Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Híradástechnikai Tanszék

Várfi László

Hangmagasság-detektálás többszólamú hangmintában

Konzulens

Firtha Gergely

Budapest, 2012

Összefoglaló

Már a jelfeldolgozás kezdeti státuszában is sűrűn foglalkoztatott téma volt a hangmagasság detektálása a zenében. Ahogy fejlődtek a jelfeldolgozó eszközök, egyre sikeresebben lehetett több szólamban is meghatározni a hangmagasságot, illetve akkordokat detektálni.

Szakdolgozatomban egy olyan program megvalósítását tűztem ki célul, amely nemcsak detektálja egy hangszer (pl. gitár) akkordjának hangjait, hanem nagy pontossággal vissza is adja, hogy milyen akkordról van szó, még hosszabb akkordmenetek esetén is. Megoldásomban megjelennek időtartománybeli módszerek is, de az algoritmus gerincét a frekvenciatartományban valósítottam meg. Emellett az akkord meghatározásához szükséges statisztikai módszerekkel is foglalkozok, majd a legalkalmasabb megoldást választva mutatom be a rendszer utolsó elemét, az akkord osztályozásához szükséges részegységet. Dolgozatomban kitérek az alkalmazott módszerek hibáira, gyengeségeire is, majd bemutatom, hogyan igyekeztem ezen ismert problémák kiküszöbölését megoldani. Végül bemutatom a megvalósított rendszert, melynek paramétereinek beállítását több minta alapján igyekeztem a lehető legáltalánosabban használhatóra hangolni. Bár a megoldásom még fejlesztésre szorul, de a kitűzött problémát a bemutatott algoritmus igen nagy pontossággal képes végrehajtani.

Abstract

Pitch detection in musical samples has been around since an early stage of signal processing. As signal processing devices had developed, detecting notes or chords of samples had advanced as well.

In my paper my aim was to implement a program that not only detects the frequencies of the chords' notes in a recorded sample (e.g. a guitar chord progression), but determines which chord the notes correspond to, even in the case of a longer chord progression. My solution also involves time-domain techniques, but most of the algorithm was implemented in the frequency domain. Besides detection, I also analyze statistical methods for chord classification, and after choosing the appropriate method, I implement the last part of my system, which is responsible for chord classification. In my paper I examine the methods' weaknesses, and how I managed to overcome those problems. Finally, I present the system itself, which I tried to implement to be versatile by setting its parameters to give satisfactory results for every sample I recorded for testing. Although the program has to be developed at some points, it solves the problem of chord detection very accurately.

Tartalomjegyzék

Ábrajegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Várfi László**, 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 hitelesített 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.

Kelt: Budapest, 2012. 12. 07.

...…….

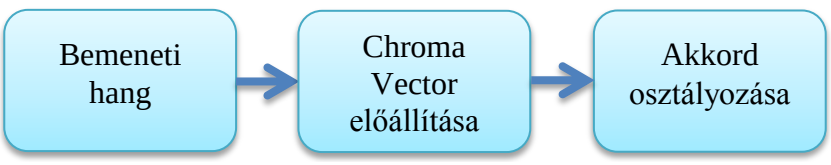
Várfi László

1 Bevezetés

A zene harmonikus tartalmának analízise különösen fontossá vált a nyugati hangzásvilágú zenék komponálása, megértése során, hiszen e zenei környezet alapkövei az akkordok, illetve váltakozásuk adja a számunkra megszokott hangzást, zenét. Az akkordok felosztása legegyszerűbben a hangsor megszólaltatott fokainak számával tehető meg, tehát két fok megszólaltatása úgynevezett üres akkordot, három foké hármashangzatot, négy foké négyeshangzatot ad [1]. Fontos megjegyezni, hogy ez a felosztás *különböző hangokra* vonatkozik, vagyis például egy hármashangzat három különböző hangból áll – ebből következik, hogy egy alaphangot és annak az oktávját azonos hangnak tekintjük az akkordfelismerésnél. Tehát ha a kromatikus skála alaphangjait egy körnek képzeljük el, akkor magát a skálát egy olyan spirálként lehet felfogni, melynek alapja az előbb említett kromatikus kör, a függőleges tengely pedig a hangmagasság. A spirálgörbén az egymás felé eső hangokat azonos osztályba soroljuk be, nem teszünk különbséget köztük az akkord felismerésekor. Így adott 12 különböző osztályba eső hang, melyek meghatározott kombinációi egy akkordot adnak eredményül. Megjegyzendő, hogy szakdolgozatom során feltételeztem az egyenletesen temperált skála használatát, a problémát ebben a környezetben oldottam meg.

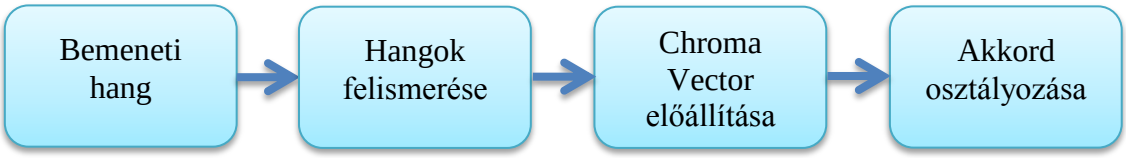
A szakdolgozatom során felmerülő problémákat hármashangzatokra, azok közül is a nagy és kis hármas (vagyis dúr és moll akkordok) esetekre oldottam meg, hiszen gitáron megszólaltatva a leggyakoribb akkordok ezek közül kerülnek ki. Azonban fontos megjegyezni, hogy az algoritmusok kisebb módosításával elérhető bonyolultabb akkordok detektálása is, de akár egyetlen hang detektálása is lehetséges. Ez esetben nyilván a legnagyobb energiájú hangot detektáljuk az adott környezetben, például egy akkordban.

Egy akkordfelismerő rendszer megvalósítása a következő séma alapján történt leggyakrabban:



1.1. ábra: Az egyszerű akkordfelismerő rendszer sematikus vázlata

Ezzel szemben szakdolgozatomban eltértem a hagyományos, 1.1. ábrán látható rendszertől, saját rendszeremben nem a bemeneti hang egészéből olvastam ki az akkordokat, hanem előbb különféle algoritmusokat alkalmaztam az alaphangok minél jobb elkülönítésére, hogy a lehető legpontosabb legyen a végső feladat, az akkordfelismerés megoldása. Ez alapján a megvalósított rendszer legegyszerűbb vázlata az alábbi ábrán látható:



1.2. ábra: A saját akkordfelismerő rendszer sematikus vázlata

Ugyan viszonylag egyszerű a rendszer vázának felépítése, ezen blokkok működése egyáltalán nem magától értetődő. Több módszert is teszteltem munkám során és igyekeztem kiválasztani a legjobban működő, legkisebb hibát eredményező eljárásokat. Dolgozatom felépítése így a következők szerint alakul:

A következő fejezetben egy elméleti összefoglalót, egy bevezetést mutatok, hogy a rendszeremhez felhasznált algoritmusok, módszerek megismerésre kerüljenek. Kezdetnek a hangmagasság meghatározására, akkordok besorolására már létező megoldásokat ismertetem. A harmadik fejezetben a saját fejlesztést, a rendelkezésre álló módszerek használatát, kiterjesztését ismertetem – hogyan oldható meg az alaphangok felismerése a HPS algoritmus többszöri alkalmazásával, hogyan tudunk pontosítani az eredményeken, végül pedig hogyan tudjuk a keresett akkordot meghatározni.

A dolgozat végén bemutatom az elkészített rendszer tesztelését nagy mennyiségű felvett mintával. Itt megmutatom, hogy bizonyos szélsőségeknek, tulajdonságoknak milyen hatása van az eredményre.

2 Elméleti áttekintés

2.1 A hangmagasság meghatározása

Ahhoz, hogy akkordokat tudjunk meghatározni, lépésről lépésre kell haladnunk, tehát először meg kell határozni a mintában szereplő hangok hangmagasságát, majd ezeket felhasználva tudjuk analizálni, milyen akkordokat adtunk a rendszer bemenetére. Ehhez azonban meg kell ismernünk a hangmagasság pontos definícióját.

Mikor nevezünk egy hangot magasnak és mikor mélynek? Ha például összevetünk egy hegedűt egy csellóval, az előbbi „magas", az utóbbi „mély" hangokat szólaltat meg. Azonban ha a hangmagasságot teljesen egyenlőnek tartanánk a frekvenciával, tévednénk. A hangmagasság az emberi hallás sajátossága, és mint ilyen, több komponenstől is függ, ezek pedig a pszichoakusztikára vezethetőek vissza. Például kísérletek kimutatták, hogy egy 6 kHz-es frekvenciájú hangnál, ha az intenzitását 60 dB-ről 90 dB-re növelnek, akár 30 centtel is nőhet az érzékelt hangmagasság [1]! Emellett befolyásoló tényező lehet még a jel harmonikustartalma, illetve a hang időtartama is. Gyakorlatban azonban az emberi hallást tökéletesen modellezni nem lehet, így elfogadott közelítés, ha a frekvenciára vezetjük vissza a hangmagasság érzékelését. Ezáltal hangmagasságnak dolgozatomban én is a spektrum alapharmonikusát definiálom, ezért az első feladat a spektrum ezen frekvenciájának meghatározása.

A probléma megoldására rendelkezésre állnak mind időtartományban, mind frekvenciatartományban módszerek. Természetesen mindkettőnek vannak előnyei, illetve hátrányai – például ugyan a frekvenciatartománybeli módszerek pontosabbak, de jóval nagyobb a számításigényük, így nehezebben implementálható egy valósidejű környezetben is működő program. A következő részben először az időtartománybeli módszereket ismertetem. Bár feladatom lényegi részét nem az időtartományban oldottam meg, fontosnak tartom ezen módszerek megismerését is, előnyeik, illetve hátrányaik felmérését egyaránt. Végül dolgozatomban a kvázi-frekvenciatartománybeli megoldás mellett döntöttem, mert jobban működőnek bizonyultak az akkordmeneteknél jellemző nagy harmonikustartalmú jelek hangmagasságának felismerésére.

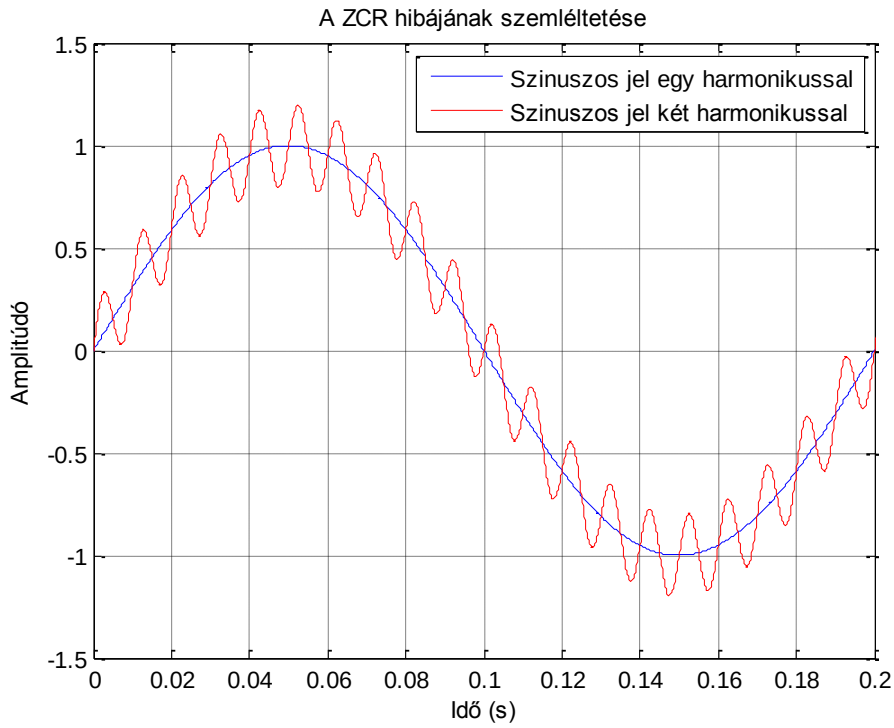
2.1.1 Időtartománybeli megoldások

A legalapvetőbb megoldás a hangmagasság meghatározására a légnyomás idő szerinti változását reprezentáló hullámforma vizsgálata – azaz az időtartománybeli analízis [3]. Dolgozatomban csak néhány lehetséges módszert mutatok be érintőlegesen, hiszen a végső megoldásomban nem ezen lehetőségek közül választottam.

Az alább vizsgált módszerek közös tulajdonsága, hogy bizonyos események bekövetkeztének gyakoriságának analizálásán alapszanak. Ezekből már tudunk következtetni a jel spektrális tulajdonságaira, így meghatározhatóvá válik a jel alapharmonikusa.

Nullátmenet-számlálás (ZCR):

Legegyszerűbb a fentebb említett módszerek közül a nullátmenet-számlálás [4]. Az alapfeltevés az, hogy a nullátmenetek számából tudunk következtetni a jel egység alatti ismétlődésének számára – s ezáltal az alapharmonikus frekvenciájára. Ez azonban nagyon egyszerűen megmutatható, hogy nem minden esetben vezet pontos eredményre. Az ábrán két szinuszos jelet láthatunk, különböző harmonikusszámmal:



2.1. ábra: A ZCR hibájának szemléltetése

Jól látható, hogy míg a tisztán szinuszos jel kétszer halad át a nulla vonalon periódusonként, egy több harmonikust tartalmazó jelnél már hibásan detektálnánk a frekvenciát. Természetesen javítható az eljárás hatékonysága – a cél a zavaró felharmonikusok eltávolítása. Azonban kihívást jelenthet a szűrő vágási frekvenciájának megválasztása, hiszen minél több felharmonikust ki szeretnénk szűrni, az alapharmonikus megőrzésével.

Csúcsérték számlálás (PR):

Az eljárás a csúcsértékek számlálásán alapszik. Az alapgondolat, hogy egy periodikus jel minden periódusa alatt felveszi a maximum és minimum értékét is – ezen események bekövetkeztének számát kell meghatároznunk, és ebből adódik a frekvencia. Emellett az algoritmust használhatjuk úgy is, hogy a maximumok közötti távolság megadja a hullámhosszt, amiből a frekvenciát ki tudjuk számolni.

Ezen módszerek fejlesztése:

Ha egy hullámforma periodikus, akkor a deriváltja (pillanatnyi meredeksége) is az. Az így kapott derivált jelben szintén vizsgálhatunk nullátmeneteket, illetve maximumokat – sőt, néhány esetben ezek az eredmények sokkal informatívabbak, mint az eredeti jellel végzett vizsgálatok.

Autokorrelációs algoritmus (ACF):

Másik módja a jel időtartományban történő vizsgálatának az összehasonlítás. A legegyszerűbb, mégis talán leggyakrabban alkalmazott módja ennek az autokorrelációs algoritmus [3]. Az eljárás a két hullámforma hasonlóságának vizsgálatán alapszik. Először a keresztkorreláció definíciójából indulunk ki:

$$R_{xy}(v) = \sum_{n=-\infty}^{\infty} x(n) \cdot y(n+v) \tag{2.1}$$

Ez alapján pedig egy függvény autokorrelációja a következőképpen definiálható egy végtelen $x(n)$ függvényre:

$$R_x(v) = \sum_{n=-\infty}^{\infty} x(n) \cdot x(n+v) \tag{2.2}$$

Ezt pedig kiterjesztve egy N véges hosszú diszkrét $x'(n)$ függvényre:

$$R_{x'}(v) = \sum_{n=0}^{N-1-v} x'(n) \cdot x'(n+v) \tag{2.3}$$

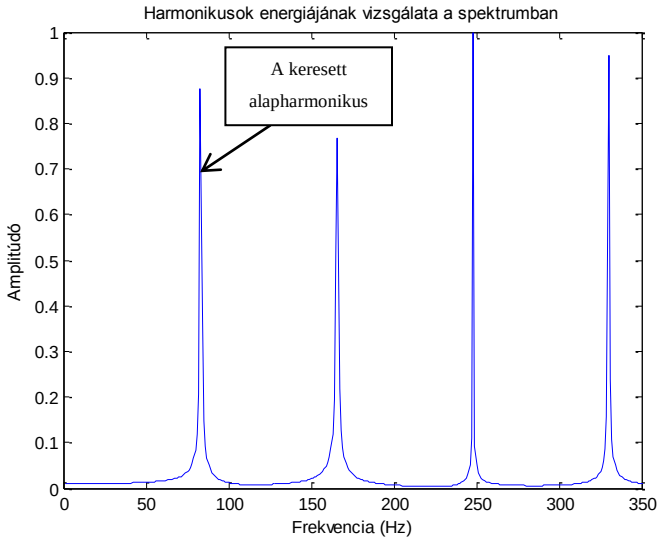
A periodikus jelek autokorrelációs függvénye szintén periodikus. A korreláció akkor minimális, ha a periódusidő felénél vizsgáljuk, és maximumát egy periódus hosszának a végénél éri el, mert ilyenkor a jel és az időben eltolt másolata fázisban van. Az autokorrelációs hullámforma első maximuma jelzi a jel periódusidejét, amiből már a frekvenciát egyszerűen tudjuk számolni. A módszer hátránya, hogy harmonikusokban gazdag jeleknél több maximum jelenik meg. Emellett az első *legnagyobb értékű* maximum ugyanúgy a periódusidejét fogja adni a jelnek, azonban így az algoritmus alkalmazhatósága romlik, hisz nemcsak a bekövetkező csúcsok számát kell vizsgálnunk, hanem azok értékét is, hogy a megfelelő eredményre jussunk.

2.1.2 Frekvencia tartománybeli megoldások

Frekvenciatartományban már jóval több információ áll rendelkezésre, amelyből a jel alapharmonikusának frekvenciára következtethetünk. Fontos különbség még az időtartománybeli analízishez képest, hogy míg ott zavaró a felharmonikusok jelenléte, itt az algoritmusok megfelelő működéséhez elengedhetetlen [3].

A következő részek megértéséhez egy fontos fogalmat kell definiálnom, mely dolgozatomban többször előfordul. A „bin” elnevezés a külföldi terminológiában használatos, és a jel diszkrét spektrumának egy-egy elemének indexét jelenti. Dolgozatomban is ebben a jelentésében használandó a továbbiakban.

Egyszerűen adódna, hogy keressük meg a spektrum legnagyobb energiájú komponensét. Azonban ez nagyon sok esetben nem vezetne helyes eredményre, hiszen a spektrumban nem szabályszerűen az alapharmonikusnak van a legnagyobb energiája, ez a megközelítés pedig ilyen esetekben arra vezetne, hogy nem az alapharmonikus, hanem egy felharmonikus frekvenciáját határoznánk meg. Ezt szemléltetem a következő ábrán:



2.2. ábra: Spektrum harmonikusainak vizsgálata

Az ábrán az az eset látható, amikor nem az alapharmonikus a legnagyobb energiájú komponense a spektrumnak. Ilyenkor természetesen a maximumkeresés hibás eredményt ad, éppen ezért különböző összetett módszerek alkalmazására van szükség. Dolgozatomban először bemutatom egy lehetséges megközelítését a probléma megoldásának, azonban mélyrehatóbban csak azzal az algoritmussal fogok foglalkozni, mellyel megoldottam az akkordban szereplő hangok frekvenciájának felismerését.

Kepsztrális analízis:

A kepsztrális analízis alapfeltevése, hogy periodikusnak tekinthetünk egy több harmonikust tartalmazó spektrumot, mivel a spektrum komponensei egyenlő, éppen akkor távolságra vannak egymástól, mint maga az alapharmonikus [4]. A kepsztrum módszere az ún. „source-filter-speech” modellhármason alapszik, hiszen a módszer alapjában véve az emberi beszéd, emberi hang felismerésére lett kifejlesztve. A *source*, azaz forrás eredetileg az emberi hangszálak rezgéséből adódó hullám. E hullámra a különböző hangképző szervek szűrőként hatnak, és ilyenkor a kimeneti jel (*speech*, azaz a beszédhang) kifejezhető a gerjesztés időfüggvényének és a szűrő impulzusválaszának konvolúciójával. Frekvenciatartományban ezt a következőképpen írhatjuk le:

$$Y(j\omega) = X(j\omega) \cdot H(j\omega) \tag{2.4}$$

Ha csak az amplitúdó spektrumokat vizsgáljuk, illetve azoknak logaritmusát vesszük:

$$\log|Y(\omega)| = \log|X(j\omega)| + \log|H(j\omega)| \tag{2.5}$$

Ha ezt inverz Fourier transzformáljuk, megkapjuk a szűrt jel kepsztrumát. Azaz:

$$\mathcal{F}^{-1}\{\log|Y(\omega)|\} = \mathcal{F}^{-1}\{\log|X(j\omega)|\} + \mathcal{F}^{-1}\{\log|H(j\omega)|\} \tag{2.6}$$

A kapott kepsztrumban az egyes csúcsok jelzik a periodikus frekvenciakomponenseket. Mivel a kepsztrum a logaritmusképzés hatásaként még inkább periodikus lesz, így az első kepsztrális csúcs helyéből meghatározhatjuk az alapfrekvenciát. A kepsztrum x tengelye idő jellegű mennyiség, de binekbe osztályozva láthatjuk az értékeket rajta. Ez alapján, ha egy csúcs az N. idő-binben jelenik meg és a mintavételi frekvencia f_s, a periódusidő a következőképpen határozható meg:

$$T_0 = \frac{N}{f_s} \tag{2.7}$$

Innen látható, hogy ha az eredeti jel alapharmonikusát kívánjuk meghatározni, a következő összefüggést kell használnunk:

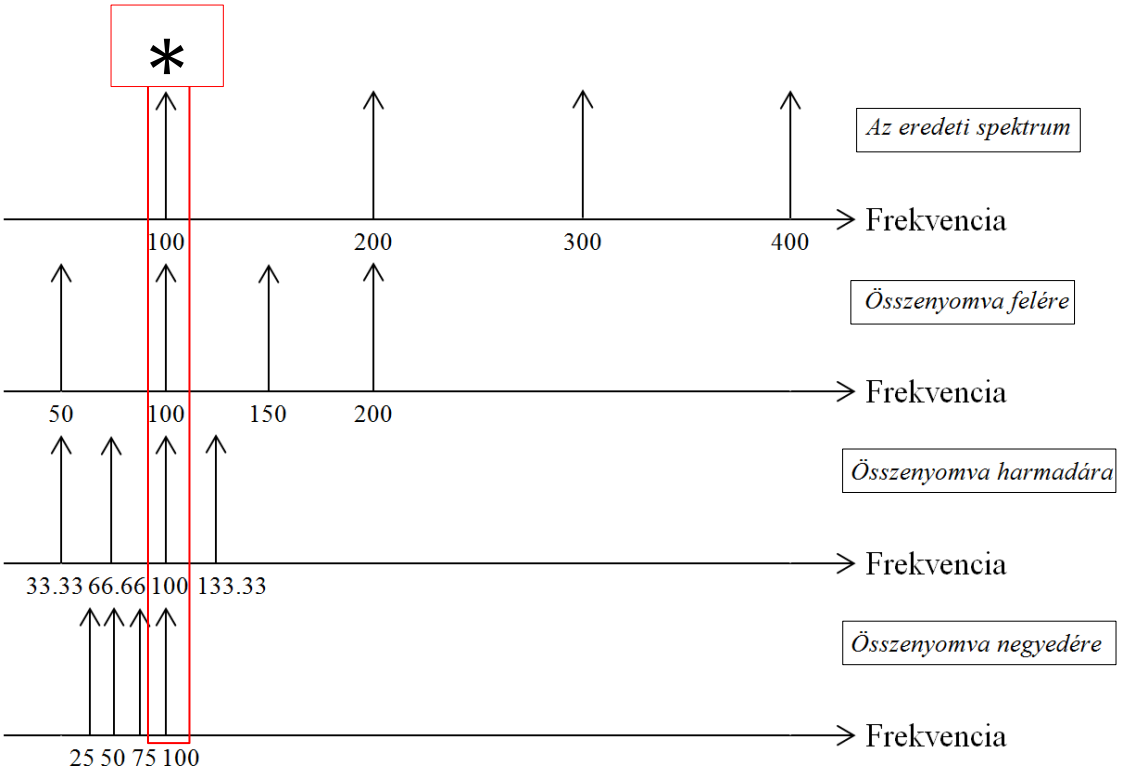
$$f_0 = \frac{f_s}{N} \tag{2.8}$$

Látható, hogy bár eredetileg nem hangszerek által készített minták kezelésére alakították ki, de a feladat megoldhatósága szempontjából az algoritmus ezen esetekre is megfelelő eredményt szolgáltathat.

Emellett viszont figyelembe kell vennünk az algoritmus hibáit is; többek között a zajérzékenységét, és a nagy számításigényét. Esetemben az első komoly problémát okoz, hiszen dolgozatomban egy rugalmas rendszer létrehozása volt a célom, amelynél olykor sajnos elkerülhetetlenül zajosak a hangminták.

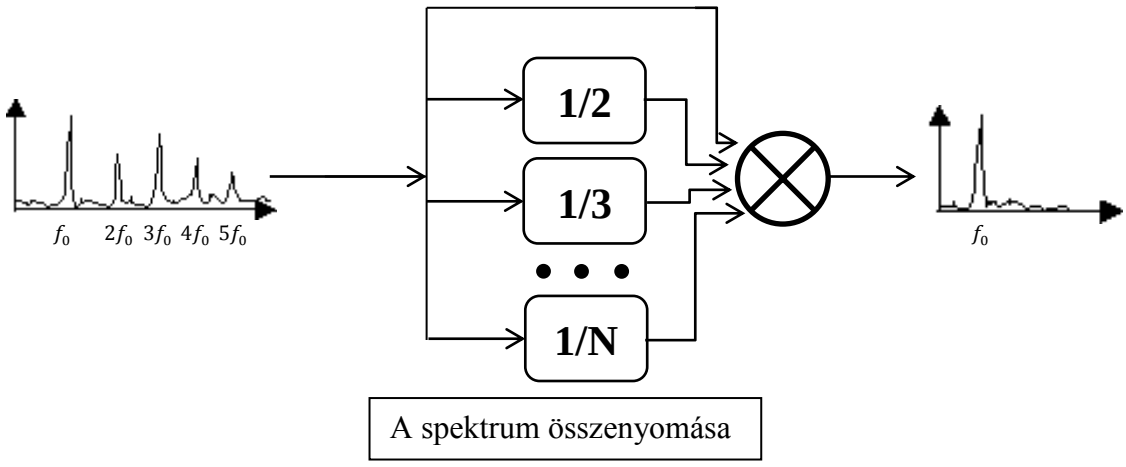
Harmonic Product Spectrum (HPS) algoritmus:

Az algoritmus alapötlete, hogy a felharmonikusokat az alapharmonikus kiindulási pozíciójába juttassuk a spektrum „összenyomásával”, majd az így kapott komponenseket összeszorozzuk. Célunk, hogy az alapharmonikus pozíciójában erősítsék egymást, a többi pozícióban pedig ne legyen számottevő az amplitúdójuk. A spektrum „összenyomását” legegyszerűbben újramintavételezésekkel érhetjük el, melyek számát helyesen megválasztva a következőképpen alakul a spektrum:



2.3. ábra: A HPS algoritmus működése

Az algoritmus megvalósításának módját a következő működési vázlaton szemléltetem:



2.4. ábra: A HPS algoritmus megvalósítása

Ezek alapján, ha helyesen választjuk meg az újramintavételezések számát, a kimeneti spektrum a 2.4. ábrán láthatóhoz hasonló lesz, és az alapharmonikus keresésének

problémáját vissza tudjuk vezetni egyszerű maximumkeresésre. A módszer mellett szól még kis számításigénye, illetve, hogy mind additív, mind multiplikatív zajra érzéketlen.

Vizsgáljuk meg az algoritmus lehetséges hibáit! Gyakori jelenség az oktávhiba. Ez abból ered, hogy ha a spektrum annyi felharmonikust tartalmaz, hogy még az algoritmus lefuttatása után is marad több jelentős komponens, előfordulhat, hogy nem az alapharmonikus lesz maximális értékű, hanem egyik többszöröse, leggyakrabban a második felharmonikus [3]. Ez zenében pontosan egy oktávnyi hibát jelent. Ugyan ez az akkordfelismerésnél nem jelent feltétlen hátrányt, de az azt követő szűrésnél, amelyet a következő fejezetben mutatok be, a legrosszabb esettel számoltam, és az alapharmonikus lehetséges többszöröseit eltávolítottam a spektrumból. Nagy probléma azonban az, hogy ugyan az algoritmus hibája kHz-es nagyságrendű frekvenciáknál elhanyagolható, de alacsonyabb frekvenciákon igen jelentős. Ez alapján belátható, hogy az algoritmus alkalmazása önmagában még nem jelent pontos megoldást a kitűzött célra, pontosítani kell a kapott eredményeket. A kidolgozott megoldásomat később, az rendszer részletes elemzésénél fogom ismertetni.

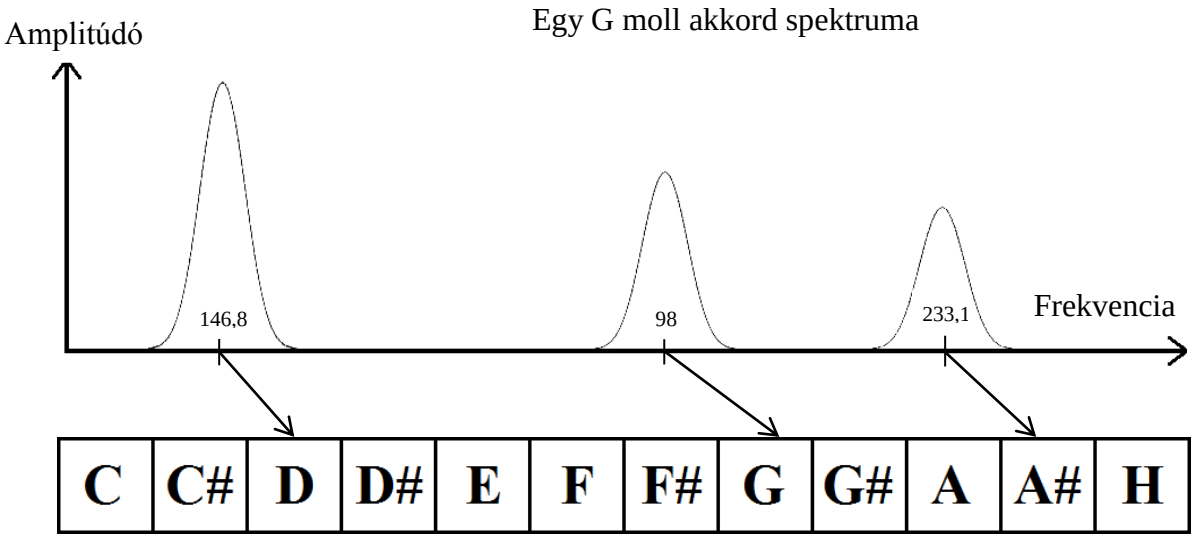
2.2 Az alaphangok tárolása – A Chroma Vector

A következő feladat az alaphangok olyan tárolása volt, hogy abból később meg tudjuk határozni, milyen akkord szól a mintánkban. Erre a legmegfelelőbb megoldásnak az úgynevezett *Chroma Vector*, magyarul *kromavektor* bizonyult, melynek lényege, hogy egy 12×1 méretű vektorban eltároljuk az egyenletesen temperált skála félhangjaihoz rendelt intenzitást [5]. Megjegyzendő, hogy a skála hangjainak megfelelő frekvenciáit, egyúttal a tárolók középfrekvenciáit a következőképpen lehet számolni:

$$f = f_0 \cdot \sqrt[12]{2^n} \tag{2.9}$$

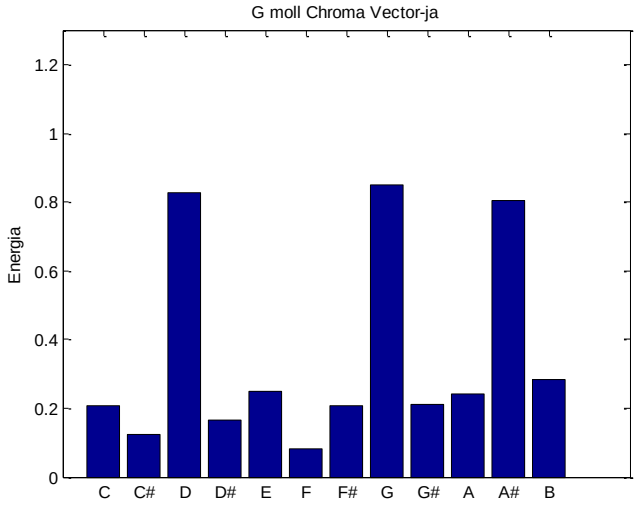
Itt f a keresett hang frekvenciája, f_0 az adott oktáv C alaphangjának frekvenciája, n pedig az a szám, amivel megadjuk, hány félhanggal legyen magasabban f, mint f_0.

A célt, a frekvenciák összekapcsolását a kromavektorral a következő ábra szemlélteti:



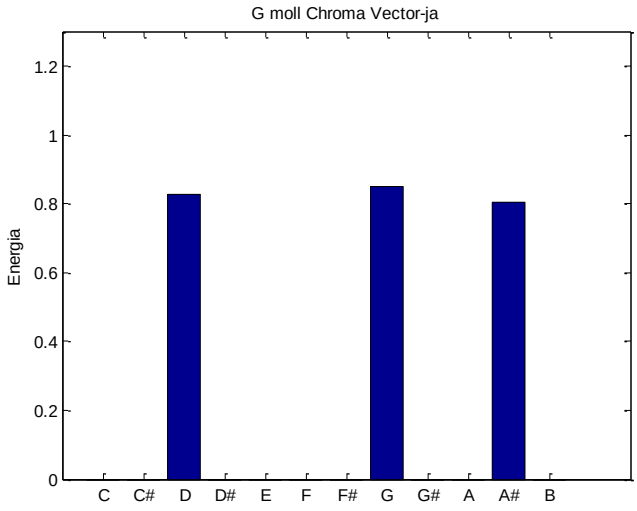
2.5. ábra: Egy G moll akkord osztályozásának szemléltetése

Miután feltöltöttük, a kromavektorunk a következőképpen alakul:



2.6. ábra: G moll kromavektorja általános esetben

Látható, hogy a várt G, D és A# hangok mellett a többi hangnak is van számottevő energiája. Az ábrázolt esetben nem jelentenek problémát az akkord felismerésénél, azonban ha túl magas az értékük, zavart okozhatnak. Ideális esetben csak a három hangot kell tartalmaznia kromavektornak, így nem lehet téves a felismerés. Ezért a célom a dolgozatban, hogy egy ilyen, az ideálishoz közelítő kromavektort töltsek fel, ezt az előbb bemutatott G moll akkordra szemléltetve:



2.7. ábra: Kromavektor ideális esetben (G moll akkord)

Természetesen egyetlen akkordra meg lehet alkotni utólag is egy ilyen ideális kromavektort a kisebb energiájú komponensek szűrésével, azonban figyelembe kell venni, hogy ha a vektort időben folyamatosan újra és újra feltöltjük, akkor több nagy energiájú zavaró komponens jelenhet meg – főleg az akkordváltásoknál, illetve csend esetén. Felfedezhetjük azt is ezen ábrázolási módban, hogy oktávokat nem különböztetünk meg, csak a 12 félhang vizsgálatát végezzük vele. Ebből következik, hogy a hangok és oktávjai összegét kell gyűjtenünk a 12 elemű vektorba. A módszernek léteznek továbbfejlesztései is, például ilyen egy 36 elemű vektor, amivel azt a hibát is tudjuk kezelni, hogy a hangszerek, melyekkel a mintát rögzítettük, esetleg eltérhetnek a standard hangolástól, és ez hibákat okozhat az akkordfelismerés során. Dolgozatomban feltételezem, hogy a minták felvételénél a gitár standard A hangolásban van, attól legfeljebb néhány centtel tér el – ekkor még az algoritmus megfelelően működik.

Bár a módszer működőképes lehetne a bemeneti jel DFT (Diszkrét Fourier Transzformáció) eredményét felhasználva is, a hiba jelentős lenne. Például az alaphang harmadik felharmonikusa növelné az energiáját a hang kvintjének – még ha az nincs is benne a bemeneti jelben [6]! Ezért dolgozatomban eltérek a „klasszikus” kromavektortól ezen jellemzőjében, és egy újabb módszert alkalmazva a HPS algoritmus alkalmazása után a detektált alaphangokkal töltöm fel a kromavektort időben folyamatosan. Az utolsó lépés az akkord meghatározásánál a kromavektor kiértékelése, illetve a felismert akkord ún. osztályozása, azaz a felismert hangok alapján a keresett akkord, illetve akkordok meghatározása.

A következőkben ezen probléma több megközelítésével foglalkozok, illetve bemutatom az egyszerű módszert is, mellyel végül nagy pontossággal meg tudtam oldani a feladatot.

2.3 A detektált hangok kiértékelése, akkord osztályozása

Akkord osztályozása alatt dolgozatomban azt a végső lépést értem, amivel meghatározzuk, kiíratjuk a bemeneti jelben felismert akkordot, akkordmenetet. Triviális megoldásnak tűnne az egyszerű összevetés már előre definiált akkordokkal, azonban fontos megemlíteni, hogy ha a vektorunk elemei közt több zavaró komponens van, és azok energiája nincs megfelelően elnyomva, vagy elkülönítve, több a jelben nem szereplő akkordot is felismernénk, hibásan. Ezért több módszer is kialakult arra, hogy miként lehetne a már feltöltött kromavektorból visszaadni azokat az akkordokat, amelyek ténylegesen a mintában szerepelnek. Az egyik nagy csoport a statisztikai módszerek alkalmazása.

2.3.1 Statisztikai módszerek

Ezen módszerek nagy része a Rejtett Markov Modelleken (HMM) alapul [5]. Eleinte ezeket a modelleket beszédfelismerésre használták, és ezen alkalmazásukra hatalmas adatbázis áll rendelkezésre, amelyekből a modell paraméterei megválasztásra kerülhetnek. Azonban lévén ez egy újfajta megközelítése az akkordfelismerésnek, ebben a témakörben nem áll rendelkezésre ilyen adatbázis. Másrészt pedig egy zenedarabon belül sokkal nagyobb különbség lehet, mint például egy beszédnél –többek között frekvenciában, hangszínben, dinamikában is [7]. Ebből következik, hogy sokkal több adatot kell gyűjtenünk, hogy a modelleket betanítsuk, és általánosítsuk a módszert. Az egyik használt módszer a modellek tanítására az Expectation Maximization (EM) algoritmus. Általában a problémát két részre bontják – az egyik a szegmentálás, tehát az akkordok határainak megtalálása, másrészt az akkordnak magának a meghatározása. Ez a módszer az előbbi feladatot viszonylag pontosan meg tudja oldani, azonban az akkord felismerésénél igen magas a hibák aránya. Ennek fejlesztésére voltak és vannak

is kísérletek, hiszen az akkordok határainak megtalálása egy-egy akkordmenetben szintén nem egyszerű feladat, és a módszer előnyét ebben ki lehetne használni. Azonban dolgozatomban ezzel a megközelítéssel többet nem foglalkozom, a következő

fejezetben bemutatott sablon-alapú megoldással közelítem meg a problémát, és igyekszek minél pontosabb frekvenciákat eltárolni már a hangmagasság felismerésnél, hogy a sablonokkal csak a probléma végső megoldására kelljen fókuszálni, az algoritmus túlságos bonyolítása nélkül.

2.3.2 Sablon-alapú módszerek

Ezen csoportba tartozó módszerek sokkal több változatban alkalmazottak, mint az előbbiek. Az alapötlet az, hogy definiálunk bizonyos sablonokat, amelyek az akkordoknak felelnek meg, majd ezeket összehasonlítjuk a saját kromavektorunkkal, és megvizsgáljuk a korrelációjukat [8]. Ezt a következőképpen lehet elképzelni:

```
chroma_sajat=[1 0 0 0 1 0 0 1 0 0 0 0]
chroma_Gmoll=[0 0 1 0 0 0 0 1 0 0 1 0]
chroma_Cdur =[1 0 0 0 1 0 0 1 0 0 0 0]
chroma_Cmoll=[1 0 0 1 0 0 0 1 0 0 0 0]
```

Fentebb négy kromavektort láthatunk; ezeket a jelöléseket úgy kell értelmezni, hogy a beolvasott akkord tartalmaz C, E, illetve G hangokat (Tehát ott „1” az érték, amelyik hangot felismertük a mintában, megjegyezve, hogy a vektor első eleme a C hang, utolsó pedig a H.), tehát egy *C dúrt* keresünk a chroma_sajat vektor értékei alapján. A következő három vektor egy-egy előre definiált sablont tartalmaz, és ezeket vetjük össze a vizsgált vektorunkkal. Látható, hogy a G mollnál is részleges egyezést tapasztalhatunk, tehát a G hang megtalálható mindkét vektorban, azonban az igazi probléma a C mollnál merül fel, ahol a 3 megtalált alaphangból kettővel egyezést tapasztalunk. Ilyenkor felmerül a kérdés, hogy a bemenetre biztos, hogy egy C dúrt tartalmazó mintát adtunk, vagy vektorunk feltöltése előtti felismerésnél történt egy félhangnyi hiba. Ezért nagyon fontos a félhangnyi tévedéseket már a hangfelismerésnél kiküszöbölni, hiszen akkor az utolsó lépésben rossz akkordot ismer fel a rendszer. Dolgozatomban a minta felismerésének pontosságára fektettem hangsúlyt, nem az itt leírtak szerinti osztályozási módszerek fejlesztésére, emellett viszont fontosnak tartom megemlíteni azokat a módszereket, melyek nem a felismerésnél, hanem az osztályozásnál végzik a hibakezelést. A most következő két algoritmus definíciószerűen használja a *bit maszk* kifejezést, ami alatt az akkordok már általam is bemutatott sablonjait kell érteni.

A legközelebbi szomszéd módszer:

A módszer a következő képlet számításán alapul:

$$\delta_i = \sum_{n=0}^{N-1} \big(T_i(n) - C(n)\big)^2 \tag{2.10}$$

Itt N=12, vagyis egy oktávon belüli félhangok száma, a kiválasztott akkord pedig az lesz, amelyiknek a távolsága (δ_i) minimális az i. maszk (T_i) és a C által reprezentált kromavektor között.

A súlyozott összeg módszer:

A másik gyakran alkalmazott módszer a súlyozott összeg módszer, amely a következő képlet használatára épül:

$$\Delta_i = \sum_{n=0}^{N-1} \big(W_i(n) \cdot C(n)\big)^2 \tag{2.11}$$

A fentiekből látszik, hogy azt az akkordot fogjuk kiválasztani, amelyik maximalizálja Δ_i-t, vagyis a skaláris szorzatot a kromavektor, illetve az i. maszk egy súlyozott értéke között. A bit maszk súlyozása elősegíti azon akkordok összehasonlítását, amiben eltérő számú hang van, így például egy dúrt össze tudunk hasonlítani akár egy négyes hangzattal, vagy egy üres akkorddal is – hiszen azok az akkordok, melyekben több hang van, természetesen nagyobb összegeket fognak adni, mint a kevesebb hangot tartalmazók. Emellett a súlyozással szintén megoldható az is, hogy a gyakoribb akkordokat nagyobb súllyal vegyük figyelembe a kevésbé gyakoriaknál. Dolgozatomban ilyen típusú módszert alkalmaztam az akkordok osztályozására.

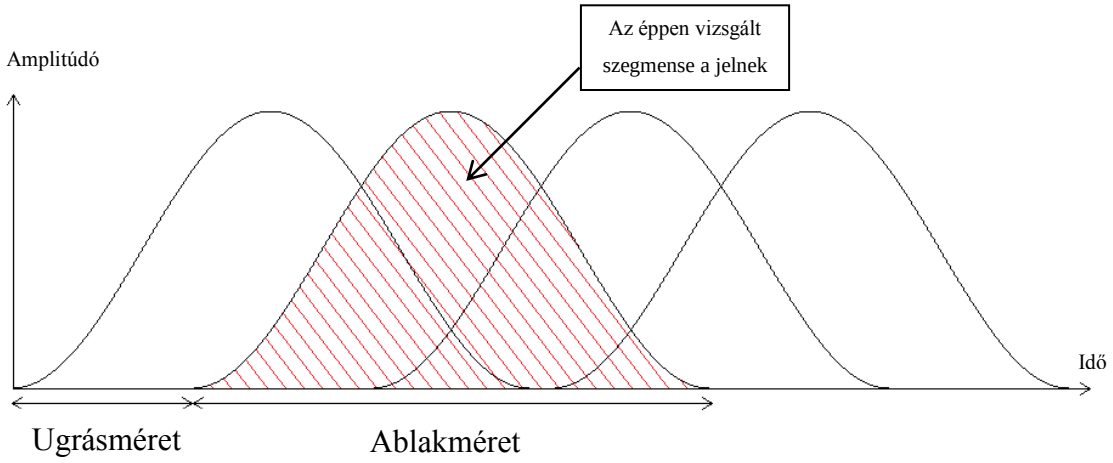
Általánosságban elmondható, hogy a legtöbb akkordosztályozó algoritmus a kromavektor alapján működik. Ez alól persze van egy-egy kivétel, melyre dolgozatomban nem térek ki, az ilyen megoldások már a hangfelismerés résznél más módszert alkalmaznak, például ún. *constant-Q transform*-ot, amely szorosan kötődik a Fourier-transzformációhoz, de a megoldás egésze különbözik a dolgozatomban bemutatottól, így az ilyen típusú módszerekre nem térek ki.

3 A saját akkordfelismerő rendszer fejlesztése

Rendszeremet MATLAB környezetben hoztam létre, a könnyű implementálhatóság, tesztelhetőség, illetve az esetleges fejlesztések megkönnyítésére. Először az egyszerű blokkvázlat részelemeinek ismertetésével kezdem a rendszerem bemutatását.

3.1 Alaphangok felismerése

Első lépésben szegmensekre kell bontanunk csúszó ablakozással a bemeneti jelet, ezután alkalmazhatjuk a HPS-t. A szegmentálást a következő ábrán szemléltetem:

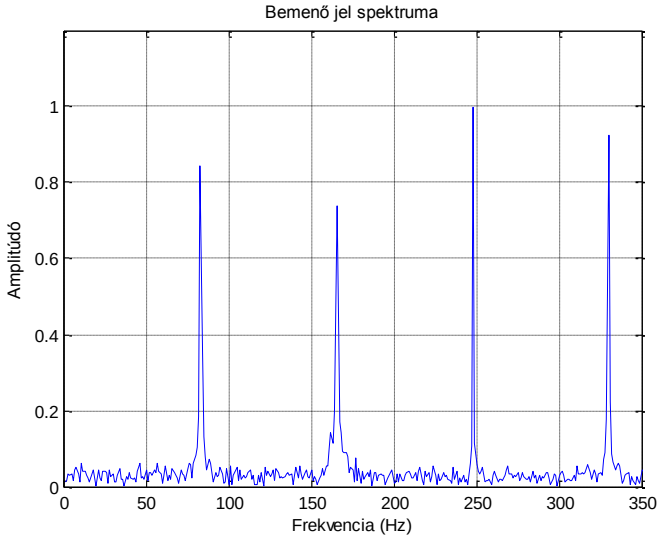


3.1. ábra: A csúszó ablakos szegmentálás

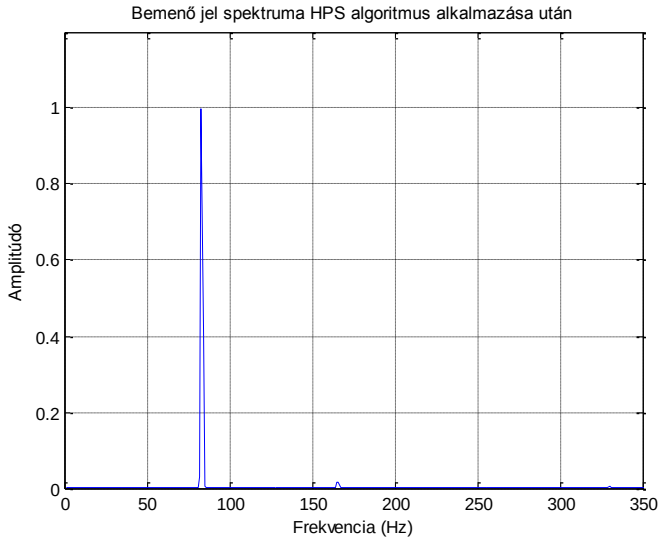
A program egy fontos pontja az ablak- és az ugrásméret megválasztása, amivel a felbontást lehet javítani. Első megközelítésben 4096-os ablakmérettel és 1024-es ugrásmérettel dolgoztam, de az ablakméretet később megnöveltem a pontosabb eredmény érdekében. A fenti értékekkel meghatározható a DFT felbontása; 44100 Hz-es mintavételi frekvenciával a következő adódik: $\frac{44100}{4096} \approx 10$ Hz. Ez alacsonyabb frekvenciákon nem elegendő. Példaként vizsgáljuk meg a C2 és a C#2 hangot; az előbbi frekvenciája 65,41 Hz, az utóbbié 69,3 Hz. Ez a két hang egyszerű DFT-t használva ugyanabba a binbe kerül, nem teszünk különbséget közöttük. Belátható, hogy a kisebb frekvenciák pontos kezelése miatt a felbontás növelése elengedhetetlen.

A szegmentálás elvégzése után következik a HPS algoritmus két fő lépése: az újramintavételezés és a szorzás, melyeket bemutattam az előző fejezetben. Az újramintavételezés alatt jelen esetben azt kell érteni, hogy lépésenként összenyomjuk a spektrumot felére, harmadára, negyedére... stb. Fontos megemlíteni, hogy a bemeneti jelet célszerű újramintavételezni, mert bár meg lehet oldani ugyan a spektrum újramintavételezésével is a problémát, de akkor az itt leírtak fordított megközelítésével kell eljárni. Végül, miután elvégeztük a szükséges újramintavételezéseket, az így kapott összenyomott spektrumokat összeszorozzuk. Szemléltetésképpen, amikor felére akarjuk összenyomni a spektrumot, az a célunk, hogy az alapharmonikus legyen fele akkora frekvenciájú, a periódusa pedig legyen kétszer akkora. Az algoritmus lépései egyszerűen implementálhatóak MATLAB környezetben, rendelkezésre áll egy függvény, ami az első lépést, az újramintavételezést, vagyis esetünkben túlmintavételezést elvégzi.

Kérdésként felmerül, hogy meddig kell folytatni a spektrum összenyomását. Ennek a paraméternek a megválasztása fontos, hiszen például egy nagyon kis értékű felharmonikus, amit lekeverünk az alapharmonikus helyére, az összeszorzás után nagyon lecsökkentheti az alapharmonikus értékét, ezzel pedig hibás eredményt kapunk. Miután összeszoroztuk az így kapott spektrumokat, már elvégezhető az egyszerű feladat, a maximumkeresés. Az algoritmus szemléltetéseként nézzük meg először egy 3 felharmonikust tartalmazó szinusz spektrumát egy egyszerű FFT, majd a HPS algoritmus alkalmazása után (Az alapharmonikust 82,3 Hz-re állítottam be):



3.2. ábra: 3 felharmonikust tartalmazó szinuszos jel spektrumának részlete



3.3. ábra: A HPS algoritmus eredménye

Jól látható, hogy a felharmonikusok az alapharmonikus kezdeti pozíciójába kerültek, mintegy erősítve azt. Az algoritmust a spektrum négyszeres összenyomásáig végeztem el, ez az eljárás bizonyult a legmegfelelőbbnek a probléma minél pontosabb megoldására, ebből már meg tudtam határozni a legnagyobb energiájú hangot. Azonban, mivel az algoritmus végeredménye a programban még nem a frekvenciáját adja meg a keresett hangnak, hanem azt a bint, amibe tartozik, ezért ezt még át kell számolnunk frekvenciába. Emellett megjegyzendő, hogy a maximális frekvenciát tartalmazó bin MATLAB környezetben $(\text{index} - 1)$. számú, mivel a DC komponens a 0. binben helyezkedik el. Ezen megfontolások alapján a maximális intenzitáshoz tartozó frekvencia a következő képlet alapján számolható:

$$f_{\max} = (\text{frekvenciabin száma} - 1) \cdot \frac{\text{Mintavételezési frekvencia}}{\text{Ablakozott jel hossza}} \qquad (3.1)$$

Látható, hogy a viszonylag alacsony (≈ 10 Hz) felbontás miatt az algoritmus ugyan az alapharmonikust felismeri, de lehetséges, hogy több Hz-es tévedéssel. Ez alacsonyabb frekvencián nagy problémát jelent, így rendszerem következő szükséges lépése a pontosítás.

3.2 A hangmagasság felismerés pontosítása

Dolgozatomban két módszert mutatok be a hangfelismerés pontosítására, melyek alkalmazása után már kifejezetten pontos eredményt tudunk elérni az akkord

osztályozásánál. A pontosítás szükségességét az indokolja, hogy alacsony frekvencián, ha pont a bin határára esik a hang, akár 10 Hz is lehet a felismert és a tényleges hang között a különbség. A most következő két módszer a dolgozatom lényeges saját eredménye, amellyel az esetleges felismerési hibákat kiküszöbölöm, illetve megteszem az első lépést az algoritmus azon irányba való fejlesztéséhez, hogy az enyhén elhangolt mintákra invariáns legyen.

Hangmagasság felismerés pontosítás keresztkorreláció alkalmazásával:

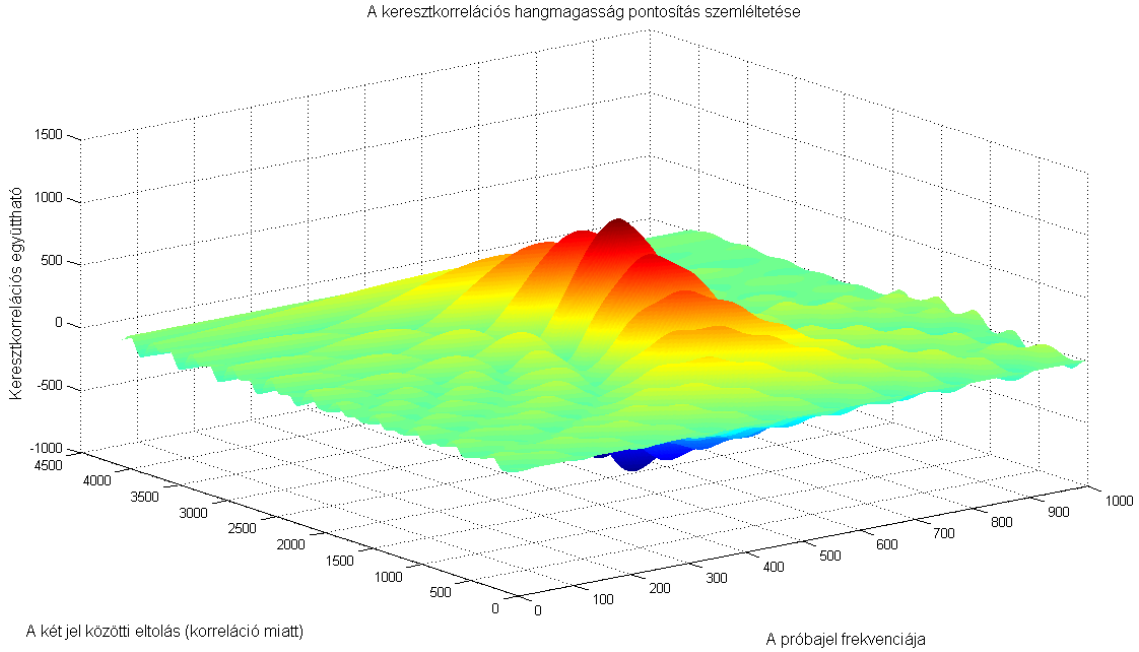
A módszer alapötlete, hogy keressük meg az eredeti jelünkkel legnagyobb hasonlóságot mutató harmonikus jelet, majd ezt felhasználva pontosítsuk az eredményt. Ezt úgy tudjuk megoldani, hogy a frekvencia lépésenkénti változtatásával keresztkorrelációt számolunk az eredeti jel és a harmonikus jelek között. Ezt a következő képlettel tudjuk megtenni:

$$R_{xy}(v) = \sum_{n=-\infty}^{\infty} x(n) \cdot y(n+v) \tag{3.2}$$

Az eredmény maximumhelye a legjobban illeszkedő harmonikus jelet jelzi.

Programomban ezt a következőképpen implementáltam:

Miután meghatároztuk a maximális frekvenciát az előző fejezetben bemutatott módon, eltároljuk annak intenzitásértékét is, majd definiálunk egy felbontást. Először az alapharmonikus frekvenciája (f_0) közelében 100 lépésben, $f_0 \pm 5$ Hz-es környezetben pásztázom a spektrumot $0{,}1$ Hz-es frekvencialépéssel. Minden iterációban létrehozok egy próbavektort, ami az éppen aktuális $f_0 + df$ frekvenciájú szinuszt tartalmazza, majd ezt összehasonlítom keresztkorreláció alkalmazásával a bemeneti jellel. Itt abból az elméleti tényből indultam ki, hogy két szinuszos jel skaláris szorzata 0, ha a frekvenciájuk nem egyenlő (Természetesen esetünkben véges hosszú jelekkel dolgozunk, így sosem lesz pontosan 0 az érték.). Másrészt azonos frekvenciájú szinuszos jelek skaláris szorzata akkor maximális, ha a fázisuk is megegyezik, vagyis a jel négyzetéről beszélünk. A leírtak szerint keressük ennek a kétdimenziós függvénynek a maximumát. Az eljárás eredményét szemlélteti ez az ábra esetemben ($f_s = 44100$ mintavételi frekvencia, $L = 2048$ felbontás, illetve $f_0 = 118{,}43$ Hz felismert frekvencia esetén):



3.4. ábra: A hangfelismerést pontosító algoritmus eredménye

Az egyik tengely mentén a két jel közötti eltolás van, ez az eredmény szempontjából nem lényeges, ebből a szinuszos jel fázisát tudnánk meghatározni. A másik tengely mentén azonban az az érték van, amellyel az iterációban beállítottuk az éppen aktuális frekvenciát. Először ezt az értéket határozzuk meg (amelynél maximális volt a korreláció értéke), ebből pedig a következő képlettel ki tudjuk számolni a keresett frekvenciát:

$$f_{\text{pontosított}} = f_0 - \left(i - \frac{N}{2}\right) \cdot \text{felbontás}, \tag{3.3}$$

ahol f_0 a felismert alaphang, i a futóváltozó azon értéke, ahol maximális volt a korreláció, N a választott lépések száma, a felbontás pedig a választott felbontás értéke. Egy példát megvizsgálva a teszt eredménye a következőképpen alakult:

$f = 113{,}5$ Hz – az eredeti szinuszjel alapfrekvenciája – ideális esetben ezt kellene kapnunk

$f_0 = 118{,}43$ Hz – az algoritmusunk által meghatározott alapharmonikus frekvenciája

$f_{\text{pontosított}} = 113{,}63$ Hz – a pontosított felismert frekvencia.

Hangmagasság pontosítás szinusz-illesztés alkalmazásával:

Ez a módszer lényegesen gyorsabb, elegánsabb megoldás, mely egy ritkábban alkalmazott technikán alapul. A célunk az, hogy a jelet közel DC, azaz 0 Hz környékére

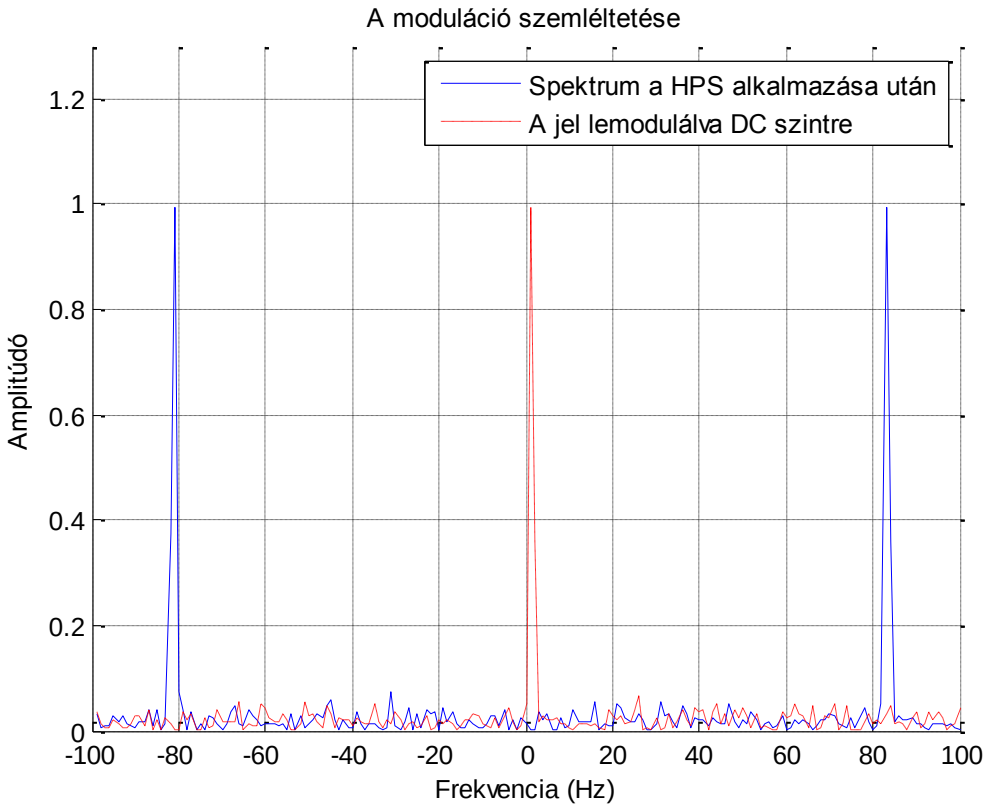
lekeverjük, majd ott elérjük, hogy csak az a nagyon kis frekvenciájú komponens maradhasson benne, amit keresünk. A következő lépésben numerikus módszerrel szinuszt illesztünk a jelre, és a kapott eredményből meghatározzuk az illesztett szinusz frekvenciáját, majd ezzel pontosítjuk az eredményt.

A lekeverést úgy tudjuk elvégezni, hogy felhasználjuk a modulációs tételt, azaz:

$$x(t)e^{-j\omega_{max}t} = X\,(j(\omega - \omega_{max})) \tag{3.4}$$

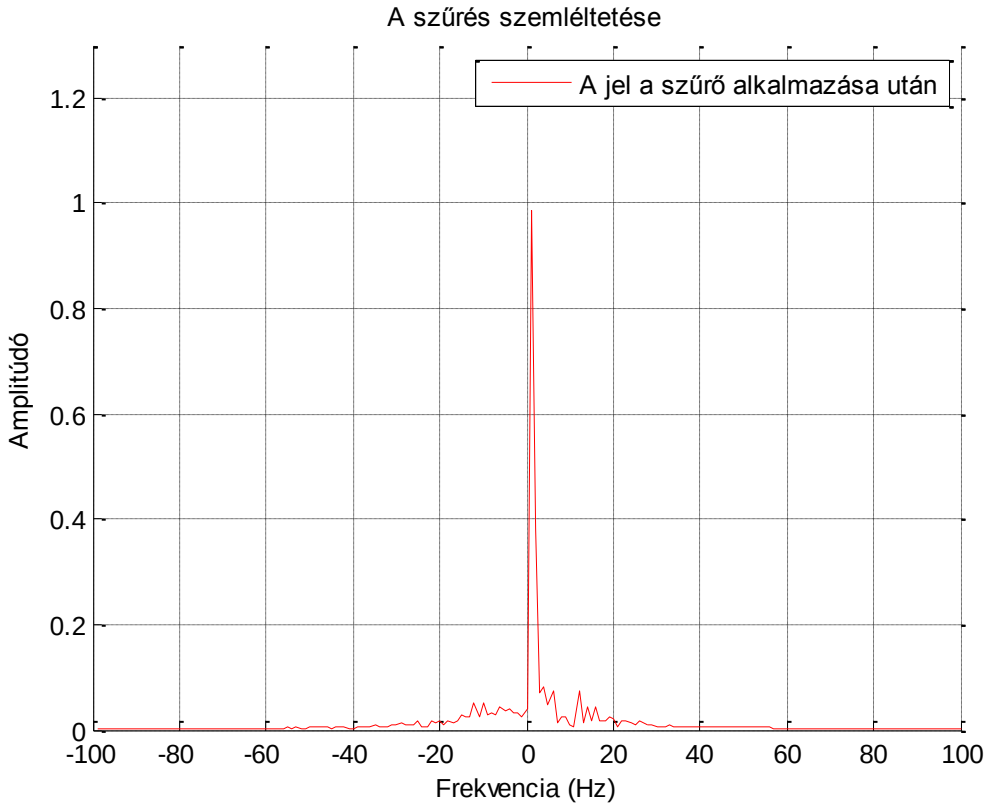
Ebből a képletből látható, hogy ahhoz, hogy a spektrumban ω_0-lal eltoljuk a jelet, időtartományban be kell szoroznunk az $e^{-j\omega_{max}t}$ tényezővel, ahol $\omega_{max} = 2 \cdot \pi \cdot f_{max}$, illetve f_{max} a HPS algoritmus által meghatározott pontosítandó alaphangunk.

Induljunk ki a következő ábrából, amely egy 82,3 Hz-es frekvenciájú, 3 felharmonikust tartalmazó szinuszos jel spektrumát ábrázolja a HPS algoritmus alkalmazása, illetve a moduláció után:



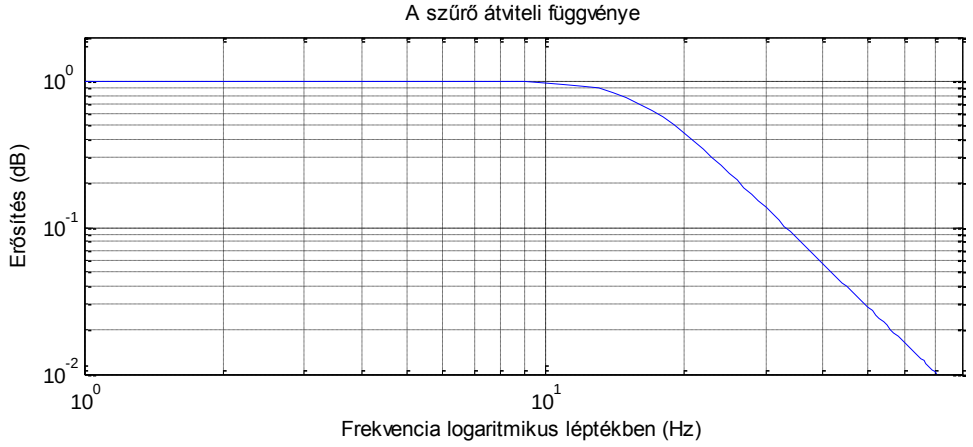
3.5. ábra: A pontosítás lépéseinek szemléltetése: moduláció

A következő ábrán a szűrő hatását láthatjuk:



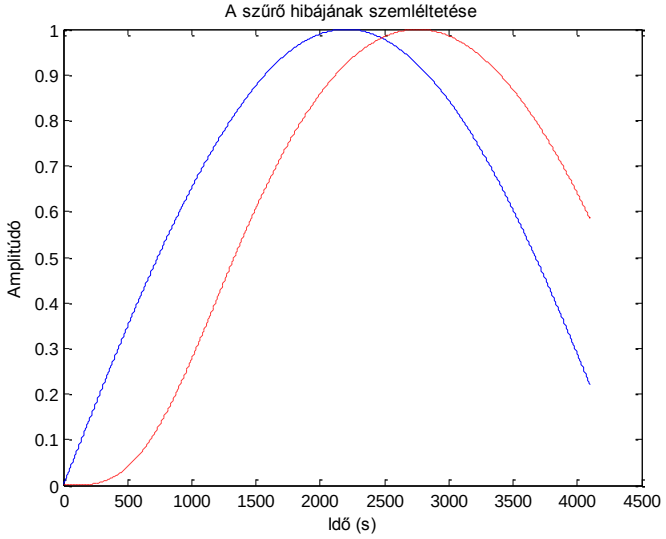
3.6. ábra: A pontosítás lépéseinek szemléltetése: szűrés

Látható, a moduláció hatására a jel 0 Hz környékére került. A következő lépés a nagy pontosságú aluláteresztő szűrés. Célunk, hogy a magas frekvenciás komponenseket kiszűrjük, és csak a nagyon kis frekvenciás szinusz maradjon. Ezt egy Butterworth-szűrővel tudjuk megoldani a legpontosabban, átvitelét mutatja a következő ábra:



3.7. ábra: Az alkalmazott szűrő átvitele

Azonban vigyázni kell, hogy nehogy túl magas fokú szűrőt tervezzünk, mert ugyan minél magasabb a fokszáma, annál meredekebb a vágás, de a fokszámmal arányos az impulzusválasz hossza is. Ez azért jelent problémát, mert a szűrő később éri el az állandósult állapotát, mint a minták bevitelének kezdete. Ebből következik, hogy az első minták erősen tranziens viselkedést mutatnak, ami elrontja a szinusz illesztés pontosságát. Hasonló okokból nem szabad a vágási frekvenciát sem túl alacsonyra választanunk. A következő ábrán ezt a jelenséget figyelhetjük meg, egy harmadfokú 50 Hz vágási frekvenciájú szűrő alkalmazásával szűrök egy 5 Hz-es szinuszt. Az elején megfigyelhető a tranziens jelenség, itt még nem a szűrt minták, hanem maga a szűrő tranziense adja a kimeneti jelet:



3.8. ábra: A szűrő tranziens viselkedésének jelensége

Miután elvégeztük a modulálást, illetve a szűrést a megfelelően megválasztott szűrőnkkel, következik a szinusz illesztés, amelyet egy lineáris egyenletrendszer megoldásával végzünk el. Általánosan nézve egy ilyen egyenletrendszer alakja:

$$\boldsymbol{A} \cdot \underline{x} = \underline{b} \tag{3.5}$$

Belátható, hogy ez az egyenletrendszer akkor és csak akkor oldható meg, ha $\boldsymbol{A}$ egy $n \times n$ -es mátrix, vagyis annyi ismeretlenünk van a b vektorban, ahány egyenletünk, illetve a mátrix rangja megegyezik a sorok számával. Alapfeltevésünk, hogy:

$$x_{n+1} = A \cdot e^{j \cdot 2\pi \cdot f \cdot dt} \cdot x_n \tag{3.6}$$

és ez minden szomszédos mintapárra teljesül. Könnyen belátható, hogy mivel a modulációt egy komplex fazorral végeztük, így a fenti feltevésünk teljesül (Más esetben egy koszinuszos tartalmú jelet kapnánk). Ez esetben az egyenletrendszerünk a következőképpen írható fel:

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \cdot \left[A \cdot e^{j \cdot 2\pi \cdot f \cdot dt}\right] = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n+1} \end{bmatrix} \tag{3.7}$$

Látható, hogy az egyenletrendszer ún. túlhatározott egyenletrendszer, hiszen az egyenletek száma nagyobb, mint az ismeretlenek száma [9]. Az $A \cdot e^{j \cdot 2\pi \cdot f \cdot dt}$ együtthatót keressük, amely a (3.5) egyenletben $\underline{x}$ vektorként szerepelt. Ezen egyenletrendszer esetén a hagyományos inverz művelet nem használható, így más módszert kell alkalmaznunk, az ún. Moore-Penrose pszeudoinverz segítségével oldhatjuk meg az egyenletrendszert [10]. Definíció szerint egy $m \times n$-es $\boldsymbol{A}$ mátrix pszeudoinverzén azt az $\boldsymbol{A}^+$-szal jelölt mátrixot értjük, amellyel a sortér minden $\underline{x}$ vektorára, illetve az oszloptérre merőleges minden $\underline{z}$ vektorra igazak a következő összefüggések,

$$\boldsymbol{A}^+ \cdot (\boldsymbol{A} \cdot \underline{x}) = \underline{x}$$

$$\boldsymbol{A}^+ \cdot \underline{z} = 0$$

illetve az egyetlen olyan $m \times n$-es mátrix, mely kiegyenlíti az alábbi négy feltételt [10]:

$$\boldsymbol{A}\boldsymbol{A}^+\boldsymbol{A} = \boldsymbol{A}$$

$$\boldsymbol{A}^+\boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{A}^+$$

$$(\boldsymbol{A}\boldsymbol{A}^+)^T = \boldsymbol{A}\boldsymbol{A}^+$$

$$(\boldsymbol{A}^+\boldsymbol{A})^T = \boldsymbol{A}^+\boldsymbol{A}$$

Amennyiben a mátrix oszlopai lineárisan függetlenek, a pszeudoinverz zárt alakban megadható a következő módon:

$$\boldsymbol{A}^+ = (\boldsymbol{A}\boldsymbol{A}^T)^{-1}\boldsymbol{A}^T \tag{3.8}$$

Amennyiben a mátrix oszlopai lineárisan függetlenek, a lineáris egyenletrendszer legkisebb négyzetek problémájának egyetlen megoldása van, melyet a pszeudoinverz ad meg:

$$\underline{x} = \boldsymbol{A}^{+}\, \underline{b} \tag{3.9}$$

A pszeudoinverz alkalmazásával bizonyíthatóan megkapjuk az egyenletrendszerünk négyzetes hibában minimális megoldását, így a jelre minimális hiba mellett tudunk szinuszt illeszteni. A pszeudoinverz implementálását MATLAB környezetben egyszerűen az ún. *mldivide* operátor alkalmazásával tudjuk megtenni. A programban az egyenletrendszer megoldásából a fázisinformációt használjuk fel, majd kiegészítjük a kapott eredményt azzal a frekvenciával, amivel lemoduláltuk az elején. Képletbe foglalva:

$$\varphi = 2\pi \cdot df \cdot dt, \text{ ahol } dt = \frac{1}{fs} \tag{3.10}$$

$$f_{\text{pontosított}} = f_0 + \frac{\varphi \cdot f_s}{2\pi} \tag{3.11}$$

ahol f_0 a felismert alaphang, φ az egyenletrendszer megoldásának fázisinformációja, f_s a mintavételezési frekvencia (44100 Hz).

Egy példa a módszer alkalmazására:

$f = 82{,}3$ Hz – az eredeti szinuszjel alapfrekvenciája – ideális esetben ezt kapnánk

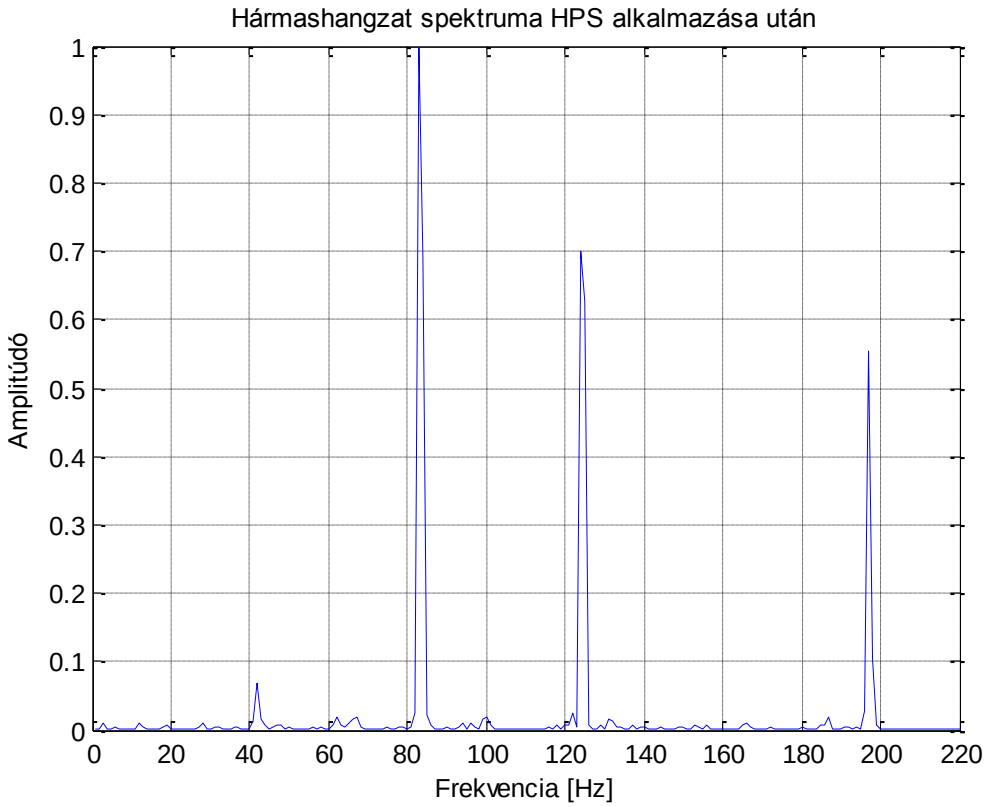
$f_0 = 96{,}9$ Hz – az algoritmusunk által meghatározott alapharmonikus frekvenciája

$f_{\text{pontosított}} = 82{,}2$ Hz – a pontosított felismert frekvencia.

Látható, hogy mindkét módszer elvégzi a kívánt pontosítást, így ezzel a pontosított jellel már a kromavektor feltöltésénél is minimalizáljuk a hibát. Dolgozatomban a gyorsasága, illetve az elegánsabb megoldás érdekében a második módszerrel dolgoztam.

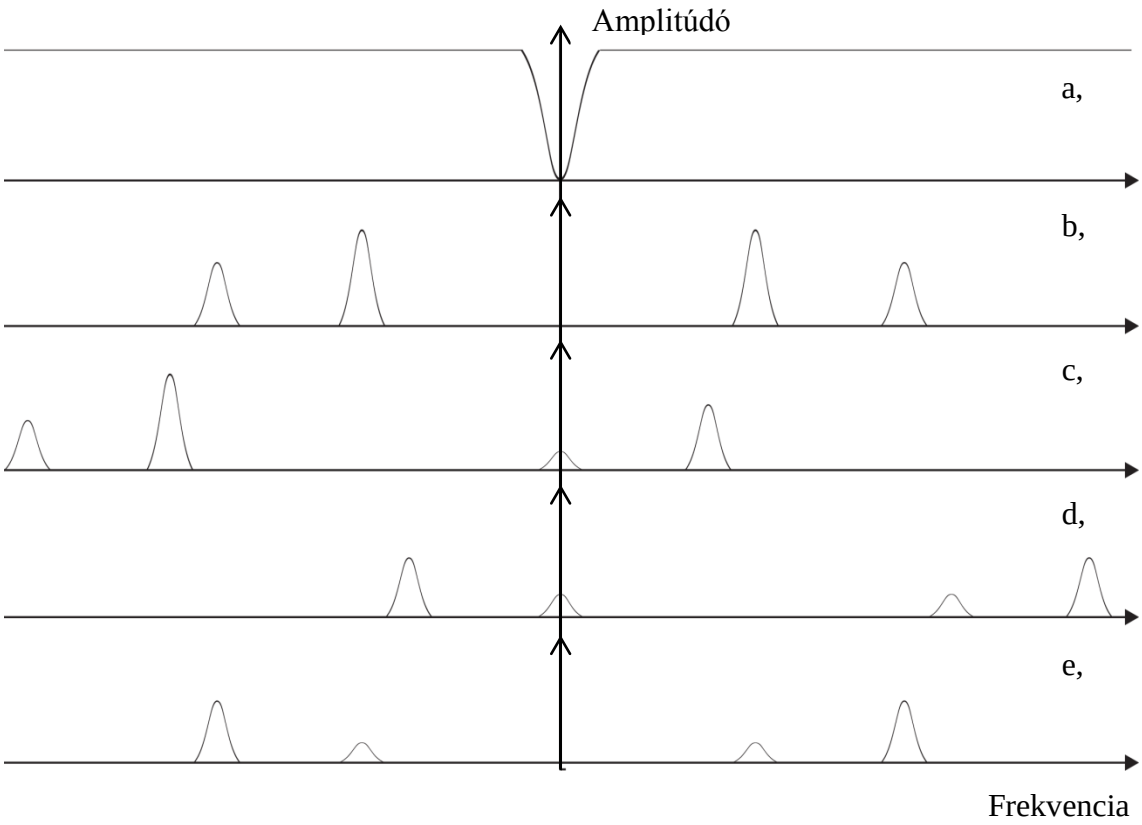
3.3 A hangmagasság felismerés kiterjesztése több hangra

Idáig sikerült létrehozni egy működő rendszert a bemeneti minta alaphangjának felismerésére. Azonban dolgozatomban akkordok felismerését tűztem ki célul, így megoldást kellett találni arra, hogy a második, illetve harmadik legnagyobb energiájú hangot is megtaláljuk a spektrumban, és ebből a hanghármasból már meghatározhatóvá váljon az akkord. Ehhez meg kellett találni a módját, hogy tudjuk elvégezni a HPS számítást többször egymás után hatékonyan. Dolgozatom most következő részében az egyik legfontosabb részegységet mutatom be, amit fejlesztésem során implementáltam.



3.9. ábra: Egy E moll akkord spektruma HPS alkalmazása után

Kezdetnek a 3.9. ábrából indulok ki, miután megtaláltuk a HPS algoritmussal a bemeneti jel legnagyobb intenzitású alaphangjának frekvenciáját. Az alapötlet az, hogy minden egyes HPS számítás után kiszűrjük a megtalált alaphangot, majd megkeressük a következő legnagyobb intenzitású hangot. Erre szakdolgozatomban hasonló módszert alkalmaztam, mint az előbb bemutatott pontosításnál, tehát első lépésben lemodulálom a jelet olyan módon, ahogy a 3.5. ábrán már láthattuk. A különbség, hogy jelen esetben felülátereszto szűrőt használunk. A cél az, hogy a DC-re lekevert alaphangunkat a lehető legpontosabban kiszűrjük, majd visszakeverjük az eredeti pozíciójába. A megoldást az alábbi ábrák szemléltetik:



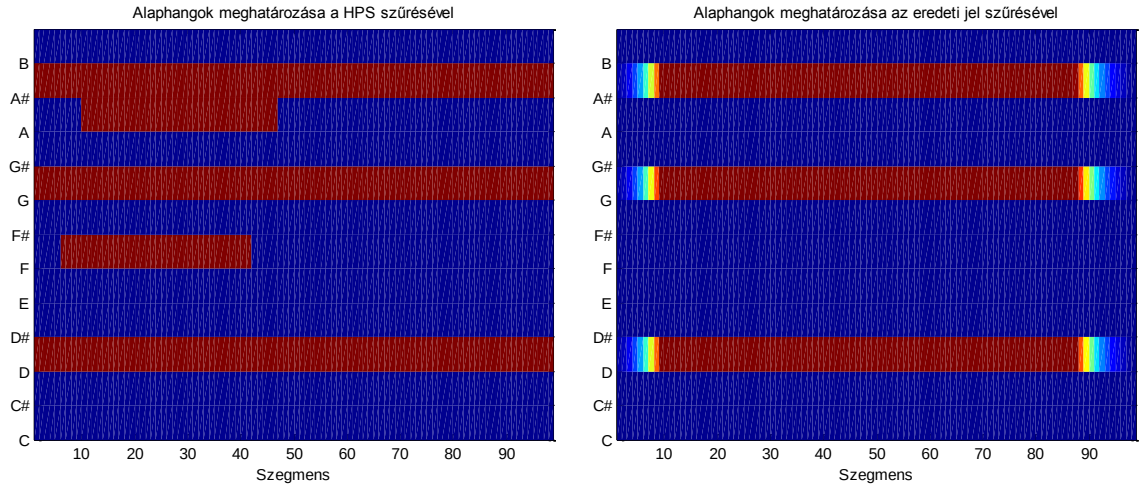
3.10. ábra: A hangmagasság felismerés kiterjesztése több hangra

a, Az alkalmazott szűrő átvitele
b, A detektált és a következő detektálandó hang frekvenciái
c, A jel lemodulálva DC szintre, szűrés után
d, A jel eltolva 2*f_0-lal, hogy a negatív komponenst is kiszűrjük
e, Visszamodulálva a kimeneti jel

Fontos felismerni, ahogy a 3.10. ábrán is látszik, hogy a spektrum szimmetriája miatt mindkét irányban el kell végeznünk a szűrést, ellenkező esetben az eltolás miatt a negatív frekvenciahelyeken megmaradhatnak a komponensek, amely zavart okoz.

Az implementált kódban, miután megkerestem a maximumot a HPS algoritmussal, a fent bemutatott szűrővel kiszűrtem az algoritmussal előállított spektrumból az alapfrekvenciát, és annak többszöröseit (felét, negyedét, illetve egész számú többszöröseit). Azonban egy lassabb, ám pontosabb módszer az, ha az eredeti jelből szűrjük ki a kívánt komponenseket, majd ezután ennek számítjuk a HPS-ét. Ez azért kifejezetten fontos különbség, mert ha a HPS által összenyomott spektrumból

szűrném ki a felismert alaphangot, mivel a frekvenciacsúcsok közelebb kerültek egymáshoz, előfordulhat, hogy a szűréssel a következő meghatározandó alaphangot is kiszűröm egyaránt. Azonban az eredeti jelben, ha az alapharmonikust megszűrjük, a többi felharmonikus az értékét jelentősen megnöveli, így felismerhető marad, ellentétben a HPS-sel kialakított jellel, ahol a felharmonikusokat az alapharmonikus binjébe juttattuk. A különbséget a két módszer között a következő, már kész kromavektorokat tartalmazó ábra szemlélteti:



3.11. ábra: Az alaphangok meghatározásának két különböző módja

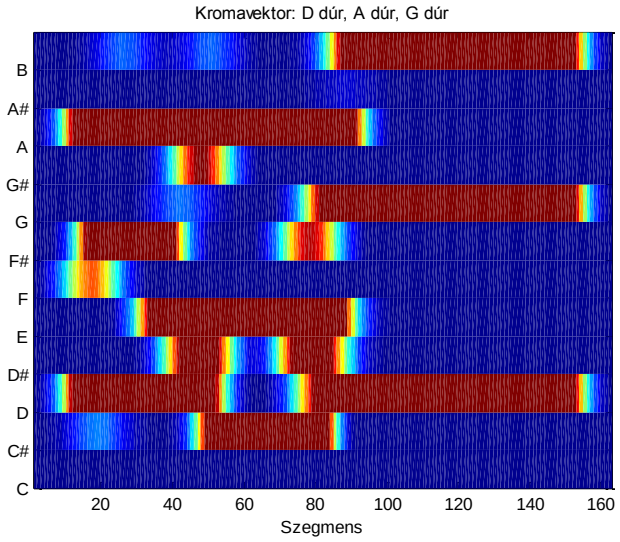
Az ábrákon egy G moll akkord látszik. Hamar észrevehető, hogy míg a második módszer ugyan lassabb, lényegesen kevesebb hibás komponenst tartalmaz, ami megkönnyíti az akkord pontos felismerését. Az első módszernél a rendszer természetesen hibás működést mutat, hiszen hiába jelennek meg végig a megfelelő alaphangjai az akkordnak, az akkord kiértékelését a hasonló energiájú tévesen érzékelt hangok ellehetetlenítik.

3.4 Az alaphangok rendszerezése

Miután nagy pontossággal felismertük az akkord hangjait, el kell raktároznunk őket egy egyszerű, viszonylag könnyen kezelhető tárolóba. Erre szolgál a 2. fejezetben bemutatott kromavektor, ami a 12 félhang energiáját tartalmazza elemenként (Megvalósítható úgy is a rendszer, hogy egyszerűen nullákkal, vagy egyesekkel töltjük fel a vektort, amitől a számítás egyszerűsödik, azonban a pontosság kárára, hiszen ilyenkor nem lehet használni azt a fajta szűrést, amit implementáltam a modellben).

Látható, hogy az oktávok között nem tesz különbséget az algoritmus, így egy elem a vektorban igazából a hangot és oktávjait is tartalmazza.

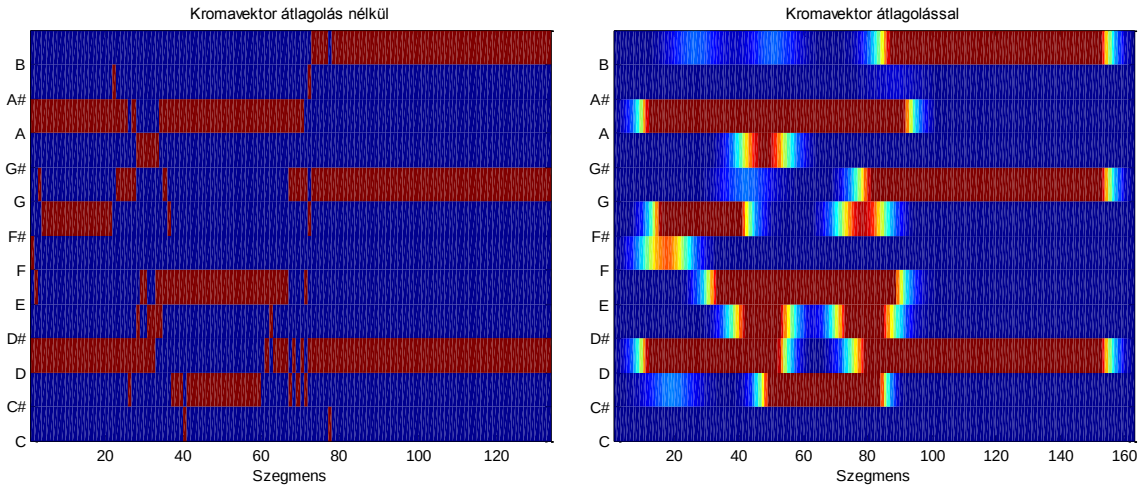
Először definiálom hangok alsó határának megfelelő frekvenciaértékeket, ezután következik a kromavektor kitöltése. Fontos megemlíteni, hogy a vektor időbeli, azaz szegmensenkénti változását is eltárolom egy mátrixba, így lehet nyomon követni, hogy a hangminta darabjai milyen energiájú komponenseket tartalmaznak egy-egy szegmensben. Ezt szemléltetem a következő ábrával, ahol egy D dúr – A dúr – G dúr akkordváltást láthatunk:



3.12. ábra: Az időben változó kromavektor

A képen látható, hogy a pontos hangok mellett megjelenik néhány hibát okozó komponens, ezek kezelését az utolsó lépésben végzem majd, ahol kikötöm, hogy azokat az értékek, amelyek a kromavektorban található legnagyobb érték felénél kisebbek, ne vegye figyelembe az algoritmus az akkord osztályozásakor. Ez a módszer ugyan hibátlan felismerést szavatol, azonban a dinamikát erősen korlátozza a vizsgálandó hang esetében, hiszen ha egy viszonylag hangosan megpengetett akkord után egy halkabb jön, akkor fennáll a lehetősége, hogy a halk akkordot egyszerűen nem érzékeli az algoritmus, így annak információját elveszítjük. Ezt meg lehetne oldani egy olyan algoritmussal, ami folyamatosan vizsgálja a maximumértékeket, és ebből következtet arra, hogy egy új akkord következik, vagy csak néhány zavaró komponens jelent meg a jelben.

Miután a kromavektort meghatároztuk, alávetjük egy időtartománybeli átlagolásnak, ami szűrési funkciót lát el, ezzel még inkább pontosítva az értékeinket. A következő ábrákon látható, hogy milyen a felismerés átlagolás nélkül, illetve átlagolással.



3.13. ábra: Az átlagolás hatásának szemléltetése

Látható, hogy a kisebb értékű zavarok teljesen eltűntek, de itt is megfigyelhető az a jelenség, hogy az akkordot a rendszer a megjelenése után egy kis késéssel érzékeli. Ez a hiba az átlagolás előnyeihez képest elhanyagolható, az akkordfelismerés minőségét nem befolyásolja.

Az eddigi algoritmusok alapján már viszonylag pontos eredményeket lehet elérni az akkordok felismerésénél, de a következő alfejezetben láthatjuk, hogy a kiíratásnál még több, hibásan érzékelt akkordot is jelezne a rendszer, így ezt is ki kell küszöbölni a tökéletes eredmény elérése érdekében.

3.5 Az akkordok osztályozása, kiíratása

Az akkordok osztályozását az ismertetett módszerek közül sablon-alapú megoldással valósítottam meg, tehát definiáltam a 24 akkordnak (12 dúr és 12 moll) megfelelő maszk kromavektorokat, ahol 0 jelzi azt, hogy az adott indexű hang nincs benne az akkordban és 1 azt, ha benne van. Ahhoz, hogy kiterjesszük az algoritmust egyéb akkordok felismerésére is, első lépésnek ezt az adatbázist kell bővíteni, ami a lehetséges akkordokat tartalmazza, amelyekkel összehasonlítjuk a későbbiekben a bemenetről érkező hangmintánkat. Ezután egyszerűen összeszorozzuk a felismert

alaphangjainkból alkotott kromavektorokat a definiált sablonokkal. A skaláris szorzat definíciója szerint ez a szorzat akkor lesz maximális, ha teljes az egyezés. Ez képletbe foglalva:

$$\Delta_i = \sum_{n=0}^{N-1} \left(X_i(n) \cdot S(n) \right)^2 \tag{3.12}$$

ahol X az éppen vizsgált kromavektort, S pedig a sablont reprezentálja. A fenti képletet úgy kell értelmezni, hogy megvizsgáljuk az összes kromavektor és a definiált sablonok hasonlóságát, majd elemezzük, mikor lesz maximális a skaláris szorzat.

Eltároljuk a maximális értéket is, amelyet az utólagos szűrésnél fogunk felhasználni. Az osztályozás utolsó lépése, hogy megvizsgálja, hogy a felismert akkord melyik sablonhoz tartozik, és ezt beírja egy változóba.

Fontos megjegyezni, hogy egy kezdetleges csend-detektálást is implementáltam a rendszerbe, hogy az akkordváltásoknál illetve szüneteknél ne kapjunk vissza hibás eredményt. Az algoritmus nem dinamikus állítású, működése alapja, hogy egy megkapott kromavektornak beállít egy alsó határt, egy előre definiált küszöbérték alatti értékeket nem vesz figyelembe. Ezt a kiíratás műveleténél vizsgáljuk időlépésenként. Ha a felismert akkord értéke egy küszöbérték alatt van, azt zavarként kezeljük és eldobjuk. Ha a küszöbérték felett van az értéke, akkor kiírjuk az akkordot, előfordulásonként egyszer. Tehát a rendszer akkor jelez új akkordot, ha változás volt az előzőhöz képest, nem írja ki minden szegmensre az információt.

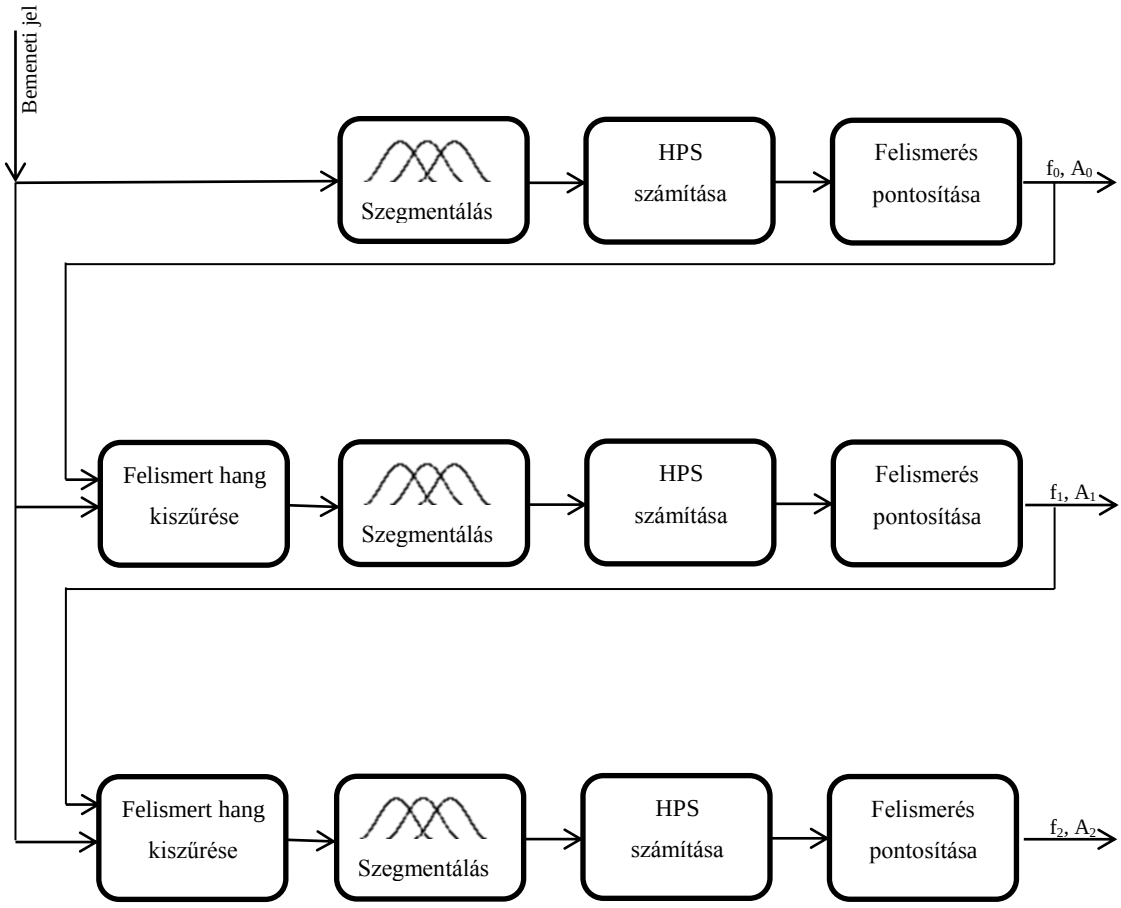
Végül a program az akkordmenet visszaadásán felül az akkordok lefogását és kottáját is megjeleníti, négyes osztásokban.

3.6 A teljes rendszer megvalósítása

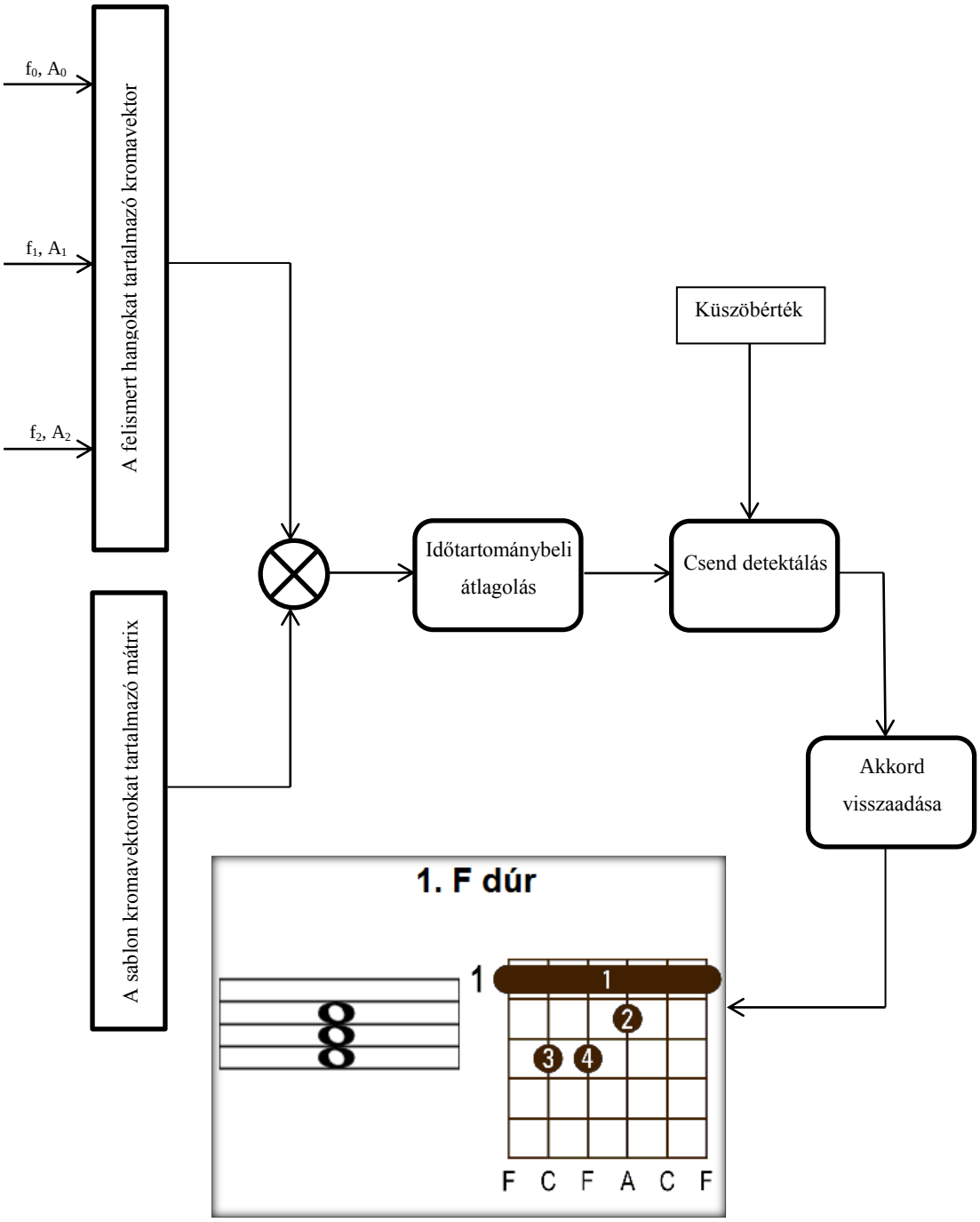
Miután az algoritmusokat implementáltam, egy működő rendszerré fűztem őket össze. Működése pár szóban összefoglalva:

A jel szegmentálásának elvégzése után minden időpillanatban HPS-t számolunk, elvégezzük a pontosítást, majd kiszűrjük a megtalált alaphangot az eredeti jelből. Minden lépés végén meghatározzuk az aktuális szegmensre jellemző akkordot, majd eltároljuk egy mátrixba. Az iteráció végén megvizsgáljuk a feltöltött kromavektor korrelációt a definiált sablonokkal, majd egy időtartománybeli átlagolás, illetve csend detektálás után kiíratjuk a keresett akkordot.

A megvalósított rendszert két részegységre bontva mutatom be, egy hangfelismerő, illetve egy akkordosztályozó részre bontva:



3.14. ábra: A megvalósított rendszer hangfelismerő részegysége



3.15. ábra: A megvalósított rendszer akkordfelismerő részegysége

4 A rendszer tesztelése

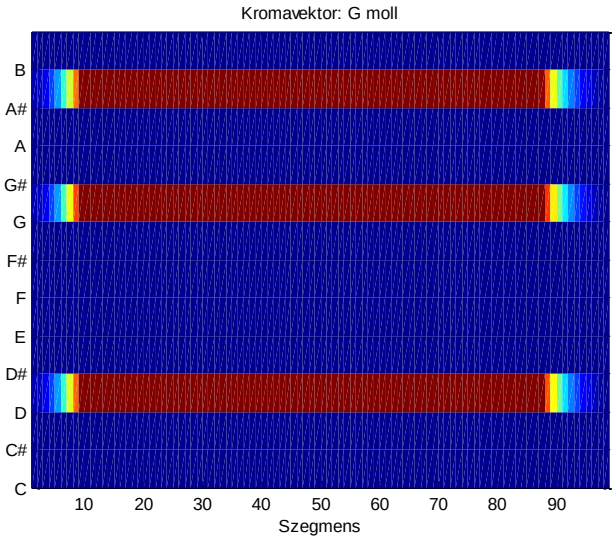
A rendszert különböző, néhol szélsőséges mintákkal teszteltem, felmértem a korlátait, illetve néhány paraméter módosításával igyekeztem alkalmassá tenni többféle hangszer, többféle hangfekvés felismerésére is. Az itt bemutatott teszteknél kitérek arra, hogy mi a rendszer várt viselkedése egyes minták esetén, illetve ennek okaira is.

4.1 A rendszer fő profilja, a gitár

Ahogy dolgozatom elején is szerepel, a teljes rendszert a gitárjáték akkordjainak felismerésére optimalizáltam, de közel sem egyforma hangzású, egyforma jel-zaj viszonnyal rendelkező mintákkal végeztem a tesztelést. A következő alfejezetben bemutatott minták többféle elektromos gitárral, többféle berendezésen készültek, így némileg hangzásban is eltérnek.

4.1.1 Elektromos gitár – tiszta és torzított hangszínek

Az első minta, mellyel még a fejlesztés során teszteltem a rendszer funkcionalitását egy elektromos gitárral feljátszott G moll akkord volt, ennek láthatjuk alább az eredményét:

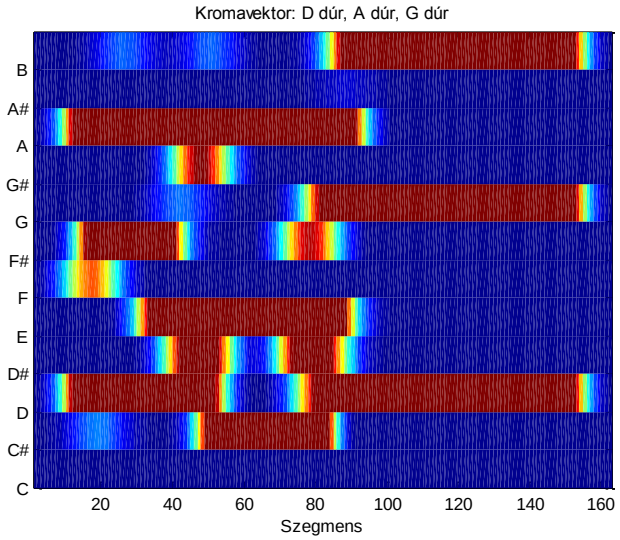


4.1. ábra: Egy G moll akkord kromavektorja

Az ábrán látszik, hogy tökéletes a felismerés. Amit már említettem korábban, a rendszer egyik kompromisszuma az átlagolásos szűrés miatt, hogy nem azonnal ismeri fel az akkordot, van egy kis késés. A szűrőt kicsit hangolni kell, ha esetleg az akkordváltások nagyon gyorsan történnek, vagy lényegesen halkabb az egyik akkord, hiszen akkor az átlagolás miatt elveszik az információ. Erre a minták között mutatok majd példát, illetve a megoldást is a problémára.

A rendszer továbbá az akkordhoz ábrázol egy lehetséges fogást gitáron, illetve a kottáját is megmutatja alapfordításban. Dolgozatomban ezt később mutatom be egy szemléletesebb példánál.

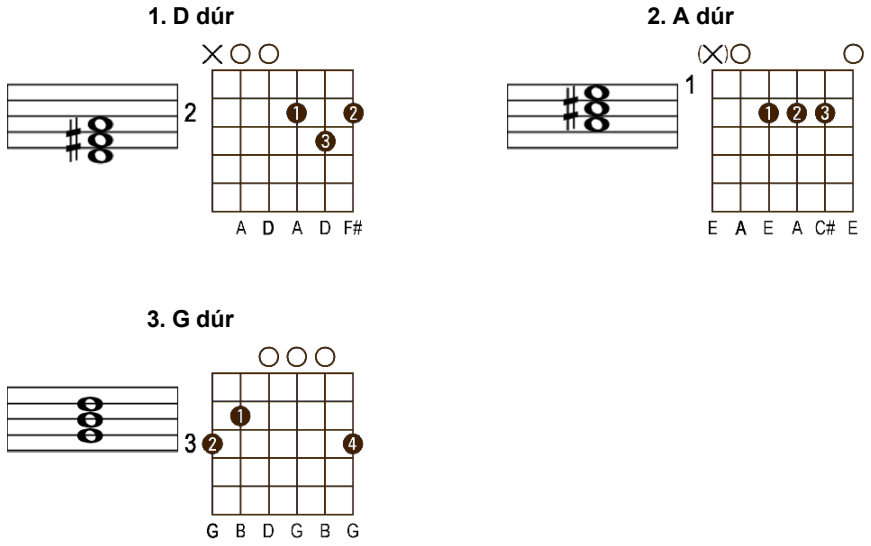
A következő minta, amivel még a fejlesztés során igyekeztem az akkordmenetek kezelését megvalósítani, szintén elektromos gitárral felvett D dúr, A dúr, G dúr akkordokat tartalmaz. Itt az átlagolás fontos szerepet játszik, mivel a rendszer a váltásoknál hibás akkordokat is felismer, hiszen nyilván nem egyszerre szólalnak meg a következő akkord hangjai, illetve szűnnek meg az előzők. Ha egy minimális átfedés is keletkezik a két akkord közt, az hibás működésre vezetne, ezért kell az előző fejezetben említett Gauss-ablakos átlagolást alkalmaznunk. A működést szemléltető ábra:



4.2. ábra: D dúr, A dúr, G dúr akkordmenet kromavektorja

A mintában megfigyelhető többek között, hogy a gitár hangolása miatt fellép néhol egy kis lebegés, azonban jól kivehetőek az akkordok alaphangjai. Itt azonban még hibás lenne az akkordkijelzés, ezért kell alkalmaznunk azt a szigorú feltételt, hogy amely hang

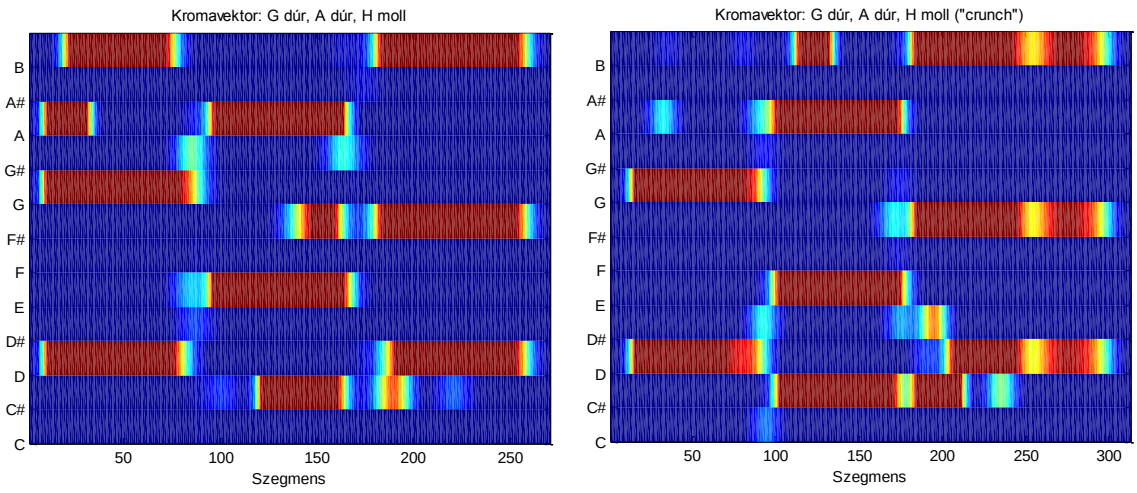
a maximum érték felénél alacsonyabb, azt eldobjuk. Ekkor már a rendszer hibátlanul felismeri az akkordokat, ahogy a program kimenetének ábrázolása is mutatja:



4.3. ábra: D dúr, A dúr, G dúr akkordok ábrázolása a programban

Ezután a rendszer sokszínűségét azzal kívántam fejleszteni, hogy a gitár effektezésére vonatkozó toleranciáját tesztelem, hiszen a gitár torzítása a modern zenében mintegy effektként mindennapos használatúvá vált. Emellett azt kívántam bemutatni, hogy a rendszer nem kíván meg pontos hangolást, a mintákat enyhén elhangolt, pontatlan gitárral vettem fel. Igyekeztem, hogy a minták összehasonlíthatóak legyenek, így metronómra, közel ugyanolyan erősséggel pengetve vettem fel a mintákat. Emellett persze a jelentkező különbségeknél figyelembe kell venni az emberi játék hatásait is, tehát a minták csak *közel* összehasonlíthatóak, azonban dolgozatomban ezeknek tanulságaiból vonok le következtetést, nem a reprezentatív tesztek , illetve azok összehasonlíthatósága a fő prioritás.

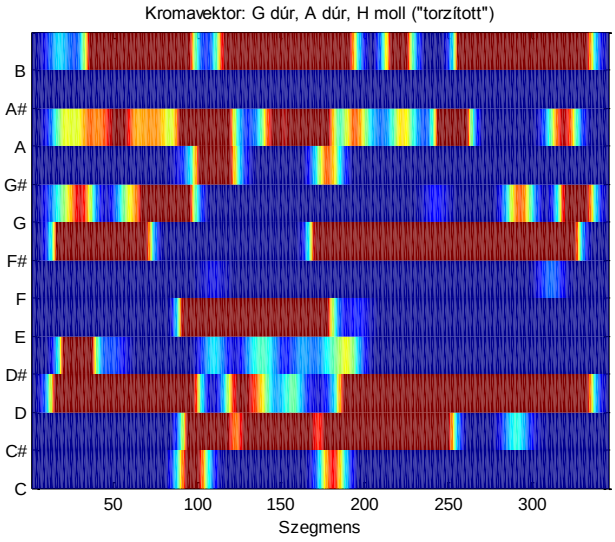
A következő két ábra egy elektromos gitáron feljátszott G dúr, A dúr, H moll akkordmenetet ábrázol tiszta, illetve enyhén torzított (úgynevezett „crunch”) hangszín használatával:



4.4. ábra: Tiszta és torzított hangszínű gitáron G dúr, A dúr, H moll váltás

Az első két ábra enyhe torzítás hatását mutatja. Látható, hogy a torzított mintában az első akkordnál a H hang jóval gyengébb energiájú, de jelen van, így ez a felismerésnél nem jelent problémát. A kompresszió jelenségét több helyen is megfigyelhetjük a mintában, a jelek „egybefüggőbbek”, hosszabbak. Fontos megemlíteni, hogy a második mintánál egy enyhe lebegés figyelhető meg H moll és H dúr felismerése közt, ezt egyszerűen ki lehet küszöbölni az átlagolás szigorításával, így a rendszer a helyes akkordokat ismeri fel.

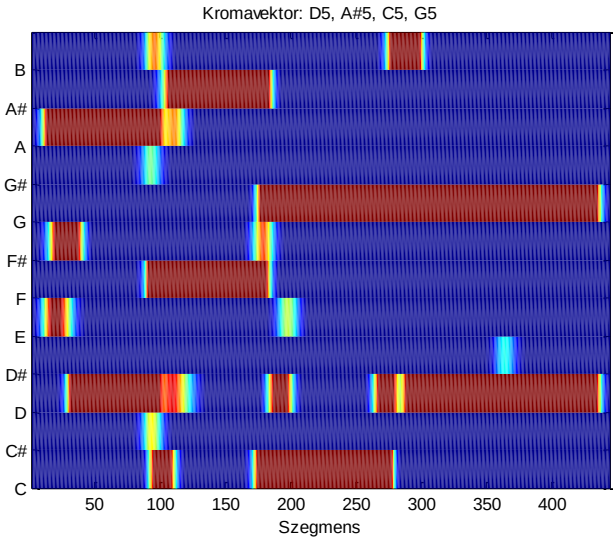
A torzítás további növelésével a következő ábra adódik:



4.5. ábra: Erősen torzított gitáron G dúr, A dúr, H moll akkordváltás

Az ábrán az előbb elmondottakat látjuk még sarkalatosabban, a durvább torzítás erősebben kompresszál, a felharmonikusok energiájának erősödésével megjelennek zavart okozó hangok.

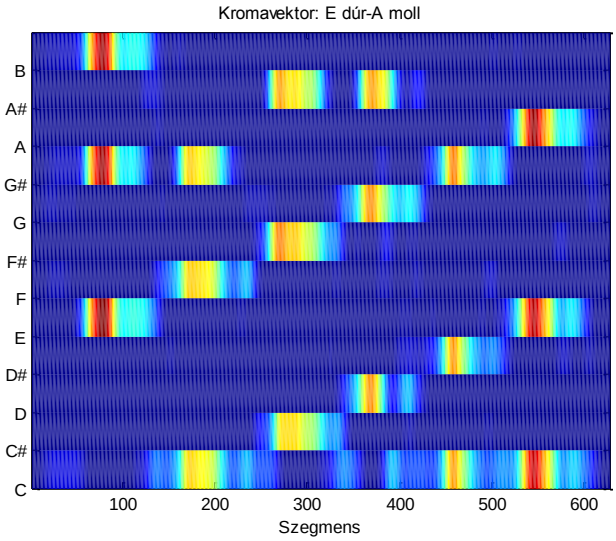
A következő tesztelt minta torzított gitárral feljátszott 4 db 5-ös (vagy más néven *üres akkord*) volt, itt azt vizsgáltam, hogy a felismert hangok számát átállítva 2-re hogy működik a rendszer (ugyanis az üres akkordok két hangból állnak – egy alaphangból, illetve annak a kvintjéből). Ezzel a rendszernek egy határvonalát mutatom be. A minta tesztelése után a következő ábrát kaptam:



4.6. ábra: D5, A#5, C5, illetve G5 akkordok kromavektorja

Látható, hogy ezen akkordok felismerését is viszonylag pontosan lehet kivitelezni, azonban itt az akkordok kirajzoltatását dolgozatomban nem oldottam meg, hiszen mint az elején említettem, az eredeti cél a hármashangzatok (dúrok és mollok) kezelése volt. Azonban a fenti ábra mutatja, hogy ilyen típusú akkordok esetén is értékelhető eredményt kapunk.

A torzítás hatása után egy zajosabb, félhangonként lépegető, moll és dúr akkordokat váltakozva tartalmazó mintát vizsgáltam. Tettem ezt egyféle stressz tesztként, hogy egy ilyen sarkalatos minta esetén történik-e hiba, a félhangos ugrások, vagy a rossz minőség miatt. A következő ábra adódott:

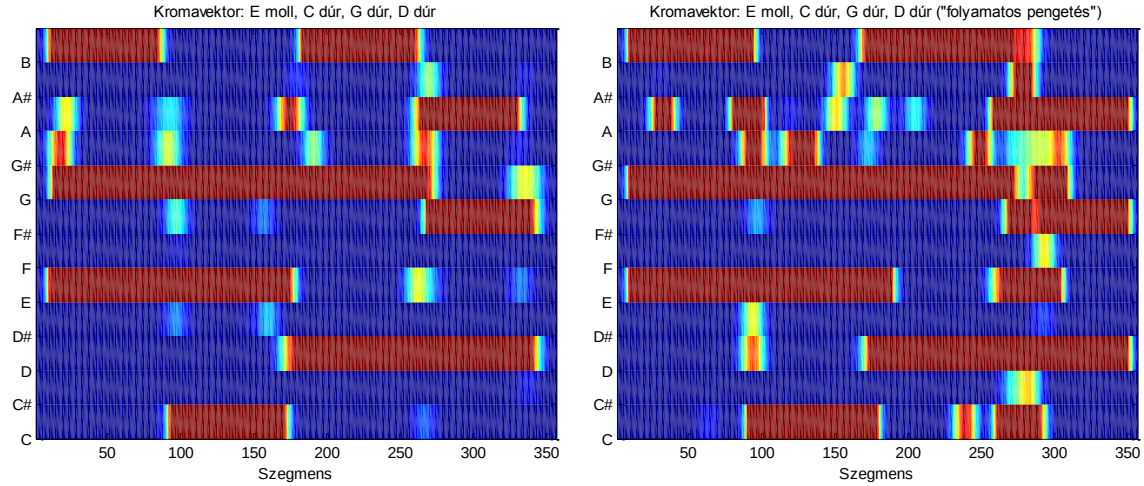


4.7. ábra: E dúr, F moll, F# dúr, G moll, G# dúr, A moll akkordváltások

Ugyan az ábrán az akkordok nagyon kis energiájúak, de az utószűrésnél az algoritmus ezen kis intenzitású jel maximumához mérten nullázza ki az 50%-nál kisebb értékű komponenseket, így ebből is meghatározhatóvá válnak az akkordok, az algoritmus a pontos eredményt ad.

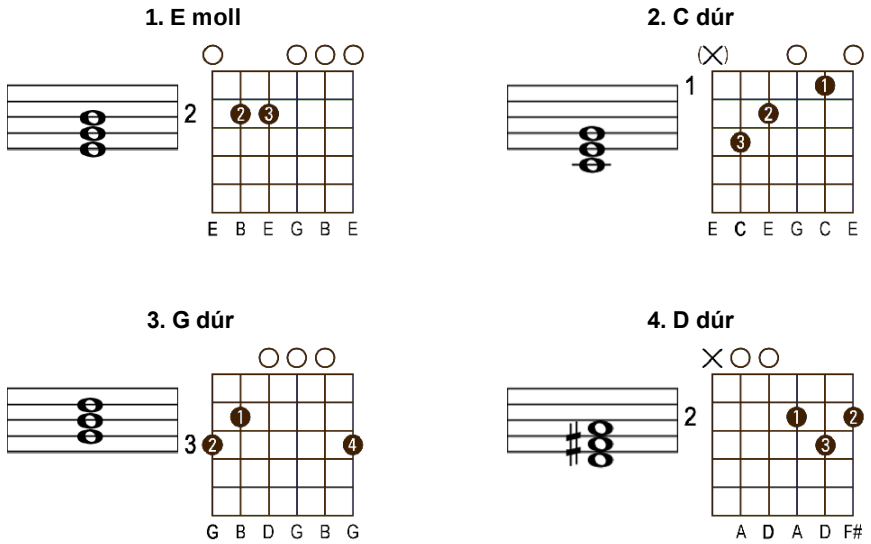
4.1.2 Akusztikus gitár – egyszeri és folyamatos pengetés

A következő teszt kérdése, hogy a rendszer hogyan reagál mikrofonozott akusztikus gitárral felvett mintákra, illetve az egyszeri, valamint folyamatos akkordpengetés esetén lesz-e eltérés, hiba a felismerésben. A teszt szintén elég szélsőséges, hiszen nem profi körülmények között rögzítettem, illetve a felvételeken szereplő akusztikus gitár sem professzionális, beállított hangszer. Az eredmény:



4.8. ábra: E moll, C dúr, G dúr, D dúr egyszeri, és folyamatos pengetéssel

Egyszeri pengetés esetén jóval kevesebb a hiba, a folyamatos pengetés hatásaként sokkal több zavaró hang jelenik meg. Emellett viszont jól kivehető a 4 akkord, a rendszer is pontos eredményt szolgáltat, a következő ábrát látjuk mindkét esetben:

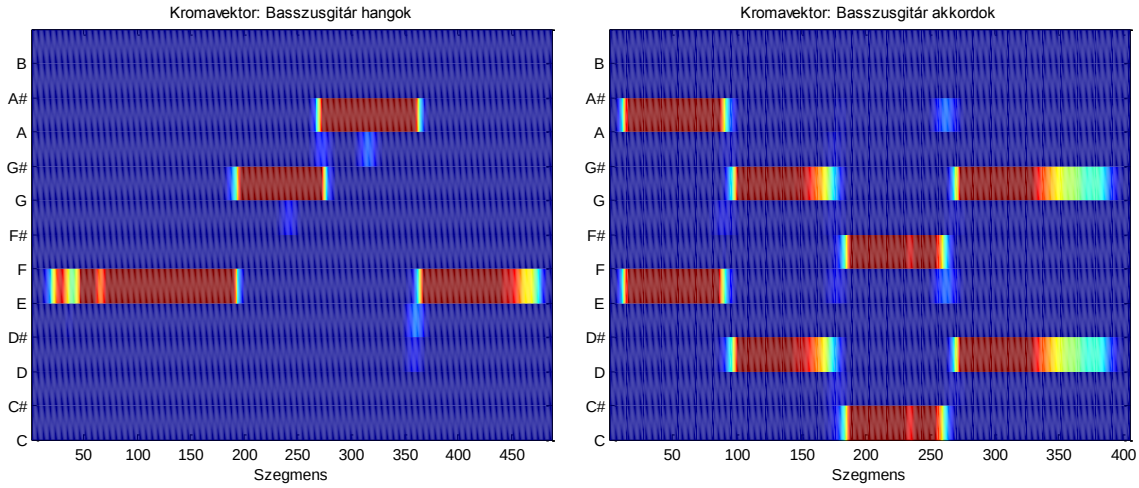


4.9. ábra: Akusztikus gitár: E moll, C dúr, G dúr, illetve D dúr váltás

4.1.3 Basszusgitár és lehangolt gitár – mély hangok hatása

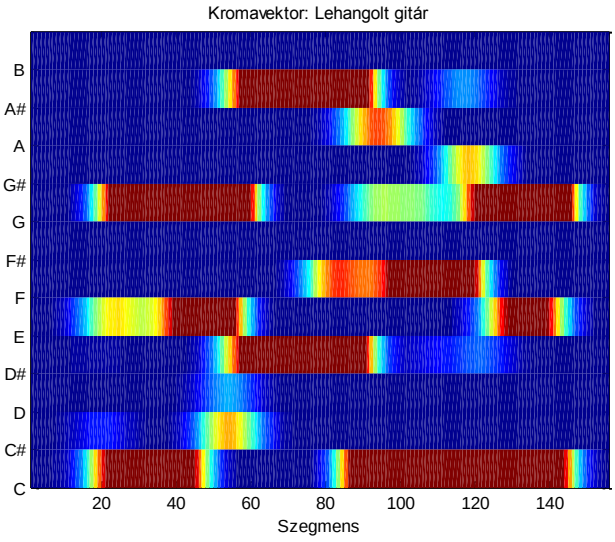
Ebben részben azt szeretném megmutatni, milyen az algoritmus viselkedése a viszonylag mély hangokra. A következőkben az algoritmus alkalmazhatóságának körét vizsgálom.

Először a basszusgitárt vizsgálom – először egy-egy alaphang pengetése esetén, majd a két hangot tartalmazó üres akkordok esetét is bemutatom. Az algoritmus pontos működését várom mindkét esetben, amiből látszik, hogy modern húros hangszerek esetén a hangszínre kvázi invariáns a rendszer, tehát ugyanúgy képes meghatározni egy mélyebb hangot basszusgitár esetén, mint több oktávval feljebb egy gitár esetén.
A basszusgitár két mintáját tesztelve:



4.10. ábra: Basszusgitár: E, G, A, E hangok és A5, G5, F5, G5 akkordok

A lehangolt gitárnál a nem megfelelő vastagságú húrok nagyobb kilengéséből adódó hangmagasság-pontatlanság miatt kevésbé pontos eredményre számítok, azonban következtetni tudunk majd belőle, hogy használható-e esetleg 7 húros, vagy bariton gitárok esetén az algoritmus. Az eredmény:



4.11. ábra: Lehangolt gitár esetén C5, D#5, C5 akkordok

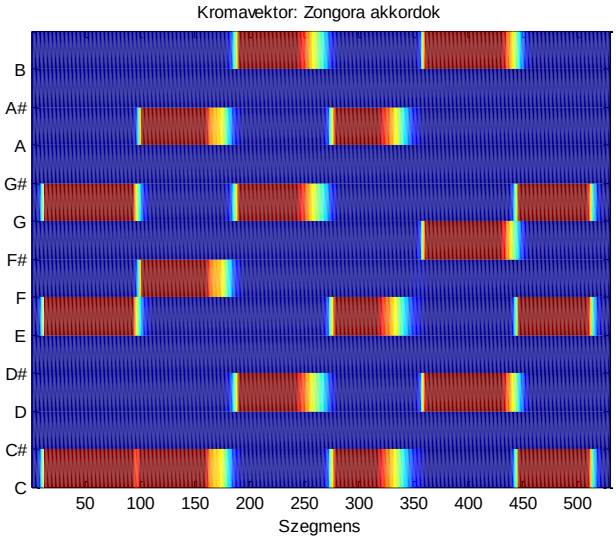
Látható az ábrán, hogy a nagyobb energiájú jelek zavart okoznak, azonban a felismerésnél probléma csak az utolsó C5 akkordnál van, amikor megjelent egy nagyobb energiájú E hang is, pedig a pengetett akkord csak C, illetve G hangot tartalmaz.

Ezek alapján az algoritmus nem érzékeny a normális tartományba tartozó mély hangokra, ugyanúgy használható basszusgitár, illetve lehangolt, vagy 7 húros gitár akkordjainak felismerésére is – természetesen a felismerni kívánt alaphangok számának beállítása után – bármilyen típusról is legyen szó.

4.2 Akkordfelismerés zongora, szintetizátor esetén

A következő néhány vizsgálat arra irányul, hogy egy hangolásban biztosan pontos zongorahangszínt hogyan tud kezelni az algoritmus. Először megvizsgálom az akkordfelismerés lehetőségeit a hangszerre, majd kitérek a határok vizsgálataira, úgy mint egy nagyon magas akkord felismerése, valamit akkordfordítások kezelése.

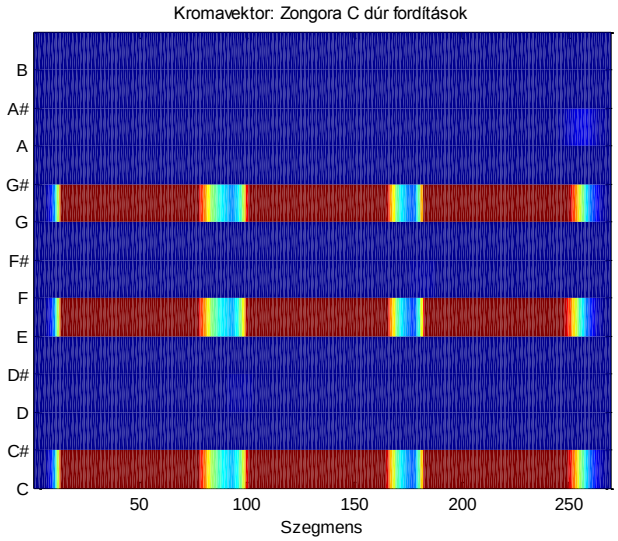
Először a zongorán feljátszott C dúr, F dúr, G dúr, A moll, H moll, C dúr akkordkörre végeztem el a tesztelést. Az eredmény a következő:



4.12. ábra: Zongorával feljátszott akkordmenet

Itt érdemes megjegyezni, hogy a G dúr és az A moll hallhatóan jóval halkabb a többi akkordnál. A kromavektoron is látszik, hogy a 3. és a 4. akkord intenzitása jóval kisebb, ezért az átlagolásnál való durva szűrést (amikor az 50%-nál kisebb értékeket eldobjuk), el kell hagyni, vagy nagyon minimálisra csökkenteni, hiszen ellenkező esetben a rendszer nem ismer fel két lényeges akkordot. Miután ezt a módosítást megtettük, a rendszer már hibátlanul kezeli a mintát. Emellett viszont láthatjuk, hogy a felismerés rendkívül jó, ebből arra következtethetünk, hogy ha a hangolás ideálisan pontos, a rendszer szinte hiba nélkül működik.

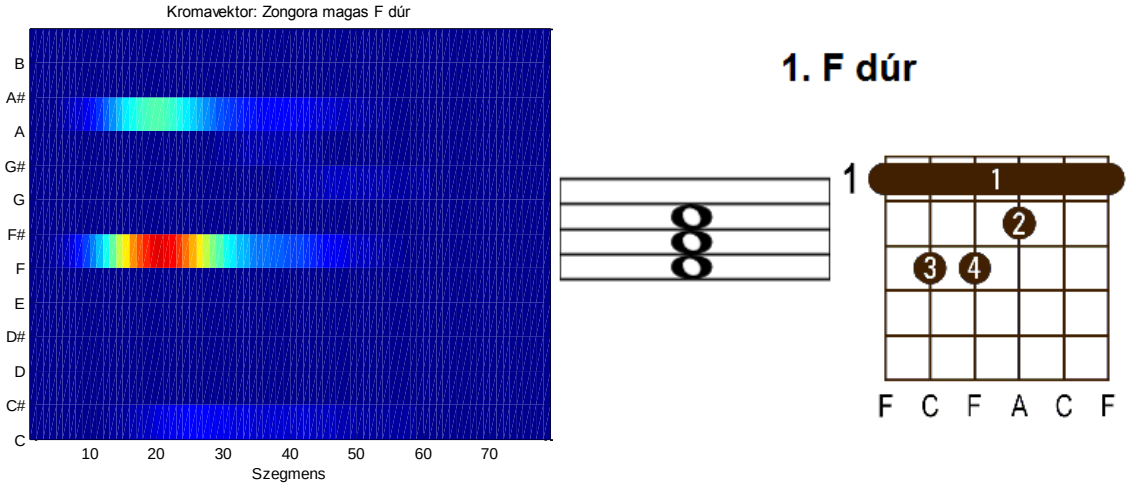
A következő minta 3 változatát tartalmazza a C dúr akkordnak, ezzel azt vizsgálom, hogy az akkordfelismerést befolyásolja-e, hogy milyen fordításban játsszuk az akkordot. A teszt eredménye:



4.13. ábra: C dúr akkord 3 fordításban

Jól látszik, hogy mind a háromszor C dúrt ismert fel, ezért a rendszer az akkordfordításokra invariáns.

A következő tesztet a nagyon magas tartomány tesztelésére végeztem el. A mintából szinte alig lehet hallani, hogy F dúr akkordról van szó, ennek ellenére a rendszer tökéletesen felismeri, a következőképpen:

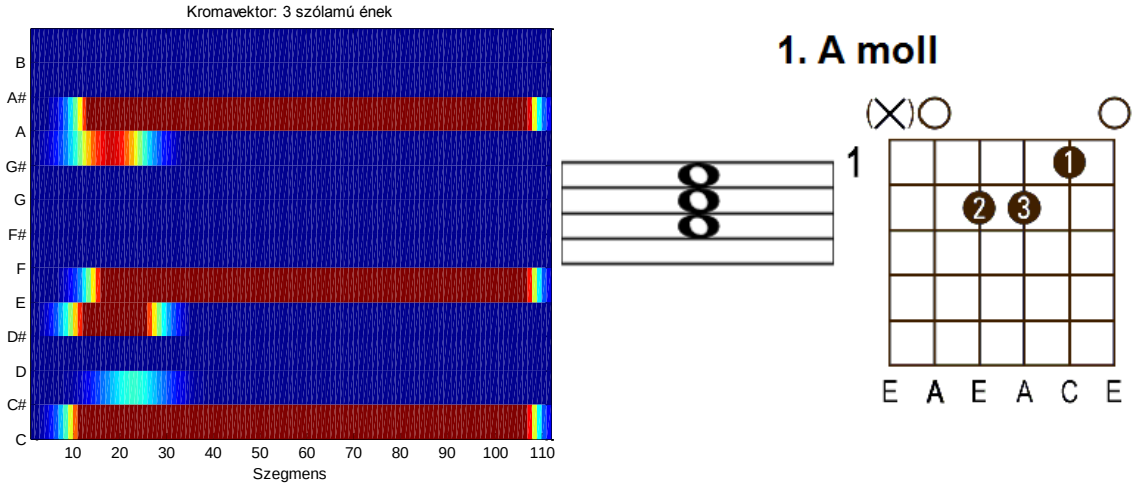


4.14. ábra: Zongorán egy nagyon magas F dúr akkord felismerése

4.3 Hangfelismerés egyéb felvett mintákból

4.3.1 Akkordfelismerés 3 szólamú énekből

Fontosnak tartottam egy olyan minta tesztelését is, ahol három énekszólam ad ki egy hármashangzatot, így készítettem egy mintát, ahol egy A moll akkord feléneklése volt a kitűzött cél. A rendszer a következőképpen működött:



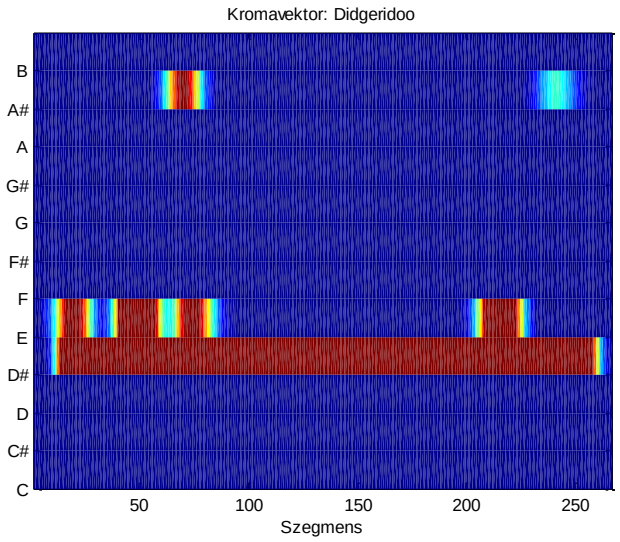
4.15. ábra: Egy felénekelt A moll akkord felismerése

Látható, hogy az elején az intonálásból eredő kis hibától eltekintve tökéletesen felismerte az énekelt akkordot. Fontos megjegyezni, hogy szöveges ének, illetve a beszéd felismerése az egy jóval komplikáltabb rendszert igényel, hiszen meg kell különböztetnünk a hangokat fajtájuk szerint is egymástól, például zöngés, illetve zöngétlenek közt is különbséget kell tennünk. Dolgozatomban erre részletesebben nem térek ki.

4.3.2 A didgeridoo hangjának felismerése

Utolsó mintám egy érdekes hangszerrel, az úgynevezett didgeridoo-val felvett hang. A didgeridoo egy ősi ausztrál hangszer, és a rézfúvósok családjába tartozik. A legősibb didgeridoo-k eredetileg egy C hang körüli frekvencián szólaltak meg, mára azonban szinte minden hangszínben kaphatóak. A mintában szereplő didgeridoo-ra az a specifikáció vonatkozott, hogy egy D# és E közötti hangot tud megszólaltatni, így a

tesztem egy terméktesztelés, hogy mi az a hang, amin a hangszer megszólal. Az eredmény:



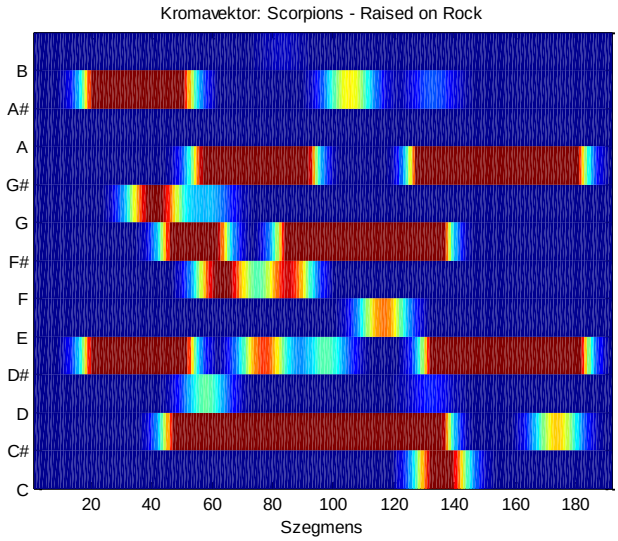
4.16. ábra: Egy didgeridoo alaphangja

A kapott eredményből egyértelműen kitűnik, hogy egy D#-hez közelebbi, de valóban nem tisztán D# hangot hallhatunk a hangszerből, így a specifikáció helyes volt.

4.4 Akkordfelismerés ismertebb dalokból

A saját mintáimon felül természetesen fontos, hogy a rendszer már meglévő dalokból ki tudja-e olvasni az éppen játszott akkordot. Megjegyzendő, hogy a tesztet olyan dalrészletekre végeztem el, ahol a gitár mellett nincs, vagy csak nagyon minimális a hangszerelés az egyszerűsítés kedvéért. Ha egy kompletten meghangszerelt darab akkordjait szeretnénk felismerni, ahhoz komplex, megfontolt szűrést kell alkalmaznunk, hogy a legtöbb számunkra zavaró hangot kiszűrjük, és a hasznos információt vizsgálja csak az algoritmus. Ettől az iránytól dolgozatomban eltekintek.

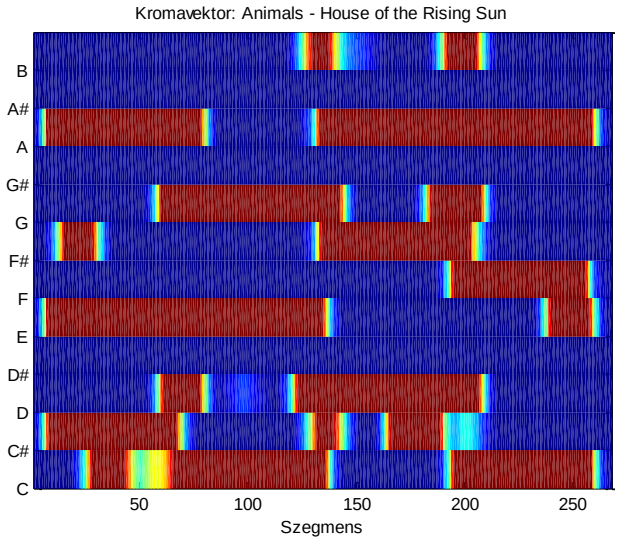
Első példaként egy új Scorpions dal kezdő akkordváltásait határoztam meg, ezek üres akkordok, amiket egy torzított gitár játszik, sorban: D#5, C#5, F#5, G#5. Az ábra:



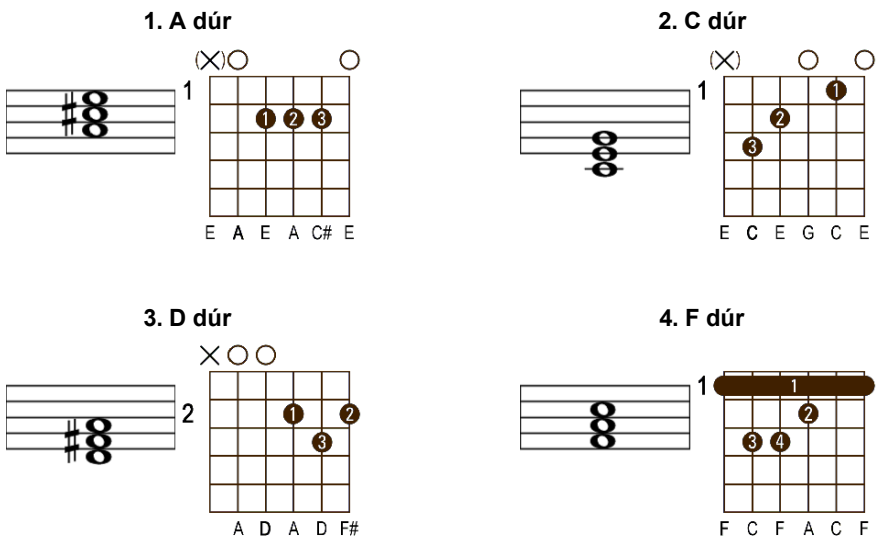
4.17. ábra: Scorpions – Raised on Rock kezdeti akkordjai

A felsorolt akkordok jól kivehetőek, néhány zavaró komponens jelen van (ezt pontosított szűréssel meg lehetne oldani), de az akkordok felismerhetőségét nem befolyásolják.

A másik és egyben utolsó tesztalanyom az Animals – House of the Rising Sun első négy akkordbontása. Itt szemléltetem egyben azt is, hogy akkordbontásból is meg tudja határozni bizonyos esetekben a rendszerem a keresett akkordot. Az ábrák:



4.18. ábra: Animals – House of the Rising Sun akkordjainak kromavektorja



4.19. ábra: Animals – House of the Rising Sun első 4 akkordja

Az első, ami az ábrákon szembetűnik, hogy megjelenik egy C# hang, ami megzavarja a felismerőt és A moll helyett A dúrt ismer fel. Emellett viszont a többi akkordra pontos a felismerés, hiába régi a felvétel, illetve nem egyszeri pengetés, hanem akkordbontás van a mintában.

5 Összefoglalás, fejlesztések

Összefoglalva az előző fejezeteket, a rendszer megvalósításához szükséges elmélet megismerése után bemutattam a rendszer egyes részegységeit, valamint működésük elvét. Végül egy nagyon átfogó teszteléssel, több, változatos mintával megmutattam, hogy az algoritmus alkalmazási területe szerint igen sokrétű, több hangszer akkordjainak felismerésére is alkalmas.

Bár a rendszer még fejlesztésre szorul, működése pontosnak mondható. Munkám során a „klasszikus" módszerekkel ellentétben nem a jel DFT-jéből állítottam össze a kromavektort, hanem a HPS, illetve egy implementált pontosítást végző algoritmussal oldottam meg a hangfelismerés problémáját, az eredmények jóval pontosabbnak bizonyultak, mint egyszerű DFT esetén. Az alaphangok felismerése után kiterjesztettem a rendszert több hangra egy saját szűrési metódussal. Végül pedig egy egyszerű összehasonlítási modellt alkalmazva meghatároztam a feltöltött kromavektor időbeli reprezentációjából, hogy milyen akkord szólhatott a bemeneti mintában.

Emellett viszont a következőkben bemutatom, hogy melyek azok a részei, amelyek fejlesztése esetén számottevő javulás érhető el.

Elsőként egy olyan algoritmus kifejlesztése, amely a mintának meg tudja határozni az ütemét, így pedig jó közelítéssel az akkordok szegmentálása megoldható. Ha külön tudnánk kezelni az akkordokat, nem pedig az egész akkordmenetet egyben, jóval pontosabb szűrést, és ebből következően pontosabb felismerést tudnánk végrehajtani. Emellett dinamikusan be tudnánk állítani az utolsó szűrést, ami után így biztosan nem szűrnénk ki egyetlen fontos akkordot sem.

Mint a 3. fejezet végén már említettem, a modellben csak egy egyszerű csend-detektálást implementáltam, így lényeges része lehetne a fejlesztésnek egy dinamikus, a vizsgált mintához legmegfelelőbb beállítású rendszer. Megvalósítása után a sokkal dinamikusabb (pl. klasszikus zene) hangminták akkordjainak felismerése is megoldhatóvá válna.

Szintén egy fontos pontja lenne még a fejlesztésnek, hogy az algoritmus

felismerje, hogy egyetlen hangról, üres akkordról, hármashangzatról, vagy másról van szó, hiszen ha egy dúrnál négy vagy csak két hangot érzékel, akkor téves lesz az érzékelés.

Bővítésre szorul az adatbázis is, elkészíteni a kromavektorait, a megjelenítendő képeit még több sablonnak, hogy többféle akkordot tudjon kezelni a rendszer.

Végül megoldandó a rendszer real-time implementációja. Viszonylag kevés változtatással el lehetne érni, hogy ne csak előre rögzített mintákat vizsgálhassunk, hanem egy interface-en keresztül valós időben játszott hangmintát adjunk a rendszer bemenetére. Az algoritmusok működése alapján ez megvalósítható, és előrelépés az akkordfelismerésben.

Mindezen felsorolt pontok ellenére a rendszer igen sokrétű, a szélsőséges mintákat, valamint pontos hangolású akkordmeneteket is tudja kezelni. A megvalósításakor nem a bonyolultságot, inkább az átláthatóságot, illetve egyszerűséget tartottam szem előtt, hogy könnyen szerkeszthető, fejleszthető legyen a rendszer. Emellett viszont ugyan némi korlátok között, de a kitűzött célt, az akkordok felismerését kitűnően megvalósítja.

Irodalomjegyzék

[1] Wikipédia, *Akkord*, http://hu.wikipedia.org/wiki/Akkord (2012. dec.)

[2] HyperPhysics: *Details about pitch*, http://hyperphysics.phy-astr.gsu.edu/hbase/sound/pitch.html (2012. dec.)

[3] Firtha Gergely: *Hangmagasság korrekciós rendszer létrehozása MATLAB környezetben*, Budapesti Műszaki és Gazdaságtudományi Egyetem, Híradástechnikai Tanszék, 2009.

[4] David Gerhard: *Pitch Extraction and Fundamental Frequency: History and Current Techniques*, University of Regina, Department of Computer Science, 2003.

[5] Adam M Stark: *Musicians and Machines: Bridging the Semantic Gap In Live Performance*, Centre for Digital Music, Department of Electronic Engineering and Computer Science, Queen Mary, University of London, 2011.

[6] Fiala Péter: *Hangszerek fizikája,* Jegyzet, Budapesti Műszaki és Gazdaságtudományi Egyetem, Híradástechnikai Tanszék, 2010.

[7] Kyogu Lee, Malcolm Slaney: *Automatic Chord Recognition from Audio Using an HMM with Superwised Learning*, Center for Computer Research in Music and Acoustic, Department of Music, Stanford University, 2006.

[8] Nanzhu Jiang, *An Analysis of Automatic Chord Recognition Procedures for Music Recordings*, Saarland University Faculty of Natural Sciences and Technology I, Department of Computer Science, 2011.

[9] Freud Róbert, *Lineáris algebra*, ELTE Eötvös Kiadó, 2005.

[10] Wettl Ferenc, *Lineáris algebra*, Budapesti Műszaki és Gazdaságtudományi Egyetem, Természettudományi Kar, 2011.