Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Zeneszámok hangnemének automatikus felismerése

Szakdolgozat

Készítette
Fülöp Tibor

Konzulens
Bank Balázs

2012. május 17.

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott *Fülöp Tibor*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2012. május 17.

Fülöp Tibor
hallgató

Kivonat

A lemezlovasok mixeik elkészítése során a "*Harmonic-Mixing*" DJ-technika segítségével egymással harmonizáló zeneszámokat kevernek egymásra. Ahhoz, hogy ez megvalósítható legyen, ismerniük kell a zeneszámok hangnemét. A dolgozat célja, hogy megtalálja a legalkalmasabb *hangnemdetektáló algoritmusokat* egy DJ-asszisztens szoftver megvalósításához. A dolgozat az irodalom néhány létező hangnemfelismerő algoritmusán keresztül mutatja be a zenei megoldások egy részét, majd az áttekintés után négy detektáló algoritmust valósít meg.

A program alapja a MATLAB környezetben megvalósított keretrendszer, mely képes a megvalósított algoritmusokat *nagy méretű zeneszám-adatbázison* lefuttatni, és mindemellett könnyen paraméterezhető. A detektálási eredményekről LaTeX formátumban jól áttekinthető mérési jegyzőkönyvet generál, és a létrehozott összefoglalás alapján az algoritmusok szabad paramétereivel egyszerűen hangolható. A detektáló algoritmusok - spektrumanalízis segítségével - a dalok hangnemének meghatározását különféle módon kísérelik meg. A módszerek között szerepel a zenei hangok intenzitásainak kinyerése az audiójelből energiaspektrum-sűrűség alapján, időablakonként a legintenzívebb hangok összegyűjtése, a zeneszám akkordmenetének lejegyzése és a talált akkordok hangjainak összegzése, illetve az amplitúdóspektrum jelleggörbével kompenzált lokális maximumainak meghatározása.

A módszerek segítségével és a szabad paraméterek megfelelő megválasztásával sikerült megközelíteni - és bizonyos feltételek mellett túlszárnyalni - egy elterjedten használt hangnemdeteketláló célszoftver találati eredményeit. A dolgozat tartalmazza a szabad paraméterek optimális megválasztásának folyamatát, és kitér a további jelfeldolgozási tecnkikák bevezetésének lehetőségeire.

Abstract

"*Harmonic-Mixing*" is a DJ technique, which enables the DJ to mix songs together based on musical keys. In order to make it achievable, the keys of the songs must be known. The aim of the thesis is to discover the most suitable keydetection-algorithms to create a DJ-assistant software in the future. The first part of the thesis describes known keydetection techniques and algorithms, then it demonstrates four methods implemented by me.

The basis of the program is a framework in MATLAB environment, whitch can analize a huge song database with the created easily customizable algorithms. The results of the measurements are exported in LaTeX format. The framework creates clear reports and a summary of the whole analyzing process, the results of the keydetections are also included in these documents. The free parameters of the algorithms can be tuned easily, based on the exported results. The algorithms are using spectral-analysis to detect the musical keys of the songs. The implemented methods are: musical note intensity extraction from audio signals by energy spectrum density analysis; collection of the most intensive notes from windowed signals; chord-detection and the summation of the identified chord notes; and determination the local maximums of the trend compensated amplitude-spectrum.

With the optimal choice of parameters, meeting certain cicumstances, the methods became nearly the same accurate as a well-known keydetection software (in case of my song database). The thesis includes the method of the free parameter tuning, and covers the future prospects of the program.

Bevezető

A zene egy olyan művészi kifejezési forma, amely a hallgatóban érzelmeket, indulatokat kelt és gondolatokat ébreszt [16]. A dalok előadása egészen a hanglemez megjelenéséig különféle zenekarok, zenészek feladata és privilégiuma volt.

Miután a technika lehetővé tette, hogy a dalokat rögzítsék és lejátsszák, megjelentek a *lemezlovasok* (*Disc-Jockey-k, DJ-k*), akik a - zenés-táncos ünneplések hangulatát irányító - zene fő felelősei lettek. Ahogy telt az idő, a technológia fejlődése egyre több DJ-technika kialakulását tette lehetővé, és a lemezlovasok tehetsége és tudása a technikák alkalmazásán mutatkozott meg a legjobban.

A 21. században a technológia olyan szinten támogatja már a DJ-k munkáját, hogy igazán kitűnni csak tehetséggel, és a legmodernebb DJ-technikák alkalmazásával lehet. Jelenleg az egyik egyre jobban terjedő DJ-technika a "*Harmonic-Mixing*", melynek lényege, hogy a dalok tempójának egyeztetésén felül a DJ ügyel arra, hogy az együtt megszólaló dalok zeneileg is harmonizáljanak egymással.

A DJ-technika alkalmazásához a jó zenei érzék és hallás mellett szükséges, hogy kellő információ álljon rendelkezésre a zeneszámok hangneméről, és a lehetséges hamonizáló kombinációkról. Mivel a dalok hangneme - legtöbb esetben - nem áll közvetlenül a DJ-k rendelkezésére, így azokat meg kell határozni.

A hangnemek meghatározása történhet hallás alapján, vagy különféle hangnemfelismerő algoritmusok segítségével. Ilyen algoritmusok fejlesztése egy olyan témakör, amely ötvözi a zeneelméletet a jelfeldolgozással, így a megfelelő megoldás megtalálásához kreativitás, szakmai-, és zenei tudás is szükséges.

Egyetemi éveim alatt mindig foglalkoztatott a zenei jelfeldolgozás, de általában nem éltem a lehetőségekkel, hogy mélyebben belemerüljek a témába, így ideális lehetőségnek tűnt, hogy a szakdolgozat alkalmával, egy szakértő konzulens segítségével a lehető legnagyobb mértékben sikerüljön megismerkednem a témakör lehetőségeivel.

A hangnemfelismerés, mint megoldandó probléma, akkor merült fel, amikor a DJ-ként tevékenykedő testvéremmel lehetőségeket kerestünk produkciója színvonalának emeléséhez, és a "*Harmonic-Mixing*" technika szóba került. Mivel testvérem a DJ pályafutása előtt nem foglalkozott mélyebben zenéléssel, én pedig 2001 óta aktívan zenélek zenekarokban,

úgy gondoltam kellő mértékben tudnám segíteni Őt. Egy kifejezetten nagy zenei adatbázis hangnemének meghatározására volt szükség, ezért ötletként felmerült, hogy egy erre a célra kifejlesztett szoftvert alkalmazzunk a dalok hangnemének detektálására. A piacon több cég is foglalkozik ilyen szoftverek fejlesztésével, de - a téma érdekessége miatt - inkább egy saját rendszer fejlesztése mellett döntöttem. Ahhoz, hogy egy ilyen célszoftvert meg lehessen valósítani, a megfelelő detektáló algoritmus megtalálása és megvalósítása szükséges.

Az algoritmus fejlesztéséhez és teszteléséhez a MATLAB környezetet választottam, mert így lehetőségem nyílt a program használatában mélyebb tapasztalatokat szerezni, és létrehozni egy komolyabb referenciamunkát.

A zeneszámok hangnemének felismerésére rengeteg ötlet és algoritmus létezik, így a saját algoritmus megvalósításához különböző szemszögekből lehetett megvizsgálni a problémát. A felmerült ötletek mennyisége kinőtte egy algoritmus kereteit, így párhuzamosan több algoritmus fejlesztésébe kezdtem bele. Ezeket az algoritmusokat tesztelni és paraméterezni kellett, így kifejlesztettem egy keretrendszert, amely képes az algoritmusokkal megvalósított módszerek futtatására úgy, hogy könnyen paraméterezhetően, automatizáltan, egy nagyobb - és változatos dalokból álló - zeneszám-adatbázis elemeit dolgozza fel. A detektálási eredményekről részletes, beszédes statisztikákat és jegyzőkönyveket készít.

A konzulensem bemutatta nekem a LaTeX környezetet, melyben elegáns dokumentumokat lehet "*programozás szerűen*" létrehozni, így - a környezetben rejlő lehetőségek miatt - a generált statisztikák, jegyzőkönyvek, és maga a szakdolgozat formátumának is a LaTeX-et választottam. Ezáltal lehetőségem nyílt kellő mennyiségű tapasztalatot szerezni a "nyelv" készség szintű elsajátításához.

A dolgozat az első három fejeztében egy rövid leírást tartalmaz a DJ-technikák fejlődéséről, majd minimális zeneelméleti áttekintést ad, végül különféle hangnemfelismerő módszereket, algoritmusokat és programokat mutat be. A negyedik fejezetben leírja a megvalósított módszerek futtatásához kifejlesztett keretrendszert. Az ötödik fejezetben a megvalósított algoritmusok működését és felépítését mutatja be. A hatodik fejezet röviden áttekinti a zeneadatbázis létrehozásának folyamatát és annak felépítését. A hetedik fejezet bemutatja a bemeneti paramétereket, és az optimális értékek megválasztásának folyamatát. A nyolcadik és kilencedik fejezet összefoglalja a szoftver által elért eredményeket, és megmutatja a lehetséges továbbfejlesztési és felhasználási lehetőségeket. A dolgozat mellékletében egy legenerált jegyzőkönyv található, illetve a program által az *Összefoglalás* fájlba automatikusan kiexportált találati eredményei, futásidők és statisktikák is itt kerülnek bemutatásra.

1. fejezet

DJ keverési technikák fejlődése

A technika fejlődése nem csak a mindennapi életünket változtatja meg, hanem a szórakozás és szórakoztatás módja is jelentősen átalakult. A klubbok, szórakozóhelyek hangulatát a zene szolgáltatója, a DJ tartja kézben. Ahhoz, hogy egy jó lemezlovas a lehető legjobb minőségű produkcióval tudja kielégíteni a közönség zenei igényeit, kifinomult érzéke kell legyen a hallgatóság ízlésének gyors felismeréséhez.

Az idő során a zeneszolgáltatás ezen fajtája mind technológiailag, mind technikailag nagyon jelentős változásokon ment keresztül. Ezt a folyamatot négy jól elkülöníthető szakaszra lehet bontani, melyek a [9] forrás alapján lesznek bemutatva.

A fejezet röviden bemutatja a keverési technikák fejlődését, illetve kitér a szükséges technológiai fejlesztésekre is, amelyek lehetővé tették a keverési technikák kialakulását.

Kezdetben a lemezlovasok legfőbb eszköze a jó hangulat megteremtéséhez csupán a közönség kéréseinek teljesítése volt, mely abból állt, hogy egyszerűen egymás után lejátszották a kért zeneszámokat.

1.1. Egyszerű átkeverés - *"Basic Fades"*

A DJ-k elkezdtek a zeneszámok átkeveréséhez két lejátszót használni, így miközben az egyik lejátszón a zeneszám a végéhez közeledett, a másik lejátszón el tudták indítani a következő dalt. Az átkeverés akkor okozott kellemetlen hangzást, ha a dalok tempói különböztek vagy nem voltak szinkronban (*és megfelelő fázisban*), illetve zeneileg sem passzoltak egymáshoz. Azt a jelenséget, amikor két dal - valamilyen módon - tempóban el van csúszva egymástól *"lódobogásnak"*, vagy *"csattogásnak"* nevezik, az angol terminológia pedig a *"train-wreck"* kifejezést használja. Mivel ez egy olyan hiba, amit a közönség is hamar észrevesz, - és emiatt az előadás a zeneiségét elveszíti, - megpróbálták a jelenséget orvosolni.

1.2. Vágás és Scratch - *"Cutting and Scratching"*

A *"Scratch-elés"* - mint DJ technika - kialakulása a rap-zenéhez köthető. A rap számok gyorsan változó szövegbetétei ideális minták voltak ahhoz, hogy a folyamatosan lejátszott zenébe effektként rövid, ismétlődő betéteket vágjanak a DJ-k. A gyakorlatban ez úgy va-

lósult meg, hogy a DJ a kezével a bakelitlemezt forgás közben megállította, és előre-hátra mozdulatokkal ismétlődő effekteket hozott létre.

1.3. Tempó egyeztetés - *"Beat Mixing"*

A technológia fejlődésének köszönhetően a bakelitlemez-lejátszókban - a régi szíj-hajtású motorok helyett - direkt hajtású motorokat kezdtek el használni. Az új motorok sokkal stabilabban tartották a forgási sebességet, így a motorokból adódó tempóingadozás szinte teljesen megszűnt. A fejlődés egy mérföldköve volt, mikor a **Technics** [15] cég 1972-ben bemutatta az **Technics SL-1200**-as bakelitlemez-lejátszót. A lejátszó - funkciói és a benne rejlő lehetőség miatt - *"sztenderd"* lett, és a Technics - terméke folyamatos fejlesztésével - 1984-ig felruházta a lejátszót mindazon funkciókkal, amelyek lehetővé tették a zeneszámok tempójának szinkronizálását.

Az SL-1200-ban megjelent a *"Pitch-Fader"*, mellyel - bizonyos tartományban - állítani lehetett a tárcsa forgási sebességét. Ez lehetőséget adott arra, hogy az eltérő tempójú zeneszámokat megegyező tempóra lehessen hozni. A lejátszó képes volt arra, hogy a Start gomb megnyomásakor a lemez azonnal a beállított sebességel kezdjen el forogni, így a DJ-k sokkal nagyobb biztonsággal tudták az aktuális zeneszámot a megfelelő időpontban elindítani. Az SL-1200 lehetőséget adott arra is, hogy a lemezt visszafelé játssza le, így a lejátszáshoz nagyon precízen meg lehetett találni a megfelelő ütem kezdőpozícióját.

A technológia fejlődése nem csak a DJ eszközöket változtatta meg, hanem a lejátszott zene hangzása, az alkalmazott hangszerek és a zeneszámok felépítése is jelentősen átalakult. A legtöbb tánczene elektronikus dobokat használ, ami - állandó tempó mellett, - ismétlődő mintákból van felépítve. A zeneszámok és a lejátszók stabil, állandó tempója így lehetővé tette, hogy a zeneszámok átkeverésekor azok tempóját össze lehessen szinkronizálni, és kialakulhasson a tempóegyeztetés technikája.

A zeneszámok - az esetek legnagyobb részében - tartalmaznak dallamhangszereket is, emiatt kellemetlen érzést szülhet a zeneileg igényesebb közönségben, amikor a DJ két olyan dalt választ, melyek zeneileg nem *harmonizálnak* egymással. Mivel a zeneszám tempója a lemez forgási sebességének függvénye, a tempó módosításakor a lejátszott dal hangmagasságában is változás történik[1]. Ahhoz, hogy elkerüljék a zeneileg hamis átkeveréseket, a zeneszerzők a dalok elején és végén több perces, csak dobból álló részeket hoztak létre, hogy a DJ-nek legyen elég ideje a zeneszámokat - a hangnemek ütközése nélkül - tempóban és fázisban is szinkronizálni.

1.4. Digitális technika a hangfeldolgozásban

A DJ-k kezébe a digitális zeneformátumok (például CD és MP3) megjelenésével újabb technikák, technológiák kerültek ahhoz, hogy produkciójuk minőségén még nagyobbat tudjanak

[1] körülbelül *0.66*% tempó változás egy félhangnyi hangmagasság változást jelent

emelni. Mivel a zene digitális jelként van tárolva és feldolgozva, lehetőség nyílt a dal tempóját - speciális algoritmusok segítségével - úgy megváltoztatni, hogy a dal hangmagasságát ez ne befolyásolja. Az ilyen célra kifejlesztett algoritmusokat *"time-stretch"* algoritmusoknak nevezik.

A technológiát beleépítették a modern DJ CD-lejátszókba, és létrejöttek kifejezetten *"DJ-zésre"* kifejlesztett számítógépes célszoftverek is. Ezeket a programokat speciális - erre a célra kifejlesztett - MIDI-kontrollerekkel is lehet vezérelni, így a DJ-k számára egy olyan kezelőfelület áll rendelkezésére, mint a CD-s vagy bakelitlemezes rendszerek esetén. Ezek a szoftverek nagyon jól rendszerezhető zeneszámlistával rendelkeznek és támogatást nyújtanak az automatikus tempóegyeztetéshez, illetve a hangmagasság kézzel történő változtatásához.

1.4.1. Hangnem egyeztetés - *"Harmonic-Mixing"*

A *hangnemek egyeztetésén* azt a DJ technikát értjük, amikor a lemezlovas olyan dalokat, dallamokat kever egymásra, melyek zeneileg *harmonizálnak egymással*[2]. A digitális technika által nyújtott lehetőségek sokkal több szabadságot és lehetőséget adnak a DJ-knek ahhoz, hogy az előadásuk alatt a hangnemek egyeztetésével, és művészibb, kreatívabb produkció létrehozásával tudjanak foglalkozni.

[2]Részletes leírás a 2. fejezetben

2. fejezet

Zenei áttekintés

> *"A zene a hangok és a csend érzelmeket kiváltó elrendezése, létezésének lényege az idő."* [16]

A fejezet egy leegyszerűsített áttekintést tartalmaz a dolgozatban használt zenei fogalmakról, és felvázolja a használt zenei jelölésrendszert. A fejezet a [17] forrás alapján készült.

2.1. A hangnem

A mai nyugati zene nagy része *tonális*, vagyis létezik egy *alaphang* (tonika) és egy alaphangra épülő *hangsor*, amiből a zenemű építkezik. A zenemű hangsorát és alaphangját együtt *hangnemnek* nevezzük [14].

A legtöbb zenei stílusban törekednek arra, hogy csak olyan zenei hangokból építsék fel a dallamokat, harmóniákat, amelyek a választott hangnemen belül találhatóak[1].

A hangsorok közül talán a legismertebb a természetes hétfokú hangsor (más néven diatónikus skála), amely a Kodály-féle szolmizációval jelölve:

dó - ré - mi - fá - szo - lá - ti - dó

Ez a jelölés leírható - a gyakorlatban is használt - *"ábécés"* hangnevekkel, például C-dúr skála[2] esetén (angol elnevezéseket használva):

C - D - E - F - G - A - B - C

Ez a hangsor egy *oktáv* terjedelmet ír le. Az oktávok egymás után ismétlődnek. A skála mind hangmagasságban, mind hangmélységben a végtelenségig folytatható, csak az emberi fül által hallható hangtartomány szab határt a folytatásnak.

A diatónikus skálán kívül létezik még *kromatikus skála* is, mely esetén az oktávot 12 félhangra osztjuk fel.

[1] Egy dalon belül előfordulhat hangnemváltás is, viszont ilyen esetekkel a dolgozatban nem foglalkozunk.
[2] jelen esetben a **dó** -nak a **C** hang felel meg

A **C** hanggal kezdődő kromatikus skála[3]:

C - C♯ - D - E♭ - F - F♯ - G - A♭ - A - B♭ - B - C

2.2. Hangközök

Két zenei hang leírható hangköz elnevezésekkel. A főbb elnevezések **prím, szekund, terc, kvart, kvint, szext, szeptim** és **oktáv**, melyeket felhasználva a teljes kromatikus skálához tartozó hangközök:

Tiszta prím, kisszekund, nagyszekund, kisterc, nagyterc, tiszta kvart, tiszta kvint, kisszext, nagyszext, kisszeptim, nagyszeptim, tiszta oktáv

Ha a hangközt alkotó hangokat megszólaltatjuk, a hangzás lehet

- **harmonikus**: ez a *konszonancia* (önmagában is teljes, harmónikus hangzat). Ilyen hangköz a tiszta prím, kisterc, nagyterc, tiszta kvart, tiszta kvint, kisszext, nagyszext és a tiszta oktáv.
- **diszharmonikus**: ez a *disszonancia*. Ilyen hangköz a kis-, és nagyszekund, kis-, és nagyszeptim.
- **neutrális**: bővített kvart

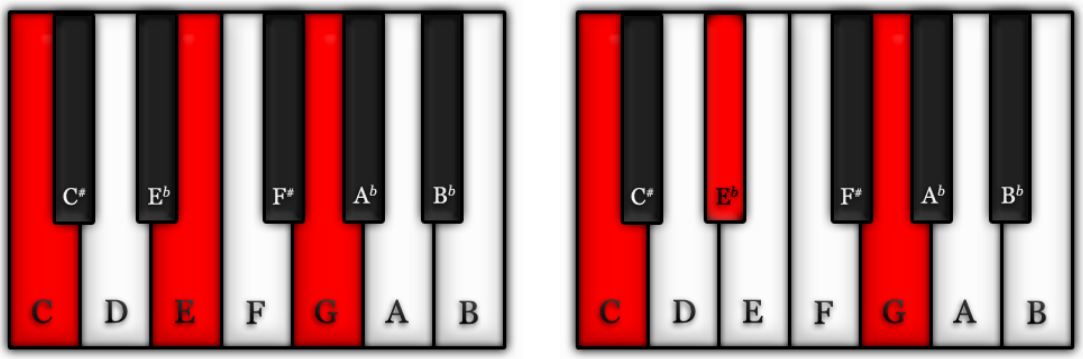
2.3. Akkordok - Harmóniák

Egy harmónia egyidőbeli megszólaltatásához legalább három zenei hang szükséges. Nem kritérium, hogy *konszonáns* legyen, az is harmónia, amely disszonáns hangok összessége.

Megkülönböztetünk *Dúr* és *moll* akkordokat[4], melyek közül a Dúr kissé vidámabb, szabadabb érzést kelt, a moll pedig melankólikusabb hangzású, szomorkás, sejtelmes. Egy C-Dúr és egy C-moll akkord látható zongora billentyűkön felrajzolva a 2.1. ábrán.

[3]A zeneelméletben a hangokra különbőző elnevezéseket is használnak. Pl.: **G♯ - *Gisz*** = **A♭ - *Ász*** (ahol "♯" *("kereszt")* a fél hanggal magasabb hangot, "♭" *("bé")* a fél hanggal alacsonyabb hangot jelöli). Mivel pl a G♯ és A♭ jelentése megegyezik, a későbbiekben a most leírt kromatikus skála hangneveivel fogom jelölni a hangokat.

[4]Természetesen sokkal több féle akkord létezik, de a dolgozatban leírtak megértéséhez elegendő ezt a két fajtát megismerni.



(a) C-Dúr akkord (b) C-moll akkord

2.1. ábra. *C-Dúr és C-moll akkordok*

Az ábrán jól látszik, hogy a hangok között Dúr esetén sorban egymáshoz képest **alaphang(1) - nagyterc(4) - kvint(8)**, moll esetén pedig **alaphang(1) - kisterc(3) - kvint(8)** a távolság[5]. Ennek alapján az összes alaphangra ki tudjuk számítani a Dúr és moll akkordokat. Azért ezek a hangok alkotnak akkordot, mert együttes leütésükkel kellemes összhangzást adnak (*konszonánsak*).

A "**C**" a *C-Dúr* akkord, a "**Cm**" a *C-moll* akkord jelölése (a többi akkordnál értelemszerűen ugyanígy: ha csak az alaphang van jelölve nagybetűvel akkor Dúr akkord, ha mellette szerepel egy kis "m" betű akkor moll akkord). Angol nyelvű kottákban/szövegekben a Dúr akkord és a Dúr skála rövidíteseként **Cma** (*Major*), a moll akkord és moll skála rövidítéseként pedig a **Cmi** (*minor*) jelölést használják[6].

2.4. Skálák

A *hangsort* más néven *skálának* is nevezik. A 2.1. és 2.2. táblázatok fekete betűkkel tartalmazzák az aktuális skála hangjait. Bármely hanggal kezdődően ha sorban beírnánk a táblázatba a kromatikus skála hangjait, megkapnánk a választott kezdőhanghoz tartozó Dúr vagy moll skálát (mivel a hangok távolsága nem változik). Mivel - a jelenleg általunk tárgyalt - tánczenére a Dúr és moll skálák a legjellemzőbbek, a dolgozatban csak ezekkel a skálákkal foglalkozunk.

[5] A hangközök mögött zárójelben lévő szám a félhangok számát jelöli az alaphanghoz képest

[6] Dúr esetén lehetséges még a kicsi háromszög jobb felső indexben, vagy a "maj" rövidítés, moll esetén pedig középen "-" jel, vagy "m" rövidítés

2.4.1. A Dúr-skála

- *Diatónikus* skála (7 hangot tartalmaz)
- **dó** kiindulási alapú, például C-Dúr esetén a **dó** hang a **C** hangnak felel meg
- Vidám, élénk hatású, szabadságérzetet kelt

C	C♯	**D**	E♭	**E**	**F**	F♯	**G**	A♭	**A**	B♭	**B**

2.1. táblázat. *A C-Dúr skála hangjai*

2.4.2. A moll-skála

- *Diatónikus* skála (7 hangot tartalmaz)
- **lá** kiindulási alapú, például A-moll esetén a **lá** hang az **A** hangnak felel meg
- Melankólikusabb, szomorkás, sejtelmes benyomást kelt

A	B♭	**B**	**C**	C♯	**D**	E♭	**E**	**F**	F♯	**G**	A♭

2.2. táblázat. *Az A-moll skála hangjai*

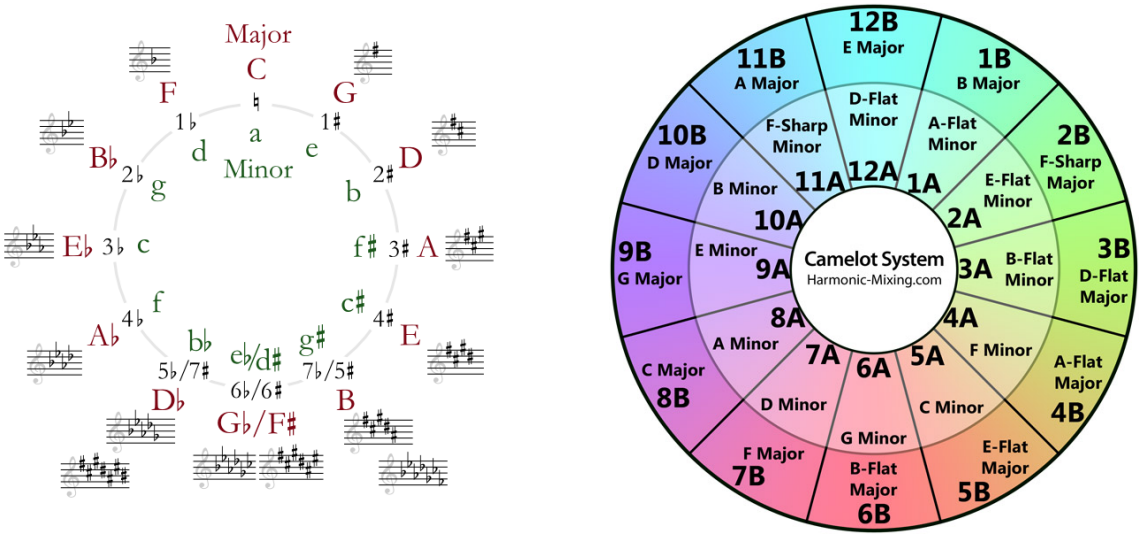
2.5. ”A Kvintkör”

A skálák egymástól való eltérésének jelölésekor is a ”♯” *(”kereszt”)* és a ”♭” *(”bé”)* jelölést alkalmazzák, viszont a hangok különbsége nem ± 1 félhangot, hanem ± 5 egész hangot - vagyis egy *kvintet* - jelent.

Ha összehasonlítjuk a 2.1. és 2.2. táblázatokban a C-Dúr és A-moll skálákat, könnyen felfedezhető, hogy a kiemelt hangok teljesen megegyeznek, de el vannak tolva egymáshoz képest pozitív irányba 8 félhanggal, vagyis egy kvint távolsággal.

A Dúr és a moll hangnemek *előjegyzés* szerinti sorrendjét az alaphangok kvinttávolsága határozza meg. Valamely kiindulóponttól számítva, a kvinttel magasabb hangnem előjegyzése egy *kereszttel* (”♯”) több (egy ♭ kevesebb), a kvinttel mélyebb hangnemé egy *bével* (”♭”) több (egy ♯ kevesebb) mint a kiindulópontté. Mindez a *kvintkörrel* ábrázolható (2.2 ábra). A kvintkörön az óramutató járásával megegyező irányban a keresztek, ellenkező irányban a bé-k száma nő. 12 kvint körüljárása után, enharmónikusan visszaérkezünk a kiinduló pontra [7].

A kvintkör - többek között - arra használható, hogy egyszerűen meg lehessen határozni két hangnem *előjegyzés eltérését*. Az eltérés ismerete esetén eldönthető, hogy a két hangnem zeneileg milyen mértékben passzol egymáshoz.



2.2. ábra. *A "Kvintkör" [13] (balra) és a "Camelot-Easymix" tárcsa [3] (jobbra)*

Két dal akkor harmonizál egymással, ha:

- Hangnemeik megegyeznek (zenei előjegyzésük azonos)
- Dúr-moll párok (vagyis egymás párhuzamos hangnemei)
- Dúr és moll-tól függetlenül, a zenei előjegyzésükben[7] nincs nagy eltérés. (A ± 1 ♯ vagy ♭ még megfelelő.)

A **Mixed in Key** [4] hangnemfelismerő szoftver (`Harmonic-Mixing.com`) készítői átalakították a kvintkör jelölésrendszerét könnyen megjegyezhető szám-betű kombinációkra, hogy a harmonizáló hangnemeket egyszerű összefüggésekkel le lehessen írni. (Ez lehetőséget ad a zeneileg nem képzett előadóknak is a hangnemek egyeztetésére.) A rendszert *"Camelot System"*-nek nevezték el (2.2. ábra). A jelölés lényege, hogy egy kört felosztottak az óralapnak megfelően 12 darab megszámozott részre. Kialakítottak két körgyűrűt, melyek közül a külső gyűrű "**B**" betűvel a Dúr-skálákat, a belső gyűrű pedig "**A**" betűvel a moll-skálákat jelöli. Két dal akkor harmonizál egymással, ha az **A** és **B** betűket figyelmen kívül hagyva a számuk megegyezik, vagy ±1 értékkel tér el[8].

A saját hangemfelismerő algoritmusok, - és az azokat futtató keretrendszer - megvalósításakor a hangnemek előjegyzés-különbségeinek jelölésére a <szám>♯ (pl. 2♯ - két kvinttel magasabb) és <szám>♭ (pl. 1♭ - egy kvinttel mélyebb) jelöléseket használtam. A megvalósított hangnemfelismerő algoritmusok a Dúr-moll párok megkülönböztetésével sem foglalkoznak, így ha az előjegyzés megegyezik, helyes találatként lesz elkönyvelve.

[7] ♯-ek és ♭-k számában
[8] Természetesen itt is léteznek bonyolultabb összefüggések, de ezekkel a dolgozatban nem foglalkozunk.

3. fejezet

Hangnemfelismerő módszerek, algoritmusok, szoftverek áttekintése

Egy zeneszám hangnemének meghatározása teljesen más kihívás egy ember, és egy algoritmusokat futtató számítógép számára. Az ember (*ha van hozzá zenei érzéke*) megpróbálja hallás után megtalálni a dal alaphangját, majd a kiváltott érzelmi hatások alapján jó becslést tud adni arról, hogy a hangnem Dúr vagy moll. Az algoritmusok a zenét, mint jelsorozatot különféle matematikai modellek és módszerek alapján feldolgozzák, és a kiszámolt eredmények alapján valószínűséget rendelnek a lehetséges hangnemmintákhoz.

A fejezet néhány hangnemfelismerő módszert, eljárást mutat be.

3.1. Hallás utáni hangnem meghatározás

Az emberek egy kisebb csoportja - mindenféle segédeszköz nélkül - képes egy hang magasságát, azon túl egy dallam hangnemét annak megszólalása után felismerni. Ezt a tulajdonságot *abszolút hallásnak* nevezzük.

Az emberek azon csoportja akik nem rendelkeznek ezzel a képességgel, - de van zenei érzékük, - egy hangszer segítségével képesek hallás alapján megkeresni az éppen megszólaló hangokat, amelyekből a zene építkezik. Ezen információk birtokában már a zenei skálákból ki lehet következtetni az aktuális hangnemet.

Jól működő módszer az is, ha a dallamnak megkeressük azt a hangját ami a legjobban harmonizál magával a dallammal. Ezt a hangot a zeneelmélet **tonikának**, vagy **alaphangnak** hívja. Az alaphang felismerése után már csak azt kell eldönteni, hogy az alaphangból kiindulva a Dúr-, vagy a moll-skála illeszkedik jobban az aktuális dallamra. Ez abból látszik, hogy a skála harmadik hangja az alaphangtól *Dúr* esetén **nagyterc**, *moll* esetén pedig **kisterc** távolságra van. Az így megkapott hangot az alaphanggal együtt figyelembe véve a zenei skálákból az aktuális hangnem már könnyedén meghatározható.

3.2. Célszoftverek

Mivel nem mindenki képes hallás után a hangnemek meghatározására, és nagyobb menyiségű zenei hanganyag esetén a hangnem felismerése olykor nehézkes, sok idő, és fárasztó,

embert próbáló feladat, ennek ellátására különféle hangnemfelismerő algoritmusokat dolgoztak ki. Néhány cég külön specializálódott ezen probléma megoldására.

3.2.1. Rapid Evolution

A **MixShare** (`MixShare.com`) egy olyan weboldal, ahol DJ-k és zenészek megoszthatják a zenei tudásukat, és egy közös zenei adatbázisba feltölthetik a zeneszámaik metaadatait[1]. Ez a csapat fejleszti a **Rapid Evolution** *ingyenes* zenerendszerező programot, ami offline hangnem-, és tempódetektáló algoritmusokkal is rendelkezik.

A **Rapid Evolution 2.** - és a jelenleg béta tesztek alatt álló **Rapid Evolution 3.** - [6] olyan *ingyenes* zenerendszerező szoftver, melyet arra fejlesztettek ki, hogy támogassa a DJ-ket mixeik elkészítése közben. Lehetőséget ad arra, hogy a zeneszámokról eltároljunk, és - jól átlátható módon - elrendezhessünk a keverés szempontjából olyan fontos információkat, mint a tempó[2] és hangnem. A program kiemeléssel jelezi az egymással kompatibilis zeneszámokat is. Lehetőséget ad arra, hogy a dalok mellé kommenteket fűzzünk, így minden ötletet azonnal a dal metaadataihoz lehet csatolni. A szoftver ideális információs adatbázis a DJ-k számára, elősegíti a zeneszámok hangnemének egyeztetését és segítséget nyújt profi minőségű, kreatív mixek egyszerű és gyors létrehozásához.

3.2.2. Mixed in Key

A **Mixed in Key** [4] (`www.Harmonic-Mixing.com`) egy *fizetős* hangnemdetektáló szoftver és zenei információ adatbázis. A program a dalokat online dolgozza fel, és a termék gyártója az eddig ellenőrzött és feldolgozott zenei adatbázist zenei stílusonként, részletekben is értékesíti.

A programban használt detektáló algoritmust a *zplane.developement* fejleszti, és a **[tONaRT]** nevet viseli.

A **[tONaRT]** [18] egy olyan detektáló algoritmus, amely automatikusan meghatározza az egy-, és többszólamú audiójelek hangnemjegyeit. Az algoritmus különlegessége, hogy - a detektálás pontosságának növelése érdekében - a feldolgozott zeneszám spektrumát speciális módon szűri. Rendelkezik egy - a cég által kifejlesztett - hangszínszűrő-adatbázissal, melyből a felhasznált szűrőket a zeneszám középső frekvenciái alapján hangolja.

[1] Ilyen metaadatok például az MP3 fájlok **ID3 tag**-jeiben tárolt száminformációk. Ez egy jól definiált információ tárolási struktúra, melyet a legtöbb nagy zenelejátszó gyártó is használ. (Bővebben: `www.id3.org`)

[2] A tempó mérőszáma a BPM. A betűszó kibontása *"Beat per Minute"*, és az értéke megmutatja, hogy egy perc alatt hány ütem tellik el.

Az algoritmust a **Mixed in Key** programon kívül több másik szoftverben is alkalmazzák:

- acoustica DJ Twist and Burn
- acoustica MixCraft 3
- algoriddim djay 4 for Mac
- DJUCED 1.x
- Scratch DJ Academy MIX!

3.3. Néhány algoritmus bemutatása

A hangnemfelismerő algoritmusok fejlesztésével már régóta foglalkoznak, viszont a legtöbb megoldás alapjául kotta vagy MIDI jelek szolgálnak, ahol a dal hangjai már ismertek. Kifejlesztettek olyan algoritmusokat is, amelyek képesek a folyamatos audió jelből a zenei információkat úgy kinyerni, hogy később azt fel lehessen használni a zeneszám hangnemének meghatározására. Ezek az algoritmusok alapvetően a spektrumanalízis, statisztikus modellezés és a hallás modellezésének módszereit használják fel. A következtkező algoritmusok leírását a [10] forrás alapján építettem fel.

3.3.1. Zenei hangnem kinyerése audió jelből

Pauws [8] egy olyan algoritmust írt le, amely képes közvetlenül kinyerni a hangnemadatokat a nyers audiójelből. Az ember hangnemfelismerési folyamatait és a zenei szabályokat vette alapul. Az algoritmusa viszonylag *"egyszerű"* és kis számítási igénnyel rendelkezik.

Az algoritmus működése a következő:
Az audiójelből kivág egy tizedmásodpercet, és alulmintavételezi azt. A számítási igény további csökkentése céljából az 5kHz feletti frekvenciákat eldobja, mivel a magas frekvenciák nem befolyásolják a mélyebb frekvenciákat. A megmaradt mintákat *Hamming-ablakkal*[3] megszorozza, majd 1024-pontos FFT-vel meghatározza az amplitúdó-spektrumot.

A spektrumból létrehoz egy 12 elemből álló - úgynevezett - *chroma vektort* (másnéven *chromagram*), ami átalakítja a spektrumot a 12 zenei hangra. Ez például **C** hang esetén 6 darab spektrumtartományt jelent, melyben a tartományok közepén a **C** hang különböző oktávjai találhatóak[4]. Ezek után a chorma vektor normalizálása következik, hogy jól látszódjon - egymáshoz képest - a hangok relatív aránya.

A zeneszám minden tizedmásodpercéhez tartozni fog egy chroma vektor. Miután az összes chroma vektort korrelálja a *Krumhansl hangnem profilokkal* [5], kiderül, hogy melyik

[3] Részletes magyarázat az 5. fejezetben
[4] Konkrét frekvencia értékek: C1 (32.7Hz), C2 (65.4Hz) C3 (130.8Hz), C4 (261.6Hz), C5 (523.3Hz), C6 (1046.5Hz)

hangnemprofilnak a legnagyobb a korrelációja. Ennek a profilnak megfelelő hangnem lesz a legvalószínűbb hangnem.

Pauws az algoritmust 237 darab CD-minőségű zongoraszonátával tesztelte, és 75,1% pontosságot ért el. Ha az eredmények közül a pontos, domináns, szubdomináns és párhuzamos (*Dúr-moll*) hangnemeket is megfelelőnek vette, akkor a pontosság értéke elérte a 94,1%-ot.

3.3.2. Akkord szegmentáció és felismerés, "elvárás maximalizálcióval" tanított rejtett Markov modell segítségével

A szövegfelismerő algoritmusok esetén a *Rejtett Markov modell* (*Hidden Markov Modell - HMM*), és a modell tanítására használt *"Elvárás Maximalizáció"* (*Expectation-Maximization - EM*[5]) népszerűen alkalmazott módszerek.

Sheh [11] megállapította, hogy ezek a módszerek minimális módosítással alkalmassá tehetőek zenei akkordok felismerésére is, mivel analógia fedezhető fel egy felvett beszédben szerepelő szavak sorozata, és egy zeneszámrészletben szerepelő akkordjegyek között.

A módszer *Fujishima* [1] által megalkotott *hangmagasság-osztály profilokat* (*Pitch Class Profile - PCP*) alkalmazza, melyek nagyon hasonlóak a 3.3.1. fejezetben leírt *chroma vektorokhoz*. Ebben az esetben is 12 félhangnak megfelelő vektorban tároljuk el a spektrumból a zenei hangok tartományainak intenzitásait.

A folyamat a jel frekvenciatartományba transzformálásával indul STFT[6] segítségével. Az STFT az átlapolódás nélküli FFT-vel szemben sokkal jobban alkalmas gyorsan változó jelek analizására, mert pontosabb képet ad a jel amplitúdó-spektrumáról.

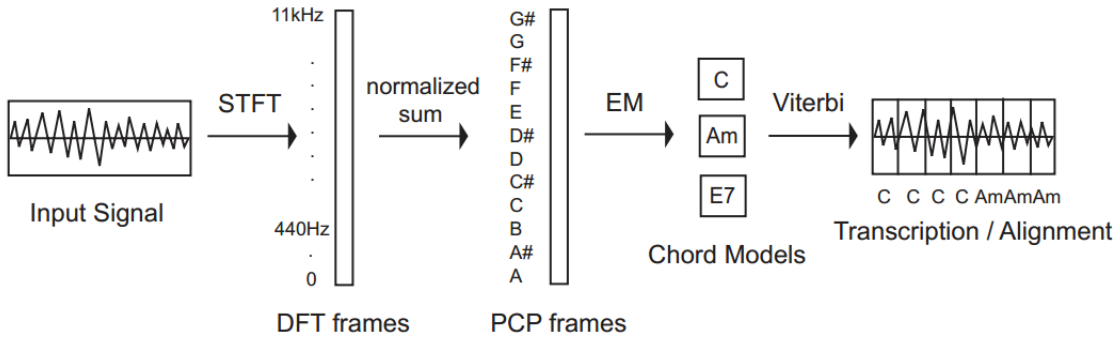
Az algoritmus következő lépésben az ablakozott jel spektrumából kinyeri és eltárolja a - hangmagasság osztályoknak megfelelő - zenei hangokhoz tartozó intenzitásokat. A tárolás 12 elemű *PCP-vektorokban* történik, és a vektorok az oktávonként összegzett, normalizált intenzitásokat tartalmazzák.

Az algoritmust először a kiválasztott tesztfájlok segítségével EM módszerrel megtanítják az akkordok mintáira, és felállítják a *Rejtett Markov Modellt*. A tesztfájlok másik felére lefuttatva az EM algoritmus meghatározza a PCP-vektorok *középértékét* és a *varianciáját*, és ezen adatok alapján ad valószínüséget az aktuális akkordokra. Az adatokat *Viterbi-algoritmussal* rendezi, és a legnagyobb valószínüségű akkord lesz a nyertes.

Az algoritmust egy dalrészletre végigfuttatva, végeredményként a dalrészlet detektált akkordmenetét kapjuk meg.

[5]Egy olyan iteratív technika, amely alkalmas egy befejezetlen adathalmaz legnagyobb valószínűségének meghatározására

[6]Short Time Fourier Transform - A jelet rövid részletekre bontja, amelyek át vannak lapolódva, és ezeken a részleteken végez FFT-t



3.1. ábra. *Folyamatábra* Sheh *algoritmusához [11]*

Ahhoz, hogy az algoritmus jó hatásfokkal működjön, - és a megfelelő modellt lehessen felállítani, - kellő mennyiségű tesztfájlra van szükség, melyek hasonló hangzással és hangszereléssel rendelkeznek. Az algoritmus hátránya, hogy számításigényes, és csak zeneszámok részleteire működik megfelelően.

Az STFT alkalmazásakor - tapasztalat szerint - a hangmagasság-osztály profilok létrehozásához sokkal nagyobb pontosságot lehetett elérni, mint az átlapolódás nélküli FFT esetében, így a módszert célszerű felhasználni egyéb hangnem detektáló algoritmusok megvalósításakor is.

3.3.3. Robosztus Predomináns-F0 becslési módszer valós idejű dallam és basszus felismeréshez

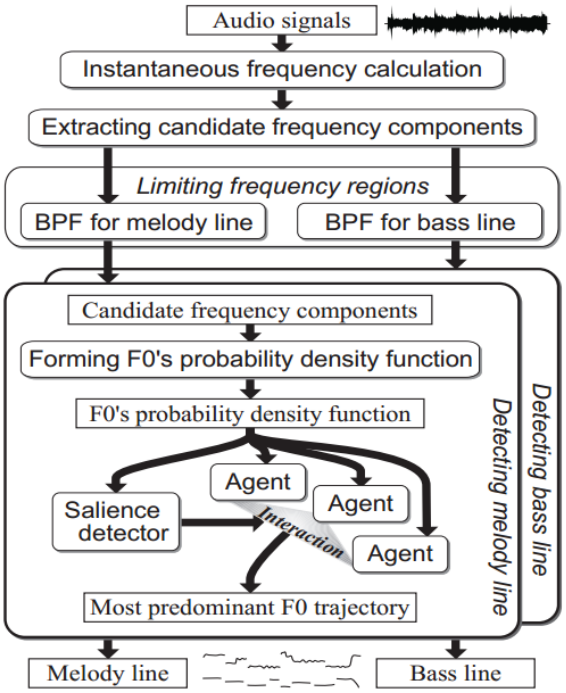
Goto [2] által megalkotott *PreFEst* (*Predominant-F0 Estimation Method*) módszer képes komplex zenei anyagok dallamának és basszusmenetének felismerésére.[7]

A *PreFEst* a zeneszámokról a következő tulajdonságokat feltételezi:

- A dallam és a basszusmenet alapvetően harmónikus hangokból építkezik.
- A dallam a legtöbb esetben a frekvenciatartomány középső és felső hányadában, a basszusmenet pedig a mélyebb tartományban tartalmazza a domináns harmóniákat.
- Mind a dallam, mind a basszusmenet folyamatos, ismétlődő mintákból építkezik

Az algoritmus első lépésben többféle jelfeldolgozási módszer segítségével kiszámítja a pillanatnyi frekvenciákat, majd egy kivonatot készít a talált frekvenciákból. A kivonat alapján megbecsüli az alaphangot (F0), és - mivel a harmonikusaihoz tarozó intenzitások az alaphang intenzitásától is függenek - kompenzálja a harmónikusok értékét. Létrehoz két sávátereszto szűrőt, melyekkel leválasztja a közép és magas frekvenciákat a dallam detektáláshoz, és a mély frekvenciákat a basszusmenet meghatározásához.

[7] A F0 a fundamentális hang, vagyis az alaphang rövidítése.



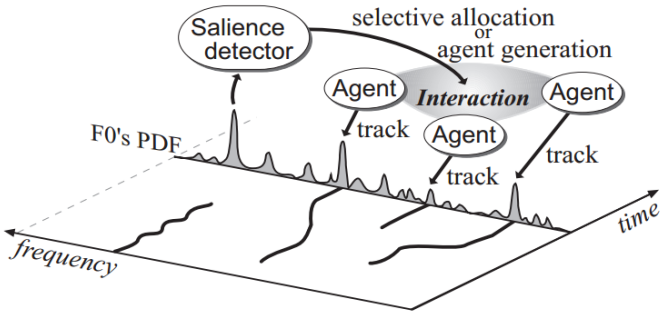
3.2. ábra. *A PreFEst folyamatábrája [2]*

A következő lépéseket mind a dallam, mind a basszusmenet tartományon elvégzi:

Meghatározza az alaphang *valószínűségi sűrűségfüggvényét*, amiből láthatóvá válik annak *relatív dominanciája* az összes harmónikus szerkezethez képest. A valószínűségi sűrűségfüggvény alapján ki lehet számítani a lehetséges harmónikus struktúrák súlyozott értékeit, melyeknek a maximuma lesz a legdominánsabb harmonikus szerkezet.

A becslés *"Elvárás maximalizálás" algoritmus* (EM) segítségével történik (EM: 20. oldal lábjegyzet).

Utolsó lépésben több *megfigyelő* követi a frekvenciák eltolódása által rajzolt pályák trajektóriáját, és a "*nyertes*" egy olyan pálya lesz, ami a lehető legstabilabb, és - amennyire lehetséges - mentes a kiugrásoktól. (Segítség a megértéshez: 3.3.. ábra)



3.3. ábra. *A pályák görbülésének követése [2]*

Az így megkapott görbék megmutatják a zeneszám dallamának és basszusmenetének a változásait, így magának a dalnak a zenei felépítését.

4. fejezet

A Keretrendszer

A fealdatkiírásban leírt követelmények teljesítéséhez többféle hangnemfelismerő-algoritmust kellett megvalósítani MATLAB környezetben. Az algoritmusok paramétereit - a statisztikák alapján - úgy kellett hangolni, hogy azok a lehető legpontosabb eredményt hozzák. A beállítások hatásfokát egy relatíve nagy zeneszám-adatbázison lefuttatott tesztek eredményeinek tanulmányozásával lehet ellenőrizni.

A feladat megoldásához szükséges volt a MATLAB-on belül egy olyan keretrendszer létrehozása, amely:

- Képes többféle algoritmus bemeneti paramétereit beolvasni és előállítani, majd képes az algoritmusokat lefuttatni,
- Az eredményekről, futási időkről részletes és könnyen áttekinthető statisztikákat, összefoglalást és jegyzőkönyvet hoz létre,
- Lehetővé teszi nagy mennyiségű zeneszám automatikus feldolgozását,
- Könnyen paraméterezhető, jól átlátható.

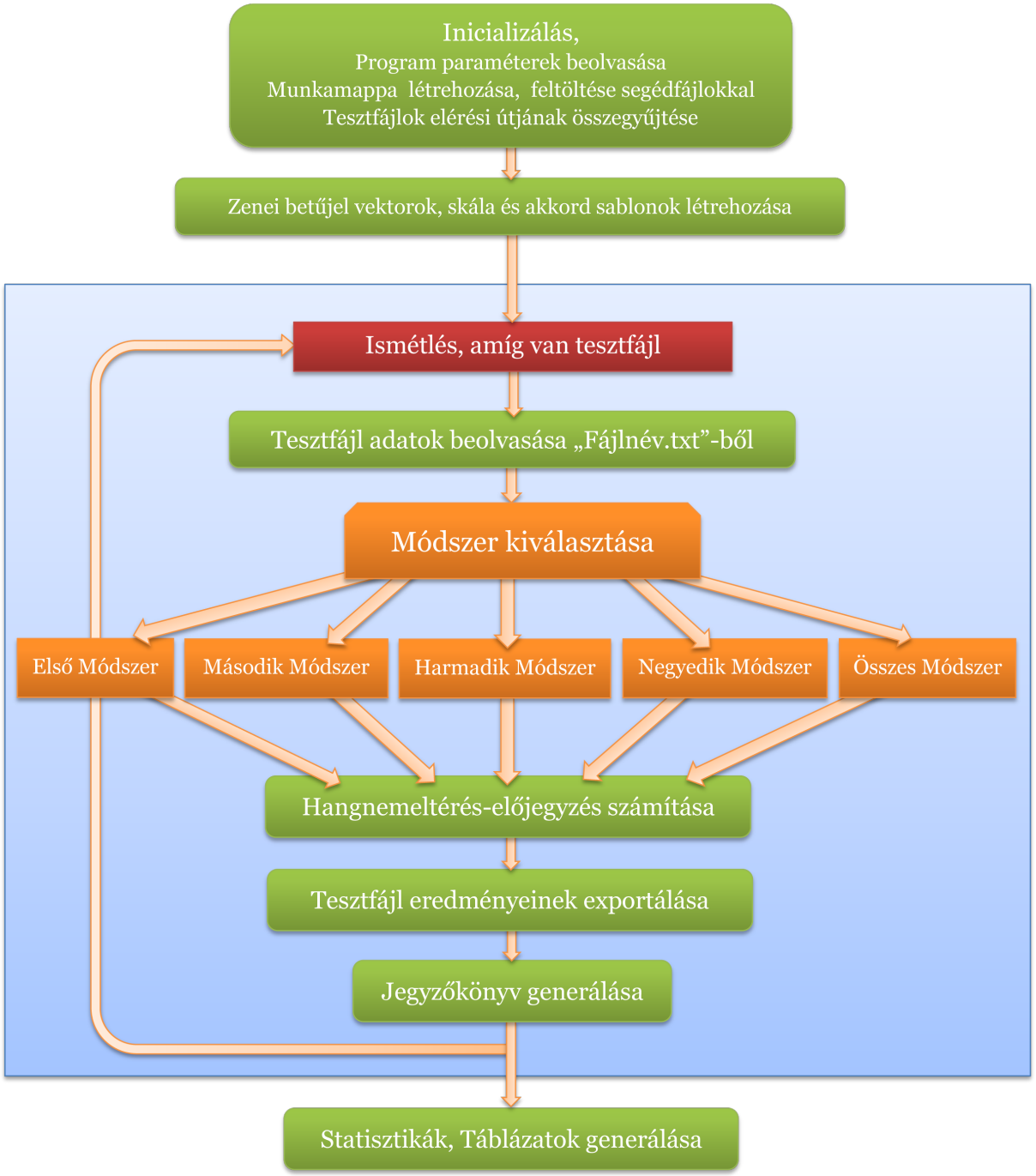
A megvalósított keretrendszer működésének vázlatát a 4.1. ábra szemlélteti.

A következő alfejezetek részletesen bemutatják a program szerkezetének felépítését, és a különböző programblokkok működését. A fejezetekben használt jelölésrendszert a 4.1. táblázat szemlélteti.

Név	**Jelölés**
Fájlnév, Mappanév	`parameters.ini, Reports`
MATLAB változó	`variable_name`
MATLAB függvény	`find_maximums()`
MATLAB ciklus	`while`
Képletben használt változónév	$F_{s_{ds}}$
Új fogalom	*Decimálási faktor*

4.1. táblázat. *A fejezetekben használt jelölésrendszer*

4.1. A keretrendszer



4.1. ábra. *Az algoritmusokat futtató keretrendszer folyamatábrája*

4.1.1. A program gyökérkönyvtárának felépítése

A program - mivel a függvények nagy része relatív elérési utakkal operál - fix felépítésű gyökérkönyvtárral rendelkezik. A könnyebb megértés elősegítéséhez célszerű letisztázni a használt mappanevek jelentését, funkcióját:

- **Functions** - A megvalósított függvények mappája (A MATLAB `addpath` függvénye segítségével a mappa fel van véve a belső függvények mellé, így a meghívásuk semmiben sem különbözik a belső függvényekétől)
- **Reports** - A munkakönyvtárak és jegyzőkönyvek célkönyvtára
- **Templates** - Sablonok gyűjtőmappája
  - **Report** - Jegyzőkönyv LaTeX sablon
  - **Summary** - Összefoglalás LaTeX sablon
- **testfiles_big** - A teljes méretű tesztfájlokat tartalmazó mappa, melyen belül stílusonként kategorizálva találhatóak a WAV fájlok és a dalokhoz tartozó információs fájlok (Bővebben: 4.1.5. fejezet).
- **testfiles_small** - A tesztfájlok egy 20-30 másodperces részletét tartalmazó mappa, melyen belül stílusonként kategorizálva találhatóak a WAV fájlok és a dalokhoz tartozó információs fájlok (Bővebben 4.1.5. fejezet).

A gyökérmappában található a keretrendszert megvalósító `main.m` fájl, és a `parameters.ini`, a konfigurációs paramétereket tartalmazó fájl. A konfigurációs fájl paramétereit a 4.2. táblázat írja le.

Paraméternév	Paraméter jelentés	Alkalmazási hely
`decimate_factor`	Decimálás értéke	Összes Módszer
`time_window`	Ablak hossza másodpercben	Összes Módszer
`window_overlap`	Ablak átlapolódása	Összes Módszer
`matched`	Maximális harmónia találat	Második Módszer
`scale_punishment`	A skálán kívüli hangok büntetésének mértéke	Keretrendszer
`testfile_type`	Tesztfájl típus (`small` - Részlet, `big` - Teljes fájl)	Keretrendszer
`method_number`	Detektáló módszer sorszáma	Keretrendszer
`plot_on`	Grafikonok engedélyezése (0 - Grafikon generálás tiltás, 1 - PDF grafikonok, 2 PNG grafikonok)	Összes Módszer
`range`	Jelleggörbe átlagolás tartománya	Harmadik Módszer
`min_peak_distance`	A minimális csúcs távolság maximumkereséskor	Harmadik Módszer
`threshold_db`	A minimális jelszint maximumkereséskor [dB]	Harmadik Módszer
`frequency_range`	A zenei frekvenciák környékének tartománya	Első és Negyedik Módszer
`min_probability`	A talált akkord minimális valószínűsége	Második Módszer

4.2. táblázat. *A* parameters.ini *változóinak rövid áttekintése*

A paraméterek részletesen az első használatuk helyén lesznek bemutatva.

4.1.2. Inicializálás blokk

A program, miután kiüríti a MATLAB munkaterületét (*workspace*), és bezárja az eddig megnyitott függvényablakokat, a teljes futási idő mérésére elindít egy timer-t. Mivel a program függvényei, jegyzőkönyv és összefoglalás sablonjai a munkamappán belül előre definiált elérési utakon találhatóak meg, a program pontos helyét eltároljuk a `workdir` változóban.

A program minden futtatás alkalmával létrehoz a `Reports` mappán belül egy munkamappát (`report_dir`), melynek a neve a program verziójából és egy időbélyegből tevődik össze[1]. Itt jönnek majd létre a tesztfájlok saját munkamappái, a statisztikák forrásfájlai, és a legenerált összegzés (`summary.pdf`) is majd itt lesznek megtalálhatóak. A mappán belül létrejön egy `Reports` mappa, ami a program lefutása után, az összes fájlhoz legenerált jegyzőkönyvet fogja tartalmazni PDF formátumban.

A következő lépésben a `get_parameters()` függvény beolvassa a `parameters.ini` fájlból a konfigurációs paramétereket. A függvény emellett LaTeX formátumban a `parameters.tex` fájlba exportálja a paraméterek értékeit, és ezt a fájlt az időbélyegezett munkamappába másolja.

A blokk utolsó feladata, hogy a `get_testfile_parameters()` függvény segítségével összegyűjtse a tesztfájlok elérési útjait.

A `parameters.ini` fájl `testfile_type` bemeneti paramétere alapján a függvényen belül dől el, hogy csak fájlrészleteket, vagy teljes zeneszámokat kell feldolgozni. A függvény a tesztfájlok elérési útjainak vektorával (`testfiles_path`) és a tesztfájlok számával (`testfiles_num`) tér vissza.

4.1.3. Zenei betűjel-vektorok, skála-, és akkordsablonok létrehozása

A blokk első lépésben - a 2. fejezetben leírt jelölésrendszernek megfelelően - inicializálja az *"ábécés"* hangok, skálák és akkordok betűjeleinek vektorát.

A Dúr és moll skálák a `major_scale` és `minor_scale` függvények segítségével generálódnak le. A függvények kiindulási alapja egy Dúr és egy moll skálának előkészített sablonvektor. A vektorokban a skálában szereplő hangok "1"-es értékkel szerepelnek[2], a többi hang értéke pedig a `parameters.ini scale_punishment` változójának értékétől függ.

Mindkét függvény esetén - a `scale_punishment` paraméterrel - be lehet állítani a skálákban nem szereplő hangok *büntetésének* mértékét. Ilyenkor a skálán kívüli hangok helyére negatív előjellel a változó értéke fog kerülni, és emiatt a téves skálák könnyebben kiszűrhetőek.

Ezután a függvények a - már megfelelően előkészített - vektorok *cirkuláris körbeforgatásával* hozzák létre a zenei hangokhoz tartozó Dúr és moll skálákat. A program a legenerált

[1] Például: *"v1-0_5__2012_5_26-19_20_13"*

[2] kivéve a moll skála esetén, ahol a Dúr-moll párhuzamos skálák megkülönböztetése érdekében az A♭ és G hangok 0.5 értéket kapnak. Az értékek választását a összehangzatos és melodikus moll skála különböző hangjai indokolják, és ilyen paraméterek mellett pontosabb eredményeket lehet elérni

skálavektorokat a `scale_templates` mátrixban tárolja el.

Az akkordok sablon mátrixának létrehozása ezután következik. A mátrix felváltva tartalmazza a hangokhoz tartozó Dúr és moll harmóniákat, hogy a legenerált grafikonokon könnyebben lehessen értelmezni az eltéréseket.

A blokk utolsó részében a detektált hangok és azok - az eredeti hangnemtől való eltérését megmutató - *előjegyzéseinek* tárolására szolgáló vektorok jönnek létre.

4.1.4. Fájlok feldolgozása

A zeneszámokat a keretrendszer ciklikusan dolgozza fel úgy, hogy egyesével végighalad a `testfiles_path` vektor elemein.

4.1.5. Tesztfájl adatok beolvasása "<fájlnév>.txt"-ből

Az aktuális tesztfájl mappájában található egy - a tesztfájl nevével megegyező - "txt" kiterjesztésű információs fájl, amely a zeneszám legfontosabb paramétereit tartalmazza:

- Fájlnév (`Filename`) - A WAV fájl neve
- Előadó (`Artist`) - A zeneszám előadója
- Dalcím (`Title`) - A zeneszám címe
- Album (`Album`) - A zeneszám albuma
- Hangnem (`Key`) - A valós hangnem[3]
- Célszoftver által detektált hangnem (`RE_key`) - *Rapid Evolution 3* szoftver által detektált hangnem

Miután a program eltárolta a megfelelő változókban a beolvasott értékeket, kijelzi a MATLAB konzolján a feldolgozás alatt álló zeneszám előadóját és a dal címét, majd létrehozza a fájlhoz tartozó - a fájl nevével megegyező - munkamappát.

4.1.6. A módszer kiválasztása

A `parameters.ini` fájl `method_number` paraméter értéke alapján - egy `switch` szerkezetben - a program kiválasztja a futtatni kívánt algoritmusnak megfelelő *Módszert.*

Öt választási lehetőség áll rendelkezésre:

- **Első Módszer** - Hangok energiájából intenzitások számítása,
- **Második Módszer** - Hangok energiájából a legdominánsabb hang kiválasztása időablakonként,
- **Harmadik Módszer** - Akkordok detektálása időablakonként,

[3]Hallás alapján gitár segítségével meghatározott hangnem

- **Negyedik Módszer** - Spektrum jellegének kompenzálása és lokális maximumok keresése,

- **Összes Módszer** - Mind a négy módszer egymás utáni végrehajtása.

A módszerek a 5. fejezet során részletesen bemutatásra kerülnek.

Mindegyik módszer a detektált hangnemmel, a futási idejével, és a detektált hangnem LaTeX formátumával tér vissza. Az értékek a megfelelő mátrixok (`detected_keys`, `elapsed_times`, `detected_keys_tex`) elemeiben tárolódnak el[4].

4.1.7. Hangnemeltérés-előjegyzés számítása

A kiválasztott módszer(ek) lefutása után, a megkapott eredmények alapján meg kell határozni a hangnemeltérés-előjegyzéseket.

Az erre szolgáló függvényben (`calculate_key_offset()`) a kvintkör (2.2 ábra) két vektorral lett megvalósítva. A vektorok - a kvintkörnek megfelelően, egymástól egy kvinttel eltolva - tartalmazzák a Dúr és moll hangnemek betűjelét, és a két vektor azonos indexű elemei a Dúr-moll párokat jelölik.

A függvény első lépésben a vektorok elemeit összehasonlítja az eredeti hangnemmel, hogy meghatározza a vektorokon belül az eredeti hangnem indexét, majd ugyanilyen módon meghatározza a detektált hangnem indexét is. A két indexet egymásból kivonva megkapjuk az egymástól való távolságukat[5]. Ha az elemek túl távol vannak egymástól, - vagyis 7-nél nagyobb az eltérés, - a függvény a szemléletesség érdekében ♭/♯ vagy ♯/♭ konverziót valósít meg. Ez - mivel a kvint-kör körbejárható - 7-nél nagyobb értékek esetén 12 kivonását, -7-nél kisebb értékek esetén 12 hozzáadását jelenti.[6]

Mituán a függvény a detektáló módszerek eredményein és a **Rapid Evolution 3** eredményén is meghatározta a hangnemeltérés mértékét, minden készen áll az eredmények exportálásához.

4.1.8. Tesztfájl eredmények exportálása

A tesztfájlok adatait és a hangnemfelismerés eredményeit a keretrendszer mind "txt", mind LaTeX formátumban `data.txt` és `data.tex` néven kiexportálja a fájl munkamappájába. "Txt" esetén a bejegyzések egyszerűen soronként bekerülnek a fájlba *"változónév = érték"* formátumban. LaTeX fájl esetén - mivel a LaTeX fordító számára néhány karakternek speciális jelentése van - konverziókra van szükség.

LaTeX esetén a "#" jel speciális funkcióval bír, ezért a beolvasott hangnemek változóit a `key_to_tex_key` függvény a "#" és "b" karaktereket a ♯ és ♭ karakterekre cseréli. Az eredeti

[4]Természetesen az "Összes módszer" kiválasztása esetén az *all_methods()* függvény a teljes vektorokat adja vissza

[5]negatív esetben ♭, pozitív estben pedig ♯

[6]Mivel csak egy pozitív vagy negatív számmal térünk vissza, a párhuzamos hangnemeket (*Dúr-moll párokat*) is helyes detektálásnak vesszük

hangnemtől való eltérés esetén a `change_export_symbol()` függvény a negatív vagy pozitív számokat az értéküknek megfelelő ♭-s vagy ♯-es értékekre cseréli, nulla eltérés esetén pedig "**Ok**"-val jelzi a találatot.

A keretrendszer ebben a blokkban fűzi hozzá a - munkamappán belül lévő - `key_summary.txt` és `key_summary.tex` fájlokhoz az aktuális fájl sorszámát, előadóját, dalcímét, a detektált hangnemeket, és azok eredeti hangnemtől való eltérését, az `elapsed_times.tex` fájlt pedig kiegészíti sorszámmal, az előadó nevével, dalcímmel és a négy módszer futási idejével.

4.1.9. Jegyzőkönyv generálása

A 4.1.1 pontban már láttuk, hogy a `Templates` mappa tartalmaz egy LaTeX formátumú jegyzőkönyv sablont. Ez a sablon arra szolgál, hogy jól átlátható formában, összeszedetten lehessen megjelníteni a kiszámolt adatokat, paramétereket, eredményeket és grafikonokat. A sablon felépítése a következő:

- Fedőlap/Címoldal
- A dal adatai
- Konfigurációs paraméterek
- Módszerspecifikus paraméterek
- Detektált és valós hangnemek (táblázat)
- A módszerek és azok eredményei, grafikonjai (Mindegyik módszerhez külön)
  - Módszer rövid áttekintése
  - Módszerhez tartozó grafikonok
  - Módszer által detektált hangnemek, és azok "valószínűségének" értékei, táblázatban összefoglalva

A függelékben egy valós adatokkal legenerált mérési jegyzőkönyv található.

A blokk a sablon munkamappába másolásával kezdődik, majd a keretrendszer kétszer meghívja a LaTeX `pdflatex.exe` fordítóprogramját. Erre azért van szükség, mert a LaTeX fordító "körökre" osztva tölti fel a változók értékeit[7]. Következő lépésben a keretrendszer kimásolja a legenerált PDF fájlt a `Reports` gyűjtőmappába, és a tesztfájl munkamappájába is.

Az aktuális tesztfájl feldolgozása itt ér véget, és a folyamat egészen addig újrakezdődik a következő tesztfájllal, amíg a `testfile_path` vektor utolsó elemét is fel nem dolgoztuk.

[7] Így garantálható például, hogy egy grafikon sorszáma hivatkozásként a szövegben a megfelelő módon jelenik meg.

4.2. Statisztikák, Táblázatok generálása

Ahhoz, hogy a módszerek eredményességét össze tudjuk hasonlítani - és a beállításokat a későbbiekben optimalizálni tudjuk - minden futtatás eredményeiről szükségünk van egy *Összefoglalásra.*

A folyamatok során a program minden eredményt LaTeX formátumban is exportált, így a gyökérkönyvtárban `Templates` mappán belül található *Összefoglalás* sablon (`Summary`) segítségével jól átlátható, beszédes formában tudjuk megjeleníteni azokat.

Az Összefoglalás sablon felépítése:

- Összefoglaló táblázat a módszerek eredményeiről (Módszerenként az eltalált/$\pm$1, $\pm$2 vagy nagyobb $\sharp$ vagy $\flat$ eltérésű hangnemek száma, és ezek százalékban viszonyítva az összes tesztfájl számához)
- A bemeneti paraméterek összefoglalása (mivel hangolás esetén a különböző mérések eredményei csak a bemeneti paraméterek értékeinek ismeretében összehasonlíthatóak)
- Összefoglaló táblázat a teljes tesztfájl-adatbázis detektált hangnemeredményeiről
- Összefoglaló táblázat a módszerek futási idejeiről, zeneszámokra lebontva

A blokkban először egy lépésben kiexportálódnak a futási idők "txt" formátumba, majd meghívódik a `statistics` függvény. A függvény létrehozza a fent leírt összefoglaló táblázatot a módszerek hangnem-detektálási eredményeiről.

Következő lépésben a `Summary` mappát átmásolja az időbélyegezett munkamappába, majd ebben az esetben is kétszer lefut a `pdflatex.exe`, és PDF formátumban létrejön a kész `summary.pdf`. A keretrendszer kimásolja az időbélyegezett mappába az összefoglalást, majd visszatér az eredeti munkakönyvtárba (`workdir`). Végül utolsó lépésként leállítja, majd megjeleníti a teljes program futási idejét a MATLAB konzolon, és kilép.

5. fejezet

A megvalósított algoritmusok bemutatása

A 4. fejezet átfogó képet adott a hangnemfelismerő algoritmusokat futtató keretrendszerről.

Az 5. fejezetben a négy megvalósított hangnemfelismerő módszer blokkjainak részletes bemutatása található.

A keretrendszer lehetőséget ad arra, hogy a különböző módszereket egyesével is le lehessen futtatni, ezért a módszereket megvalósító függvények a feldolgozáshoz szükséges összes lépést tartalmazzák. Annak ellenére, hogy a kód redundáns, és a függvények többszöri meghívása futásidő-növekedést okoz, az algoritmusok felépítése sokkal átláthatóbb marad, emellett a módszerek futási idejéről is objektívebb képet lehet kapni.

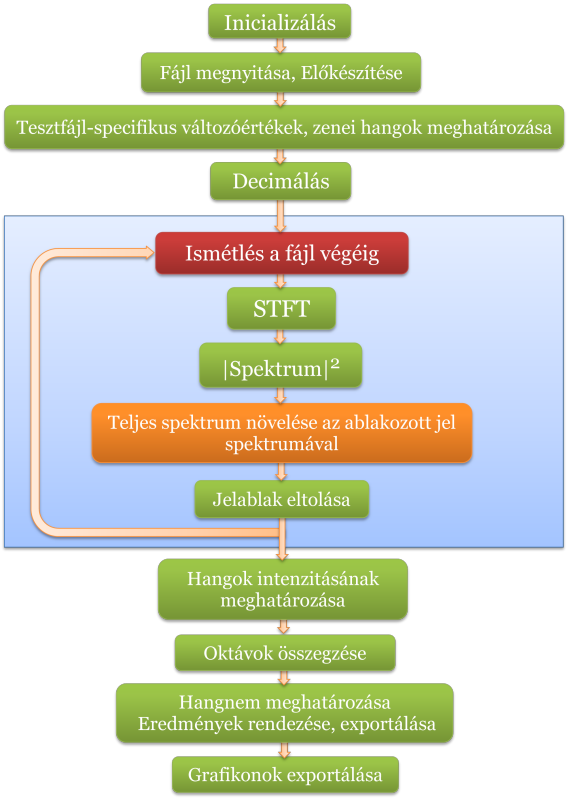
Mindegyik módszer meghívása azonos módon történik: bemenetként megkapja a feldolgozandó tesztfájl nevét (`wav_filename`). A módszer a detektált hangnemmel (`detected_key`) a hangnem LaTeX formában leírt nevével (`detected_key_tex`) és a módszer futási idejével (`elapsed_time`) tér vissza.

A négy hangnemfelismerő módszer az Első Módszeren alapul. A módszer felépítése és lépései hasonlóak a 3.3.1. fejezetben Pauws [8] által leírt módszerhez, mely - összefoglalva - a következő gondolatmenetet követi:

Kivág egy részlet a dalból, *decimálja*, 5kHz feletti frekvenciákat eldobja, megszorozza *Hamming*-ablakkal, FFT-vel meghatározza az amplitúdóspektrumot, spektrumból létrehozza a *chroma-vektorokat* és korrelálja azokat speciális hangnemprofilokkal.

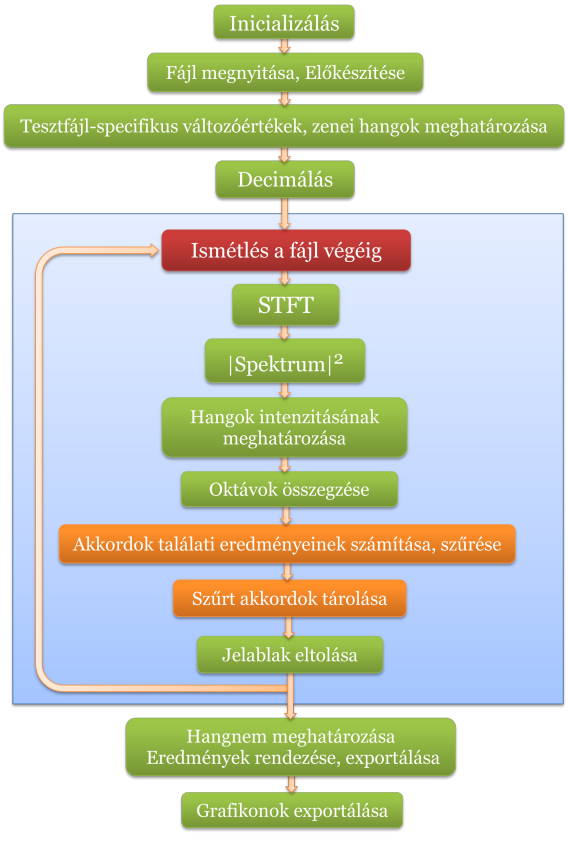
Az Első módszer a spektrum meghatározása után - a chorma-vektorok helyett - a zenei hangok intenzitását számítja ki és - speciális, különböző mértékben súlyozott hangnemprofilok funkcióinak ellátására - egyszerűbb skálasablonokat használ.

A többi módszer esetén az Első Módszer algoritmusát egészítettem ki és rendeztem át - saját ötletek alapján -, hogy minél pontosabb eredmények születhessenek. A hangnem meghatározásán túl - a Harmadik Módszer a dal akkordmenetének felvázolására is kísérletet tesz majd.

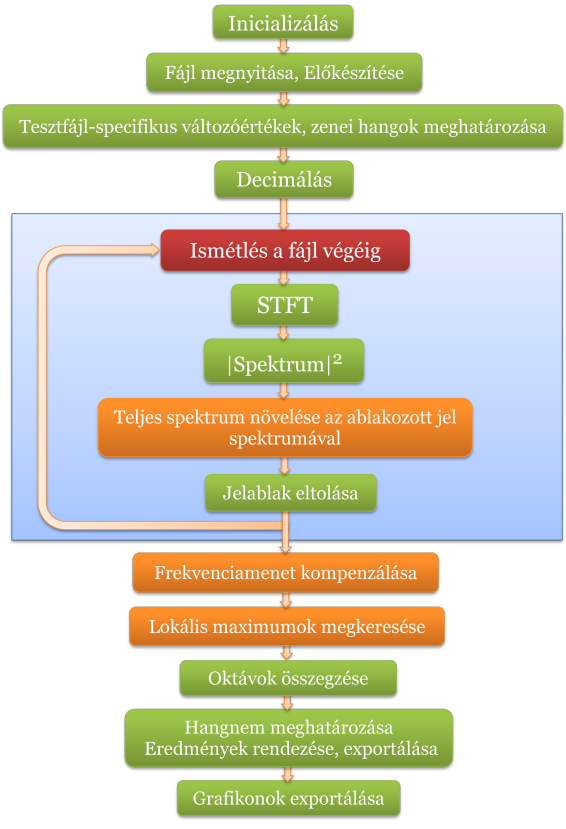


(a) Első Módszer

(b) Második Módszer



(c) Harmadik Módszer



(d) Negyedik Módszer

5.1. ábra. *A Módszerek össszehasonlítása*

A négy detektáló algoritmus két fő csoportra osztható:

1. A Második és Harmadik módszer minden időablakra meghatározza a dalrészlet hangjait, és az így kinyert értékekből következtet a hangnemre (5.1 (b) és 5.1 (c) ábrák)

2. Az Első és Negyedik Módszer az időablakonként meghatározott *energiaspektrum sűrűségfüggvényét* összegzi, és a fájl feldolgozása után a spektrumok összegéből következtet a hangnemre (5.1 (a) és 5.1 (d) ábrák)

Mivel a módszerek ugyanazokból a blokkokból és függvényekből építkeznek, - a blokkok ismertetése előtt - érdemes megfigyelni az 5.1-es ábrán - narancssárga hátterű blokkokkal jelölt - módszerek közötti különbségeket. Az ábrán látszik, hogy a lényegi különbség az algoritmusok között a zenei hangok intenzitásának összegyűjtésében van.

A következő alfejezetekben - az Első Módszer bemutatásán keresztül - értelmet nyernek az 5.1 ábrán - zöld, és piros háttérrel felrajzolt - módszerenként közös blokkok, majd a fejezet második felében a többi algoritmus blokkjairól és működéséről lesz szó[1].

5.1. Első Módszer

Áttekintés: Az Első Módszer a hangnem meghatározásához - az audiójel feldolgozása során - kiszámolja az ablakozott jelek energiaspekrum-sűrűségének az átlagát. Az így megkapott spektrumból meghatározza a zenei hangokhoz tartozó intenzitásokat, és oktávonként összegzi azokat.

Az algoritmus ábrája - a következő oldalon - az 5.2 ábrán látható.

5.1.1. Inicializálás

Első lépésben a módszer elindít egy *timer*-t a futási idejének mérésére, majd átveszi a globális változók értékeit. A globális változók használata - ebben az esetben - azért praktikus, mert így nem kell bonyolult struktúrákat vagy nagy tömböket/vektorokat használni a változóértékek átadásához, és a globális változók a függvény belső változóiként nem foglalnak plusz memóriát sem.

A módszer létrehoz a tesztfájl munkakönyvtárában (`report_dir\filename`) egy saját munkakönyvtárat (például *Első módszer* esetén `Method1` néven), és a teljes elérési utat eltárolja az `act_report_dir` változóban.

5.1.2. Tesztfájl megnyitása, előkészítése

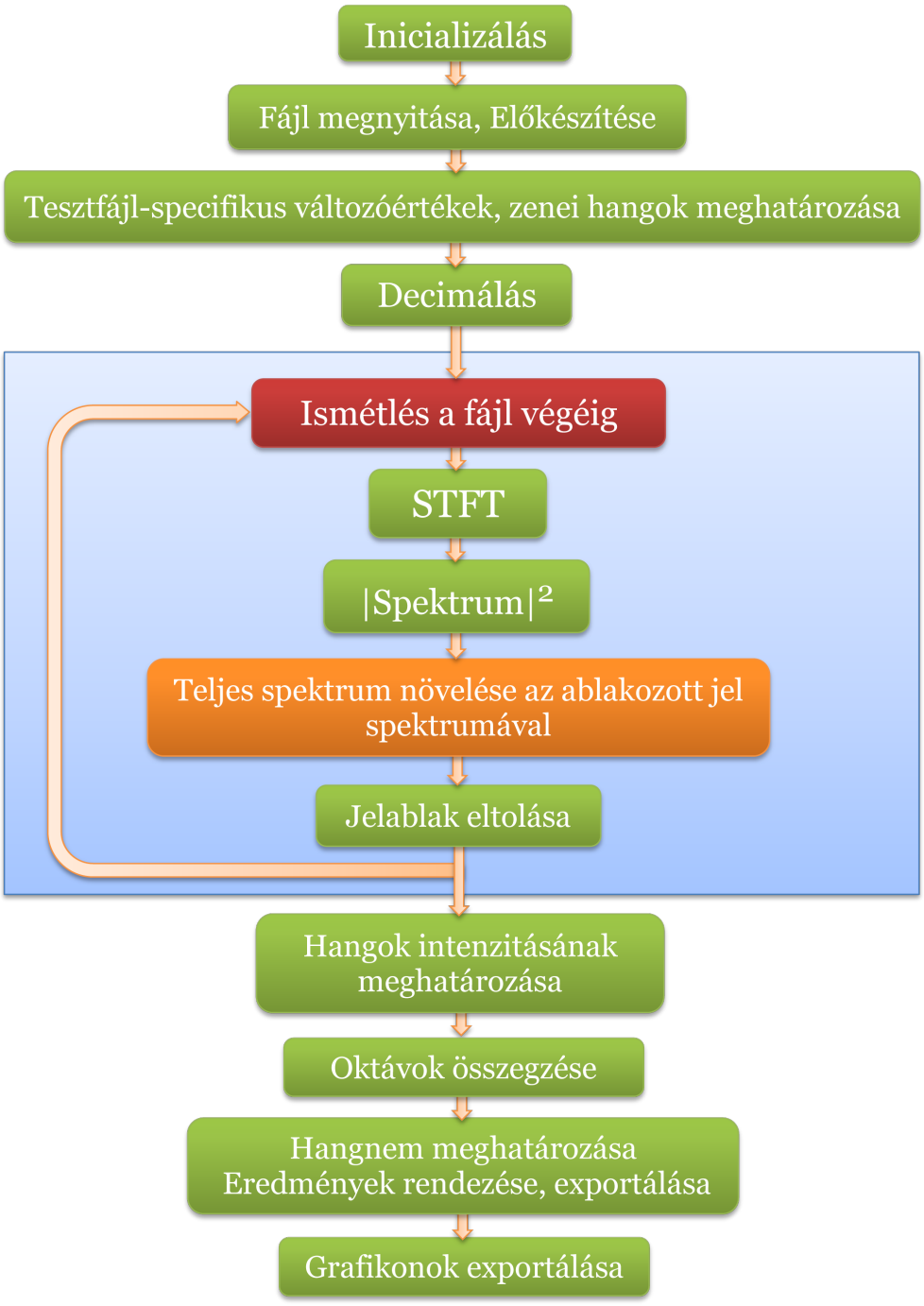
A módszer az aktuális tesztfájlt a MATLAB `wavread(wav_filename)` függvénye segítségével nyitja meg. A függvény a sztereó audiójel bal és jobb csatornájának vektorával (`stereo`), a *mintavételi frekvenciával* (F_s) és a jel felbontásával (`nbit`) tér vissza.

A módszer a következő lépésben - a két oldal mintáinak átlagát véve - a sztereó jelet *monóvá* alakítja. Az így kapott vektort a (`signal`) tárolja el. A *monósítással* a tárolt

[1] Mindegyik módszer egy rövid áttekintéssel és ábrával kezdődik, majd a működésekhez szükséges blokkok bemutatásával folytatódik

adatok mennyisége megfeleződött, és - mivel csak a monó csatornán kell a későbbiekben a műveleteket elvégezni - a számítási igény is csökkent.

A monósítás után már nincs szükség a sztereó jelre, így - mivel a tárolása sok memóriát igényel, - a `clear()` függvény segítségével a `stereo` változó törlődik.



5.2. ábra. *Az Első Módszer ábrája*

5.1.3. Tesztfájl-specifikus változóértékek feltöltése, zenei hangok frekvenciavektorának létrehozása

Az audiójel hossza (T) a `signal` hosszának (N) és az F_s értékének hányadosa (5.1).

`Fs_ds` néven létrejön a *decimált mintavételi frekvencia* ($F_{s_{ds}}$), amely az F_s és a - `parameters.ini` fájlban meghatározott - *decimálási faktor* (`decimate_factor`[2], $Decimate\_factor$) hányadosa (5.2).

Az `N_window` változóban tárolt *ablakszélességet* (ablak mintáinak számát) az $F_{s_{ds}}$ és a - `parameters.ini` fájl - *időablak-hossz* (`time_window`, $Time\_window$) értékének felfelé kerekített szorzata adja (5.3).

$$T = \frac{N}{F_s} \tag{5.1}$$

$$F_{s_{ds}} = \frac{F_s}{Decimate\_factor} \tag{5.2}$$

$$N_{window} = \lceil F_{s_{ds}} * Time\_window \rceil \tag{5.3}$$

Ahhoz, hogy a zenei hangokat azonosítani lehessen, ismerni kell azok frekvenciáit, és - a zenei intenzitások és energiák kiszámításához - a hangokhoz tartozó frekvenciatartományok kezdő-, és végértékét is meg kell határozni.

Ezeket az értékeket a `generate_note_frequency_values()` függvény segítségével lehet meghatározni.

A függvényben a zenei **A** hang frekvenciája ($f_{A_{def}}$) a `def_A_note` változóban $440Hz$-re van állítva. A zenei hangok frekvenciái ennek ismertében a 5.4. képlet alapján számíthatóak. A kiszámolt frekvenciákat (f_{notes}) a `note_frequencies` változóban tároljuk.

$$f_{notes}(i) = f_{A_{def}}\ 2^{\frac{i}{12}} \quad ahol \quad i \in [-30, 80] \tag{5.4}$$

Az i szélsőértékeit úgy választottuk meg, hogy a számításokhoz szükséges maximális frekvenciatartományt fogják át. A bemeneti paraméterek függvényében ezt a frekvenciatartományt úgy kell szűkíteni, hogy a megtartott frekvenciák ne okozzanak - például szivárgás (*leakage*) miatt - hamis mérési eredményeket.

Az első felhasznált zenei frekvencia meghatározásához figyelembe kell venni, hogy az F_s *mintavételi frekvencia* és az *időablak-szélesség* (N_{window}) hányadosa - vagyis az *FFT felbontása* - nagyobb kell, hogy legyen két egymás melletti zenei frekvencia távolságánál, vagyis

$$\Delta f_{notes} < \frac{F_s}{N_{window}} \tag{5.5}$$

[2] A paraméter értékének megválasztásáról a 7.2. fejezetben lesz szó.

A hányados értékének növelésével a mélyebb frekvenciáknál lehetséges szivárgás hatását csökkenteni lehet, ezért a függvényben az $\frac{F_s}{N_{window}}$ hányadost 16-os szorzóval vesszük figyelembe.

Az utolsó felhasznált zenei frekvencia $\frac{F_{s_{ds}}}{4}$-nél kisebb utolsó hang lesz. A magasabb zenei frekvenciákat azért dobhatjuk el, mert nagy valószínűséggel egy mélyebb hanghoz tartozó "sokadik" felharmónikusként jöttek létre, és a hangok intenzitása sem túl jelentős az alaphangokhoz képest.

A függvény a fentebb meghatározott első és utolsó hangok közötti frekvenciákat kivágja az eredeti vektorból, majd ezek lesznek a `note_frequencies` változó elemei.

Egy zenei hang intenzitását nem csak az amplitúdóspektrumban a saját frekvenciáján felvett értékből számítjuk, hanem a frekvencia egy bizonyos környezetének az értékei is hozzájárulnak. A hang körül ezt a tartományt alulról és felülről is határolni kell. A határoló frekvenciák az egymás mellett lévő zenei hangok számtani-közepei[3]. A kapott frekvenciákat a függvény a `notes_limit` változóba tárolja el.

5.1.4. Decimálás

Az erőforrásigény csökentése céljából nem az audiójel összes mintáján végezzük el a számításokat, hanem a jelet előtte *decimáljuk*. A *decimálás* azt jelenti, hogy egy diszkrétidejű sorozat mintáiból minden M-ediket őrizzük meg, és a többit eldobjuk. Ekkor M egész számra decimálunk, azaz az F_s-t M-edrészére csökkentjük, a mintavételi időközt M-szeresére növeljük.

Mivel a mintavételi tételt be kell tartani, a decimálás miatti F_s csökkenés következtében esetlegesen fellépő *aliasing* jelenség elkerülése végett a bemenő sorozat spektrumát sávkorlátozni kell egy digitális aluláteresztő szűrővel [12].

A MATLAB rendelkezik belső `decimate()` függvénnyel, mely elvégzi a sávkotlátozást is, így a módszer következő lépésében a függvény segítségével decimálja a `signal` jelet $M =$ `decimate_factor` értékkel.

5.1.5. Fájl feldolgozása

Az aktuális módszer az audiójelet egy `while` ciklus segítségével rövid időszeletekre vágva - ablakozva - dolgozza fel. A ciklus addig ismétlődik, amíg a folyamatosan eltolt - és átlapolt - időablak utolsó értékének indexe nem nagyobb az audiójel hosszánál. Még a `while` ciklusba lépés előtt a módszer létrehozza - a hangok *energiaspektrum-átlagának* tárolására szolgáló - `all_spectrum` változót, majd belép a ciklusba.

[3] A határok kiszámításához a *mértani-közép* számítása is felmerült, de mivel alig észlelhető volt az eredmények közötti változás, a kisebb számítási igény miatt a számtani-középre esett a választás.

5.1.6. STFT

Az audiójelből a `time_pos` index értékétől számítva `N_window` mintányi jel eltárolódik a `signal_part` változóban. Ez a változó a *Short-Time Fourier-Transzformáció* (`stft()` függvény) bemenete.

A függvény a kivágott jelrészletet összeszorozza a következő ablakozó jelek egyikével:

1 - *Hamming*-ablak

2 - *Rect*-ablak

3 - *Flat-Top*-ablak

Mivel a kezdeti megfigyelések és az irodalom alapján a legjobb eredményeket a *Hamming*-ablakkal való szorzással lehetett elérni, a függvény alapértelmezetten ezt az ablaktípust alkalmazza.

A függvény az ablakozás után FFT segítségével meghatározza az ablakozott jel amplitúdóspektrumát. A MATLAB `fft` függvénye a kiszámolt amplitúdóspektrumot a visszatérési vektorában úgy tárolja el, hogy a pozitív frekvenciatartomány az 1-es indextől a vektor feléig, a negatív frekvenciák tartománya pedig a vektor felétől a végéig szerepel. Mivel a további feldolgozáshoz csak a spektrum pozitív tartománya szükséges, a vektor második felét a függvény eldobja.

5.1.7. $|\text{Spektrum}|^2$

Az ablakozott jel amplitúdóspektrumának - mintánként - az abszolútérték négyzetét véve az eredmény az *energiaspektrum-sűrűség* lesz. Az így kapott értékeket az `abs_square_spectrum` változó tartalmazza, és az `all_spectrum` változó értékéhez adja hozzá.

5.1.8. Jelablak eltolása

Az audiójel ablakonkénti feldolgozásához használt `while` ciklus utolsó lépése az ablak eltolása és indexeinek beállítása. Az ablak új kezdőindexe a 5.6. képlet alapján számolható.

$$Time\_pos = Time\_pos + \lfloor N_{window} * (1 - Window\_overlap/100) \rfloor \tag{5.6}$$

Ahol $Time\_pos$ az aktuális ablak-kezdőindex, N_{window} az időablak szélessége, $Window\_overlap$ pedig a `parameters.ini` fájlból beolvasott *ablak-átlapolódás* százalékban.

Az ablakok számát a `counter` nevű számláló növelésével lehet számon tartani.

Miután a ciklus véget ért, az *energiaspektrum-átlag* meghatározásához a megkapott spektrumösszeget az ablakok számával (`counter`) el kell osztani.

5.1.9. A hangokhoz tartozó intenzitások meghatározása

A hangok *intenzitását* a `sum_notes` függvény segítségével lehet meghatározni. A függvény - a bemenetként kapott energiaspektrum-sűrűségből - meghatározza a zenei hangokhoz tartozó energiák értékét.

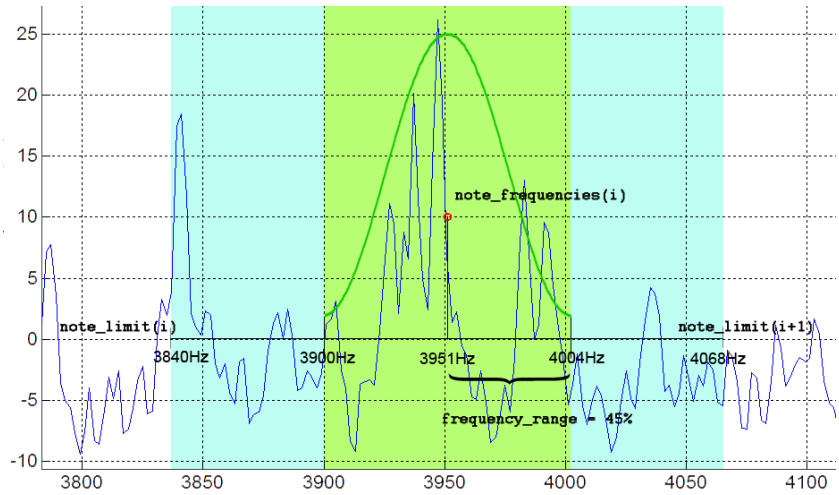
A segédváltozók deklarálása után a függvény egy `for` ciklusban sorra veszi az összes zenei frekvenciát, majd meghatározza az aktuális frekvencia energiatartományának indexeit. Az alsó és felső tartományindexeket - az egymáshoz közel lévő zenei hangok áthallásának mérsékelése érdekében - közelebb kell elhelyezni a zenei hang frekvenciájához.

Ez két lépésben valósul meg:

1. Az eredeti tartomány alsó és felső frekvenciájának és a zenei hang frekvenciájának a távolságát megszorozzuk a `parameters.ini` fájl `frequency_range` változójában tárolt százalék értékkel. Az így kapott távolságot a zenei hang indexének értékéből egyaránt kivonva és hozzáadva megkapjuk az előbb említett esetben az új alsó, utóbbiban pedig az új felső indexet.

2. Az így megkapott frekvenciatartomány értékeit összeszorozzuk egy *Hamming*-ablakkal[4]

A folyamat könnyebb megértéshez egy az 5.3. ábra alapján egy példa: A $3951Hz$-hez tartozó zenei hang eredeti frekvenciatartományát (világoskék és zöld terület) a 3840 értékű alsó indexe, és a 4068 értékű felső indexe határol. A `frequency_range` $= 45\%$ paraméter esetén a felhasznált frekvenciatartomány $3900Hz$ és $4004Hz$ közöttire csökken (zöld háttérrel jelölt terület). Az így megkapott frekvenciatartomány értékeit úgy súlyozzuk, hogy a minták értékeit összeszorozzuk a világoszöld vonallal ábrázolt *Hamming-ablak* értékeivel. Az eredmény a végleges súlyozott frekvenciatartomány.



5.3. ábra. *Példa a zenei hang frekvenciatartomány-indexeinek meghatározásához*

[4]A *Hamming*-ablak karakterisztikája miatt ideális egy frekvenciákat tartalmazó jel szélső frekvenciakomponenseinek elnyomására.

Az így megkapott frekvenciatartományt átlagolva és abszolútértékét véve a végeredmény az adott zenei hang aktuális *"intenzitása"* lesz. Miután a folyamat az összes zenei frekvenciára véget ér, a függvény az intenzitásokat tartalmazó `sum_note` vektorral tér vissza.

5.1.10. Oktávok összegzése

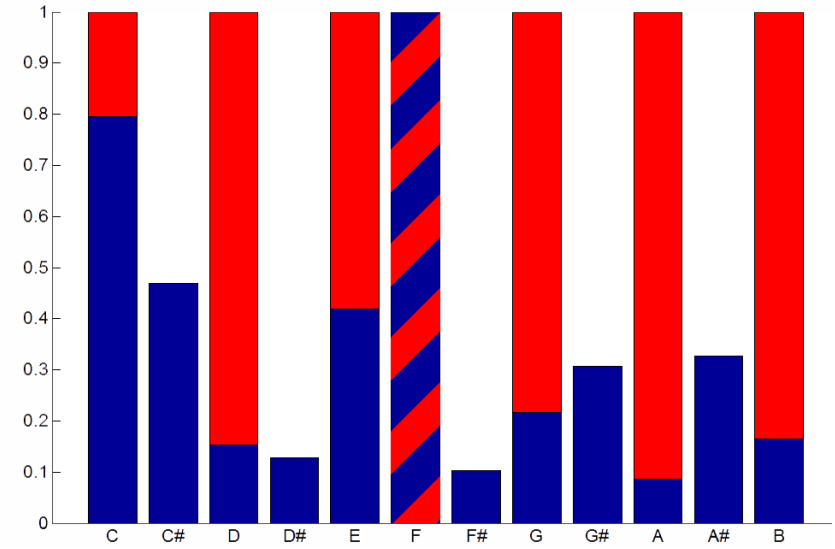
A `sum_octavs` függvény a zenei hangok oktávjainak intenzitásait egy 12 elemű `octav_sum` vektorban összegzi. A folyamat során a függvény egyszerűen `for` ciklusban tizenkettesével összeadja a bemeneti változóként megkapott `sum_note` vektor elemeit.

A végeredményt *normalizálja* úgy, hogy a vektor legnagyobb értékével elemenként leosztja a vektor értékeit.

5.1.11. A hangnem megállapítása, eredmények rendezése, exportálása

A detektált hangnemekhez rendelt *találati eredmények* a 12 zenei hang intenzitásából álló vektor (`octav_sum`) (vagy a Harmadik Módszer esetén az akkordok összegét tartalmazó vektor (`sum_chords`)) és a skálasablon (`scale_templates`) $[24 \times 12]$ mátrixának összeszorzásakor születnek meg.

Ha egy hangintenzitás-vektort (pl. `octav_sum`) skalárisan szorzunk egy skálasablon vektorával, az azt jelenti, hogy minden hangot a skálasablonban szereplő hangjának értékével súlyozunk, majd összeadjuk az így kiszámolt értékeket. A 5.4. ábra egy dal hangintenzitás-vektorát (kék oszlopok) és a C-Dúr skálasablon értékeit (piros oszlopok) mutatja (`scale_punishment`[5] `= 0` esetén)[6]. Ebben az esetben - mivel a skálasablon értékei a skálán belüli hangok esetén 1, skálán kívüli hangok esetén pedig 0 értékűek, - a C-Dúrhoz tartozó találati eredmény a piros és kék oszlopok metszetének összege lesz.



5.4. ábra. *Egy dal hangintenzitás-vektora és a C-Dúr skálasablon*

[5] A változó funkciójáról bővebben a 4.1.3. fejezetben

[6] A sraffozott **F** hang azt mutatja, hogy az intenzitásvektor és a skálasablon **F** értéke fedésben vannak egymással.

Ilyen módon mindegyik hangnemhez tartozni fog egy találati érték, melyek sorbarendezésével a legmagasabb értéket elért hangnem lesz a "*nyertes hangnem*".

A sorbarendezés úgy valósul meg, hogy létrehozunk egy $[24 \times 2]$ elemű mátrixot, melynek első oszlopában a zenei sablonokhoz tartozó találati eredmények, második oszlopában pedig azok sorszáma szerepel 1-től 24-ig. A MATLAB a `sortrows()` és `flipud()` függvényei segítségével soronként csökkenő sorrendbe rendezi az értékeket. A legmagasabb értékeket elért hangnemek a sorszámukkal együtt a mátrix tetejére kerülnek. A sorszámokhoz az *"ábécés"* hangoknak megfelő hangem társul, és a találatok a `matched` és `matched_tex` vektorokban lesznek eltárolva. A hangnemek a találati értékeikkel együtt - txt formátumban - a `key.txt` fájlba kerülnek, a nyertes hangnem neve pedig a `key` és `key_tex` változókba tárolódik el a megfelelő formátumban.

A jegyzőkönyv számára LaTeX formátumban az összes találati eredmény bekerül egy táblázatba a hozzájuk tartozó betűjellel együtt, melyet a `export_keys_table_to_tex()` függvény hoz létre, és exportál ki az aktuális módszer munkakönyvtárába `keys.tex` fájlnéven.

5.1.12. Grafikonok exportálása

A grafikonok engedélyezését és beállításait a `parameters.ini` fájl `plot_on` változójának módosításával három lehetséges értékre állíthatjuk:

0 - A grafikonok generálásának kikapcsolása

1 - Grafikonok exportálása PDF formátumban

2 - Grafikonok exportálása PNG formátumban

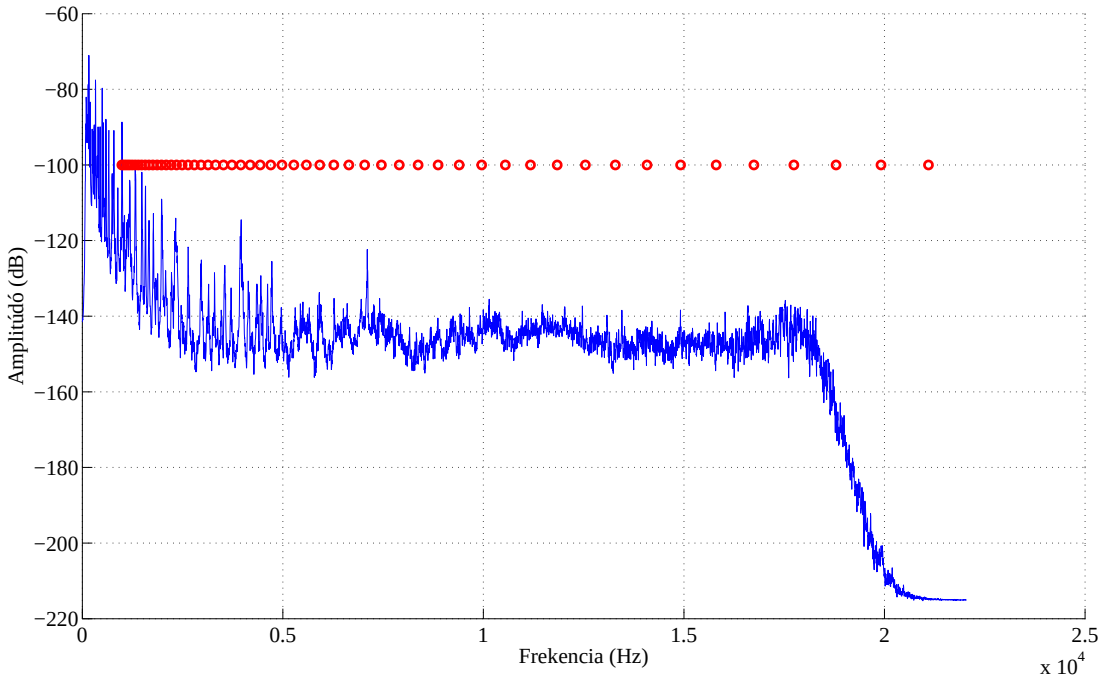
A fejezetben ismertetett grafikonokra példákat az F.2 függelékben szerepelő mérési jegyzőkönyv, és a fejezet utolsó oldala tartalmaz.

Ha az ábrák generálása engedélyezve van, a blokk előkészíti az ábrázolni kívánt görbéket az exportáláshoz.

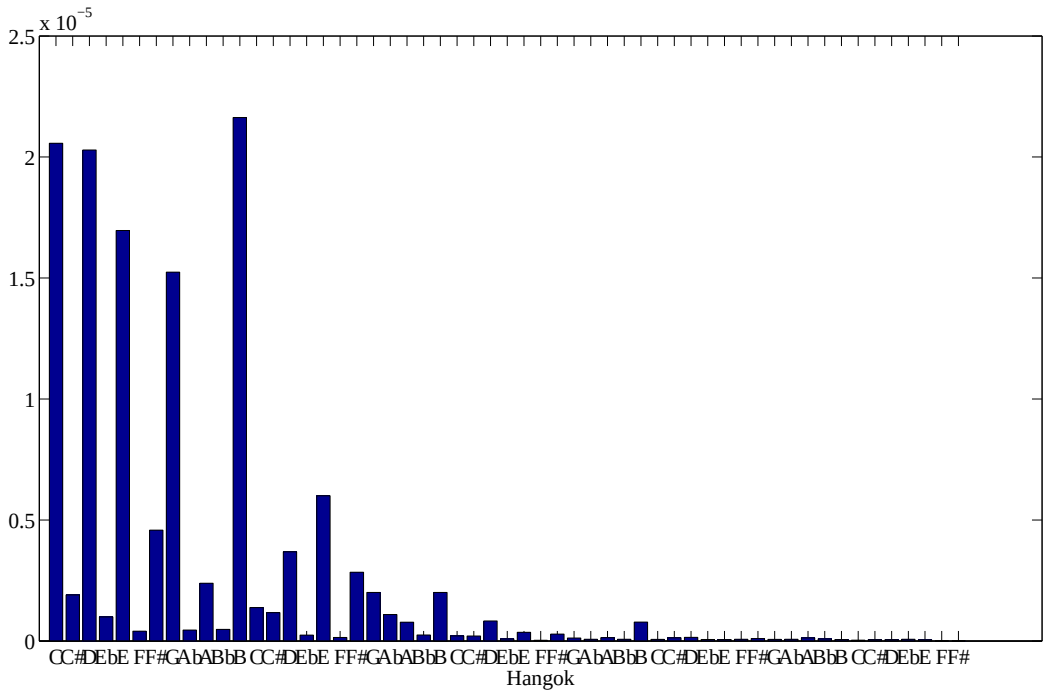
A módszer esetén az amlitúdóspektrum ábrája - piros körök segítségével - a zenei hangok frekvenciáit is megjeleníti. Ez segít eldönteni, hogy a spektrum kiugró csúcsai zenei hanghoz tartoznak-e. (Példa: 5.5. ábra)

A hangok, és az oktávonként összegzett hangok intenzitásának az ábrázolása a MATLAB - oszlopdiagrammok rajzolására kifejlesztett - `bar()` függvényével történik A generált ábrákon így jól látható, hogy az intenzitások hogyan arányulnak egymáshoz. (Példák: 5.6 és 5.7. ábrák).

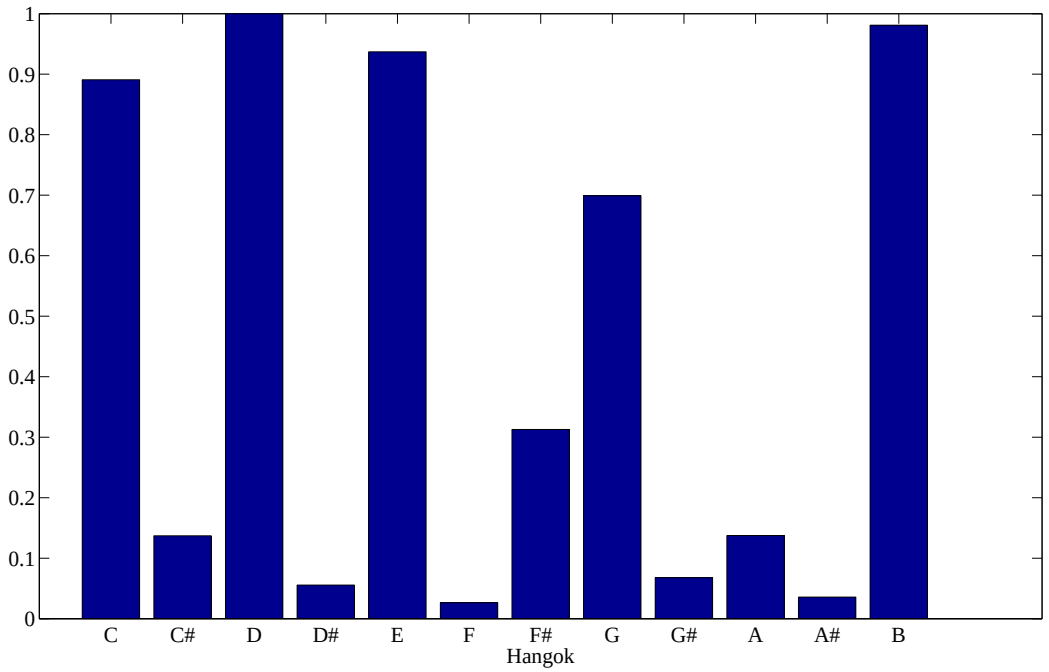
A grafikonok exportálása a MATLAB `print()` függvénye segítségével valósul meg - a beállított fájlformátumban - az aktuális módszer munkamappájába.



5.5. ábra. *Példa az amplitúdóspektrum és a zenei hangok ábrázolására*



5.6. ábra. *Oszlopdiagram példa a hangok intenzitásának ábrázolására*



5.7. ábra. *Oszlopdiagram példa az oktávonként összegzett hangok intenzitásának ábrázolására*

A függvény az utolsó lépésben megállítja a futási idő mérésére elindított *timer*-t, és a nyertes hangnem eredménnyel, és a futási idővel visszatér a keretrendszerbe.

5.2. Második módszer

Áttekintés: A Második módszer légyege, hogy időablakonként meghatározza a legintenzívebb hangot, és az értékeit egy vektorban hangonként összegzi[7].

Az algoritmus ábrája az 5.8 ábrán látható.

5.8. ábra. *A Második Módszer ábrája*

[7]Az alapkoncepció szerint a négy legintenzívebb hangot határoztuk volna meg, de az így elkészített algoritmus sokkal pontatlanabb eredményeket hozott, mintha csak a legintenzívebb hangot használtuk volna fel.

A Második módszer fő különbsége az Első módszerhez képest, hogy minden időablak esetén kiszámolja a zenei hangokhoz tartozó intenzitásokat (5.1.9), és oktávonként összegezi az eredményeket (5.1.10).

A módszerek különbségeinek könnyebb megértéséhez a folyamat kulcsszavakban:

Jelablakonként (Energiaspektrum sűrűség → Hangintenzitás → Hangintenzitások összege oktávonként → Legintenzívebb hang tárolása) → A tárolt legintenzívebb hangok összegét tartalmazó vektorból hangnem

5.2.1. A legintenzívebb hang megkeresése és tárolása

A legintenzívebb hang az `octav_sum` változó egyszerű sorbarendezésével megállapítható.

Mivel a hang az aktuális időablakon belül nem változik, az ábrázoláshoz a blokk ablakonként folyamatosan két segédvektor értékét tölti fel. A `time_pos` vektorban egymás után tárolódik az időablak elejének és végének az indexe, a `notes_vector`-ban pedig a hang intenzitásának értékei szerepelnek. Ez azért praktikus, mert a két vektor időablakonként így sokkal kevesebb elemet tartalmaz, és ha minden időpillanatban letárolnánk az aktuális hang értékét, több mint 3 nagyságrenddel nagyobb - N_{window} - tárolókapacitásra lenne szükség.

A blokk ezek után eltárolja a legintenzívebb hang értékét a `sum_note_intensities` vektorban.

A jelfeldolgozási ciklus az audiójel végéig ismétlődik, és folyamat végén a `sum_note_intensities` vektor lesz a további lépésekben használt hangintenzitás-vektor.

A hangnemdetektálási folyamat következő lépése megegyezik az Első Módszerrel, vagyis a program - a `sum_note_intensities` értékei alapján - megállapítja a hangnemet, majd rendezi és exportálája az eredményeket a 5.1.11. fejezetnek megfelelően.

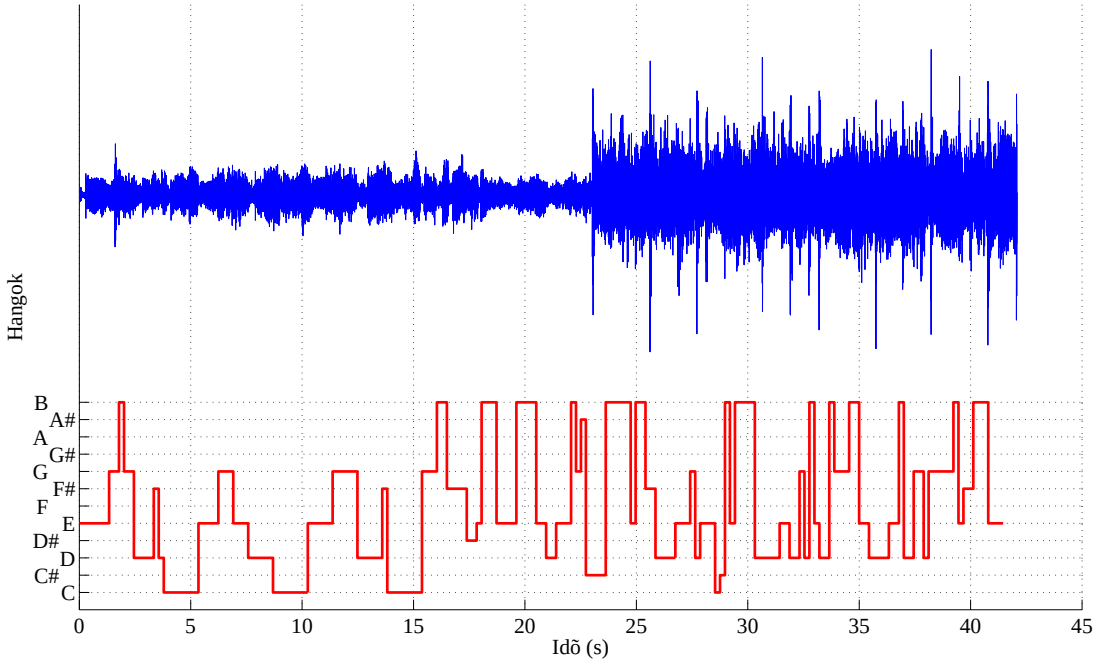
5.2.2. A grafikonok exportálása

A módszer esetén egy közös grafikonban ábrázoljuk a jel időfüggvényét, és a detektált legintenzívebb hangokat (Példa: 5.9. ábra). Ahhoz, hogy jól látszódjanak a hangváltások helyei, az időfüggvényt az y tengelyen kellő mértékben el kell tolni felfelé.

A hangok azonosításához az y tengely az *"ábécés"* hangoknak megfelelően fel van címkézve, és rácsvonalak is segítik a hangok könnyebb beazonosítását.

Mivel hosszabb zeneszámok és kis időablak-méret esetén a grafikon - a sűrű, vastag vonalak miatt - átláthatatlanná válna, a blokk 30 másodperc feletti dalhossz esetén csökkenti a vonal vastagságát.

Az időfüggvényt - mivel csak tájékozódásra szolgál - nem szükséges a teljes felbontásában ábrázolni. Tapasztalat szerint elegendő csak minden 40. mintát szerepeltetni, így a grafikonok kiexportált méretének csökkentése céljából itt is ezt az értéket alkalmazzuk.

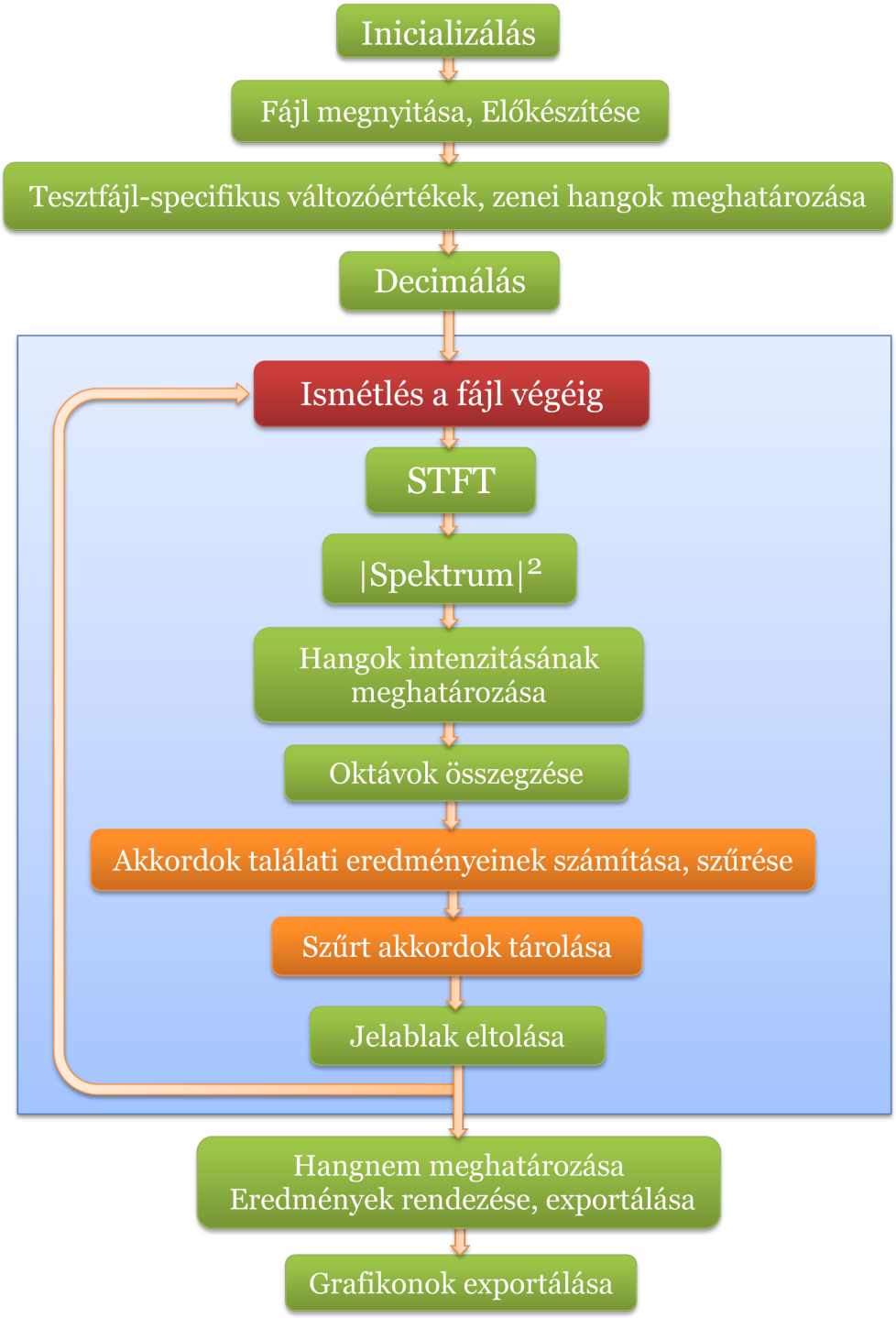


5.9. ábra. *Példa a legintenzívebb hang és az időfüggvény ábrázolására*

5.3. Harmadik Módszer

Áttekintés: A Harmadik Módszer légyege, hogy időablakonként meghatározza az aktuális akkordot[8], és az akkordok értékeit egy vektorban hangonként összegzi.

Az algoritmus ábrája az 5.10 ábrán látható.



5.10. ábra. *A Harmadik Módszer ábrája*

[8]Az akkordok bemutatása a 2.3. fejezetben található

A Harmadik Módszer - a Második Módszerhez hasnolóan - időablakonként meghatározza a hangintenzitás vektorokat, majd oktávonként összegzi azokat.

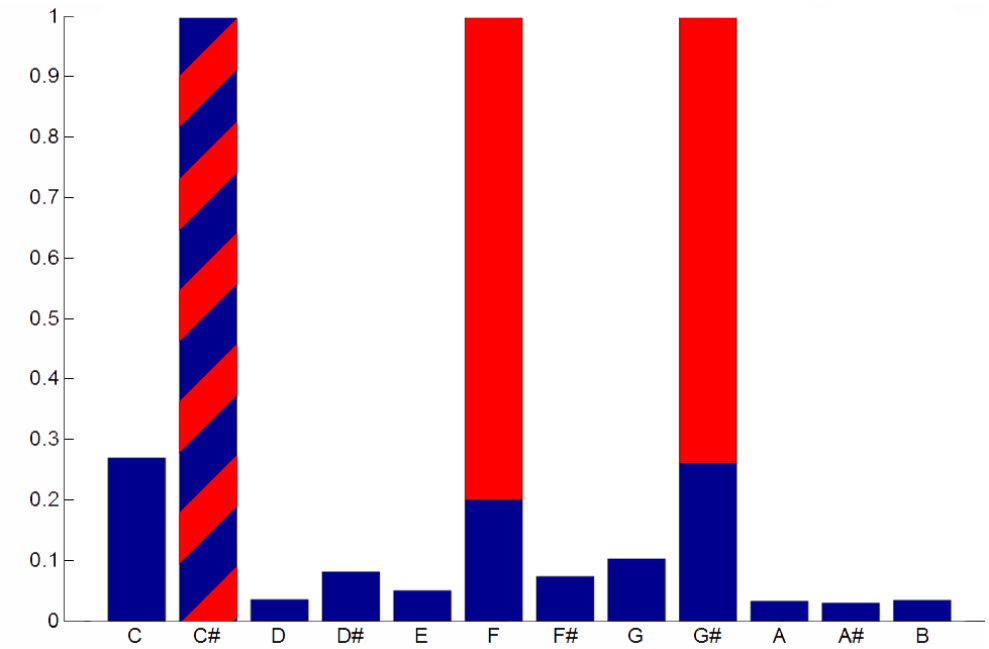
A módszerek különbségeinek könnyebb megértéséhez a folyamat kulcsszavakban:

Jelablakonként (Energiaspektrum sűrűség → Hangintenzitás → Hangintenzitások összege oktávonként → Legintenzívebb akkord hangjainak megkeresése → Legintenzívebb akkord hangintenzitásainak tárolás) → A tárolt akkordok összegét tartalmazó vektorból hangnem

5.3.1. Akkordok találati eredményeinek számítása, szűrése

Az - aktuális időablakra kiszámolt - összegzett hangintenzitás értékeket (`octav_sum`) az *akkordsablonokkal* (`chord_template`) összeszorozva egy olyan vektort kapunk ami - az aktuális mintára - megmutatja az akkordok találati értékét. A folyamat teljesen megegyezik a hangnemek meghatározásakor használt skaláris szorzással, melyet a 5.1.11. fejezet ír le. Az akkordminták illesztésére egy példa látható a 5.11. ábrán.

Az akkordokhoz ilyen módon hozzárendelt értékeket *normalizáljuk*[9]. Ha egy akkord értéke nem éri el a `parameters.ini` fájl `min_probability` változója által meghatározott - maximumhoz viszonyított - százalék értéket, azt az értéket kinullázzuk. Ha a megmaradt akkordok száma nagyobb, mint a `parameters.ini` fájl `matched` változójának értéke, akkor az túl sok találatot jelent, és ebben az esetben - a bizonytalan találatok mérsékelése érdekében - a kapott értékeket nem használjuk fel.



5.11. ábra. *Egy dal hangintenzitás-vektora és egy helyesen detektált C♯ akkord az akkordsablonok közül*

[9] A vektor minden elemének az értékét elosztjuk a vektor maximális értékével

5.3.2. Szűrt akkordok tárolása

Ha az aktuális eredmény a kritériumokat kielégítette, az oktávonként összegzett hangitenzitásokat összeszorozzuk a legnagyobb értéket elért akkordsablonnal. Az eredmény az akkordban szerepelő hangok intenzitása lesz, mely következő lépésben hozzáadódik a `sum_chords` változóhoz.

A jelfeldolgozási ciklus az audiójel végéig ismétlődik, és a ciklusból kilépve a `sum_chords` változó tartalmazza majd a jelhez tartozó hangintenzitás-vektort.

A hangnemdetektálási folyamat következő lépése megegyezik az Első Módszerrel, vagyis a program - a `sum_chords` értékei alapján - megállapítja a hangnemet, majd rendezi és exportálája az eredményeket a 5.1.11. fejezetnek megfelelően.

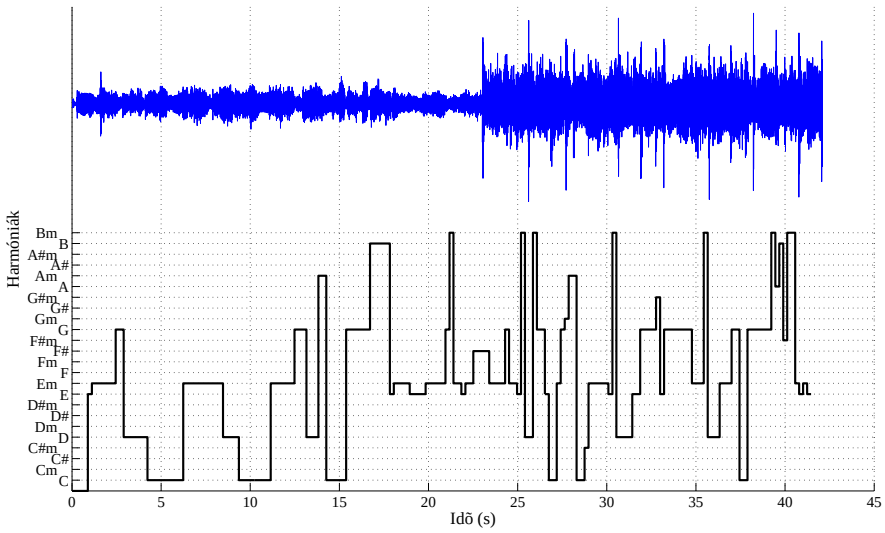
5.3.3. A grafikonok exportálása

A módszer esetén egy közös grafikonban van ábrázolva a jel időfüggvénye, és a felismert akkordok (Példa: 5.12. ábra). Ahhoz, hogy jól látszódjanak az akkordváltások helyei, az időfüggvényt az y tengelyen kellő mértékben el kell tolni felfelé.

Az akkordok azonosításához az y tengely az *"ábécés"* hangoknak megfelelően fel van címkézve, és rácsvonalak is segítik az akkordok könnyebb beazonosítását.

Mivel hosszabb zeneszámok és kis időablakméret esetén a grafikon - a sűrű, vastag vonalak miatt - átláthatatlanná válna, a blokk 30 másodperc dalhossz felett csökkenti a vonal vastagságát.

Az időfüggvényt - mivel csak tájékozódásra szolgál - nem szükséges a teljes felbontásában ábrázolni. Tapasztalat szerint elegendő csak minden 40. mintát szerepeltetni, így a grafikonok kiexportált méretének csökkentése céljából itt is ezt az értéket alkalmazzuk.

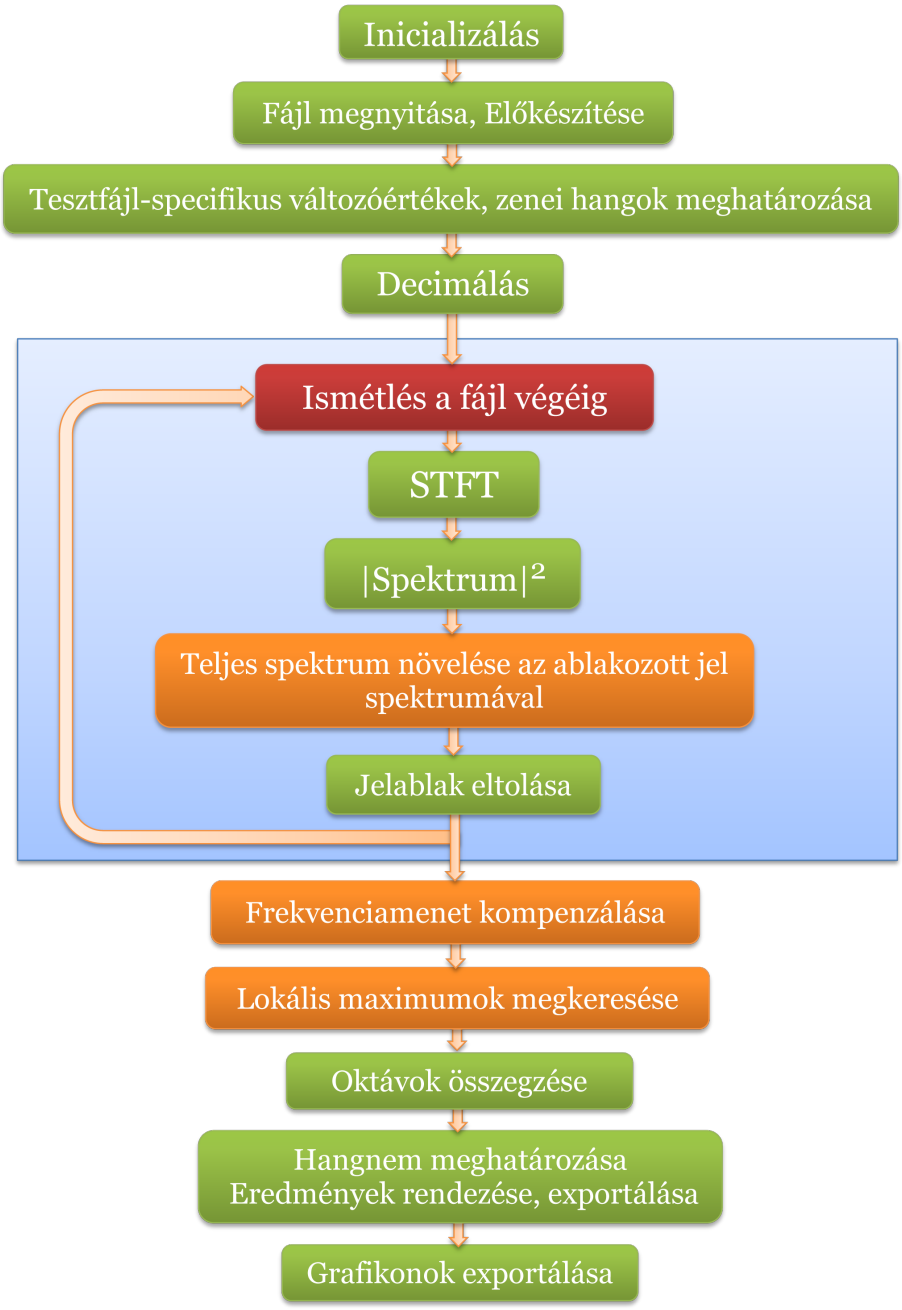


5.12. ábra. *Példa az akkordok és az időfüggvény ábrázolására*

5.4. Negyedik Módszer

Áttekintés: A Negyedik Módszer a hangnem meghatározásához, - az audiójel feldolgozása során - kiszámolja az ablakozott jelek energiaspekrumainak átlagát. Az így megkapott spektrum jelleggörbéjének meghatározása után létrehoz egy amplitúdóban kompenzált energiaspektrumot. A létrejött görbe lokális maximumértékei mutatják meg a zenei hangok intenzitásait.

Az algoritmus ábrája az 5.13 ábrán látható.



5.13. ábra. *A Negyedik Módszer ábrája*

A Negyedik Módszer az Első Módszertől csak a hangok intenzitásának meghatározási módszerében különbözik. A módszer ebben az esetben először létrehoz egy amplitúdóban kompenzált spektrumot, és a zenei hangok intenzitás-vektorát maximumkereséssel állítja elő.

5.4.1. Frekvenciamenet kompenzálása

A módszer a későbbi blokkban maximumkereséssel fogja meghatározni a zenei hangok intenzitás-vektorát, viszont mivel az amplitúdóspektrumban a mélyebb hangoknak megfelelő frekvenciák sokkal nagyobb intenzitással szerepelnek, a találatok legnagyobb része a mély tartományra korlátozódna. Ahhoz, hogy a közép és magas frekvenciatartományban lévő csúcsokat is detektálni lehessen, a spektrum frekvenciamenetét - amplitúdóban - kompenzálni kell.

A kompenzáláshoz szükséges a spektrum jellegének (*trendjének*) az ismerete. A blokk a jelleggörbe meghatározásához és a spektrum-kompenzálás elvégzéséhez a `trend_removal()` függvényt alkalmazza.

A jelleggörbe úgy jön létre, hogy a függvény veszi a spektrum minden frekvenciájának - `parameters.ini` fájl `range` változóértékeként - megadott ± jeltartományát, és meghatározza az aktuális tartomány átlagát. A `trend` változó tartalmazza a jelleggörbét, és a kiszámolt átlag lesz az aktuális tartomány középső frekvenciájának az értéke. Mivel az így kapott jel követi a frekvenciamenet változásait, az eredeti spektrumot a jelleggörbével elemenként elosztva az így kompenzált jel a közép és magasabb frekvenciáknál erősítve lesz.

A magasabb frekvenciák esetén az erősítés mértéke túl nagy, ezért a frekvenciamenet közepépső frekvenciáitól kezdődően a frekvenciák erősítése - az aktuális frekvenciával fordítottan arányosan - csökken. A lépések megértéséhez a 5.14. ábra nyújt segítséget.

5.4.2. Lokális maximumok megkeresése

A kompenzált spektrumból kiemelkedő csúcsok reprezentálják a hangok intenzitását. A csúcsokhoz tartozó hangokat a `find_maximums()` függvény alkalmazásával keressük meg. A jellegörbe meghatározásakor a mélyebb frekvenciák csak az első zenei hang környékétől indulnak, így a függvény az első zenei frekvencia alatti értékeket kinullázza. A függvény a MATLAB `findpeaks()` függvényének segítségével határozza meg a csúcsok helyét. A függvény bemeneti paramétereiként a `parameters.ini` fájlban szerepelő `min_peak_distance` és a `threshold` változók értékeit használja fel. A `min_peak_distance` értéke a minimális távolságot jelöli a talált maximumok között, a `threshold` pedig a maximális értékhez viszonyított küszöbértéket mutatja meg (dB-ben). Az így megtalált lokális maximumokhoz tartozó frekvenciák nem pontosan a zenei hangok frekvenciái, ezért meg kell keresni a maximumokhoz legközelebb álló zenei hangot, majd a `sum_note` változóban a zenei hangnak megfelő elemet kell növelni a lokális maximum értékével.

A legközelebbi zenei hang megtalálására egy jó módszer a következő:

A zenei frekvenciákat tartalmazó tömbből - az aktuálisan talált - frekvenciát kivonva megkeressük az így kapott értékek abszolútértékének minimumát. A `min()` függvény a keresett értékhez legközelebb álló zenei frekvenciát és annak indexét fogja visszaadni[10].

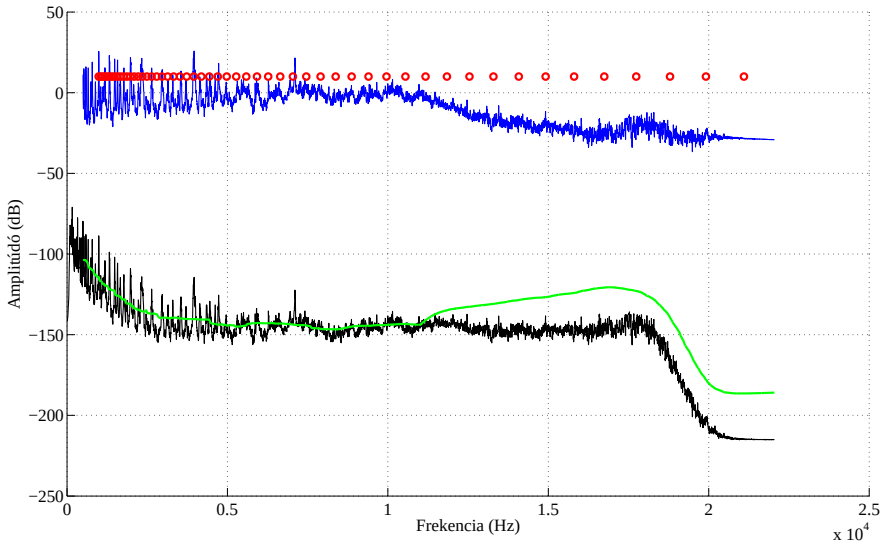
A zenei hangnak megfelelő elemet a talált maximum értékével növelni kell, majd az összes maximumra meg kell ismételni a folyamatot. A függvény a `sum_note` változó értékeivel tér vissza, és ez a változó lesz a hangintenzitás-vektor.

A hangnemdetektálási folyamat következő lépése megegyezik az Első Módszerrel, vagyis a program - a `sum_note` értékei alapján - megállapítja a hangnemet, majd rendezi és exportálája az eredményeket a 5.1.11. fejezetnek megfelelően.

5.4.3. A grafikonok exportálása

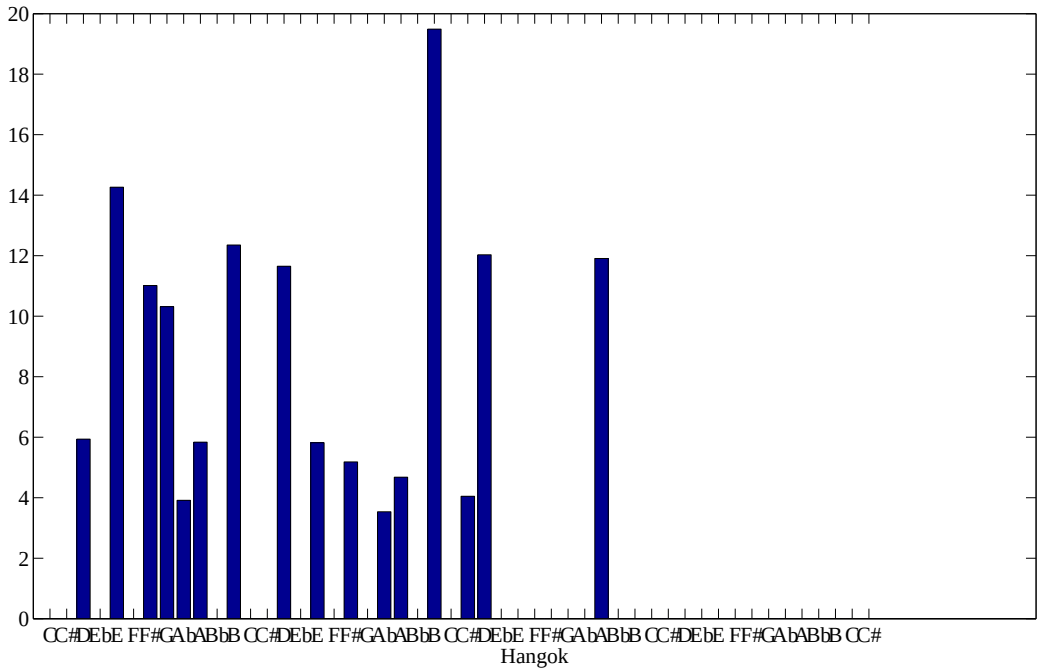
A módszer esetén az amlitúdóspektrum ábrája - piros körök segítségével - a zenei hangok frekvenciáit is megjeleníti. Ez segít eldönteni, hogy a spektrum kiugró csúcsai zenei hanghoz tartoznak-e. Az ábrában megjelenik még a spektrum - magas frekvenciáknál kompenzált - jellegörbéje és a valós és a kompenzált amplitúdóspektrum is. (Példa: 5.14. ábra)

A maximumkereséskor megtalált hangok, és az oktávonként összegzett hangok intenzitásának az ábrázolása a MATLAB - oszlopdiagrammok rajzolására kifejlesztett - `bar()` függvényével történik. A generált ábrákon így jól látható, hogy az intenzitások hogyan aránylanak egymáshoz. (Példák: 5.15. és 5.16. ábrák)

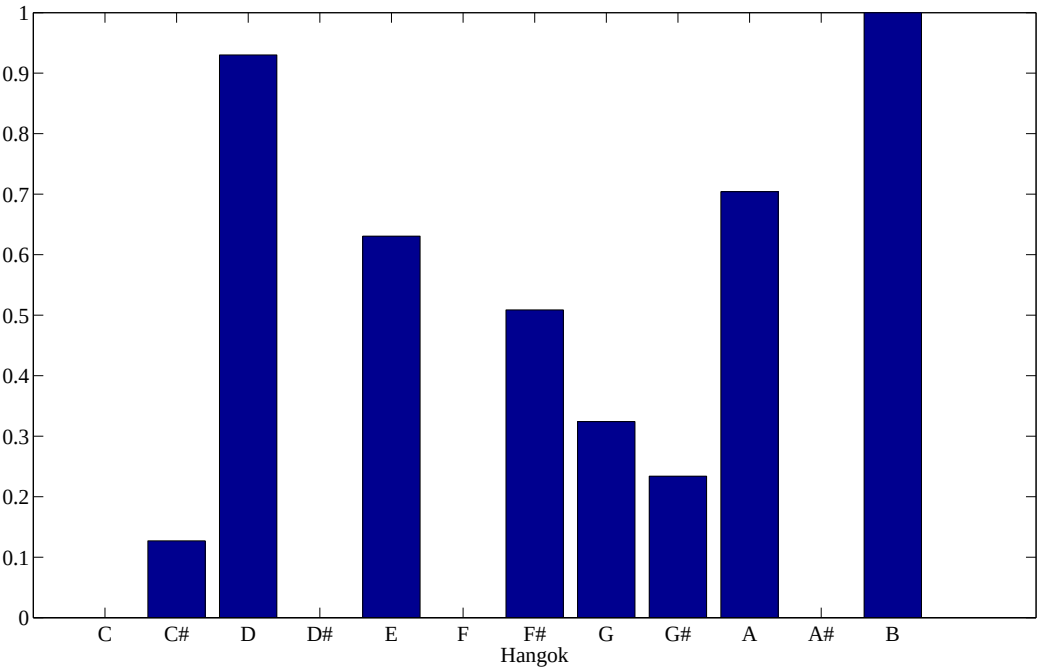


5.14. ábra. *Az eredeti (fekete) és kompenzált amplitúdóspektrum (kék), a jelleggörbével (zöld), és a zenei hangokkal (piros) (decimate_factor = 1)*

[10]MATLAB kóddal szemléltetve: `[v,ind] = min(abs(note_frequencies - act_freq))`



5.15. ábra. *Oszlopdiagram példa a hangok intenzitásának ábrázolására*



5.16. ábra. *Oszlopdiagram példa az oktávonként összegzett hangok intenzitásának ábrázolására*

6. fejezet

A tesztfájl adatbázis

Az algoritmusok tesztelésére egy relatíve nagy elemszámú zeneszám adatbázist kellett létrehozni.

A tesztfájlok megválasztásának fontos szempontjai:

- Minél változatosabb hangnemek
- Vegyesen szerepeljenek hangszeres és elektronikus dalok
- Legyenek mély, közép, és magas frekvenciatartományban dominánsabb dalok
- Monoton és változatos zenei felépítés
- A zenei stílusok minél szélesebb spektrumát fedjék le
- Rendelkezzenek valós hangnemmel
- Legyenek nehezen és könnyen felismerhető hangnemek is

6.1. Az adatbázis felépítése

A zeneszámokat hasonló stílusonként csoportosítva, külön mappákban tároltam el. Mappánként a zeneszámok sorszámozott, előadóra vagy dalcímre utaló, könnyen azonosítható, de rövid fájlneveket kaptak. Mivel a zeneszámok legnagyobb része eredetileg MP3 formátumban volt, a zeneszámokat át kellett konvertálni WAV formátumba. Az MP3 fájlok által tárolt *ID3 tag*-ekből különféle információkat lehetett kigyűjteni a zeneszámokról. Ezeket az információkat a WAV fájlok mellé egy - a WAV fájl nevével megegyező - TXT fájlba gyűjtöttem ki, és kiegészítettem a **Rapid Evolution 3.** és a hallás alapján felismert hangnemekkel. A **63** darab teljes hosszúságú hangfájl a gyökérkönyvtár `tesfiles_big` mappájában van eltárolva.

Mivel egy több perces zeneszám hangnemének analizálása akár pár percig is eltarthat és sok memóriát igényel, - a fejlesztési és tesztelési fázisok gyorsítása érdekében - az összes tesztfájlból létrehoztam egy rövidített verziót is a gyökérkönyvtár `tesfiles_small` mappájába. Ezek a fájlok az eredeti zeneszám egy jellegzetes kb. 20-40 másodperces részletét (például refrén, intro, szövegrész, stb.) tartalmazzák.

7. fejezet

Szabad paraméterek hangolása

Az algoritmusok pontossága és futási ideje függ a feldolgozott zeneszámoktól, és a beállított konfigurációs paraméterektől. A program szabad paramétereit a - a könnyű módosíthatóság érdekében - a `parameters.ini` fájl tartalmazza.

A paraméterek jelentését már a 4.2. táblázatban bemutattam, de a paraméterek értékeiről, és azok helyes megválasztásának stratégiáiról még nem esett szó. Mivel a paraméterek egymással összefüggenek, a paraméterek hangolása egy hosszú, iterációs folyamat. Az optimális paraméter értékek megválasztásához a legnagyobb segítséget a - tesztfájl-adatbázison lefuttatott, mérések végén legenerált - Összefoglalás fájl, és a fájlok mérési jegyzőkönyvei adták. Az Összefoglalás fájl tartalmazza a futtatáshoz tartozó statisztikákat, bemeneti paramétereket és találati eredményeket, ezért a különböző beállítások eredményességének összehasonlítására kiválóan alkalmas. A *Függelék* (70. oldaltól) példaként egy - a program által legenerált - *Mérési Jegyzőkönyvet*, és egy optimális paraméterekkel generált *Összefoglalás* fájlt tartalmaz.

A fejezetben a különböző paraméterek hangolási folyamata és a választott paraméterértékek indoklása található.

7.1. A paraméterek hangolásának folyamata

A sok szabad paraméter miatt a hangolási folyamat során a változók értékének meghatározását több körben érdemes elvégezni. Erre azért van szükség, mert léteznek olyan paraméterpárok, melyek összefüggnek egymással, és egymás hatását erősítik vagy kompenzálják[1]. A hangolási folyamat során miután egy paraméter értékének megtaláltam az optimumát, egy másik paraméter hangolásához fogtam hozzá. A második paraméter optimumának meghatározása után az előző paramétert - egy szűkebb értéktartományban - újra megpróbáltam hangolni, és ha így javult az eredmény, akkor azt az értéket tartottam meg.

[1] pl. Időablak átlapolódás - időablak-szélesség

A módszerspecifikus paramétereket módszerenként párhuzamosan lehetett tesztelni, mert azok csak a saját módszerüket befolyásolták.

Az összes módszerhez tartozó paraméterek hangolási sorrendje:
Decimálási faktor → Időablak szélesség → Decimálási faktor → Időablak átlapolódás → Skálán belüli hangok büntetése → Módszerspecifikus paraméterek...

7.2. A decimálási faktor

A decimálás folyamata és lényege a 5.1.4. fejezetben szerepelt. A *decimálási faktort* a `decimate_factor` változó tartalmazza.

A digitális audiójelek esetén a *mintavételi tétel* kimondja, hogy a mintavételi frekvencia legalább kétszer nagyobb kell legyen, mint a mintavételezett jel legnagyobb frekvenciája. A - CD minőségű - WAV fájlok esetén a mintavételi frekvencia 44100Hz így az átfogott frekvenciatartomány 22050Hz.

A decimálási faktor **1,2,4** és **8** értékei esetén, a legnagyobb frekvencia rendre **22050Hz, 11025Hz, 5512,5Hz** és **2756,25Hz**-re csökken.

A jel időbeli felbontása decimálás nélkül (`decimate_factor = 1`) a legnagyobb, viszont ilyen paraméterezés körülbelül 150%-os futásidő növekedést, és 2-szer annyi memória foglalást jelent, a 2-es decimálási faktorhoz képest. A hangnemdetektálás pontossága is körülbelül 10%-al rosszabb, és az összes algoritmusnál megfigyelhetővé vált, hogy hajlamosabb - ilyen paraméterezés mellett - ± 1 kvintet tévedni. Ennek az lehet az oka, hogy a magasabb - "sokadik" felharmónikusként létrejött - frekvenciák is belekerülnek a hangintenzitásvektorokba, melyek már nem az eredeti hangok oktávján jelennek meg, hanem azok, - vagy oktávjaik - kvintjének, kvartjának az intenzitását erősítik.

A 2-es és a 4-es decimálási faktor eredményeit összehasonlítva, a 4-es decimálási faktor esetén a pontosság körülbelül 10%-ot romlik, de a futásidő 25%-al rövidebb lesz és a fájlok is feleakkora memóriát igényelnek.

8-as decimálási faktor esetén - mivel a legnagyobb frekvencia értéke túlságosan lecsökkent - a detektálható frekvenciák száma nem elég az értékelhető eredmények létrehozásához, így a jól detektált hangnemek száma jelentősen kevesebb lesz.

A négy lehetséges decimálási érték közül a legpontosabb eredményeket a **2**-es decimálási faktor hozta, a többi beállításhoz képest elég gyors futási idő mellett.

7.3. Időablak szélesség

Az STFT *jelablakának szélessége* a `time_window` változójának értéke alapján jön létre (STFT: 5.1.6. fejezet). A változó értéke megmutatja *másodpercben*, hogy a folyamatos audiójelből a kivágott időablak milyen hosszú. Ahhoz, hogy az audiójel vektorából `time_window`

hosszúságú részletet ki lehessen vágni, a változó értékéből meg kell határozni, hogy az adott `decimált mintavételi frekvencia` ($F_{s_{ds}}$) mellett ez hány mintát jelent. A folyamot a 5.1.3 fejezet mutatja be.

Az FFT - az algoritmusának megvalósításából adódóan - "*gyorsabban*" meg tudja határozni egy jel spektrumát, ha az 2 hatványaival megegyező számú mintákból áll. Ebből a megfontolásból az ablakszélesség megválasztásakor olyan időértékek szerepelnek, melyekből a minták száma (N_{window}) 2 hatványai lesznek.

A tesztelt értékek (`decimate_factor = 2` esetén):

Minták száma (N_{window})	**Időablak hossz**
4096	0.1857963 s
8192	0.3715192 s
16384	0.7430385 s
32768	1.4860770 s

7.1. táblázat. *A leheséges időablakhosszak és a hozzájuk tartozó minták száma*

Az időablak szélessége tulajdonképpen a jel sávszélességét befolyásolja. Ha nagyobb értéket használunk, nagyobb lesz a frekvenciafelbontás, kis érték esetén pedig a frekvenciafelbontás csökken.

Minél kisebb az ablakszélesség, a spektrumot annál kevesebb mintából lehet létrehozni, és - mivel kisebb "szeletekre" vágjuk fel az audiójelet, - a teljes jel feldolgozásához több ablakozási és jelfeldolgozási ciklus szükséges. Emiatt a Második és Harmadik módszer esetén számottevő növekedést lehet megfigyelni a hangok és akkordok ábrázolásához eltárolt adatok méretében.

Nagy ablakszélességet alkalmazva a STFT átlapolódásából adódó időbeli pontosság romlik, és a Második és Harmadik módszer esetén az ábrákon - a felbontás csökkenéséből adódóan - a *legintenzívebb hang* és az *akkordok* időben kevésbé pontosan lesznek beazonosíthatóak. Az amplitúdóspektrum - mivel nagyobb időszakra vonatkozik, - sokkal sűrűbb lesz, és a zenében szereplő hangokhoz tatozó csúcsok az időablak bármelyik részéről származhatnak. Ezáltal lehetséges, hogy akár több egymást követő akkord vagy harmónia is egy ablakba kerül, és így - az összekeveredett hangok miatt - egy téves akkordot detektál a program. A fenti megfontolások miatt a 4096 minta alatti értékek ellenőrzésének nincs értelme, mivel túlságosan sokáig tartana az detektálási folyamat, és emiatt - az algoritmusok későbbi felhasználása során - nem lehetne alkalmazni ezeket a paraméterértékeket.

Az eredmények a minták számának csökkenésével arányosan egyre pontatlanabbak, viszont az ábrák felbontása, és a futásidő megnő.

A különböző beállítások eredményei alapján a választott érték a **16384** mintának megfelelő **0.7430385** másodperc lett.

7.4. Az időablak átlapolódása

Az időablak hosszán kívül az STFT fontos paramétere a `window_overlap` változójában tárolt *időablak átlapolódás* is. A paraméter értéke azt határozza meg, hogy az egymást követő ablakok hány százalékban legyenek fedésben egymással. Az ablakozási folyamat részletesen a 5.1.8. fejezetben található.

Az audiójelből a jelrészlet kivágása után összeszorozzuk egy ablakozó jellel. A szorzás miatt - az ablakozó jeltől függően - az jelablak széleinél a jel mintái bizonyos mértékig csökkenni fognak. Emiatt, ha az audiójel ablakozott jelrészletei átlapolás nélkül lennének kiértékelve, az ablakok széleinél információt veszítenénk.

Az ablakozott jelek átlapolódásakor jellemzően az **50%**-os értéket használják.

A kipróbált értékek: **0%** (Nincs áltapolódás), **30%, 50%, 70%, 80%**.

Az átlapolódás növelésével - mivel kisebb mértékű az időablak eltolása, - a futásidő nő, viszont pontosabb eredményeket lehetett elérni. A javulás oka az is lehet, hogy program a különböző mértékben eltolt ablakok - például egy akkordváltás pillanatát tartalmazó jelrészlet - esetén, a félredetektálásból adódó hibát az előző és a következő jelablakokban jól detektált értékeivel kompenzálja.

Az átlapolódás csökkentésével egyre több információ veszik el és pontatlanabb eredmények születnek, viszont - az ablakok számának csökkenése miatt - gyorsabb lesz a jelfeldolgozási ciklus.

0%-os átlapolódás esetén gyors, de más beállításoknál pontatlanabb eredmények születtek.

30% esetén a generált *legintenzívebb hang* és *akkord* grafikonok elnagyoltnak tűnnek, és átlagos pontosságúak az eredmények

50% értékhez korrekt eredmények és átlagos futásidő társul, viszont 70%-esetén az eredmények pár százalékkal jobbak lettek.

70% érték esetén lettek a legjobbak az eredmények, minimális futásidő növekedés mellett (50%-hoz képest)

80% estén a futási idő megnőtt, és a legintenzívebb hang és az akkord ábrák zavarosak, az eredmények pedig pontatlanabbak lettek (70%-hoz képest)[2].

A választott érték **70%** lett, mert a futási sebesség növekedés ezen az értéken még nem volt túl jelentős.

[2]A pontatlanság okára sajnos még nem találtam magyarázatot.

7.5. Skálán kívüli hangok büntetésének mértéke

A skálasablonok és az oktávonként összegzett intenzitások szorzatából alakul ki az a vektor, melynek értékei a különböző hangnemek találatainak ”jóságát” mutatják meg. A folyamat leírását a 5.1.11. fejezet, a skálán kívüli hangok büntetésének leírását pedig a 4.1.3. fejezet tartalmazza. A skálasablonokban a skálán kívüli hangok elemei negatív előjellel, a `scale_punishment` változó értékével súlyozva jönnek létre. A súlyozás miatt a skálában nem szerepelő hangok *büntetve* lesznek, így a zeneszám detektált hangnemekhez tartozó értékeki jobban tükrözik majd a találatok közötti eltéréseket.

A változó értéke $[0, \infty]$ közötti valós szám lehet, melyet a függvény negatív előjellel fog felhasználni.

A tesztelt értékek: **0, 0.5, 1, 1.5**

A változó értékeinek növelésével a találatok értékei egyre jobban eltávolodtak egymástól. A találatok sorrendjét a változó értéke nem változtatta meg, mert a hangnemek sorrendje - mint kiderült - már a ”0” változóérték esetén is kialakult.

A választott változóérték **1** lett, mert így - a Mérési Jegyzőkönyvekben - a módszerek találati táblázataiban értékeken jobban látszik, hogy milyen mértékben rosszabbak vagy jobbak egymásnál.

7.6. Módszerspecifikus paraméterek

7.6.1. Első, Második és Harmadik Módszer - A zenei frekvenciák környékének felhasznált tartománya

A `frequency_range` változó, a hangok intenzitásának meghatározásakor (5.1.9) a felhasznált minták frekvenciatartományának beállítására szolgál. A felhasznált tartományindexek úgy jönnek létre, hogy az eredeti indexek és a zenei frekvencia közötti távolságot az erre szolgáló `sum_notes` függvény a változóértéknek megfelelően csökkenti le. A folyamat részletes leírása a 5.1.9. pontban található.

Minél kisebb a paraméter értéke, annál kevesebb mintát használunk fel, és annál kisebbek lesznek az eltérések a találati eredmények között. Nagy érték választása esetén a mélyebb frekvenciákhoz tartozó tartományoknál az egymás mellett lévő hangok között áthallást lehet megfigyelni. Emiatt egy intenzívebb, mély hang esetén a szomszédos hangok intenzitása is megemelkedik és ez elrontja az eredményeket.

A kipróbált értékek: **20%, 40%, 60%, 100%**

100% felett az Második és Harmadik Módszerek esetén már nem értelmezhető a beállítás és a módszerek nem adnak eredményt. A 60% alatti értékek esetén romlik a pontosság.

A tapasztalatok szerint a legjobb eredményeket a **60**% választásával lehetett elérni.

7.6.2. Harmadik Módszer - Minimális találati valószínűség

A `min_probability` változó értéke - a maximális értékhez képest - egy küszöbértéket határoz meg százalékban a találati eredmények szűréséhez (Részletesen: 5.3.1). Így lehet garantálni, hogy csak olyan akkordok jelenhessenek meg a akkordok grafikonján, és járulhassanak hozzá a végső detektált hangnemhez, melyek elég jó eredményeket értek el.

A változó kipróbált értékei: **60%, 70%, 80%, 90%**

A legjobb hatásfokot **90%** esetén lehetett elérni, mivel a kisebb értékeknél túl sok akkord jutott keresztül a szűrőn, és ez lerontotta a találati eredményeket, és zavarossá tette az ábrát is.

7.6.3. Harmadik Módszer - Maximális harmónia találat

A Harmadik Módszer esetén, ha a harmóniák száma túl nagy, téves találatok jöhetnek létre. Ennek elkerülése érdekében, ha az egy ablakon belül talált harmóniák száma meghaladja a `matched` változó értékét, a program az eredményeket nem értelmezhetőnek definiálja, és kihagyja az összegzésből (5.3.1).

Az olyan dalok esetén, ahol az amplitúdóspektrum túl sűrű, vagy kevés kiemelkedő csúcsot tartalmaz, a paraméter értékének jó megválasztásával meg lehet gátolni, hogy - a szinte azonos intenzitású találatok félreértelmezése miatt - hiba keletkezzen.

Az értékek 1 és 24 között értelmesek, de érdemes a paraméter értékét az alábbi megfontolások alapján megválasztani:

Túl kicsi érték esetén sok akkord lesz eldobva, így kevesebb információból derül ki a hangnem, és ez általában pontatlanabb.

Túl nagy érték esetén pedig az adatok között maradnak a félreértelmezett találatok is, és így nem javul a pontosság, viszont a számítások futási időt növelnek.

A kipróbált értékek: **5, 6 , 7, 8**

Megfigyelések alapján a változó helyes megválsztása függ a `min_probability` értékétől. A hangolás során `min_probability` = 80% és `matched` = 7 értéket alkalmaztam alapértelmezettként, de a `min_probability` = 90%-ra növelése esetén a találati eredmények romlottak. `matched` értékét 6-ra csökkentve viszont az eredeti beállításnál is jobb értékeket sikerült produkálni, így a választott változóérték **6** lett.

7.6.4. Negyedik Módszer - Jelleggörbe átlagolási tartománya

A Negyedik Módszerben a jelleggörbe - a 5.4.1. fejezet alapján - úgy jön létre, hogy a `trend_removal` függvény a spektrum `range` változó értékének kétszeresével[3] megegyező

[3] ± *range* tartomány

tartományt átlagol, és az így kapott érték lesz a tartomány közepén lévő frekvencia amplitúdóértéke. A tartomány méretének növelésével - a spektrum frekvenciamenetét csak jellegre követő, - kisimult jelleggörbe lesz az eredmény. A tartomány méretének csökkentésével a jelleggörbe egyre jobban követi majd a frekvenciamenet változásait.

A megfontolások alapján, ha `range` értékét túl kicsire választjuk, akkor az intenzitások értékét elveszítjük a jelleggörbével leosztott jelből. Ha az érték túl nagy, akkor pedig a függvény nem fogja kellő mértékben csökkenteni a mélyebb frekvenciák értékeit.

A tesztelt értékek: **150, 200, 250, 300** minta, melyek **2**-es decimálási faktor esetén $300Hz, 400Hz, 500Hz, 600Hz$ frekvenciának felelnek meg.

A tapasztalatok szerint a legjobb választás értéke **250** minta, vagyis ± 500Hz lett.

7.6.5. Negyedik Módszer - Maximumkeresés paraméterei

A Negyedik Módszer a hangokhoz tartozó intenzitásokat lokális maximumok keresésével határozza meg az 5.4.2. fejezetben leírtak szerint. A `find_maximums()` függvényének az alapja a MATLAB belső `findpeaks()` lokális maximumokat kereső függvénye, melynek fő paramétereit a `min_peak_distance` és a `threshold_db` változók tartalmazzák.

A `min_peak_distance` megmutatja, hogy a detektált csúcsok között hány minta legyen a minimális távolság. A paraméter jó megválasztásával elkerülhető, hogy például - egy intenzívebb, mély hang kicsit szélesebb spektrumából - a maximum környékén (kisebb intenzitással) megjelenő lokális maximumokat a függvény újra hozzáadja a hang intenzitásához. Az ideális érték megválasztásához figyelembe kell venni, hogy a mély tartományban milyen közel lehetnek egymáshoz a csúcsok.

A `threshold_db` azt a határvonalat jelöli a globális maximumhoz képest dB-ben, amely alatt nem keresünk lokális maximumokat. Ezzel a detektált maximumok száma limitálható.

A paraméterek kipróbált értékei:

- `min_peak_distance` : 10, 15, 20 minta
- `threshold_db` : -10, -15, -20 dB

Az eredmények hangolása során a **15** minta jobb választásnak tűnt a többinél. Küszöbérték megválasztásakor pedig a -20dB túl alacsony a -10dB túl magas értékek voltak, így a középút, vagyis a **-15**dB lett a kiválasztott érték.

8. fejezet

Az eredmények értékelése és összefoglalása

A paraméterek hangolási folyamata után a négy módszer a következő eredményeket érte el a **Rapid Evolution 3.** szoftverhez képest:

Típus	**Rapid Evolution**	**Módszer 1**	**Módszer 2**	**Módszer 3**	**Módszer 4**
Jól detektált (db)	47	33	33	41	17
± 1♭ vagy 1♯ (db)	8	26	18	17	25
± 2♭ vagy 2♯ (db)	2	1	5	2	14
Maradék találat (db)	6	3	7	3	7
Jól detektált	74.60 %	52.38 %	52.38 %	65.08 %	26.98 %
± 1♭ vagy 1♯	12.70 %	41.27 %	28.57 %	26.98 %	39.68 %
± 2♭ vagy 2♯	3.17 %	1.59 %	7.94 %	3.17 %	22.22 %
Maradék találat	9.52 %	4.76 %	11.11 %	4.76 %	11.11 %
Max ± 1♭ vagy 1♯ tévedés	87.30 %	93.65 %	80.95 %	92.06 %	66.67 %

8.1. táblázat. *A módszerek eredményeinek statisztikája*

A táblázatból jól látszik, hogy ha ± 1 *kvint* tévedést is helyes találatként kezeljük, - az adott zeneszám-adatbázison - az Első és Harmadik módszer meghaladta a célszoftver pontosságát. A pontos találatok száma a Harmadik módszer esetén kevesebb, mint 10%-al marad el a célszoftvertől. A Negyedik módszer a többihez képest pontatlan eredményt hoz.

A Rapid Evolution pontosságát az indokokhatja, hogy sokkal több skála típust képes megkülönböztetni, és rendelkezik tempó detektáló algoritmussal is, amely például a helyes jelablakszélesség megválasztásához lehet ideális. A tempó ismerete az akkordváltások felismerését tehetné pontosabbá. Az eredményekből az is látszik, hogy a Rapid Evolution sokkal kevésbé téved kvintet a saját algoritmusokhoz képest.

A Rapid Evolution pontosságát a 9. fejezetben leírtak alkalmazásával meg lehetne közelíteni az Első és Harmadik módszer esetén.

A zeneszám-részleteken lefuttatott hangnem detektálás - a jelenlegi paraméterek mellett - kb 25 percig tart. Mivel csak a 20-30 másodperces zeneszám-részleteket analizáljuk, 63

tesztfájl esetén ez átlagosan 24,5 másodpercet jelent dalonként. A módszerek futásidői a mellékletben szerepelő táblázatból jól leolvashatóak, és látszik, hogy a legtöbb esetben módszerenként 3-5 másodperc kellett a hangnemek detektálásához.

Mivel ugyanazon az adatbázison hangoltam és teszteltem az algoritmusokat, valószínűleg egy másik adatbázison a program eredményeinek pontossága romlana.

A Negyedik Módszer jelentős hibaaránya a többi algoritmushoz képest valószínűleg a kompenzáló görbe nem megfelelő megvalósításának, és rossz paraméter-értékeinek tudható be. Más szabad paraméter-értékekkel az algoritmus pontosabb volt, de mivel meg sem közelítette a többi módszer pontosságát, a végső paraméterérték-módosítások során nem volt elsődleges cél, hogy ez a módszer is pontosabb eredményeket érjen el.

A program futtatásai során megfigyelhető volt az is, hogy a legelső tesztfájl első feldolgozása a MATLAB - és az operációs rendszer - memóriakezeléséből adódóan több ideig tart, így az első dal első módszerének kiemelkedően nagy futásideje nem a zeneszámból adódik.

A módszerek paraméter-értékeinek hatásait a 7. fejezetben már részletesen bemutattam, így itt ezekre nem térek ki.

A későbbi felhasználásra - az eredmények alapján - az Első és Harmadik módszer lenne a legalkalmasabb. A fejlesztés utolsó szakaszában rengeteg új ötlet merült fel ahhoz, hogy a eredményeket hogyan lehetne javítani. Az új ötleteket a 9. fejezetben foglalom össze.

A dolgozat feladatkiírásában leírtakat a megvalósított program segítségével sikerült teljeseíteni:

Létrejött MATLAB környezetben egy több algoritmust futtatni képes keretrendszer, amely négy - az irodalomban megismert és saját ötleteken alapuló - hangnemfelismerő módszer segítségével detektálja a relatíve nagy zeneszám-adatbázis fájlainak hangnemét.

Az algoritmusok szabad paramétereit a legenerált statisztikák és jegyzőkönyvek alapján egy optimális beállításra sikerült hangolni.

9. fejezet

Kitekintés a fejlesztési lehetőségekre és az eredmények felhasználására

A dolgozat alapján több olyan algoritmust is sikerült kipróbálni, kifejleszteni, melyek - további fejlesztések után - alkalmasak lehetnek zeneszámok hangnemének - elég pontos - felismeréséhez.

A módszerek eredményeinek pontosítására több ötlet is felmerült, melyeket a későbbi fejlesztések során ki lehetne próbálni.

9.1. Első és Utolsó hang, mint súlyozó tényező

A tánczenében jellemző, hogy a zeneszám a hangnem alaphangjával kezdődik, vagy azzal zárul. Mint plusz súlyozási tényezőt fel lehetne használni a hangnemek sorrendjének meghatározásakor, vagy esetleg a Dúr-moll párhuzamos hangnemek megkülönböztetésére lehetne alkalmas.

9.2. BPM detektálás a jelablak méret meghatározásához

Jelen esetben az ablakozó jel hossza - a tapasztalatok és elvek alapján meghatározott - fixen beállított érték, amely az akkordváltások pillanatait nem veszi figyelembe. Ha a dalnak rendelkezésre állna a tempója, az akkordváltások pillanatát könnyebben lehetne azonosítani. Két akkordváltás között a mintasorozatot így olyan méretű jelablakkal lehetne feldolgozni, amely pontosabban lefedi az akkorváltások közötti részt.

9.3. Módszerek eredményeinek összefűzése

Mivel a Negyedik Módszer kivételével a módszerek elég jó eredményeket hoztak, a három eredményt - *szakértői rendszer* szerűen - szavaztatni lehetne a nyertes hangnemre. A három módszer eredménye a szavazásban akár súlyozva is szerepelhet.

9.4. A statisztikák kibővítése

A módszerek objektívebb összehasonlításához a statisztikákat ki lehetne egészíteni több kinyerhető információval (például):

- Ki lehetne számolni a futásidők összegét fájlonként és módszerenként
- Zenei stílusonként (mappánként) meg lehetne jeleníteni a legtöbb jó találatot produlkáló módszert
- Több - különböző paraméterű - futtatásból egy átfogobb összehasonlító dokumentumot lehetne létrehozni, esetleg a paraméterek értékeinek hatását ábrázolni szemléletes grafikonokon

9.5. Domináns kompenzáció

Az Első Módszernél jól látszik, hogy a jó találatok mellett legtöbbször egy kvintet téved. Ennek az is lehet az oka, hogy a zenében megszólaló zenei hangok nem tisztán a saját frekvenciájukon - és oktávjaikon - tartalmaznak komponenseket, hanem a például a kvintjüknek megfelelő domináns is erősödik. Emiatt annak az értéke magasabb lesz, mint az alaphanghoz (tonikához) tartozó, ezért a detektálás egy kvintet téved. Lehetséges, hogy az alaphangokhoz tartozó domináns intenzitásának bizonyos mértékű elnyomásával a "*kvinttévedések*" száma csökkenthető.

9.6. Módszerenként saját globális paraméterek

Az egyszerűség miatt a módszerek főbb paraméterei közösen voltak állíthatóak, ezért a paraméterek módosítása az összes módszer eredményességét befolyásolta. Ha külön mindegyik módszernek minden - eddig globális - változóból saját, egyénileg módosítható lenne, a módszereket külön is optimumra lehetne hangolni.

Köszönetnyilvánítás

Köszönetet szeretnék mondani *Bank Balázs* egyetemi docensnek, aki konzulensként folyamatos odafigyeléssel és praktikus tanácsokkal, ötletekkel segítette a szakdolgozatom elkészítését, és a segítségével átfogóbb tudást és hasznos tapasztalatokat szerezhettem a jelfeldolgozás témakörében.

Ezúton szeretném megköszönni *Mészáros Tamás* jazz-énekes, zeneszerzőnek, hogy szaktudásával hozzájárult a zenei fejezetek helyes megfogalmazásához, és átfogóbb képet adott a zeneelmélet dolgozatban felhasznált témaköreiől.

Szeretném megköszönni *Mirk Nikolettnek*, hogy kitartóan támogatott a dolgozatom elkészítése ideje alatt.

Köszönettel tartozom *Édesapámnak, Édesanyámnak,* és *barátaimnak*, akik támogattak és segítettek az egyetemi éveim alatt.

Ábrák jegyzéke

Táblázatok jegyzéke

Irodalomjegyzék

[1] T. Fujishima. Realtime chord recognition of musical sound: A system using common lisp music. 1999.

[2] M. Goto. A robust predominant-F0 estimation method for real-time detection of melody and bass lines in cd recordings. pages 757–760, 2000.

[3] Harmonic-Mixing. Camelot easymix wheel. `http://www.harmonic-mixing.com/Images/camelotHarmonicMixing.jpg`.

[4] Mixed in Key. `http://www.Harmonic-Mixing.com`.

[5] C. Krumhansl. Cognitive foundations of musical pitch. 1990.

[6] Mixshare. `http://www.mixshare.com`.

[7] Frank Oszkár. *Hangzó Zeneelmélet - A tonális-funkciós zene*. Comenius Bt., 1997.

[8] S. Pauws. Musical key extraction from audio. *Processing of 5th ISMIR*, pages 96–99, 2004.

[9] Christopher Roebuck. Harmonic-mixing: Key & beat detection algorithms. pages 9–10.

[10] Christopher Roebuck. Harmonic-mixing: Key & beat detection algorithms. pages 14–19.

[11] A. Sheh and D.P.W. Ellis. Chord segmentation and recognition using EM-trained hidden markov models. *Processing of 4th ISMIR*, pages 183–189, 2003.

[12] Ugródszeka. A digitális hangtechnika alapjai - (1.rész). `http://ugrock.hu/2010-11-01/Szakbarbar/A_digitalis_hangtechnika_alapjai_-_1_resz/4/`.

[13] Wikipedia. Circle of Fdifths. `http://upload.wikimedia.org/wikipedia/commons/7/74/Circle_of_fifths_deluxe.png`.

[14] Wikipedia. Hangnem. `http://hu.wikipedia.org/wiki/Hangnem`.

[15] Wikipedia. Technics - SL-1200. `http://hu.wikipedia.org/wiki/Technics_SL-1200`.

[16] Wikipedia. Zene. `http://hu.wikipedia.org/wiki/Zene`.

[17] Zenetanoda. http://zenetanoda.mindenkilapja.hu.

[18] Zplane.developement. [tONaRT] key detection. http://www.zplane.de/index.php?page=description-tonart.

Függelék

F.1. Példa Mérési Jegyzőkönyv

A mérési jegyzőkönyvet a MATLAB automatikusan generálja, és egy ellenőrzött zeneszámrészlet eredményeit, ábráit tartalmazza. A jegyzőkönyv célja, hogy - akár önmagában - tájékoztatást nyújtson az alkalmazott módszerekről, és a bemeneti paraméterek és grafikonok alapján következtetéseket lehessen levonni magáról a zeneszámról, és a módszerek hatásfokáról.

A példa Mérési Jegyzőkönyv a következő oldalon kezdődik.

F.2. Példa Összefoglalás

Az Összefoglalás - jól átlátható formában - tartalmazza az összes információt, ami a módszerek paramétereiről és eredményességükről rendelkezésre áll. Segítségével össze lehet hasonlítani a különböző beállításokkal lefuttatott hangnemdetektálások eredményeit, és következtetéseket lehet levonni az ideális paraméterek megválasztásához.

A példa Összefoglalás a 83. oldalon kezdődik.

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Mérési jegyzőkönyv

OneRepublic - All The Right Moves (Részlet)

Hangnemének meghatározása

Készítette
Fülöp Tibor

Konzulens
Bank Balázs

Program Verzió: 1.0 - 8. Készült: 2012. május 13. - 23:31.

1. A dal adatai

- **Előadó**: OneRepublic
- **Dalcím**: All The Right Moves
- **Album**: Waking Up
- **Hossz**: 26.36 *másodperc*

2. Konfigurációs paraméterek

- **Decimált mintavételi frekvencia**: 22050 *Hz*
- **Decimálás értéke**: 2
- **Jelablak szélesség**: 0.7430 *másodperc*
- **Ablak átlapolás**: 70.0%
- **A skálán kívüli hangok büntetésének mértéke**: 1.0
- **Tesztfájlok típusa**: Részlet
- **Grafikonok exportálási beállítása**: PDF fájl

2.1. Módszerspecifikus paraméterek

- Első módszer

- **A zenei frekvenciák környékének felhasznált tartománya**: 60%

- Második módszer

- **A zenei frekvenciák környékének felhasznált tartománya**: 60%

- Harmadik módszer

- **Minimális valószínűség**: 90%
- **Maximális harmónia találat**: 6 *darab*
- **A zenei frekvenciák környékének felhasznált tartománya**: 60%

- Negyedik módszer

- **Jelleggörbe átlagolási tartomány**: ±250 *minta*
- **Minimális csúcs távolság maximumkeresés esetén**: 15 *minta*
- **Minimális jelszint maximumkeresés esetén**: -15 *dB*

3. Detektált és valós hangnemek

Módszer/Szoftver	Hangnem	Előjegyzés eltérés
Valós hangnem	**Am**	-
Rapid Evolution 3	C	**Ok**
Első Módszer	Am	**Ok**
Második Módszer	C	**Ok**
Harmadik Módszer	C	**Ok**
Negyedik Módszer	G	1♯

1. táblázat. *Hangnemek - Összefoglaló táblázat*

4. Első Módszer

4.1. A módszer áttekintése

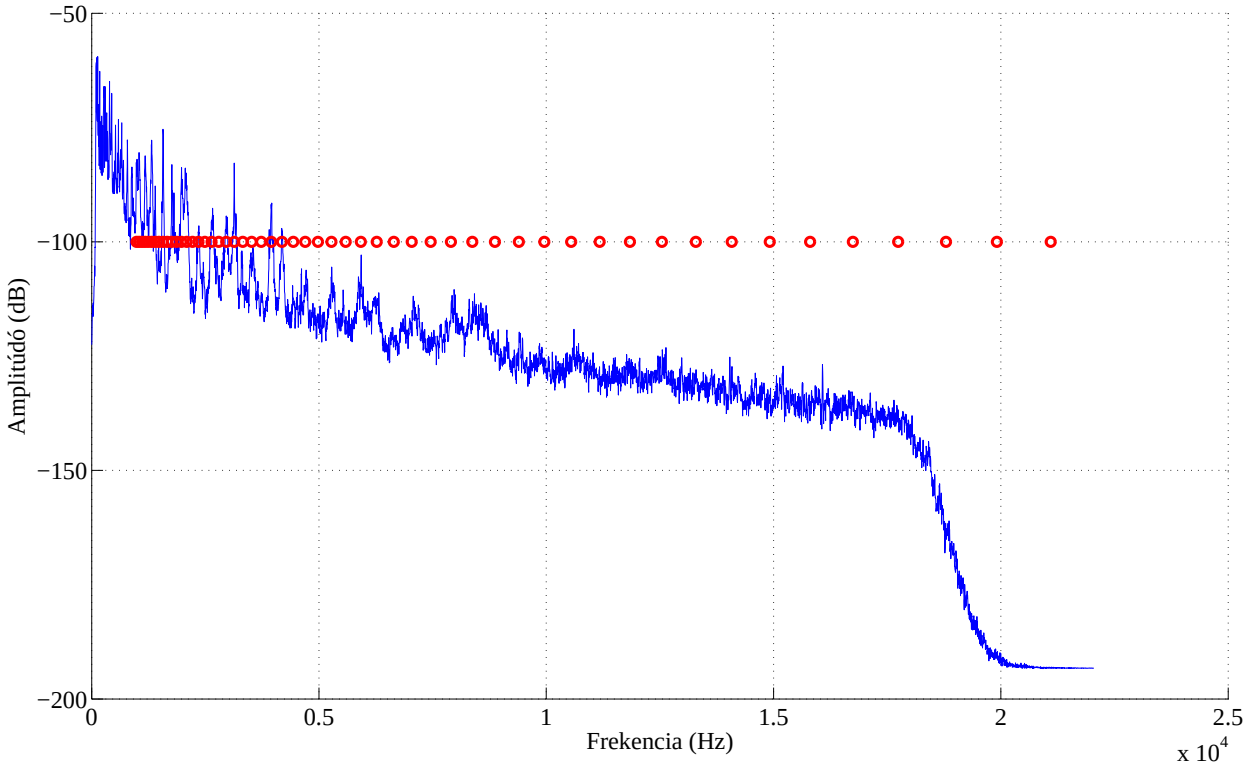
A hangfájlból monósítás és **2**. rendű decimálás után **0.7430** másodperc szélességű időablakot olvasunk ki. Az időablakot *Hamming*-ablakkal való szorzás után *Fourier*-transzformáljuk, így megkapjuk az ablakozott jel spektrumát. Az ablakozott jel spektrumával növeljük a spektrumok négyzetösszegét tartalmazó vektort. Az ablak pozícióját **70.0** % ablakszélességgel eltoljuk és a folyamatot a hangfájl végéig ismételjük.

A spektumok összegét tartalmazó vektor értékét elosztjuk az ablakok számával, így megkapjuk a hangfájl spektrumának átlagát. A zenei hangok frekvenciáinak környezetében (a hangok távolságának ±**60**%-ában) kiszámoljuk a spektrum frekvenciáihoz tartozó energiáit (*2. ábra*), majd a kapott értékeket oktávonként összegezzük (*3. ábra*).

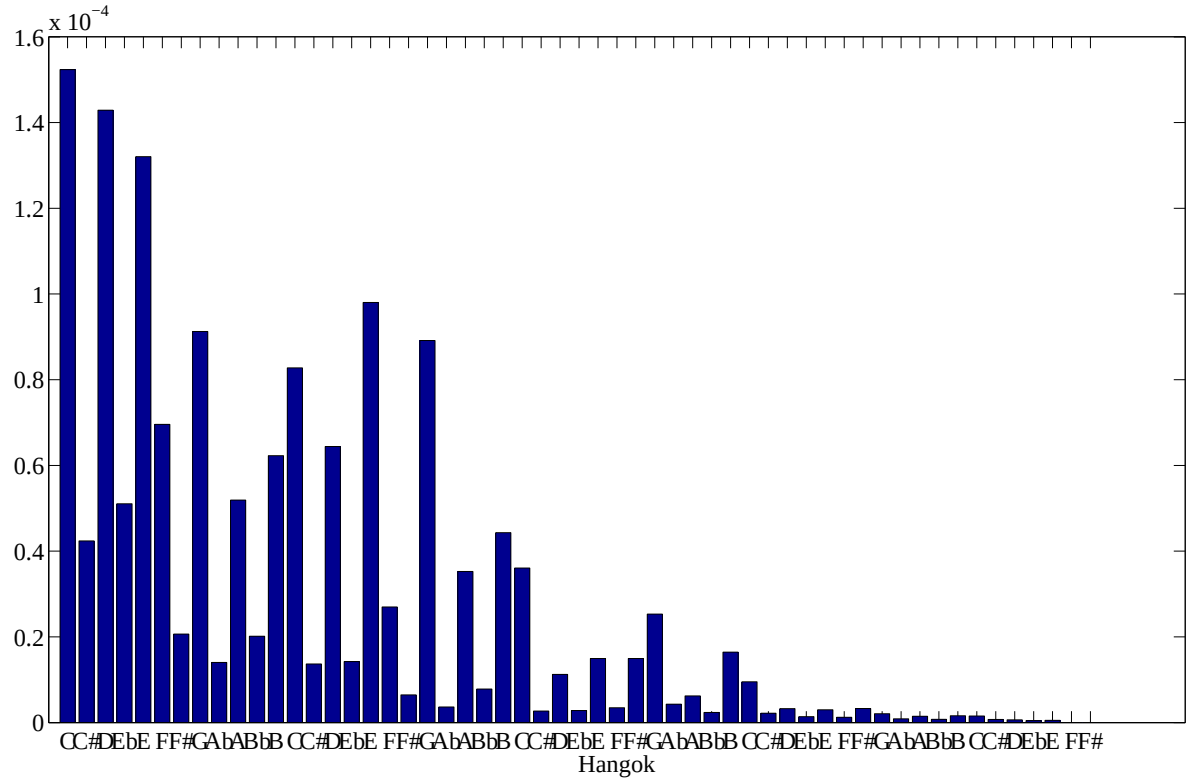
A detektált hangnem az oktávonként összegzett értékek és a *skálasablon-vektorok* szorzatának maximuma lesz.

4.2. Ábrák, Táblázatok

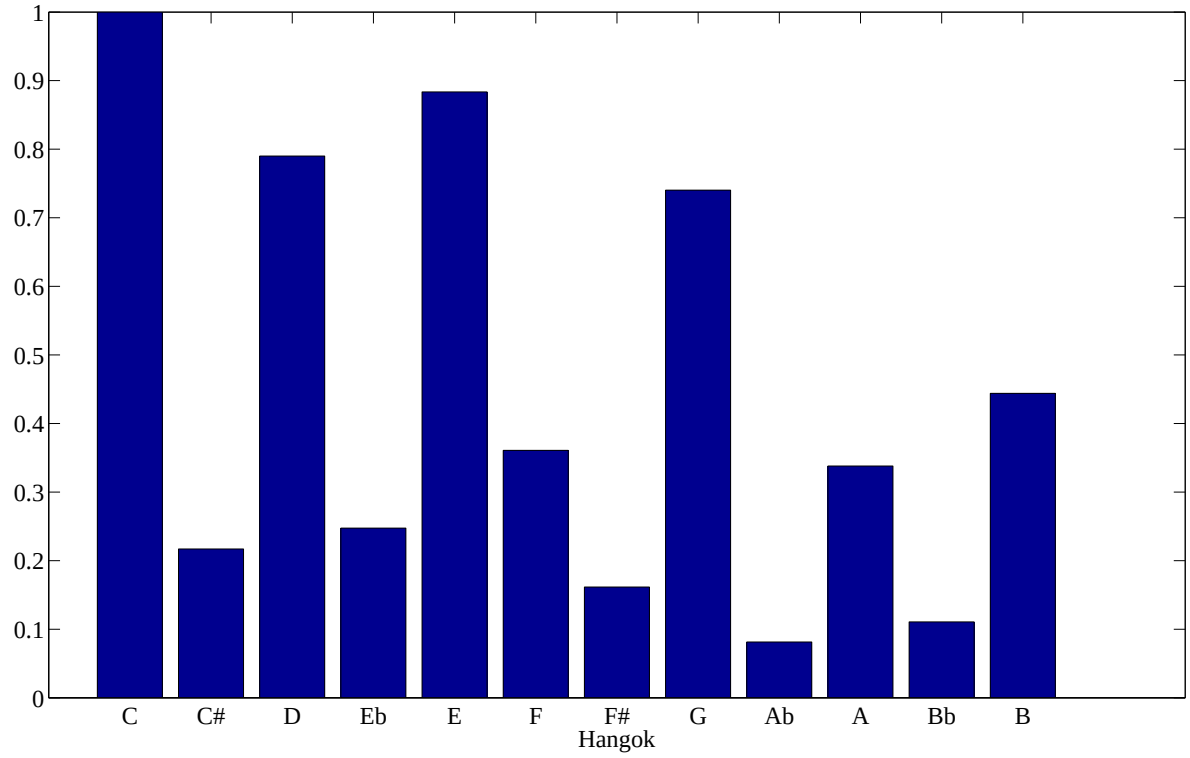
Az összegzett spektrum és a zenei hangok frekvenciái (*piros körökkel jelölve*) a *1. ábrán* láthatóak.



1. ábra. *1. Módszer - Jel összegzett spektruma a zenei hangok frekvenciáival (piros körök)*



2. ábra. *1. Módszer - A zenei hangok frekvenciáihoz tartozó energiák*



3. ábra. *1. Módszer - A zenei hangok frekvenicáihoz tartozó energiák, oktávonként összegezve*

Hangnem:	Am	C	Dm	G	Em	F	Gm	B♭
Érték:	3.8003	3.7384	3.6826	3.3399	3.1655	3.0721	3.0015	1.7998
Hangnem:	D	Cm	Bm	E♭	Fm	A	F♯m	A♭
Érték:	1.7740	1.7529	1.5260	1.2866	1.2172	0.4563	0.4325	0.1406
Hangnem:	B♭m	E	C♯m	E♭m	A♭m	C♯	B	F♯
Érték:	0.0127	-0.6289	-0.6317	-0.8510	-0.9840	-1.0169	-1.0832	-2.1291

2. táblázat. *4. Módszer - A detektált hangnemek valószínűségi sorrendben*

5. Második Módszer

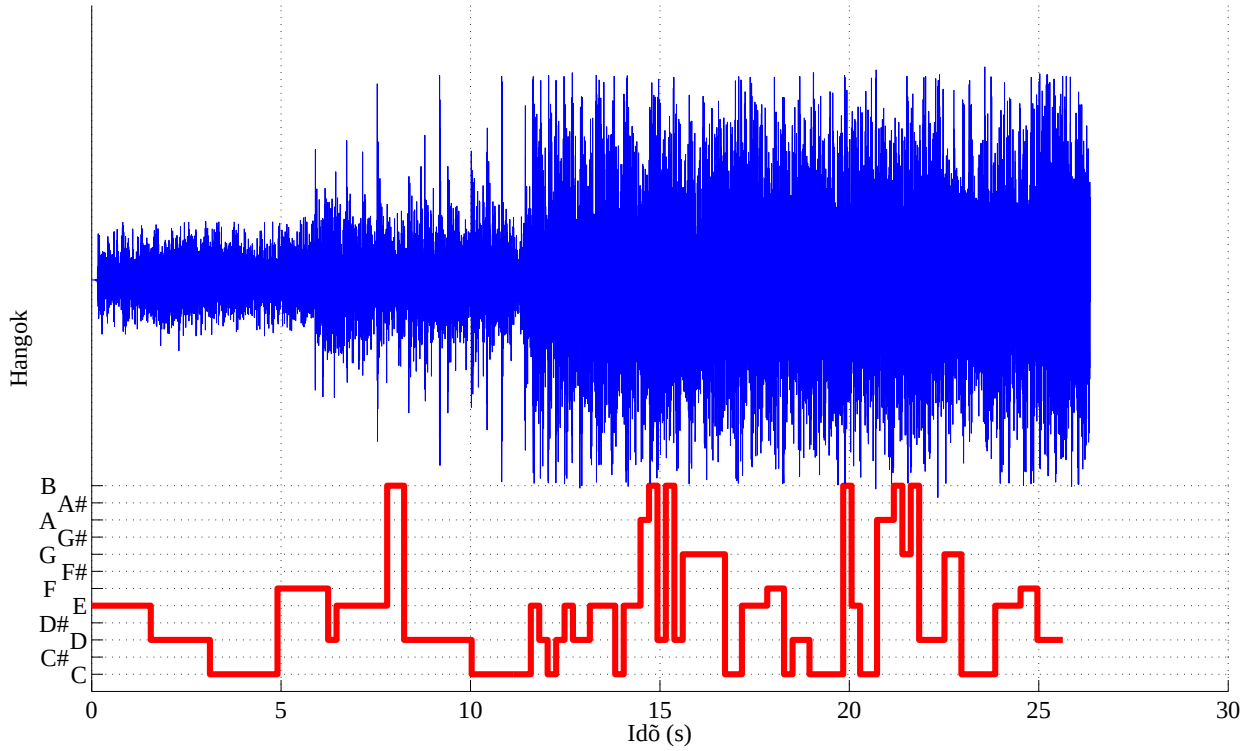
5.1. A módszer áttekintése

A hangfájlból monósítás és **2**-ed rendű decimálás után **0.7430** másodperc szélességű időablakot olvasunk ki. Az időablakot *Hamming*-ablakkal való szorzás után *Fourier*-transzformáljuk, így megkapjuk az ablakozott jel spektrumát. A zenei hangok frekvenciáinak környezetében (a hangok távolságának ±**60%**-ában) kiszámítjuk a spektrum frekvenciáihoz tartozó energiát, majd a kapott értékeket oktávonként összegezzük és a legintenzívebb hang értékét eltároljuk. Az ablak pozícióját **70.0** % ablakszélességgel eltoljuk és a folyamatot a hangfájl végéig ismételjük.

A detektált hangnem az ablakokból elátrolt legnagyobb értékek összegének és a *skálasablon-vektorok* szorzatának maximuma lesz.

5.2. Ábrák, Táblázatok

Az ablakonként tárolt értékeknek megfelelő hangok láthatóak a 4. ábrán.



4. ábra. *2. Módszer - A teljes jel időfüggvénye a detektált hangokkal*

Hangnem:	C	Dm	Am	F	G	Gm	Em	B♭
Érték:	115.0000	111.5000	110.0000	101.0000	95.0000	87.5000	80.0000	47.0000
Hangnem:	Cm	E♭	D	Bm	Fm	A	F♯m	A♭
Érték:	45.5000	41.0000	35.0000	31.0000	26.0000	19.0000	4.0000	-19.0000
Hangnem:	B♭m	C♯	E♭m	E	C♯m	A♭m	B	F♯
Érték:	-23.0000	-35.0000	-39.5000	-41.0000	-42.5000	-45.5000	-47.0000	-81.0000

3. táblázat. *1. Módszer - A detektált hangnemek valószínűségi sorrendben*

6. Harmadik Módszer

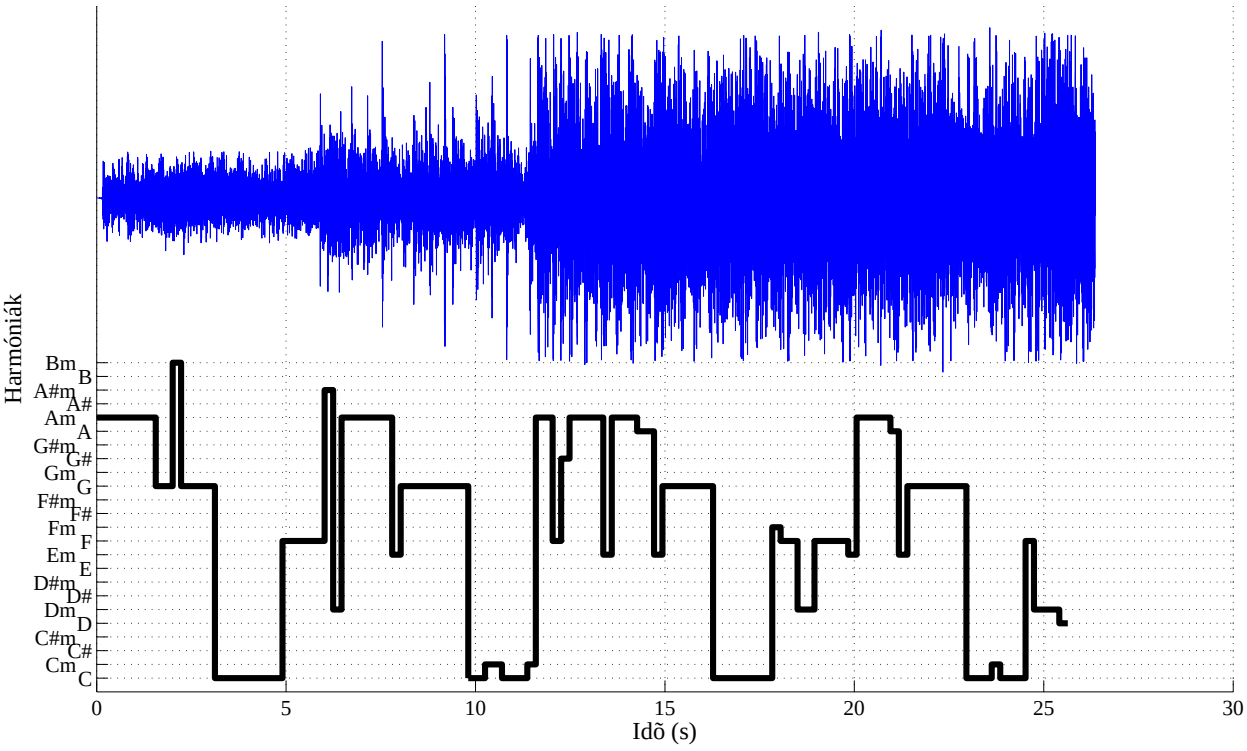
6.1. A módszer áttekintése

A hangfájlból monósítás és **2**. rendű decimálás után **0.7430** másodperc szélességű időablakot olvasunk ki. Az időablakot *Hamming*-ablakkal való szorzás után *Fourier*-transzformáljuk, hogy így megkapjuk az ablak spektrumát. A zenei hangok frekvenciáinak környezetében kiszámoljuk a spektrum frekvenciáihoz tartozó energiáját, majd a kapott értékeket oktávonként összegezzük. Összeszorozzuk az akkord chroma vectorral és normalizáljuk az értékeket. A **70%-nál** kisebb értékű hangnemeket kinullázzuk. Ha az így kapott akkordok száma kevesebb mint **6**, Az akkordot Nem felismerhetőnek definiáljuk. Ha kevesebb mint **6** akkordra igaz a feltétel, megnőveljük a hangok intenzitását tartalmazó vektor értékeit az aktuális hangok intenzitásával. Az ablak pozícióját **70.0** % ablakszélességgel eltoljuk és a folyamatot a hangfájl végéig ismételjük.

A detektált hangnem az eltárolt hangintenzitások és az akkordsablon-vektorok szorzatának maximuma lesz.

6.2. Ábrák, Táblázatok

Az ablakonként tárolt értékeknek megfelelő harmóniák láthatóak a 5. ábrán.



5. ábra. *3. Módszer - A teljes jel időtartományban a detektált harmóniákkal*

Hangnem:	C	Am	Dm	G	F	Em	Gm	B♭
Érték:	183.9781	177.8781	176.3161	158.2630	152.8102	137.0412	135.9232	77.1084
Hangnem:	Cm	D	Bm	E♭	Fm	A	F♯m	A♭
Érték:	68.0661	64.8347	53.4121	39.4983	28.5810	17.7032	6.6406	-12.7121
Hangnem:	B♭m	E	C♯m	C♯	E♭m	B	A♭m	F♯
Érték:	-24.1819	-34.7009	-44.0885	-59.9066	-60.7603	-72.7715	-72.9367	-126.1775

4. táblázat. *2. Módszer - A detektált hangnemek valószínűségi sorrendben*

7. Negyedik Módszer

7.1. A módszer áttekintése

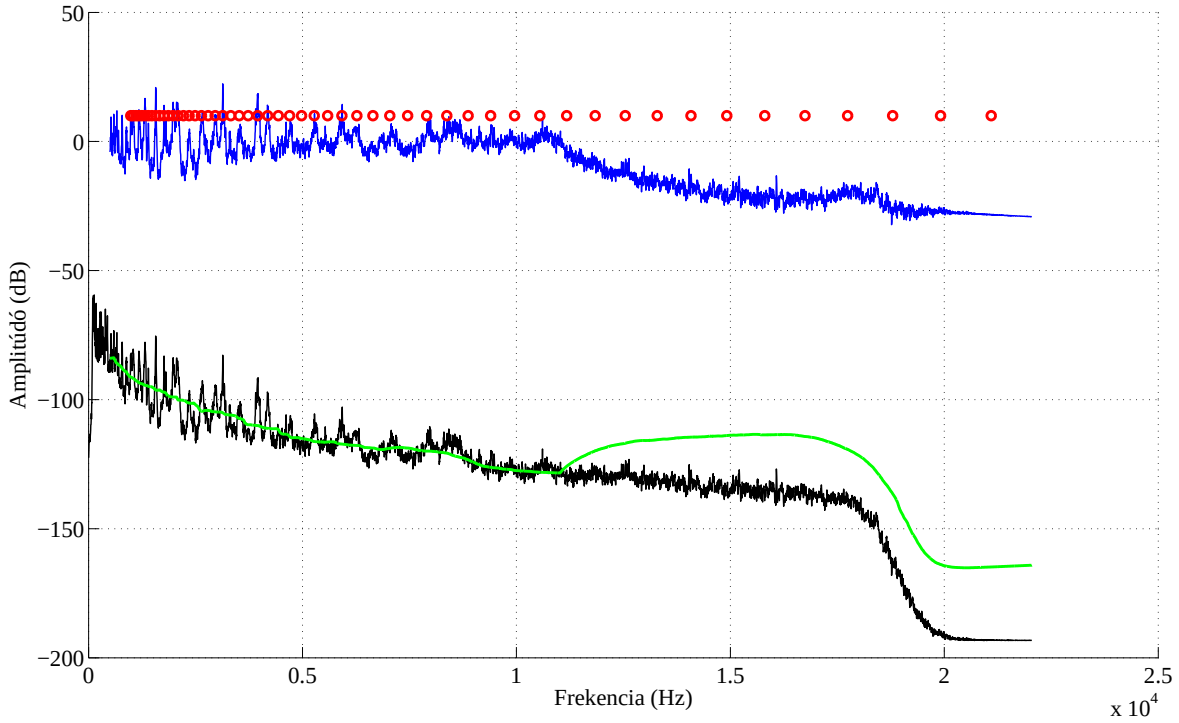
A hangfájlból monósítás és **2**. rendű decimálás után **0.7430** másodperc szélességű időablakot olvasunk ki. Az időablakot *Hamming*-ablakkal való szorzás után *Fourier*-transzformáljuk, így megkapjuk az ablakozott jel spektrumát. Az ablakozott jel spektrumával növeljük a spektrumok összegét tartalmazó vektort. Az ablak pozícióját **70.0** % ablakszélességgel eltoljuk és a folyamatot a hangfájl végéig ismételjük.

A spektumok összegét tartalmazó vektor értékét elosztjuk az ablakok számával, így megkapjuk a hangfájl spektrumának átlagát. Meghatározzuk a görbe jellegét, és kompenzáljuk vele a spektrumot. (A kompenzáló görbe úgy jön létre, hogy a környezetének az értékeit kiátlagolja ± **250** minta szélességben. Ezen felül a spektrum középfrekvenciájától a frekvenciával fordítottan arányosan elnyomjuk a magas frekvenciák amplitúdóját.) Meghatározzuk a (**-15** dB küszöbérték feletti) lokális maximumok frekvenciáját és a maximumok környezetének összegével megnöveljük a legközelebb eső zenei hang értékét a zenei hangokat tartalmazó vektorban (*7. ábra*). A kapott értékeket oktávonként összegezzük (*8. ábra*).

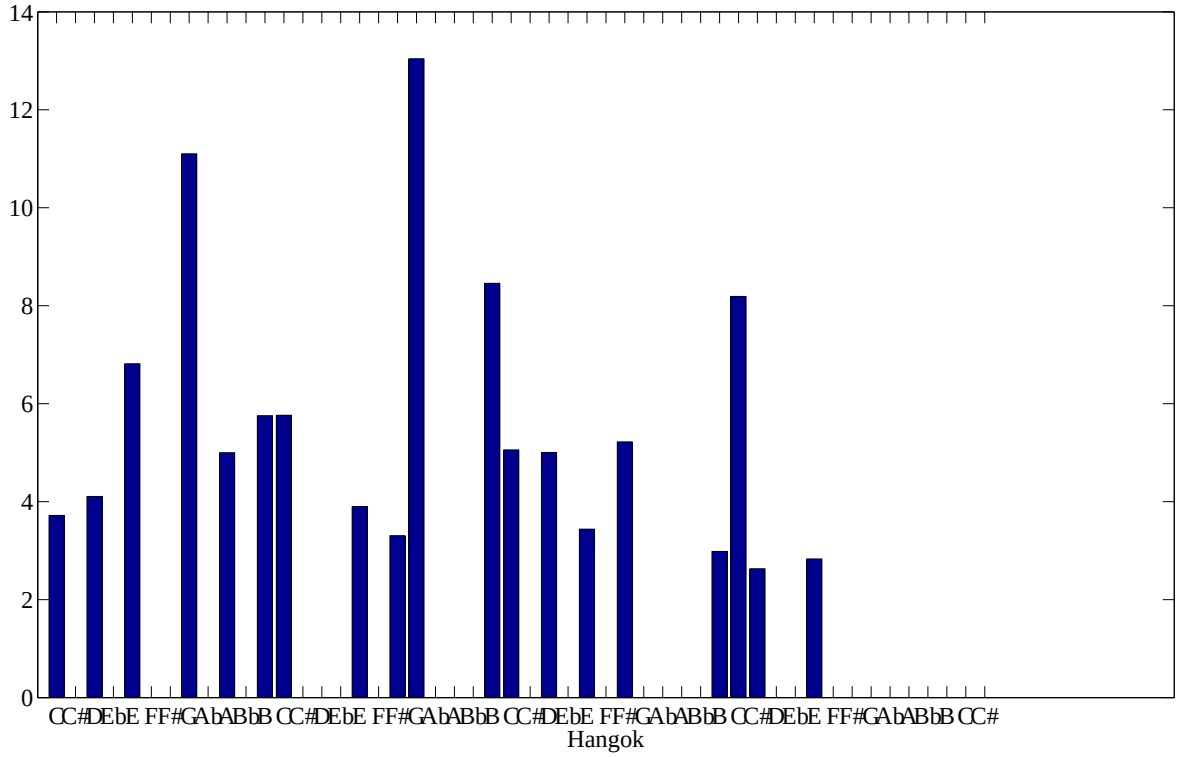
A detektált hangnem az oktávonként összegzett értékek és a *skálasablon-vektorok* szorzatának maximuma lesz.

7.2. Ábrák, Táblázatok

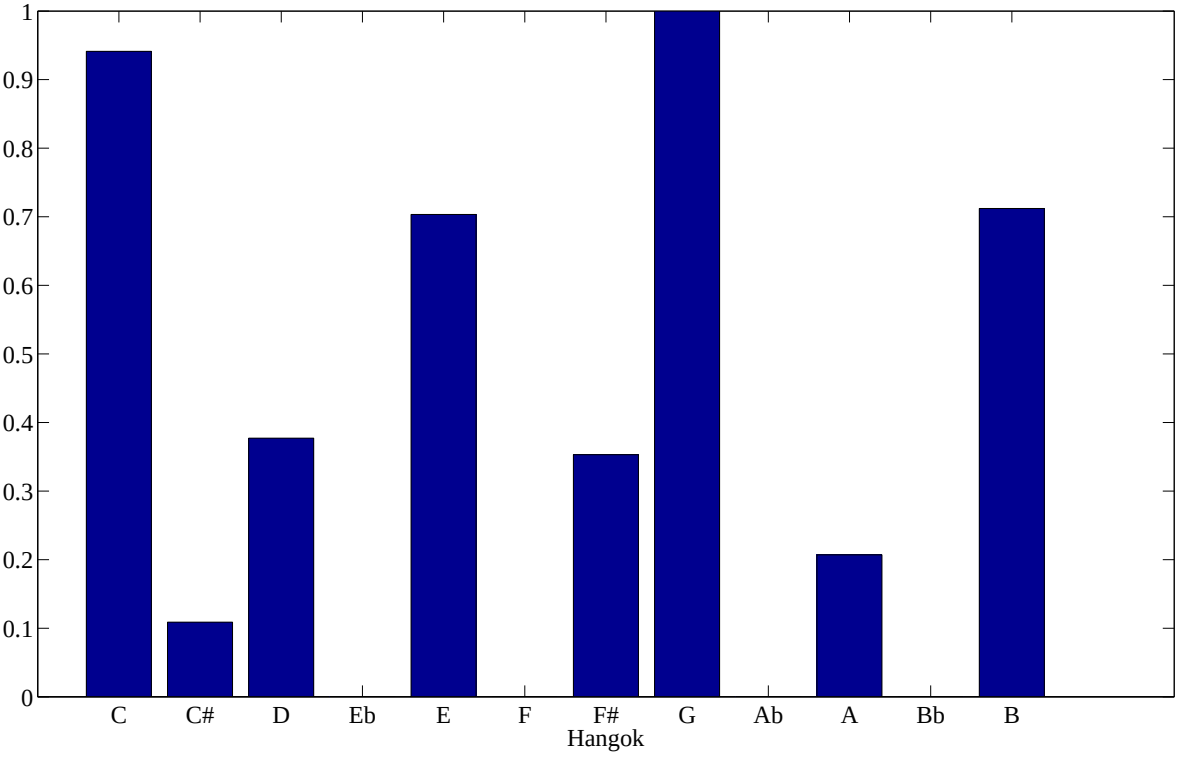
Az *6.* ábrán fekete színnel az összegzett spektrum, piros körökkel a zenei hangok frekvenciái, zöld színnel a kompenzáló görbe, kék színnel pedig a kompenzált spektrum láthatóak.



6. ábra. *4. Módszer - Jel összegzett spektruma a zenei hangok frekvenciáival és kompenzációval*



7. ábra. *4. Módszer - A zenei hangok frekvenciáihoz tartozó energiák*



8. ábra. *4. Módszer - A zenei hangok frekvenicáihoz tartozó energiák, oktávonként összegezve*

Hangnem:	G	Am	Em	C	Dm	D	F	Bm
Érték:	4.1853	4.0088	3.8781	3.4792	3.1231	2.5207	2.0549	2.0207
Hangnem:	Gm	B♭	Cm	A	F♯m	E♭	Fm	E
Érték:	1.7031	0.6480	0.5444	0.5207	0.3320	0.2338	0.2086	-0.2338
Hangnem:	B♭m	A♭	C♯m	B	E♭m	A♭m	C♯	F♯
Érték:	-0.2732	-0.3028	-0.3374	-0.6480	-0.9991	-0.9997	-1.5967	-2.0549

5. táblázat. *3. Módszer - A detektált hangnemek valószínűségi sorrendben*

Összefoglalás

Program Verzió: 1.0 - 8. **2012. május 15. - 15:47.**

1. Statisztikák

Típus	Rapid Evolution	Módszer 1.	Módszer 2.	Módszer 3.	Módszer 4.
Jól detektált (db)	47	33	33	41	17
± 1♭ vagy 1♯ (db)	8	26	18	17	25
± 2♭ vagy 2♯ (db)	2	1	5	2	14
Maradék találat (db)	6	3	7	3	7
Jól detektált	74.60 %	52.38 %	52.38 %	65.08 %	26.98 %
± 1♭ vagy 1♯	12.70 %	41.27 %	28.57 %	26.98 %	39.68 %
± 2♭ vagy 2♯	3.17 %	1.59 %	7.94 %	3.17 %	22.22 %
Maradék találat	9.52 %	4.76 %	11.11 %	4.76 %	11.11 %
Max ± 1♭ vagy 1♯ tévedés	87.30 %	93.65 %	80.95 %	92.06 %	66.67 %

1. táblázat. *Statisztikák*

2. Paraméterek

- **Decimálás értéke**: 2
- **Jelablak szélesség**: 0.7430 *másodperc*
- **Ablak átlapolás**: 70.0%
- **A skálán kívüli hangok büntetésének mértéke**: 1.0
- **Tesztfájlok típusa**: Részlet
- **Grafikonok exportálási beállítása**: PDF fájl

2.1. Módszerspecifikus paraméterek

- **Első módszer**
  - **A zenei frekvenciák környékének felhasznált tartománya**: 60%
- **Második módszer**
  - **A zenei frekvenciák környékének felhasznált tartománya**: 60%
- **Harmadik módszer**
  - **Minimális valószínűség**: 90%
  - **Maximális harmónia találat**: 6 *darab*
  - **A zenei frekvenciák környékének felhasznált tartománya**: 60%
- **Negyedik módszer**
  - **Jelleggörbe átlagolási tartomány**: ±250 *minta*
  - **Minimális csúcs távolság maximumkeresés esetén**: 15 *minta*
  - **Minimális jelszint maximumkeresés esetén**: -15 *dB*

3. Hangnem és előjegyzés

SN	Előadó - Dalcím	OR	RE	M1	M2	M3	M4	RE	M1	M2	M3	M4
1.	Doctor P - Big Boss	Fm	Fm	B♭m	Dm	Dm	Cm	**Ok**	1♭	3♯	3♯	1♯
2.	Dubsidia - Here comes trouble	Em	Em	Bm	Em	Em	A♭	**Ok**	1♯	**Ok**	**Ok**	5♭
3.	Vaski - Hurricane	Gm	Gm	B♭	B♭	B♭	A♭m	**Ok**	**Ok**	**Ok**	**Ok**	5♭
4.	Modestep - Sunlight	Am	Am	Dm	Dm	C	C	**Ok**	1♭	1♭	**Ok**	**Ok**
5.	Steve Aoki - Earthquakey People	Bm	Bm	Em	Em	Em	A	**Ok**	1♭	1♭	1♭	1♯
6.	Pendulum - Witchcraft	Fm	Fm	Fm	Fm	Fm	Gm	**Ok**	**Ok**	**Ok**	**Ok**	2♯
7.	Dodge and Fuski - Come Again	Fm	F	B♭m	Gm	Fm	Fm	3♯	1♭	2♯	**Ok**	**Ok**
8.	Flux Pavilion - Bass Cannon	A♭m	A♭m	C♯m	C♯m	C♯m	A	**Ok**	1♭	1♭	1♭	2♭
9.	Nero (Skrillex Remix) - Promises	Am	Am	Dm	C	C	Dm	**Ok**	1♭	**Ok**	**Ok**	1♭
10.	Pendulum - Hold Your Colour	Gm	G	Gm	F	F	Dm	3♯	**Ok**	1♯	1♯	1♯
11.	Skrillex - My Name Is Skrillex	Fm	Fm	B♭m	B♭m	B♭m	B♭m	**Ok**	1♭	1♭	1♭	1♭
12.	Infected Mushroom - Becoming Insane	Am	Am	Dm	Dm	Dm	Dm	**Ok**	1♭	1♭	1♭	1♭
13.	Delirium ft. Sarah McLachan - Silence	Am	Am	C	C	C	Am	**Ok**	**Ok**	**Ok**	**Ok**	**Ok**
14.	Gui Boratto - Haute Couture	Bm	Bm	A	Bm	Bm	A	**Ok**	1♯	**Ok**	**Ok**	1♯
15.	Domateck - After the Fall	Am	Am	G	G	G	Em	**Ok**	1♯	1♯	1♯	1♯
16.	David Guetta Ft. Kid Cudi - Memories	B♭m	B♭	Fm	C♯	Fm	Fm	3♯	1♯	**Ok**	1♯	1♯
17.	Inna - Amazing (Radio Version)	E♭m	E♭m	F♯	F♯	F♯	A♭m	**Ok**	**Ok**	**Ok**	**Ok**	1♭
18.	Dj Marcus - Atlantis Summer Mix	Gm	D	Gm	Gm	B♭	Am	4♯	**Ok**	**Ok**	**Ok**	2♯
19.	Alexandra Stan - Mr. Saxo Beat	Bm	Bm	Bm	A	Bm	B	**Ok**	**Ok**	1♯	**Ok**	3♯
20.	Avicii - Levels (Original Mix)	E	E	F♯m	C♯m	E	E	**Ok**	1♭	**Ok**	**Ok**	**Ok**
21.	Chris Lake - Only One (Original Mix)	F♯m	F♯m	B♭m	C♯	C♯	Fm	**Ok**	4♯	4♯	4♯	7♭
22.	Kid Cudi vs. Crookers - Day n Night	Bm	G	Em	Em	F♯m	C♯m	1♭	1♭	1♭	1♯	2♯
23.	Wawa (Lauer and Canard) - Sombrita	Bm	F♯m	F♯m	A	F♯m	Bm	1♯	1♯	1♯	1♯	**Ok**
24.	Yolanda Be Cool - We No Speak Americano	B♭m	B♭m	B♭m	B♭m	B♭m	E♭	**Ok**	**Ok**	**Ok**	**Ok**	2♯
25.	Britney Spears - Till the world ends	Cm	A♭	Cm	Cm	Cm	Gm	1♭	**Ok**	**Ok**	**Ok**	1♯
26.	E-Type - True Believer	F♯	A♭m	B♭m	Bm	B♭m	Fm	1♭	1♯	4♭	1♯	2♯
27.	Kelly Clarkson - My life would suck without U	A	A	F♯m	A	F♯m	F♯m	**Ok**	**Ok**	**Ok**	**Ok**	**Ok**
28.	30H!3 - My First Kiss ft. Kesha	Em	Em	Am	G	G	C	**Ok**	1♭	**Ok**	**Ok**	1♭
29.	Pitbull Ft. Marc Anthony - Rain Over Me	Am	Am	C	Bm	C	Dm	**Ok**	**Ok**	2♯	**Ok**	1♭
30.	Don Omar - Danza Kudoro	B♭m	B♭m	Gm	Gm	Gm	B♭m	**Ok**	3♯	3♯	3♯	**Ok**
31.	Example - Changed The Way You Kiss Me	Em	Em	Em	Em	Em	F♯m	**Ok**	**Ok**	**Ok**	**Ok**	2♯
32.	Fatboy Slim - The Rockafeller Skank	G	G	Gm	Gm	Dm	Cm	**Ok**	3♭	3♭	2♭	4♭
33.	K'naan - Wavin' Flag	C	C	Dm	C	C	C	**Ok**	1♭	**Ok**	**Ok**	**Ok**
34.	Busted - What I Go to School For	A	A	F♯m	A	A	D	**Ok**	**Ok**	**Ok**	**Ok**	1♭
35.	Good Charlotte - Misery	A♭m	E♭m	A♭m	B	B	E♭m	1♯	**Ok**	**Ok**	**Ok**	1♯
36.	Simple Plan - Welcome to My Life	B♭m	C♯	B♭m	Bm	B♭m	F♯	**Ok**	**Ok**	7♯	**Ok**	1♭
37.	Green Day - Holiday	Fm	Fm	Fm	Fm	Fm	Fm	**Ok**	**Ok**	**Ok**	**Ok**	**Ok**
38.	Blink-182 - First Date	C	C	Am	Bm	Am	Am	**Ok**	**Ok**	2♯	**Ok**	**Ok**
39.	Avril Lavigne - My happy ending	Bm	D	Em	D	D	D	**Ok**	1♭	**Ok**	**Ok**	**Ok**
40.	30 Seconds To Mars - From Yesterday	B♭	B♭	Gm	B♭	Gm	Gm	**Ok**	**Ok**	**Ok**	**Ok**	**Ok**
41.	Zebrahead - Jenny From The Block	A♭m	A♭m	E♭m	F♯	E♭m	A♭m	**Ok**	1♯	1♯	1♯	**Ok**
42.	Fall Out Boy - Dance, dance	Bm	D	Em	D	Bm	F♯m	**Ok**	1♭	**Ok**	**Ok**	1♯
43.	Demi Lovato - Get Back	G	C	Dm	Dm	Dm	Dm	1♭	2♭	2♭	2♭	2♭
44.	Panic At the Disco - I Write Sins Not Tragedies	Am	F	Am	C	C	Dm	1♭	**Ok**	**Ok**	**Ok**	1♭
45.	Bob Marley - No Woman No Cry	C♯	C♯	B♭m	B♭m	B♭m	C♯	**Ok**	**Ok**	**Ok**	**Ok**	**Ok**
46.	Reel Big Fish - I Want Your Girlfriend	G	G	Em	Bm	G	Am	**Ok**	**Ok**	1♯	**Ok**	1♭
47.	Cozombolis - Fülig ér a száj	G	G	Em	Em	Am	Am	**Ok**	**Ok**	**Ok**	1♭	1♭
48.	Elton John, Tim Rice - Hakuna matata	C	C	Am	Gm	Am	Gm	**Ok**	**Ok**	2♭	**Ok**	2♭
49.	Dub Fx - Love Someone (Another version)	Em	A	Em	Em	Em	F♯m	2♯	**Ok**	**Ok**	**Ok**	2♯
50.	Eminem - The Real Slim Shady	Cm	E♭	Cm	Fm	Cm	Dm	**Ok**	**Ok**	1♭	**Ok**	2♯
51.	Fort Minor - Remember the name	Cm	E♭	Fm	E♭	E♭	Gm	**Ok**	1♭	**Ok**	**Ok**	1♯
52.	Missy Elliot - Work It (Remix)	Cm	Am	Gm	G	Gm	A♭m	3♯	1♯	4♯	1♯	4♭
53.	Ne-Yo - So Sick	E♭m	F♯	E♭m	F♯	E♭m	C♯	**Ok**	**Ok**	**Ok**	**Ok**	1♯
54.	Pussycat Dolls and Snoop Dogg - Buttons	Dm	Dm	Dm	F	Dm	Dm	**Ok**	**Ok**	**Ok**	**Ok**	**Ok**
55.	OneRepublic - All The Right Moves	Am	C	Am	C	C	G	**Ok**	**Ok**	**Ok**	**Ok**	1♯
56.	Nova Prospect - Szabadon	A♭m	B	A♭m	C♯m	B	A♭m	**Ok**	**Ok**	1♭	**Ok**	**Ok**
57.	Limp Bizkit - Shotgun	C♯m	C♯m	A♭m	A♭m	A♭m	C♯m	**Ok**	1♯	1♯	1♯	**Ok**
58.	Depresszió - Közeli helyeken	Em	D	Em	G	G	C♯m	1♯	**Ok**	**Ok**	**Ok**	3♯
59.	HIM - Join me in Death	A♭m	A♭m	B	B	B	F♯m	**Ok**	**Ok**	**Ok**	**Ok**	2♭
60.	Linkin Park - In The End	E♭m	E♭m	E♭m	C♯	C♯	A♭	**Ok**	**Ok**	1♯	1♯	2♯
61.	Metallica - Nothing Else Matters	Em	Em	G	G	G	Bm	**Ok**	**Ok**	**Ok**	**Ok**	1♯
62.	Chad Kroeger, Josey Scott - Hero	Bm	B	F♯m	F♯m	F♯m	C♯m	3♯	1♯	1♯	1♯	2♯
63.	No doubt - Don't Speak	Cm	F	A♭	A♭	A♭	Fm	2♯	1♭	1♭	1♭	1♭

2. táblázat. *Detektált hangnemek és előjegyzések összefoglaló táblázat*

4. Hangnem detektálások időtartama

Sorszám.	Előadó - Dalcím	Módszer 1.	Módszer 2.	Módszer 3.	Módszer 4.
1.	Doctor P - Big Boss	5.2975 s	5.4061 s	5.3908 s	4.6018 s
2.	Dubsidia - Here comes trouble (Original Mix)	3.6761 s	8.3797 s	5.5561 s	3.9222 s
3.	Vaski - Hurricane	4.8640 s	5.3943 s	5.5340 s	3.8785 s
4.	Modestep - Sunlight	3.9264 s	6.8214 s	5.5484 s	4.0297 s
5.	Steve Aoki, Rivers Cuomo - Earthquakey People	4.2918 s	7.4657 s	6.9798 s	9.0445 s
6.	Pendulum - Witchcraft	3.7273 s	4.6187 s	4.5502 s	3.8165 s
7.	Dodge and Fuski - Come Again (Original Mix)	4.2380 s	3.1422 s	3.2279 s	3.6946 s
8.	Flux Pavilion - Bass Cannon	3.7315 s	5.4624 s	6.7347 s	3.9317 s
9.	Nero (Skrillex Remix) - Promises	4.8830 s	5.4930 s	5.2485 s	3.7463 s
10.	Pendulum - Hold Your Colour	3.3602 s	4.4941 s	4.6326 s	3.7273 s
11.	Skrillex - My Name Is Skrillex (Skrillex Remix)	4.1775 s	6.2488 s	5.8954 s	5.6688 s
12.	Infected Mushroom - Becoming Insane	4.3301 s	7.4006 s	7.3313 s	4.6324 s
13.	Delirium (ft. Sarah McLachan) - Silence	4.8090 s	6.2475 s	6.3202 s	4.6067 s
14.	Gui Boratto - Haute Couture	3.8384 s	5.8811 s	5.9972 s	4.0265 s
15.	Domateck - After the Fall	5.2766 s	5.2427 s	5.2846 s	3.8419 s
16.	David Guetta Ft. Kid Cudi - Memories	3.7143 s	5.8644 s	5.7086 s	4.1638 s
17.	Inna - Amazing (Radio Version)	2.8216 s	3.5265 s	3.1464 s	3.1833 s
18.	Dj Marcus - Atlantis Summer Mix	2.8903 s	3.2955 s	4.0276 s	3.5630 s
19.	Alexandra Stan - Mr. Saxo Beat	3.0468 s	3.2986 s	3.2637 s	3.3946 s
20.	Avicii - Levels (Original Mix)	3.4210 s	4.4314 s	4.3140 s	3.5116 s
21.	Chris Lake - Only One (Original Mix)	3.3889 s	4.1945 s	4.3328 s	4.0079 s
22.	Kid Cudi vs. Crookers - Day n Night	5.2729 s	5.7686 s	5.7487 s	3.9684 s
23.	Wawa (Lauer and Canard) - Sombrita	4.3178 s	7.3438 s	7.3052 s	5.1487 s
24.	Yolanda Be Cool - We No Speak Americano	4.0380 s	6.2985 s	6.2356 s	4.3589 s
25.	Britney Spears - Till the world ends	3.5664 s	3.2653 s	3.4640 s	3.1708 s
26.	E-Type - True Believer	4.0184 s	5.6645 s	5.5820 s	4.1703 s
27.	Kelly Clarkson - My Life Would Suck Without You	3.7398 s	5.7032 s	5.3263 s	3.8387 s
28.	30H!3 - My First Kiss ft. Kesha	3.9620 s	6.1123 s	5.3632 s	4.5198 s
29.	Pitbull Ft. Marc Anthony - Rain Over Me	4.6505 s	6.2525 s	6.0883 s	4.5553 s
30.	Don Omar - Danza Kudoro	3.5254 s	4.5878 s	4.5310 s	4.0547 s
31.	Example - Changed The Way You Kiss Me	6.9076 s	6.2826 s	6.3652 s	4.5167 s
32.	Fatboy Slim - The Rockafeller Skank	3.4140 s	5.0446 s	5.9698 s	3.8064 s
33.	K'naan - Wavin' Flag	4.5272 s	4.9575 s	5.0028 s	3.8380 s
34.	Busted - What I Go to School For	3.6988 s	5.4809 s	5.3268 s	3.6665 s
35.	Good Charlotte - Misery	3.2979 s	4.8916 s	4.4747 s	3.4324 s
36.	Simple Plan - Welcome to My Life	4.0157 s	4.7966 s	4.7756 s	4.1857 s
37.	Green Day - Holiday	3.3328 s	5.0352 s	4.9642 s	3.6102 s
38.	Blink-182 - First Date	3.8143 s	4.9884 s	4.9634 s	4.3166 s
39.	Avril Lavigne - My happy ending	3.2227 s	4.9167 s	4.7965 s	3.5272 s
40.	30 Seconds To Mars - From Yesterday	3.5314 s	5.4688 s	6.7456 s	4.0076 s
41.	Zebrahead - Jenny From The Block	3.0016 s	3.7323 s	3.7675 s	3.2545 s
42.	Fall Out Boy - Dance, dance	4.1341 s	4.8828 s	4.8704 s	3.5965 s
43.	Demi Lovato - Get Back	5.5101 s	8.8182 s	8.5475 s	4.8281 s
44.	Panic At the Disco - I Write Sins Not Tragedies	3.2776 s	4.6451 s	4.7439 s	3.9975 s
45.	Bob Marley - No Woman No Cry	3.6416 s	4.6654 s	5.6758 s	3.6573 s
46.	Reel Big Fish - I Want Your Girlfriend	4.2057 s	6.0266 s	5.9209 s	4.5078 s
47.	Cozombolis - Fülig ér a száj	4.7259 s	6.8528 s	6.7620 s	4.2522 s
48.	Elton John, Tim Rice - Hakuna matata	3.3676 s	4.9574 s	4.8797 s	3.7066 s
49.	Dub Fx - Love Someone (Another version)	3.5428 s	5.1600 s	5.1038 s	3.7296 s
50.	Eminem - The Real Slim Shady (Album version)	2.8777 s	3.4730 s	3.4905 s	4.4385 s
51.	Fort Minor - Remember the name	2.8094 s	3.7463 s	3.4710 s	3.2310 s
52.	Missy Elliot - Work It (Remix)	3.6074 s	3.4571 s	3.4863 s	3.5622 s
53.	Ne-Yo - So Sick	3.0458 s	4.6187 s	4.1420 s	3.7048 s
54.	Pussycat Dolls and Snoop Dogg - Buttons	3.0327 s	3.8599 s	3.7596 s	3.3596 s
55.	OneRepublic - All The Right Moves	3.4897 s	5.1604 s	5.0550 s	3.6950 s
56.	Nova Prospect - Szabadon	3.1446 s	3.4637 s	3.3207 s	3.2704 s
57.	Limp Bizkit - Shotgun	2.8018 s	3.9526 s	3.3220 s	3.6142 s
58.	Depresszió - Közeli helyeken	3.2317 s	4.7495 s	5.1017 s	3.8003 s
59.	HIM - Join me in Death	3.6905 s	5.7332 s	5.6685 s	4.2003 s
60.	Linkin Park - In The End	2.9245 s	4.0517 s	4.1252 s	3.4649 s
61.	Metallica - Nothing Else Matters	4.5638 s	8.0053 s	7.6706 s	4.5248 s
62.	Chad Kroeger, Josey Scott - Hero	4.8506 s	8.9235 s	9.3455 s	4.7791 s
63.	No doubt - Don't Speak	3.3884 s	5.3452 s	5.0010 s	4.0355 s

3. táblázat. *A hangnem detektálások időtartamának összefoglaló táblázata*