimentalno s preverjanjem uspešnosti na učni množici. Preverjanje sem zaradi siceršnje nepraktičnosti in zahtevnosti preiskovanja večparametričnega prostora izvajal le na točnih sledeh (ki so se ujemale z referenčnimi podatki), uspešnost pa ocenjeval glede na število najdenih točnih naslednikov in sicer le tistih, ki so bili izbrani kot prvi ali drugi glede na skupno oceno. Parametrični prostor sem sicer preiskal le omejeno, a kljub temu precej izboljšal osnovno iskanje prvotnih nastavitev z enakovrednimi utežmi. Sprva je algoritem našel primernega naslednika (med vsemi najdenimi kandidati) v povprečju v 83,2% primerov, v 24,3% je bil primeren naslednik kandidat z najvišjo oceno, v 48,9% primerov pa je bil primeren kandidat vsaj eden izmed prvih dveh. S popravljenimi utežmi se je uspešnost iskanja primernih kandidatov zvišala na 97,4%, s 83,9% in 96,9% za prvega in prva dva kandidata po oceni. Tu velja opomniti, da je bilo to ocenjevanje opravljeno na zelo omejeni množici točnih sledi in naslednikov, kar pomeni, da so te verjetnosti v splošnih (netočnih in neidealnih) primerih precej manjše. Pridobljene najboljše ocenjene naslednike sem uporabil v enem izmed korakov pridobivanja končne hipoteze.

2.5 Metode pridobivanja končne hipoteze

2.5.1 Osnovna hipoteza

Začel sem s preprosto metodo, ki za vsak časovni okvir izmed vseh sledi, ki imajo vrednost v tem okviru, poišče tisto z najboljšo povprečno oceno, ter njeno vrednost za ta okvir vključi v končno hipotezo. Ob najboljši hipotezi se shranijo tudi vse nadaljnje s slabšimi ocenami, ki se lahko uporabijo pozneje ob izboljševanju hipoteze.

2.5.2 Odpravljanje odstopanj

V tem koraku se poskušajo najti in odpraviti vrednosti v hipotezi, ki močno odstopajo od okolice (imajo frekvenčne vrednosti bistveno višje ali nižje od večine sosedov). To poteka tako, da se preveri vpliv posamezne vrednosti na povprečje okolice. Za vsak časovni okvir se poskuša najti okno velikosti 100 okvirov, ki vsebuje preverjani okvir in kjer se ta najmanj razlikuje od povprečja. Če je razlika njegove vrednosti od povprečja, deljena z velikostjo okna, manjša od 5, se ta vrednost sprejme kot primerna. Če okna, kjer bi to veljalo, v vsej okolici ne najdemo, se poskuša izmed sekundarnih hipotez v tem časovnem okviru najti primernejšo, ki nato zamenja prvotno. Ta korak se lahko ponovi večkrat za odpravo morebitnih dodatnih odstopanj.

2.5.3 Izravnava hipoteze z najboljšimi nasledniki

Odpravljanje odstopanj v prejšnjem koraku je le delno uspešno (ne odpravi daljših zveznih območij napačnih hipotez), zato se v tem koraku poskuša še uravnotežiti hipotezo glede na ocenjene najboljše naslednike posameznih delnih sledi. Te naslednike dobimo po postopku, opisanem v enem prejšnjih poglavij. Nato se izračuna matrika naslednikov (velikosti N_{tr}xN_{tr}, kjer je N_{tr} število vseh sledi), tako da za vsako sled pridobimo verigo najboljših naslednikov (tj. vseh nadaljnih najboljših sledi do konca hipoteze oz. dokler se veriga ne prekine, ker katera izmed sledi nima naslednika), v matriko pa se v vrstico vsake sledi na mesta vseh njenih najboljših naslednikov vpiše 1.

Z izračunano matriko nato izvedemo glasovanje, tako da vsaka sled, ki je trenutno v hipotezi, glasuje za svoje naslednike. Poleg tega še vsaka sled, ki ima med svojimi nasledniki sledi iz hipoteze, dobi dodatek k svojim glasovom, ki je sorazmeren dolžini hipotezne sledi in obratno sorazmeren z razdaljo med sledema. Nato vse sledi (tudi tiste, ki niso v hipotezi) uredimo po skupnem številu glasov.

Število glasov za vsako sled nato uporabimo podobno kot oceno sledi pri gradnji osnovne hipoteze, tako da v vsakem časovnem okviru sled, ki ima največ glasov, poskuša zamenjati sled, ki je trenutno v hipotezi. Pri tem mora za zamenjavo ta sled izpolnjevati vsaj enega izmed treh pogojev:

- Imeti povprečno utež (ustreznih uravnoteženih modelov iz prvih stopenj algoritma) večjo od štirikratnika povprečne uteži trenutne sledi v hipotezi
- Imeti povprečno vrednost vrhov v spektrogramu večjo od osemkratnika povprečnih vrednosti trenutne sledi

- Hkrati imeti povprečno utež večjo kot 80% povprečne uteži trenutne sledi in povprečno vrednost vrhov v spektrogramu večjo kot 10% povprečnih vrednosti trenutne sledi

Meje sem postavil glede na analizo običajnih oblik frekvenčnih spektrov, modelov in njihovih uteži. Velikosti vrhov osnovnih harmoničnih komponent v frekvenčnih spektrih so lahko namreč precej manjše od velikosti višjih komponent, zato mora za zadostitev samo enega pogoja (uteži ali vrednosti v spektrogramu) zamenjava imeti veliko večje vrednosti od trenutne hipoteze. Meji pri tretjem (kombiniranem) pogoju približno ustrezata spodnjim mejnim vrednostim osnovnih harmoničnih komponent (ki imajo vrednosti v frekvenčnem spektru lahko majhne, uteži pa so običajno še vseeno velike).

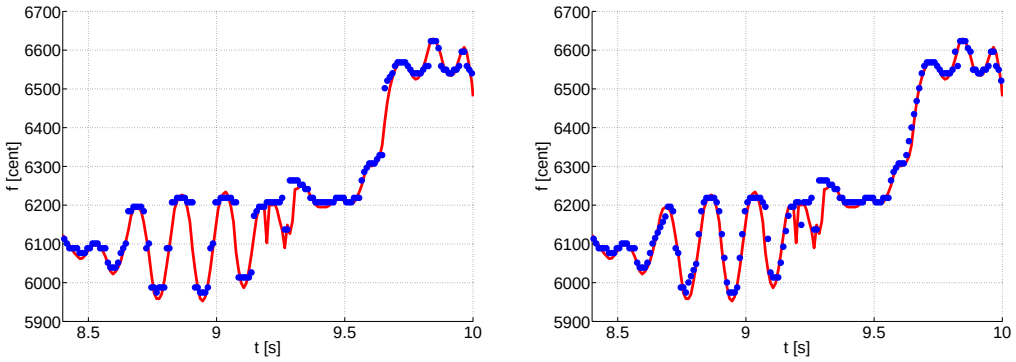
2.5.4 Prepoznavanje prehodov

Zaradi razmazanih vrednosti pri STFT je vključevanje prehodov v sledi težavno. Magnitude spektra so pri prehodih običajno precej manjše kot pri vzdrževanih tonih, ki se jim frekvenca ne spreminja, vključevanje šibkejših vrhov k močnejšim sledem pa lahko privede do združevanja glavne melodije z glasbeno podlago. To je zelo nezaželeno, saj želimo, da sledi čim bolj gotovo predstavljajo spektralne vrhove, ki pripadajo istemu viru. Bolje je, da na razdrobljenih sledeh uporabimo metodo, ki posebej presodi, če med njimi obstajajo prehodi. Ker so prehodi običajno več ali manj ravni, sem se odločil, da za ta namen preizkusim metodo za iskanje črt s področja računalniškega vida - Houghovo transformacijo.

Houghovo transformacijo sta leta 1971 razvila Richard Duda in Peter Hart po patentu Paula Hougha. Podroben opis transformacije se nahaja v njunem članku iz leta 1972 [4], osnovna ideja transformacije pa je, da vsaka točka v sliki, ki bi bila lahko del kakšne črte (npr. rob ali lokalni maksimum), glasuje za vse možne črte, katerim bi lahko pripadala. Črte, ki dobijo največ glasov, so najverjetneje tudi prisotne in opazne v sliki. V splošnem se lahko Houghova transformacija uporablja tudi za kroge in druge geometrijske like, a v našem primeru iščemo le črte.

Transformacijo sem uporabil tako, da za vsak par sosednjih časovnih okvirov, ki se razlikujeta za več kot 25 centov, najprej določim območje možnega prehoda. To območje je široko največ 20 časovnih okvirov (200ms) oz. največ polovico sledi, katerim pripadata oba časovna okvira. Na tem območju nato v posameznih vrsticah detektiram lokalne maksimume, ki so maksimumi tudi v oknih širine 7 (tako se znebimo manjših šibkejših maksimumov). Ti naj bi v večini pripadali prehodu. Na dobljenih vrhovih izvedem Houghovo

transformacijo. Najbolj izstopajočo detektirano črto podaljšam do zgornje in spodnje meje območja, saj mora prehod potekati čez celotno območje, nato pa preverim ujemanje črte z vrhovi, na katerih sem računal transformacijo. Vsi vrhovi, ki so od črte oddaljeni več kot 20% časovne širine območja, se štejejo kot neujemanja in če je teh več kot 5%-35% vseh vrhov v območju (meja je obratno sorazmerna s frekvenčno širino prehoda do 1200 centov), se prehoda ne sprejme. Prav tako se ne sprejme, če so ujemajoči vrhovi prisotni v manj kot 40%-80% vrstic (linearno sorazmerno s frekvenčno širino prehoda).



(a) (b)

Slika 2.5: Primerjava dela hipoteze testne datoteke opera_fem4.wav pred in po zaznavanju prehodov. Rdeča črta prikazuje referenčne (točne) vrednosti, modre pike pa ugotovljeno hipotezo v posameznih časovnih okvirih.

Ob analizi spektrogramov sem opazil, da so prehodi običajno bolj razpoznavni pri višjih harmoničnih komponentah (še posebej v primeru vibrata), zato sem pri zaznavanju prehodov dodal možnost, da se takrat, kadar obstajata dovolj močni magnitudi vrhov, med katerima računamo prehod, v spektrogramu tudi eno oktavo (1200 centov) višje, prehod preveri in izračuna na višji oktavi, nato pa prestavi nazaj na osnovno višino. Tako sem dosegel rahlo večjo točnost prehodov, še posebej, kadar so ti pri precej nizkih frekvencah (ki so v spektrogramu običajno bolj razmazane).

2.5.5 Ugotavljanje točnih frekvenc

Hipoteza melodije, ki jo vrne algoritem, je vezana na omejeno množico frekvenčnih košev, ki so predpisani s Fourierovim transformom z začetka obdelave. Frekvenčne vrednosti teh so razmeroma goste, a še vedno diskretne. Sicer

se zaradi združevanja sledi frekvence v hipotezi tudi oddaljujejo od vrednosti frekvenčnih košev, a to ne zagotavlja ujemanja s spektrogramom. Zato se v zadnjem koraku frekvenčne komponente iz hipoteze ponovno poišče v spektrogramu, ter se jih glede na okolico v spektrogramu poskuša premakniti na bolj verjetno vrednost. To poteka tako, da se absolutnim vrednostim frekvenčnih komponent v spektrogramu v okolici hipotezne frekvence poišče primeren polinom (kot to stori ukaz spline v Matlabu), ter v tem polinomu na večji ločljivosti poišče frekvenco, ki ustreza maksimumu tega polinoma. Frekvenco v hipotezi se nato spremeni na novo vrednost. Polinom sem računal na treh točkah.

2.6 Ugotavljanje prisotnosti melodije

Glavni vzrok za slabo skupno oceno Gotovega algoritma na tekmovanju MIREX 2005 je bila nezmožnost algoritma, da bi prepoznal območja, kjer melodija ni prisotna. Ker algoritem sicer precej dobro deluje, sem presodil, da bi bilo ugotavljanje teh območij ena izmed pomembnejših razširitev, ki bi jo bilo dobro preizkusiti, tudi če metoda za to ne bi bila zelo dobra. V ta namen sem algoritmu dodal stopnjo, ki grobo oceni prisotnost glasov (vokala in inštrumentov) v posameznih časovnih okvirih.

Ta deluje tako, da izračuna Fourierovo tranformacijo vsakega okvira posebej in nato še njeno vsoto (enačba 2.7).

$$P(t) = \sum |F(s(t))| \tag{2.7}$$

Tako dobimo neke vrste moč spektra, ki upošteva tudi periodičnost vrhov (značilno za harmonične glasove). Nato za vsako stezo v hipotezi izračuna njeno povprečno moč (enačba 2.8) in ugotovi globalni maksimum in minimum teh moči. Iz teh dveh vrednosti se določita dve meji. Prva je relativna glede na maksimum, druga pa relativna v območju med minimumom in maksimumom. Steze, ki imajo povprečje pod obema mejama, se določijo kot področja, kjer melodija ni prisotna (enačba 2.9). Obe relativni meji sem določil s testiranjem na učni množici.

$$\overline{P(T)} = \sum_{t \in T} \frac{P(t)}{L_{tr}} \tag{2.8}$$

$$v(T) = \begin{cases} 1, & \text{če } \overline{P(T)} \geq \max\limits_{T'} \overline{P(T')} * 0,42 \\ 1, & \text{če } \overline{P(T)} \geq \left(\max\limits_{T'} \overline{P(T')} - \min\limits_{T'} \overline{P(T')} \right) * 0,31 + \min\limits_{T'} \overline{P(T')} \\ 0, & \text{sicer} \end{cases} \tag{2.9}$$

Zatem se območja še povežejo, tako da se vsako zvezno območje z ugotovljeno prisotnostjo, ki je dolgo 10 okvirov ali manj (100 ms), popravi na območje brez melodije, vsako območje z ugotovljeno neprisotnostjo, dolgo 5 okvirov ali manj, pa na območje z melodijo.

Poglavje 3

Rezultati in primerjave

3.1 Ocenjevanje točnosti

Pri merjenju točnosti algoritmov za detekcijo melodije se uporablja več metod, praktično vse pa potrebujejo poleg vhodnih zvočnih datotek še referenčne zapise, ki predstavljajo točno detekcijo na vhodnih datotekah in s katerimi lahko primerjamo hipoteze algoritmov. Referenčni zapisi so običajno ustvarjeni ali iz simboličnega zapisa umetno ustvarjene izhodne datoteke (npr. standardni zapis MIDI) ali iz večslednega zapisa skladbe, kjer ima glavni vokal ali inštrument (ki predstavlja glavno melodijo) posebno sled. Iz te sledi se nato pridobi referenčna melodija s pomočjo namenskih orodij za točno merjenje frekvenc v signalu, avtomatsko generirani zapisi pa se nato še ročno preverijo in popravijo. Leta 2004 so na prvem tekmovanju v iskanju melodije določili osnovno metodo, po kateri izračunamo dve oceni točnosti [12]:

- **Absolutna višina (ang. pitch)** - Točnost detektirane frekvence v posameznem časovnem okviru se določa kot razlika med detektirano in referenčno frekvenco (nižja ocena je boljša). Frekvenci se primerjata glede na njuno vrednost v centih, največja upoštevana razlika pa je 100 centov (po enačbi 3.1). Tu je pomembno tudi zaznavanje prisotnosti melodije, saj v časovnih okvirih, kjer melodije ni, algoritem dobi oceno 100. Prav tako velja obratno, če algoritem ne zaznava melodije, a je ta prisotna.

$$\mathrm{err}_n = \begin{cases} 100, & \text{če } |f^{\mathrm{hip}}_{\mathrm{cent}}[n] - f^{\mathrm{ref}}_{\mathrm{cent}}[n]| \geq 100 \\ |f^{\mathrm{hip}}_{\mathrm{cent}}[n] - f^{\mathrm{ref}}_{\mathrm{cent}}[n]|, & \text{sicer} \end{cases} \tag{3.1}$$

  Skupna točnost algoritma na vhodnih podatkih se nato izračuna kot povprečje po vseh časovnih okvirih po enačbi 3.2 (ocena v odstotkih -

višja je boljša).

$$\mathrm{Acc} = 100 - \frac{1}{N}\sum_{n=1}^{N}\mathrm{err}_n \tag{3.2}$$

- **Kromatična višina (ang. chroma)** - Pri iskanju najboljše hipoteze v posameznem časovnem okviru se pogosto zgodi, da se algoritem zmoti točno za oktavo (1200 centov) ali njen večkratnik, saj te vrednosti ustrezajo višjim ali nižjim harmoničnim komponentam osnovne (točne) frekvence, ki so praktično vedno prisotne. Takšno hipotezo lahko deloma označimo kot točno, saj pravilno določa ton po kromatični lestvici (sestavljeni iz 12 tonov, med katerimi je pol tona ali 100 centov razlike), zato drugi način ocenjevanja oprosti oktavne napake, tako da vse frekvence v hipotezi in referenčnem zapisu preslika na eno oktavo (deljenje po modulu 1200), na teh pa nato izračuna točnost podobno kot prvi način, le da upošteva cikličnost oktave (absolutna razlika 1150 centov je pravzaprav razlika 50 centov).

Leta 2005 so nato opisano metodo spremenili zaradi mnenja, da bi bilo manjše napake potrebno popolnoma oprostiti. Zato se od takrat naprej uporablja druga metoda, kjer se vsaka vrednost v hipotezi šteje za popolnoma točno, če je njena razdalja od referenčne manj kot 50 centov (četrtina tona). Točnost algoritma je nato enostavno delež pravilno določenih vrednosti glede na ta kriterij, s tem da se posebej ocenjuje točnost hipoteze in zaznavanje prisotnosti melodije. Neodvisnost obeh ocen se doseže tako, da algoritem poda hipotezo tudi za časovne okvire, v katerih ne zaznava melodije. Še vedno se uporabljata dve oceni, absolutna in kromatična, skupna točnost (glavno merilo na tekmovanju) algoritma pa se izračuna kot delež okvirov, za katere je pravilno določena absolutna višina oz. neprisotnost melodije. Obe metodi sem opisal, ker sem za primerjavo rezultatov uporabil tudi rezultate iz leta 2004.

3.2 Testni podatki

Edini testni podatki s tekmovanj ISMIR in MIREX, ki so prosto dostopni, so vhodne datoteke s prvega tekmovanja ISMIR 2004. Poznejše testne množice so ostale skrite zaradi preprečevanja prilagajanja algoritmov in ohranjanja primerljivosti rezultatov. Svoj algoritem sem tako testiral na množici ISMIR 2004, ki sem jo lahko primerjal z rezultati tekmovanj leta 2004, 2006, 2008 in 2009, ko so algoritme testirali na tej množici. Ker so po letu 2004 spremenili

način ocenjevanja (opisano v prejšnjem poglavju), sem svoj (oz. prirejen Gotov) algoritem ocenjeval na dva načina in ga v tabelah primerjal z rezultati, ustreznimi obema načinoma.

Testna množica iz leta 2004 je sestavljena iz 20 posnetkov dolžine prib. 20 sekund s frekvenco vzorčenja 44,1 kHz pri 16 bitih na vzorec. Ti so razdeljeni v 5 skupin po 4 posnetke (tabela 3.1).

SKUPINA	**ŽANR**	**NOSILEC MELODIJE**
daisy	pop	umetni ženski vokal
jazz	jazz	saksofon
midi	2x pop, 2x ljudska glasba	inštrumenti MIDI
opera	opera	2x moški vokal, 2x ženski vokal
pop	pop	moški vokal

Tabela 3.1: Glavna testna množica, sestavljena iz 20 posnetkov, razdeljenih v 5 skupin.

Gotov algoritem je narejen tako, da določa hipotezo z ločljivostjo 10 ms, medtem ko imajo referenčni podatki podane vrednosti vsakih 256 vzorcev, kar ustreza času približno 5,8 ms pri 44,1 kHz. Popraviti osnovni algoritem na takšno ločljivost ni tako enostavno, saj je prilagojen za obdelavo 16 kHz posnetkov, predvsem zaradi večstopenjske množice filtrov, ki za ujemanje različnih stopenj potrebuje frekvenco vzorčenja, deljivo s 16. Zato sem Gotov algoritem ohranil v osnovni obliki ter rezultate prilagodil referenčnim časovnim okvirom, tako da sem rezultate med sosedoma referenčnega okvira, ki sta dovolj blizu (manj kot 100 centov), interpoliral, pri bolj oddaljenih sosedih pa vzel vrednost najbližjega po času.

Poleg testne množice ISMIR 2004 sem uporabil še učno množico s tekmovanja MIREX 2005, ki jo sestavlja 13 datotek in na kateri sem tudi prilagodil večino parametrov.

3.3 Rezultati

Rezultati so urejeni tako, da najprej primerjam uspešnost najboljše različice algoritma z uspešnostjo algoritmov leta 2004. Podrobneje komentiram najbolj in najmanj uspešno delovanje te različice na datotekah in opišem splošno uspešnost na testni množici. Nato primerjam še uspešnost algoritma s konku-

OZNAKA	DEL ALGORITMA	OPIS
Goto	Osnovni algoritem	Implementacija osnovnega Gotovega algoritma brez sledenja
ZS	Združevanje spektrov	Uporaba dveh širin oken hkrati pri računanju spektrograma
FL	Razširitev množice filtrov	Uporaba dodatne stopnje (500Hz) pri računanju spektrograma
TR	Sledenje	Gradnja sledi in uporaba najboljših v hipotezi
OO	Odpravljanje odstopanj	Odpravljanje vrednosti, ki odstopajo od lokalnega povprečja
NN	Izravnava hipoteze	Glasovanje sosednih sledi (najboljših naslednikov) za boljše ujemanje hipoteze
PP	Prepoznavanje prehodov	Prepoznavanje in izračun prehodov med sosednimi vrednostmi v hipotezi s Houghovo transformacijo
AD	Izračun točnih frekvenc	Izračun možnih točnejših frekvenc v maksimumih krivulj, prilagojenih na okolico hipoteznih vrednosti
VX	Ugotavljanje prisotnosti	Določanje prisotnosti oz. neprisotnosti melodije za posamezne sledi

Tabela 3.2: Seznam vseh delov algoritma (osnovnega in možnih izboljšav) ter njihove oznake, uporabljene v tabelah rezultatov.

renčnimi algoritmi tekmovanj leta 2006, 2008 in 2009, preizkusim pa tudi najbolj uspešne kombinacije dodatkov (seznam teh je v tabeli 3.2) za posamezne datoteke ter uspešnost algoritma na učni množici tekmovanja MIREX 2005. Celotne tabele uspešnosti algoritma pri drugih kombinacijah izboljšav so zaradi urejenosti v dodatku (A.1-A.9), v tem poglavju izpostavim le nekatere vrednosti in največje razlike v delovanju.

3.3.1 Primerjava z rezultati tekmovanja ISMIR 2004

V tabeli 3.3 je primerjava najbolj uspešne različice izboljšanega Gotovega algoritma (Goto+ZS+TR+OO+NN+VX) z rezultati tekmovanja ISMIR 2004. Iz primerjave uspešnosti na posameznih testnih datotekah je razvidno, da je v večini primerov (9 od 20) predlagani algoritem boljši. S sedmimi primeri mu po uspešnosti tesno sledi Paivov algoritem. V enakem razmerju sta si algoritma tudi po povprečni oceni, kjer je moj algoritem dobil najvišjo oceno 66,8, Paivov pa 65,0. Najslabše primerljive rezultate sem dosegel pri datotekah midi1, pop1 in pop2, najboljše pa pri jazz1 in midi2. Hipoteze teh rezultatov tudi analiziram ob slikah 3.1, 3.2, 3.3, 3.4 in 3.5 v naslednjem podpoglavju. Zanimiva izjema v tabeli je opera z moškim vokalom (datoteki op_male), kjer je daleč najboljše rezultate dosegel program za monofonično sledenje frekvence signala, ki so ga na tekmovanju uporabili kot osnovno primerjavo. Razlog za to je morda v tem, da algoritmi zaradi širokih oken pri računanju frekvenčnih spektrov težje računajo manj stabilne glasove, kot je močan vibrato pri operi, izogibajo pa se tudi nizkim frekvencam, ki običajno pripadajo basovski spremljavi. "Mono" morda uporablja precej krajša okna in se osredotoča tudi na nizke frekvence, kar bi mu dalo prednost v primerih moške opere. Če izvzamemo osnovno primerjavo, je sicer tudi pri teh dveh datotekah moj algoritem dosegel rezultate, primeljive z ostalimi.

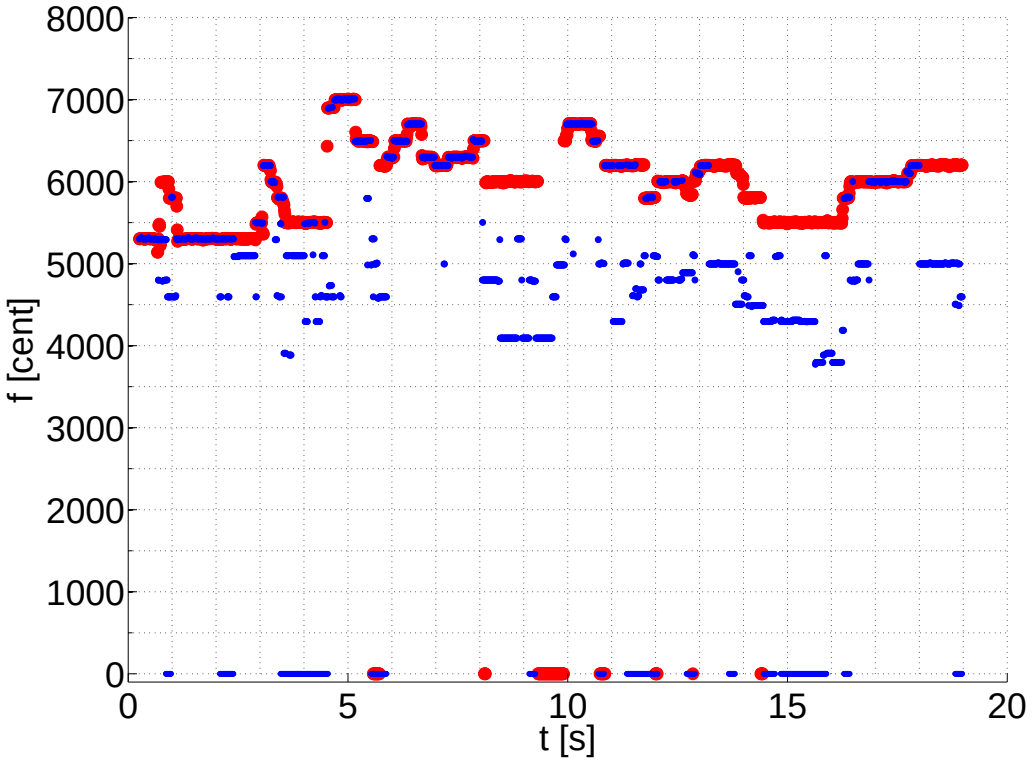
	Paiva	Tappert	Poliner	Bello	Mono	Goto	GotoTR	Kuder
daisy1	66,6	56,6	61,6	77,2	61,4	65,8	68,6	**84,7**
daisy2	75,2	53,9	78,5	78,4	70,0	74,5	74,9	**84,0**
daisy3	**91,1**	80,3	87,0	79,6	15,3	75,8	82,3	85,4
daisy4	89,6	74,6	66,4	64,0	48,0	88,8	89,3	**90,2**
jazz1	61,6	48,6	49,9	66,1	54,1	78,2	79,5	**80,2**
jazz2	68,2	38,4	75,0	63,8	52,0	71,9	73,2	**79,3**
jazz3	56,1	64,6	80,8	73,9	38,7	57,6	59,2	**83,1**
jazz4	**78,3**	44,8	47,0	64,1	29,7	58,6	60,4	71,7
midi1	**76,2**	39,8	66,7	24,7	9,5	49,5	51,8	44,3
midi2	74,0	75,9	78,5	77,7	25,8	59,0	89,1	**92,9**
midi3	64,2	62,9	51,2	50,3	35,6	60,2	82,0	**73,3**
midi4	**73,3**	50,1	38,7	29,3	8,4	55,0	58,9	63,1
op_fem2	35,5	**45,8**	35,7	44,7	41,3	36,7	34,1	44,1
op_fem4	47,0	56,1	21,8	44,4	31,2	53,0	51,2	**61,6**
op_male3	26,6	24,4	33,9	21,7	**48,9**	31,5	28,9	32,6
op_male5	46,9	34,1	29,9	21,5	**72,2**	46,1	45,0	46,6
pop1	**61,0**	28,2	55,4	30,4	16,4	28,1	29,6	36,9
pop2	**64,0**	28,9	57,9	32,6	33,3	38,2	40,7	41,9
pop3	**73,4**	34,8	46,2	68,0	27,3	53,0	59,7	72,8
pop4	70,8	37,8	70,9	**73,1**	30,8	55,1	56,1	67,6
	65,0	49,0	56,6	54,3	37,5	56,8	60,7	**66,8**

Tabela 3.3: Primerjava skupnih ocen (povprečij absolutnih in kromatičnih ocen) algoritmov na tekmovanju ISMIR 2004 z rezultati izboljšanega Gotovega algoritma (stolpec Kuder). Vrednosti v stolpcu "Mono" predstavljajo uspešnost programa za monofonično sledenje frekvence signala. V stolpcu "Goto" so ocene uspešnosti osnovnega Gotovega algoritma brez sledenja, v stolpcu "GotoTR" pa z mojim sledenjem (oba brez dodatnih izboljšav). Vrednosti, označene s krepkim tiskom, predstavljajo najboljše rezultate na posameznih datotekah.

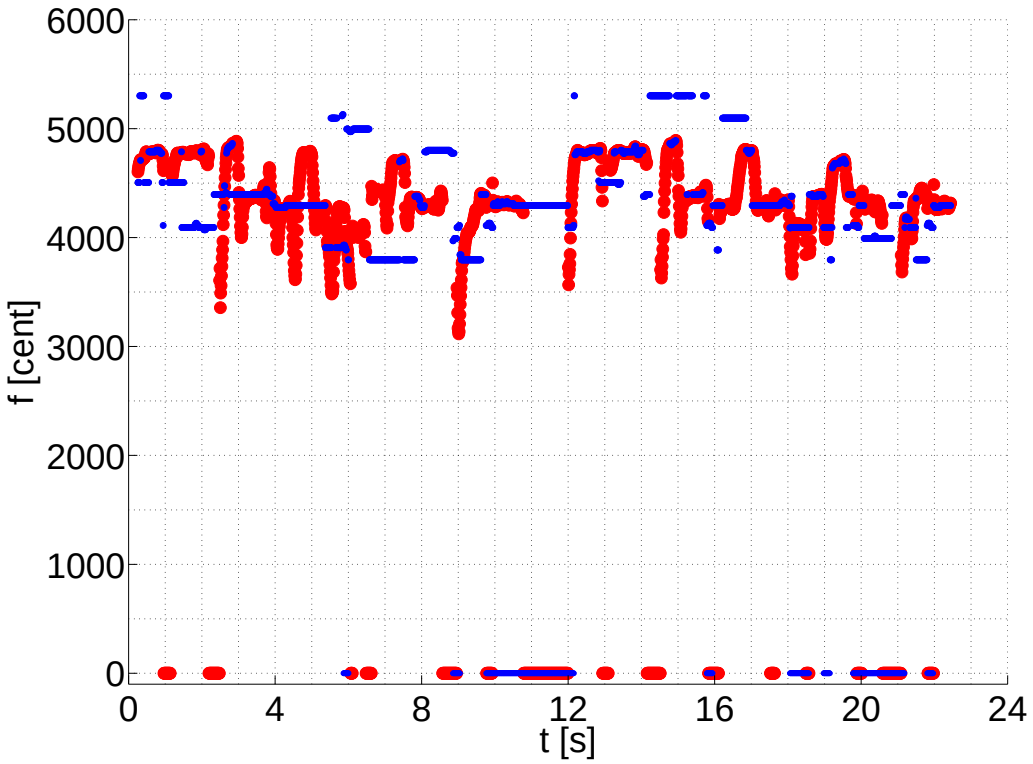
3.3.2 Analiza najslabših in najboljših hipotez

V tem poglavju so zbrani grafi petih hipotez, pri katerih je bil algoritem precej manj ali precej bolj uspešen od konkurenčnih algoritmov s tekmovanja ISMIR 2004. V teh primerih je bila uporabljena kombinacija Goto+ZS+TR+OO+ NN+VX (brez zaznavanja prehodov, razširitve množice filtrov in računanja točnih frekvenc med frekvenčnimi koši), ki je bila v povprečju najuspešnejša

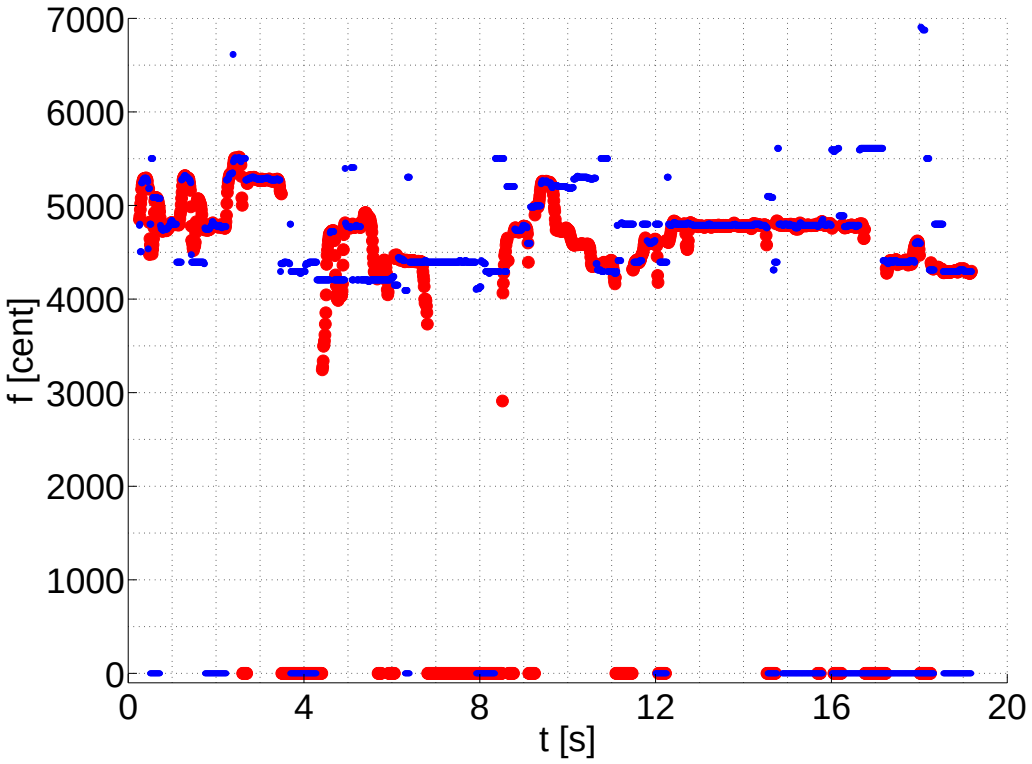
na testni množici. Komentarji uspešnosti so podani ob slikah, kjer modri znaki predstavljajo izračunano hipotezo, rdeči pa referenčne (točne) vrednosti. Dodatni znaki pri frekvenci 0 predstavljajo detekcijo in referenco področij, kjer melodije ni.



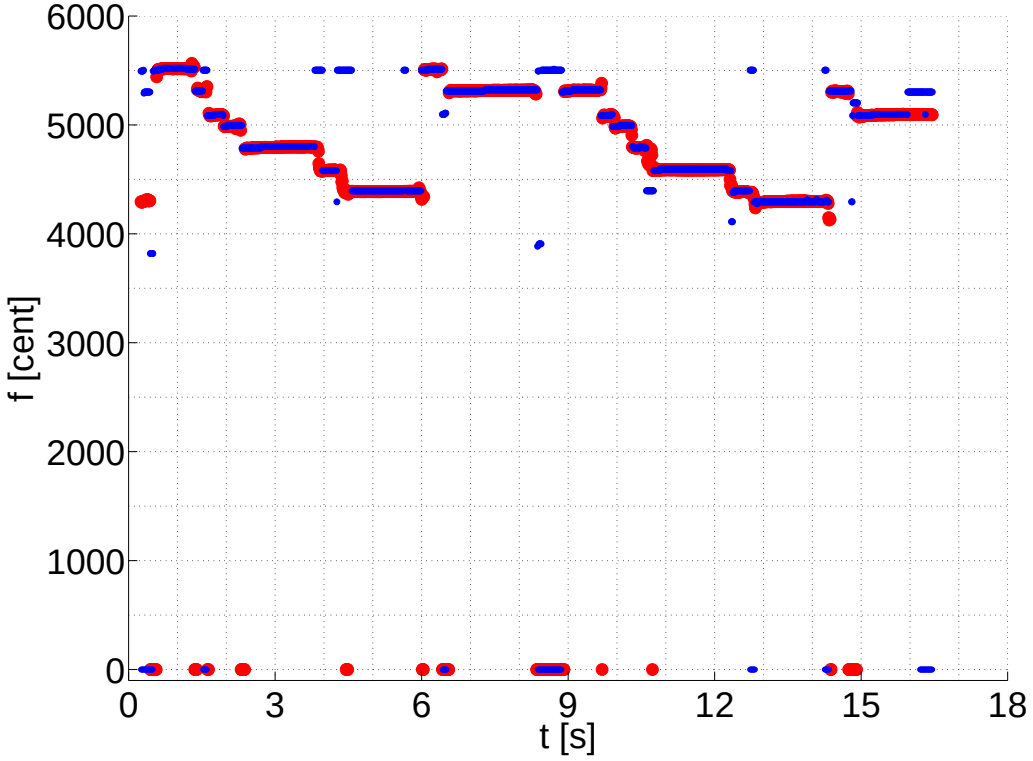
Slika 3.1: Hipoteza datoteke midi1.wav (44,3% točnost, Paiva 76,2%). Takoj je opaziti slabo detekcijo prisotnosti melodije, razdrobljenost hipoteze in nekaj oktavnih napak okoli 9 in 14 s posnetka. Ta posnetek ima precej močno basovsko spremljavo, ki prevzame hipotezo višine 4100 centov (ton F) okoli 9 s. Tu, pa tudi v območju okoli 14 s, je glavni inštrument tišji in ga je zato težje pravilno detektirati, kar morda razloži oktavne napake. Razdrobljenost hipoteze je najbrž predvsem posledica močnega basa in ritmične spremljave oz. šibkejšega glavnega inštrumenta, kar razloži tudi večje število območij, kjer melodija po hipotezi ni prisotna. Algoritem je bil nekoliko uspešnejši pri kombinaciji Goto+ZS+TR+VX (brez dodatnih popravljanj hipoteze), kjer je dosegel oceno 48,2%, torej pri zelo razdrobljeni hipotezi izravnavanje lahko tudi škoduje.



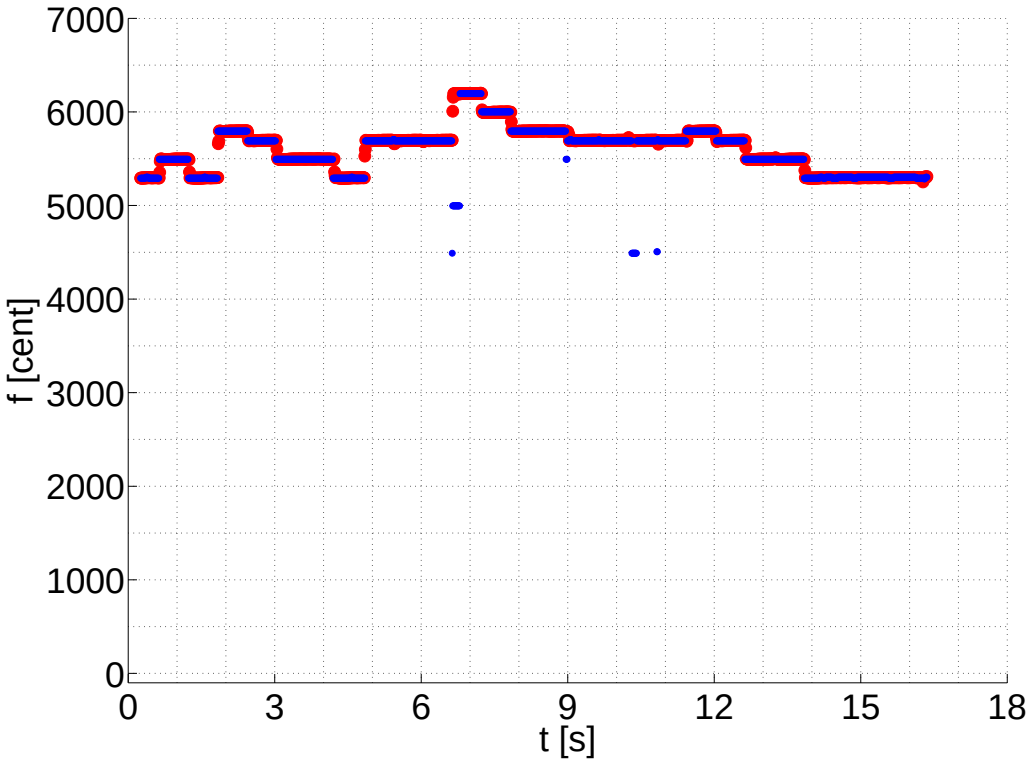
Slika 3.2: Hipoteza datoteke pop1.wav (36,9% točnost, Paiva 61,0%). Ta datoteka je bila težavna za moj algoritem zaradi ostrih prehodov ob začetkih in koncih not, kjer zaznavanje prehodov ne deluje (ki ga v tej različici algoritma tudi nisem uporabil), saj nima dveh opornih točk, med katerima bi bilo območje detekcije. Stabilnih not je malo, vseeno pa pravilno zazna tudi nekaj krajših sledi (praktično brez oktavnih napak). Ena izmed lastnosti Paivovega algoritma (najuspešnejšega pri tej datoteki) je ravno zaznavanje značilnosti ob vstopih [11], kar mu omogoča boljše računanje prehodov, ki se pojavljajo v tem izseku. Algoritem je dobil za 2% višjo oceno, če sem izključil združevanje spektrov, kar je skoraj zanemarljiva izboljšava, glede na to, da je bil algoritem brez ZS sicer v povprečju po vseh datotekah za 3% slabši od tu uporabljene različice.



Slika 3.3: Hipoteza datoteke pop2.wav (41,9% točnost, Paiva 64,0%). Težave tu so podobne kot pri pop1.wav, le da tu še zmotno označi območje po 14,5 s kot območje brez melodije. V tem predelu se glasnost izseka zniža (še posebej na koncu), izgine večina spremljave (ostane le tiha kitara), samoglasniki v besedilu (ki so bolj razločni) pa so krajši. Nobena izmed različic algoritma ni bila pri tej datoteki opazno uspešnejša.



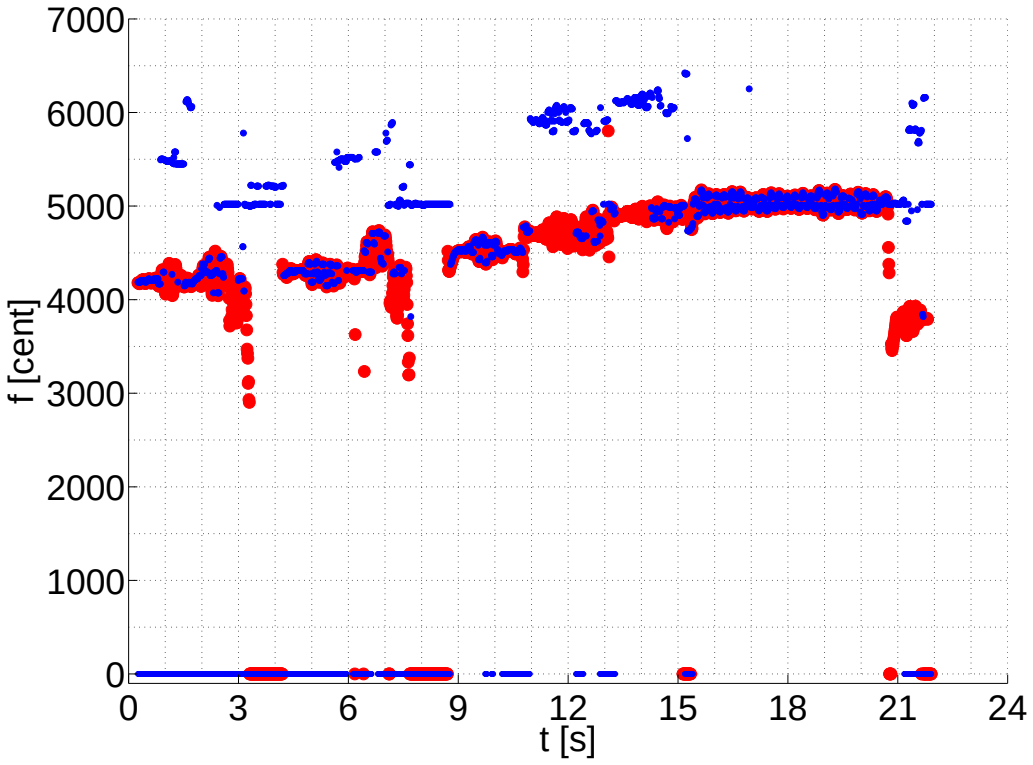
Slika 3.4: Hipoteza datoteke jazz1.wav (80,2% točnost, Paiva 61,1%). Algoritem je bil pri tej datoteki, kjer je glavni inštrument saksofon, zelo uspešen, saj je dosegel 87,6% točnost absolutne višine (po metriki iz leta 2005, kjer se odpušča manjše odmike), napake je delal skoraj izključno pri prehodih (ki se pri tej različici algoritma niso detektirali) in pol sekunde pred koncem, kjer spremljajoči klavir ohrani glasnost in za trenutek (ko saksofon zaključi melodijo) preide v ospredje. Najdaljši premor, kjer melodija ni prisotna, je dobro zaznal, ni pa zaznal večine kratkih, pri zaznavanju premorov je bil tako uspešen le v 46,0%, kar pa ni veliko vplivalo na oceno. Algoritem je bil še uspešnejši, če sem uporabil dodatno stopnjo filtrov (FL), kjer se je absolutna ocena po metriki 2005 dvignila s 87,6% na 92,0%.



Slika 3.5: Hipoteza datoteke midi2.wav (92,9% točnost, Paiva 74,0%). Primer z zelo uspešno detekcijo po metriki iz leta 2005 - absolutno oceno 96,1%. Opazne napake so le pri prehodih in enem stabilnem tonu (kjer je napaka oktavna). Paivov algoritem je najbrž zmotila spremljava, ki jo je moj algoritem zaradi dovolj stabilnih tonov (in posledično sledi) glavnega inštrumenta lahko izločil iz hipoteze.

3.3.3 Uspešnost algoritma na posameznih datotekah

V tabeli 3.4 je prikazana uspešnost najbolj uspešnega algoritma Goto+ZS+ +TR+OO+NN+VX na posameznih datotekah. Algoritem ima očitno najmanj težav s sledenjem sintetičnega ženskega glasu, saj ima skupina "daisy" po metriki iz leta 2005 absolutno točnost v povprečju nad 95%. Zelo uspešno poišče melodijo tudi v primeru jazza, torej dobro loči saksofon v ospredju pred ostalimi inštrumenti. V skupini "midi" se je uspešnost ločila po žanru. Z "midi1" in "midi4", ki sta pop skladbi, je imel kar nekaj težav, ki jih lahko pripišemo krajšim tonom in posledično večji razdrobljenosti hipoteze. Nasprotno imata "midi2" in "midi3" daljše tone, kar se je pokazalo tudi v uspešnosti algoritma.



Slika 3.6: Hipoteza datoteke opera_male5.wav (46,6% točnost, Paiva 46,9%). Opazna je slaba detekcija prisotnosti melodije (posledica glasnejše melodije na koncu posnetka), ter oktavne napake od 11 do 15 s. Sicer je točnost hipoteze razmeroma visoka.

	2004		2005				
	abs.	krom.	abs.	krom.	skupna	vxRec	FA
daisy1	84,5	84,9	95,7	96,4	89,8	99,8	30,0
daisy2	84,0	84,0	95,5	95,5	89,6	95,4	24,0
daisy3	85,4	85,4	97,2	97,2	89,4	92,1	—
daisy4	90,1	90,3	94,6	94,8	94,6	100,0	—
jazz1	80,0	80,4	87,6	88,6	84,1	94,2	46,0
jazz2	79,3	79,3	85,9	85,9	82,8	96,1	38,9
jazz3	82,9	83,4	96,0	97,2	85,6	83,9	8,7
jazz4	69,3	74,0	81,5	88,0	73,6	95,9	50,8
midi1	37,7	50,8	44,4	68,6	39,2	73,3	67,8
midi2	92,1	93,6	96,1	97,7	95,9	100,0	100,0
midi3	71,6	75,0	83,6	87,8	74,1	84,5	88,5
midi4	60,8	65,4	68,4	75,3	62,5	84,4	25,9
opera_fem2	43,5	44,8	62,7	64,3	46,8	69,3	47,5
opera_fem4	61,6	61,6	74,5	75,3	67,1	79,2	2,8
opera_male3	30,0	35,3	42,5	52,7	32,6	50,7	3,5
opera_male5	40,9	52,3	59,3	78,6	44,0	56,7	4,0
pop1	36,8	37,1	42,4	43,0	39,9	85,0	42,8
pop2	41,8	42,0	65,2	65,3	44,3	71,0	50,7
pop3	72,4	73,1	84,7	85,8	77,5	88,9	24,0
pop4	66,5	68,8	81,9	85,1	71,8	93,5	54,4
	65,6	68,1	77,0	81,1	69,3	84,7	39,5

Tabela 3.4: Tabela uspešnosti algoritma Goto+ZS+TR+OO+NN+VX na posameznih datotekah. Prikazane so točnosti po metrikah iz leta 2004 in 2005, stolpec "vxRec" predstavlja delež referenčnih vrednosti s prisotno melodijo, ki jih je kot takšne ocenil tudi algoritem, "FA" predstavlja delež referenčnih vrednosti z neprisotno melodijo, ki jih je algoritem označil napačno, d' pa sposobnost ločevanja med obema razredoma prisotnosti.

Opera, pri kateri ima večina algoritmov težave, je bila težavna tudi za moj algoritem, a vseeno je ta pokazal relativno dobre rezultate in je znal povezati tudi področja vibrata. Pri datoteki opera_male5 je večja razlika med kromatično in absolutno oceno, ker je algoritem pri daljšem vibratu naredil večje število oktavnih napak (slika 3.6). Neuspešnost pri datotekah "pop1" in "pop2" je podrobno opisana v prejšnjem podpoglavju, pri datotekah "pop3" in "pop4" pa je algoritem dosegel precej visoke ocene. Za ta dva posnetka je značilna izrazita (in hitra) ritmična spremljava, ki algoritma pri določanju hipoteze ni posebno zmotila, kar je očitno ena dobrih lastnosti algoritma, saj tudi pri drugih datotekah običajno ni bilo opaziti izrazitih napak zaradi ritmične spremljave.

3.3.4 Primerjava z rezultati tekmovanj MIREX 2006, 2008 in 2009

V tabeli 3.5 so zbrani rezultati vseh algoritmov na tekmovanjih MIREX 2006, 2008 in 2009. Moj algoritem je bil razmeroma uspešen, s kromatično oceno le 6,5% za trenutno najboljšim in 5,1% pred Gotovim z dodanim sledenjem, kar je dokaj dobro, glede na to, da so pri vrhu večinoma algoritmi, ki jih avtorji razvijajo že od začetka tekmovanj. Če primerjamo moje rezultate z vrstico "poliner", lahko vidimo, da ima na tej stopnji razvoja dobro zaznavanje prisotnosti melodije že precejšen vpliv, saj je bil moj algoritem uspešnejši v določanju višine, a slabši po skupni oceni ravno zaradi razlike v zaznavanju prisotnosti. Algoritmi v zgorni polovici tabele imajo tudi precej bolj boljšo absolutno oceno glede na kromatično, kar nakazuje, da bi bilo možno hipoteze mojega algoritma še nekoliko prečistiti.

Če bi tabelo uredili po drugih stolpcih (namesto skupni oceni), bi po kromatični oceni pridobil eno mesto, po absolutni dve, po kvaliteti razločevanja prisotnosti d' pa bi padel za dve mesti, torej je osnovni položaj v tabeli primeren splošni kvaliteti algoritma. Še največja razlika od najboljših je v zaznavanju prisotnosti, kjer najboljši algoritmi pri višjem priklicu napačno označijo do štirikrat manj okvirov, kjer melodija ni prisotna.

	leto	vxRec	FA	d'	abs.	krom.	skupna
kd	2009	91,6	15,8	2,38	87,1	87,6	86,3
pc	2008	88,0	9,8	2,47	85,1	85,9	85,1
pc	2009	84,6	15,5	2,04	82,9	83,4	82,5
dressler	2006	90,9	10,5	2,59	82,9	84,0	82,5
drd2	2008	91,7	31,6	1,86	85,7	86,2	81,5
rk	2008	88,9	26,2	1,86	82,4	83,5	78,8
ryynanen	2006	84,4	12,6	2,16	80,6	82,3	77,3
cl1	2009	95,6	69,3	1,21	85,1	86,3	76,6
dr1	2009	90,3	41,2	1,52	81,4	83,4	75,7
cl2	2009	86,4	46,2	1,20	85,1	86,3	75,4
jjy	2009	86,8	35,8	1,48	83,3	87,0	75,1
dr2	2009	84,2	26,6	1,63	81,2	83,6	74,7
poliner	2006	89,9	36,3	1,63	73,2	76,4	71,9
mw	2009	99,9	93,8	1,70	82,3	86,4	70,8
rr	2009	92,0	47,9	1,46	76,9	85,1	70,7
vr	2008	82,3	20,0	1,77	77,1	85,2	70,1
Kuder		**84,7**	**39,5**	**1,29**	**77,0**	**81,1**	**69,3**
clly2	2008	84,6	49,1	1,04	75,3	76,7	68,0
Goto+TR		100,0	100,0	0,0	71,9	78,3	61,7
drd1	2008	93,8	57,9	1,34	65,9	75,0	59,6
sutton	2006	73,2	24,9	1,30	62,6	65,4	58,2
Goto		100,0	100,0	0,0	66,4	75,0	56,7
toos	2009	99,9	93,2	1,62	61,0	71,8	52,5
clly1	2008	55,7	29,7	0,68	75,3	76,7	50,2
brossier	2006	99,7	88,4	1,55	57,4	68,7	49,6
hjc1	2009	45,7	17,3	0,84	63,9	73,6	47,4
hjc2	2009	45,7	17,3	0,84	50,5	64,9	44,6

Tabela 3.5: Primerjava ocen algoritmov na tekmovanjih MIREX z rezultati izboljšanega Gotovega algoritma (vrstica Kuder). Tudi v tej tabeli sem dodal uspešnosti osnovnega Gotovega algoritma brez vseh izboljšav (Goto) oz. samo s sledenjem (Goto+TR). Vrstice so urejene po skupni oceni, stolpec "leto" pomeni leto sodelovanja na tekmovanju MIREX.

3.3.5 Uspešnosti drugih kombinacij izboljšav

V tabeli 3.6 so zbrani rezultati najbolj uspešnih kombinacij izboljšav za posamezne datoteke. V skupinah "daisy" in "jazz" je algoritem pridobil na točnosti, če sem dodatno uporabil še prepoznavanje prehodov in računanje točnih frekvenc (PP+AD), medtem ko sta ti dve izboljšavi pri drugih datotekah imeli nepredvidljiv vpliv. Očitno v splošnem lahko z njima na točnosti pridobijo tiste hipoteze, ki so tudi že prej precej dobre in stabilne (z manj odstopajočimi vrednostmi), medtem ko pri drugih hitro lahko povzročita dodatne napake. "Jazz" je bila edina skupina, ki je pridobila z uporabo dodatnega nivoja v množici filtrov, kar je zanimivo, saj ti posnetki niso imeli ekstremno nizkih frekvenc v hipotezi. Očitno je dodatna stopnja (in posledično drugačen spekter v nizkih frekvencah) vplivala na ravnotežje modelov in morda povzročila manjši vpliv spremljave. Združevanje spektrov je imelo v večini primerov pozitiven vpliv, izjeme so bile le v skupinah "midi" in "pop", morda zaradi močnejše spremljave in križanja višjih harmoničnih komponent te z melodijo. Ugotavljanje prisotnosti je koristilo pri vseh skladbah, razen pri nekaterih, pri katerih se glasnost melodije med izsekom zelo spremeni. To se potrjuje kot šibkost tega algoritma. Sledenje je razumljivo koristno pri vseh skladbah, v večini primerov pa tudi odpravljanje odstopanj in izravnava hipoteze. Ti škodujeta kvečjemu v primerih, kjer je hipoteza zelo razdrobljena in je izravnava zato težavna.

	2004		2005					
	abs.	krom.	abs.	krom.	skupna	vxRec	FA	različica algoritma
daisy1	85,5	86	96,5	97,4	91,1	99,3	27,9	Goto+ZS+FL+TR+ OO+NN+PP+AD+VX
daisy2	84,5	84,5	95,7	95,7	89,8	95,4	24	Goto+ZS+TR+OO+ NN+PP+AD+VX
daisy3	87,4	87,4	98,4	98,4	90,6	92,1	—	Goto+ZS+TR+OO+ NN+PP+AD+VX
daisy4	91,5	91,7	95,2	95,4	95,2	100	—	Goto+ZS+TR+OO+ NN+PP+AD+VX
jazz1	84,8	85	92	92,2	87,6	100	59,1	Goto+ZS+FL+TR+ OO+NN+PP+AD+VX
jazz2	80,9	81,3	86,5	86,9	83,9	97,1	38,9	Goto+ZS+FL+TR+ OO+NN+PP+AD+VX
jazz3	86	86,1	95,1	96,7	88,4	92,2	14,4	Goto+ZS+FL+TR+ OO+NN+PP+AD+VX
jazz4	79	79,1	84,3	84,4	81,4	88,1	27,8	Goto+ZS+FL+TR+ OO+NN+PP+AD+VX
midi1	46,1	61,3	51	68	48,1	100	100	Goto+ZS+TR
midi2	94	95,5	96,1	97,7	95,9	100	100	Goto+ZS+TR+OO+ NN+AD+VX
midi3	80,7	85,3	85,4	90,2	83,8	99	100	Goto+TR+OO+ NN+VX
midi4	66,4	71,5	69,3	77,9	68,5	92,8	26,5	Goto+TR+VX
op_fem2	44,4	45,7	64,4	66,1	48	69,3	47,5	Goto+ZS+TR+OO+ NN+PP+VX
op_fem4	63,2	63,2	78,7	79,5	70,1	79,2	2,75	Goto+ZS+TR+OO+ NN+PP+VX
op_male3	32,5	36,8	44,9	53,6	35,4	59,7	29,4	Goto+ZS+TR+OO+ NN+VX
op_male5	46	60,1	58,7	76,3	51,2	100	100	Goto+ZS+TR
pop1	38,8	39	41,2	41,4	41,9	93,8	43,5	Goto+TR+OO+NN+ PP+VX
pop2	44,1	44,8	65,3	66	47,9	100	100	Goto+ZS+TR
pop3	72,1	75	80,8	85	77,2	99,7	35,9	Goto+TR+VX
pop4	66,5	68,8	81,9	85,1	71,8	93,5	54,4	Goto+ZS+TR+OO+ NN+VX

Tabela 3.6: Najboljše kombinacije izboljšav za posamezne datoteke.

3.3.6 Uspešnost algoritma na učni množici MIREX 2005

V tabeli 3.7 so zbrani rezultati algoritma Goto+ZS+TR+OO+NN+VX na učni množici s tekmovanja MIREX 2005. Ta množica je naravnana pretežno k pop skladbam, kar naj bi predstavljalo realnejšo porazdelitev glasbenih zvrsti. Algoritem je bil najuspešnejši na datotekah "train01" in "train05" iz zvrsti jazz, kjer glavno melodijo tvori ženski vokal, najslabši pa pri midi datotekah od "train10" do "train13". Pri teh ima očitno večkrat težave, če je melodična spremljava premočna. Pri ostalih datotekah (zvrsti pop in R&B) je dobil oceno okoli 70%, kar je razmeroma dober rezultat. Zopet se je pokazalo, da tudi izrazita ritmična spremljava ne vpliva veliko na uspešnost algoritma.

	2004		2005				
	abs.	krom.	abs.	krom.	skupna	vxRec	FA
	abs.	krom.	abs.	krom.	skupna	vxRec	FA
train01	70,9	70,9	89,5	89,5	76,0	94,7	41,1
train02	48,4	49,8	57,6	60,4	52,6	89,5	50,0
train03	51,4	51,5	65,9	65,9	55,1	86,7	50,1
train04	57,8	59,0	65,5	68,7	60,9	68,5	16,0
train05	65,3	65,3	87,2	87,2	70,8	94,6	53,6
train06	50,9	50,9	59,2	59,2	52,7	58,3	33,1
train07	46,4	47,2	72,9	74,6	50,0	60,4	29,6
train08	75,1	75,2	83,2	83,5	82,0	94,1	18,0
train09	70,5	71,8	80,0	82,0	75,3	87,0	14,5
train10	29,8	29,9	31,5	31,5	30,8	52,6	0,0
train11	47,2	48,9	49,9	51,9	49,2	92,2	—
train12	34,5	38,4	35,8	38,7	35,8	98,3	—
train13	18,9	37,7	21,4	41,6	19,6	91,4	—
	51,3	53,6	61,5	64,2	54,7	82,2	30,6

Tabela 3.7: Rezultati algoritma na učni množici MIREX 2005.

Poglavje 4

Zaključek

Algoritem sem implementiral v programu MATLAB, kjer se je Gotov del algoritma izkazal za računsko precej zahtevnega, saj je za testne posnetke, dolge okoli 20 sekund, potreboval tudi več kot uro (na sistemu s procesorjem Sempron 2600+ in 1GB pomnilnika). Originalna Gotova implementacija je sicer lahko delovala v realnem času (na distribuiranem sistemu), osnovna stopnja moje implementacije algoritma pa ima še veliko možnosti paralelizacije, ki bi ga teoretično lahko pohitrila tudi za faktor 2000 (testne datoteke so bile dolge okoli 2000 časovnih okvirov, ki bi jih bilo možno povsem neodvisno obdelati). Gotovo bi lahko ta del algoritma tudi sicer še dodatno optimiziral, a ker je bil večji del mojega dela usmerjen v izpopolnjevanje poznejših stopenj, prve stopnje nisem potreboval večkrat izvajati in je za ta namen delovala dovolj hitro. Druga stopnja algoritma s predobdelavo vrhov in gradnjo sledi je potrebovala od 4,5 s do 984 s (v povprečju 241 s), kar je bilo odvisno od števila vrhov, ki so ostali po predobdelavi. Pogoje pri tej bi lahko še dodatno zaostrili in izločili večje število majhnih vrhov, kar bi ob majhni izgubi točnosti lahko zelo pohitrilo izvajanje in zmanjšalo hitrost obdelave na manj kot 10 s. Preostanek algoritma z vsemi izboljšavami (odpravljanjem odstopanj, iskanjem najboljših naslednikov, prepoznavanjem prehodov, ugotavljanjem prisotnosti melodije) je potreboval na različnih testnih datotekah od 6 do 207 sekund (s povprečjem 46 s), odvisno predvsem od števila sledi, zgrajenih v prejšnjih stopnjah, kar je najbolj vplivalo na čas izračunavanja najboljših naslednikov. Brez iskanja najboljših naslednikov in prehodov (najuspešnejša različica) je bil čas izvajanja od 2 do 19 sekund, s povprečjem 8 sekund. Iz tega sledi, da bi ob dovolj hitri prvi stopnji in deloma omejeni predobdelavi algoritem lahko še vedno deloval v realnem času.

Dodelani algoritem glede na izvedene teste deluje precej dobro, a ti testi

niso zelo obsežni, zato bi ga bilo vsekakor zanimivo prijaviti na bodoče tekmovanje MIREX, kjer bi ga morda lahko ponovno preveril na testni množici iz leta 2005 (na kateri je v izvirni obliki že bil testiran) in tako pridobil natančnejše podatke o kvaliteti izboljšav. Algoritem bi moral skoraj gotovo pridobiti na skupni oceni zaradi dodanega zaznavanja prisotnosti melodije, sprememba točnosti same hipoteze pa bi bila odvisna predvsem od kvalitete gradnje sledi originalnega in mojega algoritma, ki pa ju zaradi grobega opisa Gotove metode nisem mogel dokončno primerjati.

Vsekakor bi bilo v algoritem možno dodati še precej izboljšav, ki jih v časovnem okviru izdelave te diplomske naloge (tudi zaradi težav ob implementaciji osnovnega algoritma) žal nisem mogel preizkusiti. Precej neizpopolnjeni sta metodi za iskanje naslednikov in glasovanje na njihovi podlagi, katerim bi bilo potrebno bolje nastaviti parametre. To bi pripomoglo k bolj izravnanim hipotezam, ki imajo v trenutni različici algoritma običajno še precej krajših izstopajočih (opazno nepravilnih ob pogledu na hipotezo) sledi. Primerjava z konkurenčnimi algoritmi kaže, da bi bilo odločanje o prisotnosti melodije možno še precej izboljšati, predvsem v smislu odpornosti proti spremembi glasnosti skladbe, kar je bil vzrok mnogih napak, ki bi jih lahko morda odpravili z delitvijo skladbe na dele z večjo in manjšo glasnostjo in ločeno obdelavo na teh delih. Odpraviti bi bilo potrebno tudi nekaj še vedno prisotnih oktavnih napak.

Poleg popravkov trenutnih metod bi lahko algoritem verjetno dodelali še z upoštevanjem modela človeškega sluha in melodičnih zakonitosti v glasbi (npr. struktur akordov in verjetnosti prehodov med toni), izpopolnitvijo glajenja sledi, zaznavanjem vibrata, upoštevanjem prehodov ob nastopu in koncu tonov, morda bi lahko izboljšali tudi gradnjo osnovnega spektrograma z uporabo drugačnih oken pri STFT. Možnih izboljšav je mnogo, z njimi pa bi ta algoritem morda postal konkurenčen celo najboljšim.

Dodatek A

Tabele uspešnosti različnih kombinacij algoritma

	2004		2005					
	abs.	krom.	abs.	krom.	skupna	vxRec	FA	d'
daisy1	84,9	85,4	96,4	97,1	90,4	99,8	30,0	
daisy2	84,1	84,1	95,7	95,7	89,8	95,4	24,0	
daisy3	86,3	86,3	98,4	98,4	90,6	92,1	—	
daisy4	90,6	90,8	95,1	95,3	95,1	100,0	—	
jazz1	79,6	80,0	87,0	87,8	83,7	94,2	46,0	
jazz2	79,0	79,0	85,7	85,8	82,7	96,1	38,9	
jazz3	81,8	82,3	94,3	95,4	84,5	83,9	8,7	
jazz4	68,8	73,5	80,8	87,4	73,1	95,9	50,8	
midi1	37,4	50,1	43,9	67,6	38,9	73,3	67,8	
midi2	91,8	93,3	95,8	97,4	95,6	100,0	100,0	
midi3	70,9	74,2	83,0	87,1	73,5	84,5	88,5	
midi4	60,0	64,6	67,3	74,1	61,7	84,4	25,9	
op_fem2	44,4	45,7	64,4	66,1	48,0	69,3	47,5	
op_fem4	63,2	63,2	78,7	79,5	70,1	79,2	2,8	
op_male3	29,8	35,3	41,4	52,0	32,3	50,7	3,5	
op_male5	41,7	53,4	60,1	79,7	44,9	56,7	4,0	
pop1	37,0	37,3	42,7	43,2	40,1	85,0	42,8	
pop2	41,9	42,1	65,6	65,7	44,4	71,0	50,7	
pop3	71,7	72,3	83,9	84,9	76,8	88,9	24,0	
pop4	65,9	68,2	80,7	83,9	70,8	93,5	54,4	
	65,5	68,0	77,0	81,2	69,3	84,7	39,5	1,29

Tabela A.1: Rezultati algoritma Goto+ZS+TR+OO+NN+PP+VX (najboljša kombinacija z dodanim prepoznavanjem prehodov).

	2004		2005					
	abs.	krom.	abs.	krom.	skupna	vxRec	FA	d'
daisy1	84,6	85,0	95,6	96,4	89,8	99,8	30,0	
daisy2	84,4	84,4	95,6	95,6	89,6	95,4	24,0	
daisy3	86,7	86,7	97,4	97,4	89,6	92,1	—	
daisy4	91,1	91,3	94,7	94,9	94,7	100,0	—	
jazz1	81,3	81,7	87,6	88,6	84,1	94,2	46,0	
jazz2	80,0	80,0	85,8	85,9	82,7	96,1	38,9	
jazz3	83,6	84,0	95,9	97,1	85,6	83,9	8,7	
jazz4	70,7	75,4	80,9	87,5	73,2	95,9	50,8	
midi1	36,4	49,5	44,5	68,6	39,3	73,3	67,8	
midi2	94,0	95,5	96,1	97,7	95,9	100,0	100,0	
midi3	72,6	76,0	83,6	87,8	74,2	84,5	88,5	
midi4	58,7	63,2	68,5	75,4	62,5	84,4	25,9	
op_fem2	42,8	44,1	62,1	63,8	46,5	69,3	47,5	
op_fem4	60,8	60,8	74,5	75,3	67,2	79,2	2,8	
op_male3	29,6	34,7	42,7	53,1	32,8	50,7	3,5	
op_male5	39,4	50,6	58,4	77,6	43,8	56,7	4,0	
pop1	36,0	36,4	41,7	42,2	39,4	85,0	42,8	
pop2	40,9	41,1	64,1	64,1	43,4	71,0	50,7	
pop3	70,5	71,1	83,6	84,7	76,9	88,9	24,0	
pop4	65,3	67,5	81,5	84,8	71,5	93,5	54,4	
	65,5	67,9	76,7	80,9	69,1	84,7	39,5	1,29

Tabela A.2: Rezultati algoritma Goto+ZS+TR+OO+NN+AD+VX (najboljša kombinacija z dodanim izračunom točnih frekvenc).

	2004		2005					
	abs.	krom.	abs.	krom.	skupna	vxRec	FA	d'
daisy1	84,9	85,4	95,6	96,4	90,3	99,3	27,9	
daisy2	81,9	82,2	95,0	95,4	87,3	92,9	23,6	
daisy3	71,3	83,6	80,2	96,1	74,7	91,5	—	
daisy4	90,2	90,5	94,8	95,1	94,8	100,0	—	
jazz1	83,7	83,9	92,0	92,2	87,7	100,0	59,1	
jazz2	80,5	80,8	86,6	87,0	84,0	97,1	38,9	
jazz3	86,0	86,1	96,4	98,0	89,3	92,2	14,4	
jazz4	78,8	78,9	84,4	84,5	81,5	88,1	27,8	
midi1	24,4	42,3	27,6	56,6	25,4	74,6	76,5	
midi2	91,8	93,5	95,8	97,5	95,5	100,0	100,0	
midi3	76,8	80,1	83,2	87,1	79,6	90,1	100,0	
midi4	56,0	62,7	59,9	71,6	57,5	81,2	17,2	
op_fem2	42,2	43,1	60,5	61,6	45,5	68,2	47,5	
op_fem4	60,5	60,5	72,8	73,7	65,8	79,2	2,8	
op_male3	32,5	35,8	44,9	51,0	35,4	58,5	27,3	
op_male5	34,3	46,4	45,6	75,5	36,7	46,3	1,3	
pop1	28,6	28,8	35,5	35,9	30,5	68,7	37,4	
pop2	32,3	32,3	60,4	60,4	33,3	53,8	40,6	
pop3	70,4	70,7	84,7	85,3	74,8	84,0	22,9	
pop4	61,5	63,6	79,6	82,5	66,0	95,8	80,7	
	63,4	66,6	73,8	79,2	66,8	83,1	41,4	1,17

Tabela A.3: Rezultati algoritma Goto+ZS+FL+TR+OO+NN+VX (najboljša kombinacija z dodano stopnjo v množici filtrov).

	2004		2005					
	abs.	krom.	abs.	krom.	skupna	vxRec	FA	d'
daisy1	84,6	85,0	95,6	96,4	89,8	99,8	30,0	
daisy2	84,4	84,4	95,6	95,6	89,6	95,4	24,0	
daisy3	86,7	86,7	97,4	97,4	89,6	92,1	—	
daisy4	91,1	91,3	94,7	94,9	94,7	100,0	—	
jazz1	81,3	81,7	87,6	88,6	84,1	94,2	46,0	
jazz2	80,0	80,0	85,8	85,9	82,7	96,1	38,9	
jazz3	83,6	84,0	95,9	97,1	85,6	83,9	8,7	
jazz4	70,7	75,4	80,9	87,5	73,2	95,9	50,8	
midi1	36,4	49,5	44,5	68,6	39,3	73,3	67,8	
midi2	94,0	95,5	96,1	97,7	95,9	100,0	100,0	
midi3	72,6	76,0	83,6	87,8	74,2	84,5	88,5	
midi4	58,7	63,2	68,5	75,4	62,5	84,4	25,9	
op_fem2	42,8	44,1	62,1	63,8	46,5	69,3	47,5	
op_fem4	60,8	60,8	74,5	75,3	67,2	79,2	2,8	
op_male3	29,6	34,7	42,7	53,1	32,8	50,7	3,5	
op_male5	39,4	50,6	58,4	77,6	43,8	56,7	4,0	
pop1	36,0	36,4	41,7	42,2	39,4	85,0	42,8	
pop2	40,9	41,1	64,1	64,1	43,4	71,0	50,7	
pop3	70,5	71,1	83,6	84,7	76,9	88,9	24,0	
pop4	65,3	67,5	81,5	84,8	71,5	93,5	54,4	
	65,5	67,9	76,7	80,9	69,1	84,7	39,5	1,29

Tabela A.4: Rezultati algoritma Goto+ZS+TR+OO+VX (najboljša kombinacija brez izravnave z najboljšimi sosedi).

	2004		2005					
	abs.	krom.	abs.	krom.	skupna	vxRec	FA	d'
daisy1	79,2	80,8	89,9	92,4	84,1	99,1	33,8	
daisy2	81,3	81,9	93,5	94,3	86,6	95,5	32,8	
daisy3	81,0	91,0	84,4	95,0	84,4	100,0	—	
daisy4	80,6	80,6	95,5	95,5	84,2	88,7	—	
jazz1	78,4	80,3	91,0	93,2	82,0	97,3	82,4	
jazz2	80,9	80,9	87,3	87,3	84,6	100,0	40,5	
jazz3	76,1	76,1	97,0	97,8	79,4	96,8	47,4	
jazz4	75,2	75,6	86,3	87,0	79,8	96,5	40,3	
midi1	30,4	54,8	32,4	60,8	31,5	97,8	80,9	
midi2	92,8	93,9	96,9	98,0	96,6	100,0	100,0	
midi3	80,7	85,3	85,4	90,2	83,8	99,0	100,0	
midi4	56,6	62,9	59,6	68,4	58,4	91,0	28,5	
op_fem2	36,6	38,2	48,2	50,6	38,5	63,6	36,7	
op_fem4	51,6	51,6	63,2	63,3	55,8	77,4	3,7	
op_male3	17,9	24,9	27,7	41,8	19,1	42,0	1,7	
op_male5	34,4	45,7	44,3	63,7	36,9	59,0	2,1	
pop1	38,7	38,9	41,1	41,3	41,8	93,8	43,5	
pop2	32,4	33,7	51,7	56,3	33,8	65,8	46,2	
pop3	72,4	73,3	81,7	83,1	77,3	99,4	38,5	
pop4	61,9	63,8	80,8	83,4	66,5	91,1	62,4	
	62,0	65,7	71,9	77,2	65,2	87,7	45,6	1,27

Tabela A.5: Rezultati algoritma Goto+TR+OO+NN+VX (najboljša kombinacija brez združevanja spektrov).

	2004		2005					
	abs.	krom.	abs.	krom.	skupna	vxRec	FA	d'
daisy1	85,1	85,6	95,7	96,4	90,5	99,8	27,1	
daisy2	83,7	83,7	95,2	95,4	89,3	95,2	24,9	
daisy3	81,9	84,9	91,9	97,1	85,6	91,8	NaN	
daisy4	89,3	89,4	93,7	93,9	93,7	100,0	NaN	
jazz1	79,2	79,9	87,6	88,4	83,3	97,4	58,4	
jazz2	78,2	78,8	84,5	85,1	81,7	96,8	38,9	
jazz3	83,7	83,9	96,9	97,4	86,5	84,3	7,9	
jazz4	58,8	71,2	71,9	88,7	62,2	86,7	42,4	
midi1	37,8	50,0	45,2	65,7	39,3	74,3	65,6	
midi2	89,4	93,6	93,2	97,7	92,9	100,0	100,0	
midi3	71,6	75,0	83,6	87,8	74,1	84,5	88,5	
midi4	59,0	62,0	65,3	73,9	60,7	76,9	15,3	
opera_fem2	43,3	45,0	62,2	64,6	46,6	69,3	47,5	
opera_fem4	60,0	60,5	71,6	75,0	65,5	77,7	1,8	
opera_male3	20,5	27,2	38,2	51,4	22,1	34,7	3,5	
opera_male5	33,2	44,3	56,4	76,6	35,4	44,6	1,9	
pop1	33,4	33,7	41,5	42,1	35,6	76,9	36,0	
pop2	35,6	35,8	62,8	62,9	37,4	64,0	50,3	
pop3	67,8	70,6	79,6	83,6	72,5	88,3	28,1	
pop4	62,9	67,6	78,9	85,5	68,1	95,2	67,4	
	62,7	66,1	74,8	80,5	66,2	81,9	39,2	1,19

Tabela A.6: Rezultati algoritma Goto+ZS+TR+NN+VX (najboljša kombinacija brez odpravljanja odstopanj).

	2004		2005					
	abs.	krom.	abs.	krom.	skupna	vxRec	FA	d'
daisy1	63,1	68,6	88,2	96,1	68,1	100,0	100,0	
daisy2	74,2	74,8	93,9	94,8	79,6	100,0	100,0	
daisy3	66,7	85,0	69,1	88,4	69,1	100,0	—	
daisy4	88,8	88,9	92,7	92,7	92,7	100,0	—	
jazz1	75,3	81,2	86,6	93,2	79,0	100,0	100,0	
jazz2	71,1	72,6	82,5	84,3	74,4	100,0	100,0	
jazz3	56,5	58,7	93,5	97,2	59,6	100,0	100,0	
jazz4	53,0	64,1	73,9	89,0	56,8	100,0	100,0	
midi1	40,3	58,7	44,6	65,1	42,1	100,0	100,0	
midi2	49,9	68,1	51,7	70,4	51,6	100,0	100,0	
midi3	55,9	64,4	58,8	67,6	57,7	100,0	100,0	
midi4	51,2	58,9	63,7	73,6	53,1	100,0	100,0	
op_fem2	34,7	38,7	49,3	54,6	38,4	100,0	100,0	
op_fem4	51,4	54,6	62,6	66,1	56,8	100,0	100,0	
op_male3	26,3	36,8	32,7	45,6	29,4	100,0	100,0	
op_male5	39,4	52,8	50,2	67,2	43,8	100,0	100,0	
pop1	27,3	28,9	37,8	40,0	30,4	100,0	100,0	
pop2	38,0	38,5	55,9	56,2	41,0	100,0	100,0	
pop3	50,2	55,9	69,6	77,8	54,5	100,0	100,0	
pop4	51,8	58,4	70,5	79,4	56,2	100,0	100,0	
	53,3	60,4	66,4	75,0	56,7	100,0	100,0	0,0

Tabela A.7: Rezultati algoritma Goto (brez sledenja in izboljšav).

	2004		2005					
	abs.	krom.	abs.	krom.	skupna	vxRec	FA	d'
daisy1	68,4	68,7	95,7	96,2	73,9	100,0	100,0	
daisy2	74,9	74,9	94,87	94,9	80,4	100,0	100,0	
daisy3	74,7	89,9	77,6	93,7	77,6	100,0	—	
daisy4	89,3	89,4	93,3	93,3	93,3	100,0	—	
jazz1	78,2	80,8	89,9	93,0	81,9	100,0	100,0	
jazz2	72,3	74,0	84,0	86,0	75,7	100,0	100,0	
jazz3	59,2	59,2	98,2	98,3	62,6	100,0	100,0	
jazz4	56,1	64,8	78,5	90,1	60,3	100,0	100,0	
midi1	44,2	59,3	48,9	65,8	46,2	100,0	100,0	
midi2	84,1	94,1	87,9	98,2	87,7	100,0	100,0	
midi3	78,4	85,7	83,0	90,6	81,5	100,0	100,0	
midi4	55,6	62,3	69,3	77,9	57,7	100,0	100,0	
op_fem2	31,7	36,5	44,4	50,7	34,6	100,0	100,0	
op_fem4	49,8	52,6	60,5	63,8	54,9	100,0	100,0	
op_male3	23,7	34,1	29,3	41,9	26,3	100,0	100,0	
op_male5	38,1	52,0	48,4	65,9	42,2	100,0	100,0	
pop1	29,3	29,8	40,7	41,4	32,7	100,0	100,0	
pop2	40,6	40,8	59,5	59,8	43,7	100,0	100,0	
pop3	58,2	61,2	80,8	85,0	63,3	100,0	100,0	
pop4	53,3	59,0	72,6	80,3	57,8	100,0	100,0	
	58,0	63,4	71,9	78,3	61,7	100,0	100,0	0,0

Tabela A.8: Rezultati algoritma Goto+TR (osnovni algoritem s sledenjem).

	2004		2005					
	abs.	krom.	abs.	krom.	skupna	vxRec	FA	d'
daisy1	83,3	83,6	95,7	96,2	88,8	100,0	34,8	
daisy2	81,7	81,7	94,9	94,9	87,1	96,4	39,0	
daisy3	74,7	88,7	77,6	93,7	77,6	98,8	—	
daisy4	78,3	78,3	93,3	93,3	81,8	88,5	—	
jazz1	80,3	80,8	89,9	93,0	84,0	97,3	76,4	
jazz2	78,8	80,5	84,0	86,0	82,2	98,1	33,9	
jazz3	75,4	75,4	98,2	98,3	78,7	97,4	51,1	
jazz4	69,6	78,2	78,5	90,1	73,7	92,2	40,8	
midi1	43,4	57,1	48,9	65,8	45,2	91,6	70,5	
midi2	68,7	78,6	87,9	98,2	71,4	83,2	0,0	
midi3	78,4	85,7	83,0	90,6	81,5	99,3	100,0	
midi4	66,4	71,5	69,3	77,9	68,5	92,8	26,5	
op_fem2	34,7	38,6	44,4	50,7	36,4	64,4	40,0	
op_fem4	54,9	56,0	60,5	63,8	59,4	84,2	3,7	
op_male3	18,6	25,8	29,3	41,9	19,6	42,4	5,8	
op_male5	36,2	47,5	48,4	65,9	38,9	60,1	2,3	
pop1	35,2	35,7	40,7	41,4	37,9	88,0	47,0	
pop2	29,5	29,7	59,5	59,8	30,6	57,1	51,7	
pop3	72,1	75,0	80,8	85,0	77,2	99,7	35,9	
pop4	55,4	61,1	72,6	80,3	59,7	95,0	73,0	
	60,8	65,5	71,9	78,3	64,0	86,3	40,7	1,33

Tabela A.9: Rezultati algoritma Goto+TR+VX (osnovni algoritem s sledenjem in ugotavljanjem prisotnosti).

Slike

Tabele

Literatura

[1] A. Askenfelt, "Automatic notation of played music: the VISA project," *Fontes Artes Musicae*, št. XXVI, zv. 2, str. 109-120, 1979.

[2] J. Stephen Downie, "The Music Information Retrieval Evaluation Exchange (2005-2007): A window into music information retrieval research,"v zborniku *Acoustical Science and Technology* št. 29, zv. 4, str. 247-255, 2008.

[3] K. Dressler, "Extraction of the melody pitch contour from polyphonic audio,"v zborniku *MIREX Melody Extraction Abstracts*, London, Združeno kraljestvo, 2005, str. 320-323.

[4] R. O. Duda, P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Association of Computing Machinery*, št. 15, str. 11-15, jan. 1972. Dostopno na:

http://www.ai.sri.com/pubs/files/tn036-duda71.pdf

[5] Masataka Goto, Satoru Hayamizu, "A Real-time Music Scene Description System: Detecting Melody and Bass Lines in Audio Signals,"v zborniku *Working Notes of the IJCAI-99 Workshop on Computational Auditory Scene Analysis*, str. 31-40, avg. 1999.

[6] Masataka Goto, "A Robust Predominant-F0 Estimation Method for Real-time Detection of Melody and Bass Lines in CD Recordings,"v zborniku *Proceedings of the 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2000)*, zv. II, str. 757-760, jun. 2000.

[7] Masataka Goto, "A Predominant-F0 Estimation Method for CD Recordings: MAP Estimation using EM Algorithm for Adaptive Tone Models,"v zborniku textitProceedings of the 2001 IEEE International Conference on

Acoustics, Speech, and Signal Processing (ICASSP 2001), zv. V, str. 3365-3368, maj 2001.

[8] Masataka Goto, Keiji Hirata, "Recent studies on music information processing," v zborniku *Acoustical Science and Technology (Acoustical Society of Japan)*, št. 25, zv. 6, str. 419-425, nov. 2004.

[9] Masataka Goto, "A real-time music scene description system: Predominant-F0 estimation for detecting melody and bass lines in real-world audio signals," v zborniku *Speech Communication*,št. 43, zv. 4, str. 311-329, 2004.

[10] J.A. Moorer, "On the transcription of musical sound by computer," *Computer Music Journal*, št. 1, zv. 4, str. 32-38, 1977.

[11] R. P. Paiva, T. Mendes, A. Cardoso , "On the Detection of Melody Notes in Polyphonic Audio," v zborniku *Proceedings of the 6th International Conference on Music Information Retrieval*, London, Združeno kraljestvo, sept. 2005.

[12] G. Poliner, D. Ellis, A. Ehmann, E. Gómez, S. Streich, B. Ong, "Melody transcription from music audio : Approaches and evaluation," v zborniku *IEEE Transactions on Audio, Speech, and Language Processing*, št. 14, zv. 4, str. 1247-1256, 2007.

[13] L.R. Rabiner, M.J. Cheng, A.E. Rosenberg, C.A. McGonegal, "A comparative performance study of several pitch detection algorithms," *IEEE Transactions on Acoustics, Speech and Signal Processing*, št. 24, zv. 5, str. 399-418, 1976.