

TDK dolgozat

Almási Dénes Attila

SZEGEDI TUDOMÁNYEGYETEM
TERMÉSZETTUDOMÁNYI ÉS INFORMATIKAI KAR
INFORMATIKAI TANSZÉKCSOPORT
SZÁMÍTÓGÉPES OPTIMALIZÁLÁS TANSZÉK

Automatikus kottalapozó alkalmazás megvalósítása

Almási Dénes Attila
Programtervező informatikus MSc szakos hallgató
Almasi.Denes.Attila@stud.u-szeged.hu

Témavezető: Dr. Csendes Tibor egyetemi tanár

2014

KIVONAT:

Számos hangszer esetében problémát jelenthet a zenész számára a kotta lapozása oly módon, hogy az megfeleljen a játszott műben történő előrehaladásnak. Jelen dolgozat célja egy, a jelfeldolgozás és a mesterséges intelligencia eszközeit felhasználó szoftver fejlesztésének bemutatása. A program képes egy rövid tanulási folyamat után pusztán akusztikus jelek segítségével tájékozódni a hallott zeneműben, megjeleníteni a kottát a zenész számára, és a megfelelő pillanatokban külső segítség nélkül lapozni a kotta digitális képként tárolt változatát a zenész játéka során.

Kidolgoztunk egy jelfeldolgozási módszert, amely mellett, hogy kiemeli a hang spektrumából a zeneiségben fontos információkat, lehetőséget nyújt távolságfüggvények egyszerű módon történő definiálására. Egy ilyen távolságfüggvény segítségével következtetünk két jel zenei értelemben vett hasonlóságára, továbbá megfeleltetést hozunk létre jelsorozatok között dinamikus idővetemítések egymást követő alkalmazásával. Az utolsó néhány idővetemítés illesztésének arányából számított mozgó átlag segítségével megbecsüljük, hogy a zenész játéka során egy adott pillanatban a mű mely részénél jár egy korábbi előadásához viszonyítva, lehetővé téve, hogy bármely időpontban a kotta megfelelő oldalát mutassuk számára.

Sikereket értünk el klasszikus és modern művek lapozásánál zongora- és orgonajáték esetében. Számos egyéb teszt, többek között partitúra lapozási kísérlet is várat még magára.

Az alkalmazott távolságfüggvények, a dinamikus idővetemítés különböző, lokális és globális megkötések tartalmazó variánsa, további jelfeldolgozási módszerek kipróbálása újabb kutatásoknak adhat teret.

KULCSSZAVAK: *Zene, kotta, jelfeldolgozás, spektrum, dinamikus idővetemítés*

Tartalomjegyzék

Bevezetés	1
1.1. Az automatikus lapozás feladata	1
1.1.1. Jelenlegi megoldások	2
1.1.2. A kifejlesztett megoldás	3
A zene lejegyzése, fizikája	4
2.1. A kotta időbelisége	4
2.2. A zenei hang fizikai jellemzése	5
Az algoritmus	7
3.1. Előfeldolgozás	7
3.1.1. Bemenet	7
3.1.2. Kimenet: cent-rekeszelt energia spektrum	8
3.1.3. Rövidtávú diszkrét Fourier-transzformáció	9
3.1.4. Az előfeldolgozó eljárás	10
3.2. Tanítás	20
3.3. Lapozás	20
3.3.1. Dinamikus idővetemítés	21
3.3.2. A megfelelő kottalap mutatása	24
A megvalósítás	27
Eredmények, további lehetőségek	29
Bibliográfia	34

Bevezetés

A zene fejlődéstörténetét vizsgálva már az ókortól fogva találhatunk olyan leleteket, amelyek zeneművek lejegyzésének korai kísérleteit testesítik meg^[1,2]. Az igény, hogy melódiákat, frázisokat, dalokat rögzítsünk, máig nem változott, azonban a lejegyzés formája rendkívül sokat fejlődött, mire eljutottunk ahhoz, amit ma kottának nevezünk. Napjainkban jellemzően papíron rögzítik, és a legtöbb mű hossza több oldalon átívelő leiratok létrejöttét követeli meg. A kotta a zenészek számára kulcsfontosságú lenyomatát őrzi a zeneműveknek, amelynek segítségével képesek megtanulni, reprodukálni azokat. Használata történhet mind a művek tanulása során, mind pedig az előadásuk alkalmával.

A hangszerek túlnyomó többségének megszólaltatásához, a rajtuk való játékhoz a művészek mindkét kezét használnia kell. Ebből adódóan gondot jelenthet a kotta lapozása a játék során. Noha a legtöbb zenész számára nem okoz problémát a művek memorizálása és a fejből történő játék, mégis – eltérő okokból ugyan – számos alkalommal láthatjuk, hogy előadásokon, koncerteken kottát használnak. Ezekben a helyzetekben a lapozást különféle módokon oldják meg: Zenekari művek esetében gyakran lehetséges az egyes szólamok kottáinak oly módon való tördelése, hogy az adott szólamnak éppen ne legyen semmiféle teendője az oldalak végeinek közelében. Ekkor a muzsikusz kézzel lapoz, önmagának. Előfordulhat, hogy az előadó megkér egy zeneértő embert (leggyakrabban egy kollégát, tanárt, tanítványt), hogy a megfelelő pillanatokban lapozzon. A lapozás feladatának ezen megoldására a legtöbb esetben billentyűs hangszereknél láthatunk példát. A karmesterek jellemzően maguknak lapozzák a partitúrát, ugyanis az előadás szempontjából kevesebb problémát jelenthet, ha egy rövid időre az egyik kezük nem a vezénnyelssel foglalkozik.

Az utóbbi években megjelentek különböző számítógépes megoldások, amelyek speciális hardverek, pedálok segítségével lehetővé teszik a kotta digitalizált formáinak lábbal történő lapozását^[3]. Ilyenkor a kottát általában egy táblagép jeleníti meg. A módszer előnye, hogy a lapozás külső segítség nélkül történik, a zenész megszakítás nélkül haladhat előre a műben, függetlenül annak tördelésétől. Kevésbé egyértelmű ugyanakkor a módszer azon hátránya, amely szerint a muzsikusz figyelmét ki kell, hogy terjessze a játék során a pedál használatára. Ez a feladat nehéznek bizonyulhat olyan előadók számára, akik a több éven át folytatott zenei tanulmányaik során hozzászoktak, hogy a hangszeres játék során csupán a kezüket kell, hogy használják. Azon hangszerek esetén, amelyek megkövetelik a láb használatát is (dobok, orgona), ez a módszer egyértelműen kudarcot vall.

1.1. Az automatikus lapozás feladata

A számítógépek feldolgozási sebességének, kapacitásának növekedésével, a jelfeldolgozás és a mesterséges intelligencia fejlődésével érdemessé vált annak vizsgálata, hogy lehetséges-e a kottalapozás feladatának teljes automatizálása. A probléma ekképpen történő megoldásának

alkalmazásakor a kottát számítógép jeleníti meg és lapozza, azonban az előbbi eljárással ellentétben ez az előadó bárminemű beavatkozása nélkül történik.

Bármiféle automatizált megoldás fejlesztése számos olyan kérdést felvet, amelyek megválaszolása nagyban befolyásolja a kifejlesztett módszer alkalmazhatóságát. A két legfontosabb tisztázandó kérdés a következő:

- **A digitalizált kotta formátuma.** Elképzelhetőek olyan megoldások, amelyek megkövetelik, hogy a kotta digitalizált változata speciális, a feladat végrehajtásához az alkalmazott módszer szempontjából elengedhetetlen információkat tartalmazzon. Ezek lehetnek többek között különféle időmegjelölések, akusztikus modellek, MIDI szekvenciák. Olyan megoldások készítése is lehetséges, amelyek bármilyen képi formátumú kotta lapozását megengedik (akár kéziratokét is), megkötések, különleges metainformációk használata nélkül.
- **Tanítás szükségessége.** Minden bizonnyal a legtöbb megoldási mód valamelyik ponton alkalmazza a gépi tanulás tanítópéldákon és teszteseteken alapuló módszertanát. Érdeemi különbség lehet azonban a végeredmény szempontjából, hogy a tanítást a felhasználónak, vagy a fejlesztőknek kell-e elvégezniük.

Mielőtt megismernénk, hogy a jelen dolgozatban bemutatásra kerülő alkalmazás hogyan viszonyul a fenti két tényezőhöz, tekintsük át, hogy jelenleg milyen automatizált kottalapozási megoldások érhetőek el a felhasználók számára!

1.1.1. Jelenlegi megoldások

Viszonylag kevés teljesen automatizált megoldás ismeretes ezidáig. Két ilyen alkalmazás a *Tonara*^[4] és az *AutoFlip*^[5]. Mindkét program az *Apple iPad* táblagépekre készült.

A Tonara rendkívül széleskörű eszköztárat nyújt a felhasználó számára, úgy mint: automatizált lapozás, a kottában való pontos helyzet megjelenítése, metronóm, statisztikák kimutatása az előadásmódról, a mű belső szintetizátorral történő megszólaltatása, hangfelvétel készítése. A szoftver csupán olyan kották lapozására képes, amelyek a fejlesztő által fenntartott boltból érhetőek el. A kottáikat nem képi formátumban tárolják, valamint ellátják őket számos annotációval. A Tonara zárt forrású szoftver, az általa alkalmazott technológia pedig nem publikus. Így csupán valószínűsíthető, hogy a megoldásukban a letölthető kottacsomagok előre felvett akusztikus modelleket is tartalmaznak, lehetővé téve a program számára a tájékozódást a hallott műben anélkül, hogy a felhasználónak előre el kellene játszania a műveket a program tanításának céljából.

Az AutoFlip tetszőleges, *pdf* formátumban tárolt kotta lapozására alkalmas szoftver. A lapozáshoz a jelen dolgozatban ismertetett módszerhez hasonló eljárást fejleszthettek ki, azonban a program szintén zárt forrású, az általuk használt algoritmusról nincsen elérhető információ. A dolgozat írásának idejében a szoftver utolsó frissítését 2012 márciusában kapta, a fejlesztő

weboldala 2011-ben frissült utoljára. Az Apple alkalmazásboltjában található számos pozitív értékelés ellenére valószínűsíthető, hogy a program fejlesztését felfüggesztették.

1.1.2. A kifejlesztett megoldás

Az itt bemutatott automatikus kottalapozó megoldás képes tetszőleges képi formátumban (*jpg*, *png*, *bmp*, stb...) tárolt kották lapozására. Az ismertett eljárás gond nélkül kiterjeszthető más reprezentációkban való tájékozódásra is, így például *pdf* dokumentumok vagy kottaként megjelenített *MIDI* szekvenciák lapozására is alkalmas lehet.

A lapozás végrehajtásához a felhasználó egyszeri alkalommal történő tanítójátéka szükséges, amely során a program számára jelezni kell, hogy a hallott műhöz képest mely időpontokban kell lapozni. A program az őt futtató eszköz mikrofonja vagy vonalbemenete segítségével a játék során figyeli a zene (és a környezet) akusztikus jeleit, és ebből valós időben épít fel egy *cent-rekeszelt energia spektrum* elnevezésű *akusztikus modellt*. A felhasználó megfelelő pillanatokban történő, lapozási igényt kifejező interakcióit a szoftver az akusztikus modell adott pillanathoz tartozó információihoz társítja. Miután a tanítás lezajlott, a felhasználó kezdeményezheti egy *lapozási menet* indítását. Ekkor a program megjeleníti a kotta első oldalát, és megkezdődhet az automatikus lapozási folyamat.

A zene lejegyzése, fizikája

Az algoritmus bemutatása előtt ismerjük meg a kotta tárgyalásunk szempontjából lényeges vonásait, továbbá a zenei hangok fizikai jellemzésével kapcsolatos legfontosabb fogalmakat!

2.1. A kotta időbelisége

A konvencionális, ötvonalas kotta a nyugati zenei rendszerekbe illeszkedő művek lejegyzésére alkalmas írásmód. A zene egy közelítőleges, esemény alapú reprezentációja, amely elsősorban a ritmika és a hangmagasságok megőrzését, másodsorban pedig a tempó, a dinamika, díszítések és egyéb, az előadásmódot szabályozó szubjektív utasítások rögzítését támogatja^[6,7]. A ritmus és a hangmagasságok leírásának szigorú volta miatt kevés teret ad az improvizációra, azonban a játék sebességére és a hangerőre vonatkozó utasítások interpretációjában nagy szabadságot hagy az előadóra.

Míg egy mű eljátszása során a megszólaló hangok együtthangzása és egymásutániséga kötött, addig időtartamuk csupán egymáshoz viszonyítva rögzített. Tehát az egyes zenei események manifesztációi önmagukban bármennyi ideig tarthatnak, mindössze az számít, hogy a mű többi részéhez mérten mennyi időt ölelnek fel. Ekképpen a zenei események egymásutániséga csak egy relativisztikus időkeretben értelmezhető. Ezen időkeret a mű elejétől a végéig tart, és különböző egységekre szokás bontani. A mű egészétől az apróbb részletek felé haladva számos időbeli tagozódás ismeretes, amelyek nem feltétlenül hierarchikus rendszert alkotnak^[8]. Ilyenek lehetnek a teljesség igénye nélkül: *tételek, periódusok, frázisok, ütemek*. Az ütemek kulcsszerepet játszanak a tagolásban – míg a felsorolt többi szerkezeti egység meghatározása sokszor önkényes, addig az általánosan kijelenthető, hogy a zene (eltekintve néhány különleges kivételtől) egyértelmű módon ütemekre bontható. Egy ütem hosszát egységnyiinek, egésznek tekintjük a zene relativisztikus időkeretében.

Azon kérdés megválaszolása, hogy egy mű viszonylagos időbeli rendszerét hogyan abszolutizáljuk, az előadó szabadságát gyarapítja. Ezt a fajta abszolutizálást a mű egy adott pillanatában annak *tempója* jellemzi, amely meghatározza, hogy egy percen belül hány negyed ütem hosszúságú esemény történhet a zenében. A tempó egyetlen előadás során is változó lehet: számos esetben éppen a kotta utasítja az előadót annak megváltoztatására, valamint az is elképzelhető, hogy a zenész bármiféle lejegyzett utasítás nélkül is módosítani kíván rajta a játék során.

Fontos megjegyezni, hogy a kotta nem feltétlenül szekvenciálisan határozza meg a zenei események egymásutániségát. A programozási nyelvek vezérlési szerkezeteivel analóg módon különféle utasítások befolyásolhatják a lejátszandó hangok sorrendjét. A legegyszerűbb ilyen nemszekvenciális viselkedést leíró szerkezet az ismétlés. Az ismétlés bizonyos számú (ismétlődőjelek közé foglalt) ütemek kétszer egymás után történő eljátszására utasítja a zenészt. Ezen túlmenően nagyszámú egyéb, sokszor szövegesen adott, nemszekvenciális végrehajtást meghatározó utasítás létezik^[9]. Némelyik kifejezetten bonyolult, mint például: „*ugrás a kotta elejére*,

majd onnan ismétlések nélküli játék a fine jelzésig” (Da capo al fine senza ripetizione).

A jelen dolgozat keretében leírt eljárás közvetlenül nem támogatja ezen, kottán belüli navigációt előíró utasítások feldolgozását, értelmezését. A felhasználó azonban minden esetben megadhatja a kottalapok olyan (akár ismétléses) sorrendjét, amely megfelelő lesz a lapozás szempontjából.

2.2. A zenei hang fizikai jellemzése

Az ember által érzékelt hang a légnyomás lokális, hullámszerű ingadozásaiból ered. A nyomás ingadozását valamilyen test mozgása, rezgése, alakjának kellően gyors változása idézi elő. Amennyiben ez a fluktuáció periodikus komponenseket hordoz magában, úgy a keletkezett hangot zenei hangként, ellenkező esetben zörejként azonosítjuk^[10]. A továbbiakban elsősorban a zenei hangok elemzésére szorítkozunk.

Minden hang valamilyen forráshoz köthető, amelynek rezgése kiváltotta azt. A hangforrás milyensége nagyban befolyásolja a keletkezett hanghullámok frekvenciatartalmát. A hang létrejöttéhez annak kívánt forrásával energiát kell közölnünk. Egy hangforrás lehet egy húr, amelynek megpendítésével energiát adunk át neki, lehet egy félig zárt síp, amelybe levegőt fújunk, és a ki- és beáramló levegő mozgása rezgésbe hozza azt, vagy lehet például az emberi hangszallag, amelyet a közötte áthaladó levegő késztet rezgő mozgásra. A hangforrás, annak fizikai tulajdonságaiban eltérő részei, valamint az azt körülölelő közvetítő közeg (zömmel a levegő) együttesen egy *akusztikus rendszert* alkot. Egy akusztikus rendszer felépítéséből, szerkezetéből adódóan bizonyos frekvenciákon való rezgésével lényegesen hatékonyabban képes a vele közölt energiát a közvetítő közegen keresztül leadni. Ezekon a frekvenciákon az akusztikus rendszer *rezonál*^[11]. Általában egy rendszer több ilyen frekvenciával rendelkezik, amelyeket *rezonanciafrekvenciának* nevezünk. A rezonanciafrekvenciák jellemzően a legnagyobb energialeadást biztosító, legerősebb rezonanciafrekvencia többszörösei. A legerősebb ilyen frekvenciát (amely sok esetben a legalacsonyabb is egyúttal) *alaphangnak*, a többszöröseit pedig *harmónikusoknak*, *felharmónikusoknak* nevezik.

Általánosan kijelenthető, hogy az emberi fül egy zenei hangot hallva annak alaphangját a hang magasságának érzéséhez társítja, míg a felharmónikusok intenzitásának aránya a hangszín meghatározásában játszik fontos szerepet. Kiemelendő, hogy egy hang megszólalása során mind az alaphang, mind a felharmónikustartalom csupán időben lokálisan tekinthető állandónak. Egy hangszeren való hang megszólalását négy egymást követő fázisra^[12] szokás osztani: a hang megszólalásának első pillanatai, a hang tartós rezgése, elhalkulása, továbbá teljes megszűnése¹. A felharmónikusok erőssége más és más függvény szerint alakulhat mind a négy fázisban, ezenfelül az egyes fázisok hossza is változó lehet. Ezek a tényezők jellemzően egyetlen hangszer hangját vizsgálva is függenek az alaphang magasságától, hozzájárulva a különböző zeneszerszámok hangszíneinek rendkívüli sokszínűségéhez.

¹Ezeket az angol nyelvű szakirodalomban rendre *attack*, *sustain*, *decay* és *release* nevekkel illetik.

Az ember által érzékelt hangmagasság koncepciójának pontos definiálása a pszichoakusztika tárgykörébe tartozik. Amíg e terület bővelkedik a fogalmat aspektusok sokaságából vizsgáló, azt precízen meghatározó művekben^[13,14], kérdéskörünk számára elégséges pusztán egy egyszerű összefüggés rögzítése: Az ember lineáris hangmagasság-érzete az alaphang frekvenciájának közelítőleg logaritmikus függvénye^[15]. Ezen felismerésnek megfelelően definiálható a $d_{cent} : \mathbb{R}^+ \times \mathbb{R}^+ \rightarrow \mathbb{R}_0^+$ függvény, ahol adott f_1 és f_2 frekvenciák esetén $d_{cent}(f_1, f_2)$ megadja az f_1 és f_2 frekvenciák érzékelésünk szempontjából közel lineáris mennyiségként felfogható hangmagasság-különbségét. A függvény a következő összefüggés által definiáltatik:

$$d_{cent}(f_1, f_2) = 1200 \cdot |\log_2(f_1/f_2)|.$$

A d_{cent} függvény két frekvencia zenei *centben* mért távolságát határozza meg. A zenei cent a zenészek, zenetudósok számára egyaránt pontos jelentéstartalommal bíró mértékegység: 1200 cent távolsága pontosan egy tiszta oktávot definiál – ekkor az egyik frekvencia éppen a kétszerese a másikénak. A legtöbb esetben a félhangok közötti távolságokat pontosan 100 centként határozzák meg. Így a 12 félhangra osztott oktávban a félhangok ezen mérték szerint egyenletesen, éppen 100 centenként helyezkednek el. A félhangok oktávon belüli relatív elhelyezésének kérdését *temperálásnak*, az ekképpen történő elhelyezésüket pedig *egyenletes temperálásnak* nevezik. Az egyenletes temperálás a nyugati zenei rendszerek leggyakoribb temperálási formája.

d_{cent} definíciója alapján meghatározható egy $up(f, d)$ és egy $down(f, d)$ függvény, amelyek megadják egy f frekvenciától d cent távolságra található magasabb és alacsonyabb frekvenciákat:

$$up(f, d) = f \cdot 2^{d/1200}, \quad down(f, d) = \frac{f}{2^{d/1200}}.$$

A dolgozat további részében élek azzal a feltételezéssel, amely szerint a bemenetként kapott zene egyenletesen temperált, a zongora negyedik oktávjában található A hang (A4) pedig 440 Hz-re hangolt. Az ismertetett eljárás megfelelő paraméterezéssel bizonyos határokig működhet más fajta temperálások, illetve az A4 hang frekvenciájának 440 Hz-től eltérő megválasztása mellett is. Az algoritmus leírásának egyszerűsítése érdekében azonban célszerű élni az iménti feltételezésekkel.

A következő szekcióban bemutatásra kerül a lapozást megvalósító algoritmus működése.

Az algoritmus

A módszer két lépésből épül fel. Mindkét lépés online módon szerzi bemeneteit. Az első szakasz a tanítás, amely során akusztikus jelből kivont információkat és lapozási eseményeket rögzítünk. A második szakasz a lapozás, amikor az akusztikus jelből hasonló módon kinyert adatokat a tanítás alkalmával rögzítettekhez illesztjük, és ez alapján a megfelelő pillanatokban lapozunk.

A két fázis azonos jelfeldolgozási eljárással vonja ki a bemenet megoldáshoz szükséges lényeges sajátosságait (*features*). Tekintsük át ezen előfeldolgozási eljárás lépéseit, kezdve a bemenet és a kimenet specifikációjával.

3.1. Előfeldolgozás

3.1.1. Bemenet

A bemenet két független csatornán (sztereó) érkező *LPCM* (*Linear Pulse Code Modulation*) jelsorozatból áll. Az *LPCM*^[16,17] a digitális hangrendszerekben szabványosnak mondható audió ábrázolási mód. Ezt a reprezentációt választva a bemenet formátumának megadjuk annak lehetőségét, hogy a legtöbb konvencionális digitális audió rendszerhez illeszkedő, a fogyasztói eszközök által nyújtott környezetben könnyebben implementálható algoritmust hozzunk létre.

Az *LPCM* jelek egy adott s mintavételezési gyakoriságnak megfelelő, időben egyenletesen elosztott t időpontokhoz társítanak x_t értékeket. Egy x_t érték közelítőleg, kvantált ábrázolása egy akusztikus jel amplitudójának a t időpillanatban. (Ez lehet például mikrofonba beérkező hanghullámok összességének amplitúdója – a mikrofon membránjának kitérése annak nyugalmi állapotából.) Az *LPCM* meghatározásának lineáris mivoltából adódóan az x_t értékek alakulása a hang amplitudójának lineáris függvénye. A mintavételezési gyakoriság azt szabályozza, hogy egy másodpercen belül hány alkalommal történik az akusztikus jel rögzítése.

Az eljárás a bemenetet online módon olvassa be. Ennek megfelelően képezhetünk a bemenetként kapott *LPCM* értékekből tetszőleges, w hosszúságú X jelsorozatokat, hiszen bármekkora darabokban érkezzenek a jelek az algoritmus számára, mindig összevárhatunk azokból w darab értéket. Amennyiben több, mint w darab érkezik, úgy az értékeket tagolhatjuk w hosszúságú számsorozatokká, eltekintve valamekkora esetleges maradék résztől, amelyeket az újonnan érkező jelek elé helyezhetünk. Ezzel a lépéssel konstans mennyiségű késleltetéssel terheljük a feldolgozást, azonban a legtöbb konvencionális mintavételezési gyakoriság mellett ($44\,100\text{ Hz}$, $48\,000\text{ Hz}$, $96\,000\text{ Hz}$) akár $w \propto 10^3$ választással is csupán jelentéktelen mértékű késéssel kell számolnunk. A késés mértékét az $r_1 = s/w$ formula adja meg, másodpercekben. Figyelembe véve, hogy r_1 pusztán az algoritmus paramétereitől függ, a bemeneteitől pedig független, az eljárás későbbi részei akár a késést korrigáló lépéseket is tehetnek az érték ismeretében.

A továbbiakban egy, a t időpontban kezdődő, w hosszúságú *LPCM* jelsorozatra $X^{(t)}$ -ként

hivatkozom. Ekképpen a bemenet egy nem átfedő sorozata megadható $X^{(0)}, X^{(w)}, X^{(2w)}, \dots$ sorozatként. Egy $X^{(t)}$ input tömb i -edik elemére $X_i^{(t)}$ -ként hivatkozom. ($0 \leq i < w$) Az eljárás fejtegetésének további részeiben – amennyiben külön nem jelölöm – egy adott pillanatot mindig egy, a mintavételezési gyakoriság időfelbontásában értendő, $t = kw$ ($k \in \mathbb{N}$) időponthoz társított, w elemet tartalmazó, a bemenet többi részével nem átfedő $X^{(t)}$ tömb, illetve annak tartalma jellemző. Tehát a feldolgozás szempontjából különbözőnek tekintett időpontok távolsága legalább w mintavételezést foglal magába, és az egyes események között w többszöröseinek megfelelő számú mintavételezés történik a bemeneti hangcsatornában.

3.1.2. Kimenet: cent-rekeszelt energia spektrum

A tanulás és a lapozás során begyűjtött akusztikus jelek egymáshoz való illesztése során kulcsfontosságú, hogy olyan információkkal dolgozzunk, amelyek a bemenet zeneiségét igyekeznek megragadni. A zeneiség kétségtől az emberi érzékeléshez köthető fogalom, így kézenfekvőnek bizonyult olyan információtartalmat keresni a hangban, amely az emberi muzikalitásnak megfelelően egyértelműen különbözteti meg az eltérő hangzatokat, és azonosítja a hasonlókat. Az előállított reprezentáció megalkotása során éltem azon heurisztikus feltevessel, miszerint az ember a zenei minták hasonlóságát sokkal inkább a bennük foglalt hangok relatív idő- és magasságbeli váltakozásainak egyezése szerint határozza meg, mint azok dinamikai, hangszínbeli hasonlósága, vagy például a hangforrások térbeli elhelyezkedésének közelítő egyezése alapján.

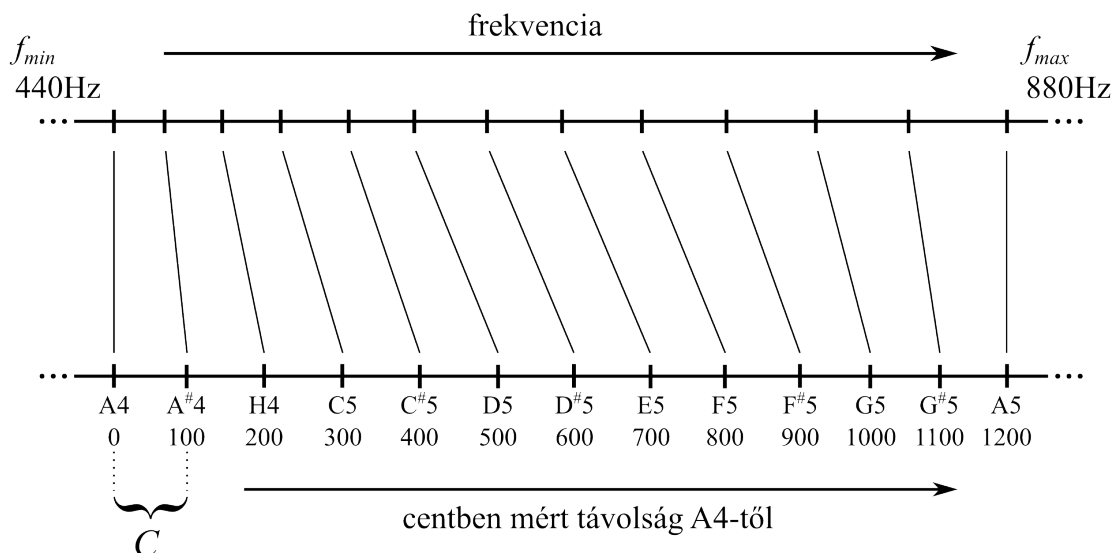
A kifejlesztett reprezentáció a *cent-rekeszelt energia spektrum* (CRES, *cent-binned energy spectrum*). Ez az LPCM bemenet minden w hosszú darabjához egy n dimenziós feature vektort rendel, amely közelítőlegesen meghatározza, hogy az adott pillanatban két f_{min} és f_{max} referenciafrekvencia közé szorított, centekben mérve konstans, C szélességű, n darab rekesz között hogyan oszlott meg a bemeneti jel energiaszintje. Egy Y CRES vektor elemeit CRES együtthatóknak nevezem. A j -edik CRES együttható jelölése Y_j , ahol $0 \leq j < n$.

Az f_{min} és az f_{max} közötti frekvenciák viszonyulását a CRES n darab rekeszéhez az 1. ábra mutatja be. Figyelembe véve, hogy tetszőleges f_{min} és f_{max} frekvenciák között nem feltétlenül egész számú C cent szélességű rekesz található, n értéke a következő formula szerint alakul:

$$n = \left\lceil \frac{d_{cent}(f_{min}, f_{max})}{C} \right\rceil$$

A CRES egy adott időpontban lokálisnak mondható energia spektrum adatait a zenei cent skálához rendelve ábrázolja. Adott j -re Y_j a spektrum $[up(f_{min}, jC), up(f_{min}, (j+1)C)]$ sávjába beleeső frekvenciáinak aggregált energiaszintjét adja meg.

Megfelelő f_{min} , f_{max} és C választással az együtthatókból egyenesen következtethetünk az adott pillanatban megszólaló hang zenei tartalmára. f_{min} megválasztása során célszerű ügyelni arra, hogy a frekvencia a lehető legtávolabb essen két zenei hangtól, egyenletes temperálás esetén éppen 50 cent-tel. A rekeszek centben adott méretét (C) jó választás lehet 100-nak állítani – így minden CRES együttható pontosan egy félhang energiaszintjéről hordoz informáci-



1. ábra: A CRES rekeszek viszonyulása a spektrum frekvenciáihoz $f_{min} = 440$, $f_{max} = 880$ és $C = 100$ értékek mellett

ót. f_{max} értéke bármilyen magas frekvencia lehet, amelyről állítható, hogy az annál magasabb hangok már nem lehetnek lényegesek az alkalmazás szempontjából. Egy lehetséges választás $f_{max} = 9600\text{ Hz}$ lehet. Ez a frekvencia jócskán nagyobb a legtöbb zeneműben legmagasabbként előforduló hang alapfrekvenciájánál.

A CRES a beszédfeldolgozásból ismeretes *MFC (Mel-frequency cepstrum)*^[18] audió ábrázolási móddal rokonítható, azonban míg az MFC kepsztrális jellegű együtthatók sorozatából áll, illetve a bemeneti jel frekvenciaterét a mel-skálára vetemíti, addig a CRES a bemeneti jeltől származtatott energia spektrum együtthatóinak egy részhalmazát vetemíti egy zenei jelentéstartalommal bíró skálára, majd a transzformált együtthatókat kepsztrális átalakítás nélkül aggregálja. A CRES nem köti meg sem ez utóbbi aggregáció módját, sem pedig, hogy előállítása során a bemeneten milyen időbeli vagy spektrális szűréseket alkalmazunk.

3.1.3. Rövidtávú diszkrét Fourier-transzformáció

A CRES együtthatók meghatározásához feltétlenül szükséges a bemenet időben lokális energia spektrumának meghatározása. Ez az input egy alkalmas szakaszolásának darabonkénti diszkrét Fourier-transzformációjával (*DFT*) állítható elő. A DFT ekképp történő sorozatos alkalmazását *Short-time Fourier Transformation (STFT)* névvel illeti a jelfeldolgozó szakirodalom. Az STFT eredménye egy mátrix, amelynek oszlopindexei időpontokhoz, sorindexei frekvenciarekeszekhez köthetők.

Az *STFT* megfelelő alkalmazása érdekében a diszkrét Fourier-transzformációk nem csupán az egyes szakaszokon, hanem azokon, és az őket megelőző k ($k \in \mathbb{N}_0^+$) szakasz együttesén történik. Ezzel megakadályozható, hogy valamely, a szakaszhatárok közelében jelenlévő frekvenciakomponensek helytelenül, vagy ne jelenjenek meg a transzformáció kimenetében. Az *STFT* előállításához szükséges szakaszolás sűrűségének megválasztására kézenfekvő választás

az LPCM jel korábbi, w szélességű darabolásának felhasználása. Ilyen módon egy t időponthoz kapcsolható egy $\mathcal{X}^{(t,k)}$ vektor, amely a t időpont $X^{(t)}$ inputjából és az előző k darab hasonló input-tömb megfelelő sorrendű konkatenációjából állítható elő:

$$\mathcal{X}^{(t,k)} = X^{(t-k+1)} X^{(t-k+2)} \dots X^{(t)}.$$

A *DFT* eljárás bemeneteként átadandó $\mathcal{X}^{(t,k)}$ vektorok szakaszolt mivolta hamis spektrális komponensek megjelenését okozhatja, amennyiben nem alkalmazunk valamilyen, a jelfeldolgozásból ismeretes *ablakfüggvényt* a transzformáció során. Ezt a jelenséget spektrális szivárgásnak (*spectral leakage*) nevezik. A hamis komponensek létrejöttének oka, hogy a transzformált LPCM jel véges hosszúságú, és annak a bemeneti tömb széleinél történő hirtelen megszűnése is feldolgozásra kerül a Fourier-transzformáció által.

A jelfeldolgozásban számos, ezen probléma kezelését célzó *ablakfüggvény* ismeretes, úgy mint: *rectangular*, *Hann*, *Hamming*, *Blackman*, *Blackman-Harris*, *Parzen*, *Nuttall*. Ezen függvények általában valamilyen $\text{win} : \mathbb{R} \rightarrow [0, 1]$ leképezés által definiálhatók. Kiköthető továbbá, hogy $\forall x : (x < 0 \vee 1 < x)$ esetén $\text{win}(x) = 0$. Egy *ablakfüggvény* alkalmazása a diszkrét eseményekre tagolt bemeneti jel és a függvény egy alkalmas mintavételezésének elemenkénti szorzatának előállítását jelenti. A $\text{win}(x)$ *ablakfüggvény* egy q ($q \in \mathbb{N}^+, 1 < q$) hosszú mintavételezését a win^q vektor, ennek i -edik elemét pedig win_i^q jelöli. win^q a következőképp állítható elő:

$$\text{win}_i^q = \text{win}\left(\frac{i}{q-1}\right) : 0 \leq i < q.$$

Az előbbi ismeretében az *ablakfüggvény* alkalmazása egy kiterjesztett $\mathcal{X}^{(t,k)}$ bemenetre a következőképp történik:

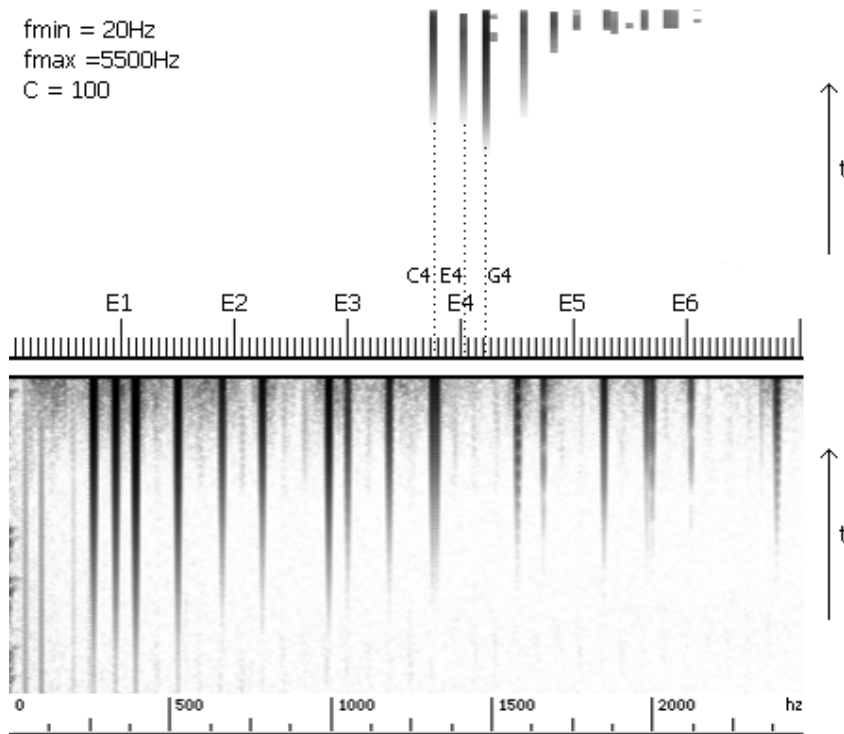
$$\hat{\mathcal{X}}_i^{(t,k)} = \text{win}_i^{kw} \cdot \mathcal{X}_i^{(t,k)} : 0 \leq i < kw.$$

A jelfeldolgozó szakirodalomban elterjedt *ablakfüggvények* különböző módon befolyásolják az STFT eljárás kimenetét. Általánosan kijelenthető, hogy az *ablakfüggvény* megválasztása a spektrális kép által ábrázolható dinamikai tartomány (amplitúdók) nagysága és a hasonló frekvenciák megkülönböztethetősége közötti kompromisszum megkötésével jár. Alkalmazásunk szempontjából ezen két tényező között az egyensúlyt a legjobban a szűksávú alkalmazások (pl. telefónia) során gyakorta alkalmazott *Hann* és *Hamming* ablakok, illetve az azokhoz hasonló *Blackman*, *Nuttall*, *Blackman-Nuttall* *ablakfüggvények* teremthetik meg.

A CRES és az STFT egymáshoz való viszonyulását szemlélteti a 2. ábra.

3.1.4. Az előfeldolgozó eljárás

Egy t pillanathoz tartozó w hosszúságú $X^{(t)}$ bemenethez társított $Y^{(t)}$ CRES vektor előállítása általánosan a következő módon történik:



2. ábra: Egy zongorán játszott C4-E4-G4 hármashangzat CRES (fent) és STFT (lent) elemzése.

- Az utolsó k LPCM input tömbből konkatenációk sorozatával előállítjuk az $\mathcal{X}^{(t,k)}$ kiterjesztett input tömböt.
- Előállítjuk $\mathcal{X}^{(t,k)}$ diszkrét Fourier-transzformáltjának frekvenciarekeszekhez tartozó magnitúdóértékeit, amelyet $\mathcal{F}^{(t,k)}$ -val jelölünk. Ekkor $\mathcal{F}^{(t,k)}$ egy $m = 1 + kw/2$ elemű vektor, amelynek valamely i ($0 \leq i < m$) eleme az $\mathcal{X}^{(t,k)}$ által hordozott jel alábbi frekvenciasávjának energiaszintjét jellemzi (Hertzben):

$$\left[i \frac{s}{2m}, (i+1) \frac{s}{2m} \right].$$

Az eredményként kapott DFT együtthatók mindegyike egy $\frac{s}{2m} Hz$ széles frekvenciatartomány energiaszintjét jellemzi.

- A spektrum f_{min} és f_{max} által definiált tartományát centskálára vetemítjük.
- A kapott cent-spektrumot n darab, egymás után elhelyezkedő, C hosszúságú rekeszre bontjuk. (A vektor végén esetlegesen előálló $n+1$ -edik töredékrekeszt eldobjuk.)
- A rekeszek tartalmát valamilyen eljárással aggregáljuk. (Átlagolás, maximum stb...)
- Az n aggregált érték rendezett sorrendje megadja a bemenet CRES együtthatóit, amelyekből felépül a CRES vektor.

Speciálisan, az együtthatók jelen alkalmazási célját figyelembe véve, azok előállításához egy összetettebb algoritmust fejlesztettem ki. Ez a módszer további szűréseket, nemlineáris

transzformációkat, és a CPU terhelés csökkentését célzó lépéseket tartalmaz. Ekképpen a ki-nyerendő CRES együttthatók gyorsabban állíthatóak elő, és hitelesebben közelítik meg az adott pillanatban megszólaló zenei félhangok energiaszintjét. Az algoritmus különböző részeit psze-udokóddal ismertetem. Az algoritmus korábbi szekciókban definiált C, w, n, k, m paraméterei hasonló nevekkkel szerepelnek a kódban is. f_{min} és f_{max} a nekik megfelelő f_min és f_max változókban tárolódnak. A bemenet mindig egy t időpontra vonatkoztatva értendő L és R lebegőpontos vektor, amelyek megfelelnek a korábban definiált $X^{(t,k)}$ vektornak a sztereó kép bal és jobb csatornáján érve. A lebegőpontos 0.0 érték a tökéletes csendnek, az 1.0 érték pedig a maximálisan elfogadható amplitúdónak felel meg. (Számos professzionális digi-tális audió rendszerben ez implementációs szinten szabványosnak mondható.) Feltételezem, hogy L és R adottak: a tetszőleges darabolásban érkező inputok kw hosszú, $(k-1)w$ méretű átfedő részt tartalmazó szakaszokra történő összevágásért felelős algoritmus létét adottnak ve-szem. Egy ilyen algoritmus megvalósítása nagyban függene az implementációs környezettől, egyszerűsége pedig nem követeli meg részletes ismertetését. Az itt bemutatott algoritmus egy folytonos feldolgozást biztosító ciklus állandó futtatásával valósítható meg. Implmentációfüggő, hogy lehetséges-e blokkoló I/O használata az adatok olvasásához. Az eljárás a következő:

```

1  function CREATE_CRES()
2  begin
3      while(!exit) do
4          if(GET_INPUT(L, R) == SUCCESS) then
5              X = MIX(L, R)
6              X = FILTER(X, lowpass_filter)
7              X = DOWNSAMPLE(X, rate)
8              F = DFT_MAGNITUDES(X)
9              F = WRAP_TO_CENTS(F, f_min, f_max,
10                             resolution)
11             F = CONVERT_TO_DB(F, level_min, level_max)
12             F = CENTWIDE_GAUSS_FILTER(F, gauss_width)
13             F = MAX_EMPHASIZER(F, cent_width1,
14                               relative_referencel,
15                               global_thresh1)
16             F = MAX_EMPHASIZER(F, cent_width2,
17                               relative_reference2,
18                               global_thresh2)
19             F = TIME_EROSION(F, erode_width)
20             RESULT = MAX_AGGREGATE_TO_CRES(F, C, resolution)
21             CRES_READY(RESULT)
22         end
23     end

```

24 end

Az egyes, nagybetűvel szedett részlépések szerepe a következő:

GET_INPUT(L, R). Megkísérel a következő w darab LPCM jel és az előző $(k-1)w$ darab LPCM jel együttesét előállítani a bal és a jobb bemeneti csatornákat reprezentáló L és R tömbökbe. Ez az algoritmus valósítja meg a bemenet szakaszolását, amely implementációfüggő. Amennyiben visszatérési értéke `SUCCESS`, úgy a beolvasás sikeresnek bizonyult, L és R pedig érvényes bemenetet tartalmaznak.

MIX(L, R). Előállítja L és R sztereó keverését. Esetünkben ez a két tömbben található jelsozrat átlagának meghatározását jelenti. (Alkalmazásfüggő, hogy a keverés során milyen arányban veszik a két jelet – ez lehet dinamikus is akár.)

```
1 function MIX(L, R)
2     X = new_array(length(L))
3     for i=0 to k*w-1 do
4         X[i] = L[i]/2.0 + R[i]/2.0
5     end
6     return X
7 end
```

FILTER(X, filter). Előállítja az X jel egy frekvenciaszűrt változatát. Ez az X és egy `filter` szűrőmaszk konvolúciójának kiszámítását jelenti. A `CREATE_CRES` eljárás kódjában található hívására azért van szükség, hogy a lépést követő `DOWNSAMPLE` futása során ne léphessen föl a jel- és képfeldolgozásból ismeretes *aliasing*^[19] jelensége. Ebből kifolyólag a használatos szűrő egyértelműen aluláteresztő kell, hogy legyen, azonban a konkrét filter kiválasztását, annak megtervezését az implementálóra bízom. Fontos, hogy a filter a `DOWNSAMPLE` lépést követően előálló jel új mintavételezési gyakoriságának felénél (*Nyquist-frekvencia*^[20]) magasabb frekvenciájú komponenseket ne engedjen át.

A konvolúció kiszámításához az implementáció során célszerű egy extra puffert használni, amely memorizálja a konvolvátlan bemenet utolsó elemeit az alkalmazott szűrő hosszának megfelelően. Ekképpen lehetséges a jel egy folytonos (a szűrő hosszával késleltetett) konvolúciójának meghatározása, amelyben az egyes X inputok minden eleme számára biztosított a hozzájuk rendelt konvolvált érték kiszámítása. (Így a tömb széleinél is, ahol általában ezt problémás lehet információvesztés nélkül megvalósítani.)

DOWNSAMPLE(X, rate). A függvény egy X jel mintavételezési gyakoriságának egész számú osztásával (`rate`) előálló alulmintavételezést valósít meg:

```

1 function DOWNSAMPLE(X, rate)
2     i = 0
3     V = new_array(length(X)/rate)
4     while(i < length(V)) do
5         V[i] = X[i*rate]
6         i++
7     end
8     return V
9 end

```

A bemenet DOWNSAMPLE eljárással történő alulmintavételezésének célja az azt követő diszkrét Fourier-transzformáció teljesítményének javítása. Tekintettel arra, hogy a CRES együtthatók előállításához csupán az f_{min} és az f_{max} frekvenciák közötti sáv energiatartalmának ismerete szükséges, egy magas mintavételezésű jel transzformációja feleslegesen terhelne a CPU-t. Ez a terhelés különösen alacsony w -hez társuló magas k értékek esetében a legjelentősebb – ekkor ugyanis nagyon rövid időszelletekre szakaszoljuk az inputot, azonban ezekből nagyon sokat számon tartunk a kimenet számításához, ekképp sűrűn kell nagy teljesítményigényű számításokat futtatni. Sajnálatosan éppen az alacsony w melletti magas k érték adja a legpontosabb eredményeket, ugyanis az alacsony w értékből fakadó gyakori újraszámítások magas időbeli felbontást eredményeznek, a magas k érték pedig az $m = 1 + kw/2$ értéket növelve a DFT eljárás frekvenciafelbontását javítja.

DFT_MAGNITUDES(X). Ez az eljárás a kw hosszúságú X bemeneti tömb diszkrét Fourier-transzformáltjának frekvenciarekeszeihez rendelt magnitúdóit állítja elő. Tekintettel a teljes eljárás periodikusságára, az eredményként előálló, m hosszúságú F vektor a teljes bemenet egy lokális STFT vektorának tekinthető. Az eljárás implementációjára számos előre elkészített, függvénykönyvtárként használható megvalósítás található.

WRAP_TO_CENTS(F, f_min, f_max, resolution). A részlépés az F -ben tárolt STFT együtthatók cent-skálára történő vetemítését végzi az f_{min} referenciafrekvenciától számított f_{max} frekvenciáig kiterjedően. A megvalósításához definiáljuk a korábban meghatározott $up(f, d)$ függvényt megvalósító UP szubrutint.

```

1 function UP(f_ref, d)
2     return (f_ref * pow(2, d/1200))
3 end

```

Az WRAP_TO_CENTS eljárás egy további paramétere a vetemítés felbontását szabályozó resolution. Ez egy egész érték, amely meghatározza, hogy az eredményvektor egyetlen eleme a bemenetet hány cent szélességű darabokra osztja. A resolution paraméter meghatározása szándékosan különbözik a hasonló célt szolgáló, a teljes eljárás végleges kimenetét meghatáro-

zó C rekesszélességtől. Jelen paraméter C -től független hangolásával lehetséges a `CREATE_CRES` procedúra további részlépéseinek információ többlettel történő ellátása. A `resolution` értéket C valamely osztójának választva potenciálisan egy finomított centfelbontáson dolgozhatnak a `CREATE_CRES` eljárás további részei, egészen az algoritmus végéig, ahol a végleges, C szélességű rekeszek kialakításra kerülnek. Az `resolution` és a C közötti oszthatósági reláció megkövetelésével garantálható, hogy a `WRAP_TO_CENTS` szubrutin kimeneteként előálló vektor bármely eleme a végeredményként előálló `CRES` együtthatók közül pontosan egyre legyen hatással.

```

1 function WRAP_TO_CENTS(F, f_min, f_max, resolution)
2     # Az eredménytomb merete
3     len = d_cent(f_min, f_max) / resolution
4     V = new_array(len)
5
6     # Egy frekvenciarekesz szélessége Hz-ben
7     f_width = s / (2*m)
8
9     for i=0 to len-1 do
10         bin_from = UP(f_min, i*resolution) / f_width
11         bin_to = UP(f_min, (i+1)*resolution) / f_width
12         sum = 0.0
13         for j=bin_from until j <= bin_to do
14             sum = sum + F[j]
15         end
16         V[i] = sum
17     end
18     return V
19 end

```

CONVERT_TO_DB(F, level_min, level_max). Ezen lépés célja a kimeneti jelszint emberi hangerőérzethez viszonyított linearizálása. Ez éppen a jel energiaszintjének egy referenciaszinthez viszonyított decibelben való kifejezését jelenti^[21]. Az itt választott referenciaszint a maximális energialeadást reprezentáló 1 érték. (A minimálisat a 0 érték jelenti.) A decibel ábrázolás a tökéletes csendhez $-\infty$, a maximális energiaszinthez pedig 0 értéket rendel.

A következő eljárás a bemenet decibelbe történő konverziója után az eredményt a következő módon küszöböli: Azok az értékek, amelyek valamilyen `level_min` *dB* szintnél kisebbek, a `level_min` értéket veszik fel, a `level_max` *dB* értéknél nagyobbak pedig `level_max`-ot. A kapott értékek ezután min-max normalizáció segítségével a $[0, 1]$ intervallumba kerülnek. A *dB* értékek számítása a digitális audió szoftverekben megszokott módon történik^[22].

```

1 function CONVERT_TO_DB(F, level_min, level_max)
2     V = new_array(length(F))
3     for i=0 to length(F)-1 do
4         V[i] = 20 * log(F[i])/log(10)
5         V[i] = max(min(V[i], level_max), level_min)
6         V[i] = (V[i] - level_min) / (level_max - level_min)
7     end
8 end

```

CENTWIDE_GAUSS_FILTER(F, gauss_width). Az eljárás az F bemeneti vektorán végrehajt egy Gauss-szűrést. Ez megfelel a bemeneti jelsorozat és egy szűrőmaszk konvolúciójának. A szűrőmaszk egy $\text{gauss_width}/2$ középpértékű Gauss-függvény gauss_width elemű mintavételezéseként állítható elő. A mintavételezett függvény szórásának meghatározása az implementáló szabadságát gyarapítja. A konvolúcióval történő szűrések során általában véve lényeges, hogy a szűrőként használt jelsorozat (maszk) elemeinek összege 1 legyen – így biztosítható, hogy a kimenet jelszintje azonos skálán mozogjon a bemenetével. Ennek megfelelően jelen filter konstrukciójának utolsó lépéseként szükséges annak normalizációja.

Megfelelő elemszámú bemeneti vektor, és ahhoz viszonyítva kellően kicsi gauss_width érték mellett (pl. 106 vs. 3) a konvolúció által az F vektor széleinél adott értékek megválasztásának módja elhanyagolható jelentőséggel bír.

MAX_EMPHASIZER(F, cent_width, relative_ref, global_thresh). Az eljárás egy összetett, nemlineáris szűrőt határoz meg. A CRES együtthatók zenei információval való erős korrelálhatósága és a kimenetben található alacsony zajszint javarészt az ezen egyedi szűrő által végrehajtott transzformációnak tudható be. A filter a bemenetet $\text{length}(F)/\text{bin_width}$ rekeszre tagolva vizsgálja. Feladata, hogy az egyes rekeszek lokális maximumértékeihez közel álló értékeket kiemelje, az attól lényegesen eltérőeket pedig csillapítsa, feltéve, hogy az adott rekesz és a teljes bemenet maximumértékeinek hányadosa elér egy küszöbszintet.

Definiáljuk a $h(x)$ segédfüggvényt a következő módon:

$$h(x) = \frac{2}{1 + 2^{-(8\frac{x}{\alpha l_{\max}} + 8)}}.$$

A filter az F bemeneti vektor i -edik elemét a következő módon transzformálja:

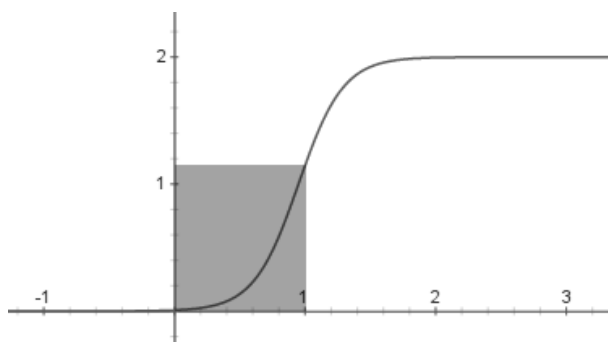
$$F'_i = \begin{cases} 0, & \text{ha } l_{\max} < \beta g_{\max} \\ F_i \cdot h(F_i) & \text{különben} \end{cases}$$

$(0 < \alpha, \beta \leq 1).$

A kimenetként keletkező F'_i -t definiáló kifejezésben g_{max} az F vektor maximuma (globális maximum a teljes inputon). A $h(x)$ definíciójában található l_{max} az i -edik vektorelemet tartalmazó, i -től függő, bin_width széles rekesz (lokális) maximumértéke. α az algoritmus $relative_ref$ paraméterének, β pedig a $global_thresh$ paraméternek felel meg.

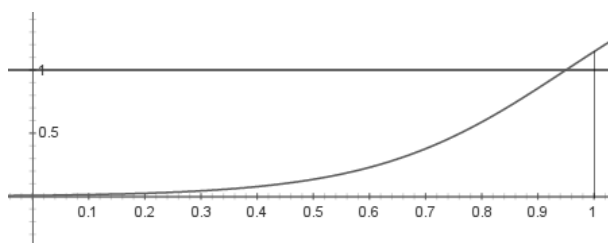
Amennyiben egy adott F_i bemenethez tartozó F'_i kimenet kiszámítása a $h(x)$ függvény segítségével történik, úgy $h(x)$ egy együtthatót ad meg, amely meghatározza F_i csillapításának vagy erősítésének mértékét. Azon x_0 pontot, amelyre $h(x_0) = 1$ teljesül, $h(x)$ *semleges pontjának* nevezhetjük. A szűrő a semleges pont helyével megegyező erősségű bemeneti jelet változatlanul hagy.

A $h(x)$ egy transzformált szigmoid függvényt határoz meg. A függvény 8-szoros x tengely mentén történő zsugorítása, valamint az eltolásának mértéke empirikusan meghatározottak. $h(x)$ képét $\alpha = 0.95, l_{max} = 1$ választás mellett a 3. ábra szemlélteti. A képen sötétítéssel kiemelt a függvény jeltranszformáció szempontjából releváns tartománya.



3. ábra: $h(x)$ függvény, $\alpha = 0.95$.

A sötétített terület egy nagyított változata látható a 4. ábrán. Itt az $y = 1$ egyenes szintén ábrázolt, kiemelve $h(x)$ semleges pontját a két görbe metszéspontjában.



4. ábra: $h(x)$ a $[0, 1]$ intervallumon ($\alpha = 0.95$), valamint az $y = 1$ egyenes.

Az α paraméter segítségével szabályozható a semleges pont helyzete. A magasabb α értékkel rendelkező függvények kevesebb jelet erősítenek fel, és többet csillapítanak. Kellően magas α választással csupán a legmarkánsabb, legerősebb jelek maradnak meg egy rekeszen belül, azok azonban erősítve jelennek meg a kimeneten. Tekintve, hogy a $h(x)$ által leírt transzformáció bármely F_i elem esetén csupán az i -hez tartozó lokális maximumhoz viszonyított (nem pedig a teljes F bemeneti vektorhoz), megfelelő paraméterezéssel a szűrő kimenetében nem vesznek

el az alacsonyabb értékű lokális jellegzetességek valamely globális maximum nagyságának hatására.

Annak érdekében, hogy elkerüljük a túlságosan alacsony értékkel rendelkező jellegzetességek aránytalan fölerősítését (amely megtörténhet, amennyiben a rekeszükben ezek a legmagasabb értékek), szükséges valamilyen globális információ alapuló szűrés is. E célból a szűrő azokat a rekeszeket, amelyek helyi maximuma alacsonyabb a teljes jel globális maximumának β -szorosánál, egyszerűen kioltja.

A MAX_EMPHASIZER pszeudokódja a következő:

```

1  function MAX_EMPHASIZER(F, bin_width, relative_ref,
2                                global_thresh)
3      global_max = max(F)
4      i = 0
5      while(i < length(F)) do
6          local_max = 0
7          for j = i to i+bin_width-1 do
8              if(F[j] > local_max) then
9                  local_max = F[j]
10             end
11         end
12         if(local_max < global_max * global_thresh)
13             for j = i to i+bin_width-1 do
14                 F[j] = 0
15             end
16         else
17             ref = relative_ref * local_max
18             for j = i to i+bin_width-1 do
19                 h = 2.0 / (1.0 + pow(2.0, -(8*F[j]/ref-8)))
20                 F[j] = F[j] * h
21             end
22         end
23         i = i + bin_width
24     end
25 end

```

A MAX_EMPHASIZER megfelelő paraméterezésével elérhető többek között, hogy az egy félhanghoz tartozó centrekeszek összességében megszólaló különböző erősségű hangok közül jó eséllyel csupán egy maradjon. Szimulálható vele a jelenség, amely szerint hallásunkban a hangosabb jellegzetességek hajlamosak elnyomni a halkabbakat. A fent vázolt CREATE_CRES algoritmusban látható, hogy a MAX_EMPHASIZER algoritmus kétszer, közvetlenül egymás után kerül

hívásra, különböző paraméterekkel. Egy lehetséges (és a megvalósításomban használt) paraméterezés először félhang szélességű rekeszek mentén erősít, majd nagyterc rekeszmérettel dolgozik. Itt az első futtatás célja inkább a zajszűrés és kevésbé a feature-elnyomás, a második futás feladata pedig az első szűrés után megmaradt közeli jellegzetességek versenyeztetése.

TIME_EROSION(F, erode_width). Ezen lépés a jel időbeli szűrését végzi, a morfológiai erózió műveletét F és önmaga előző erode_width-1 bemenete fölött elemenként megvalósítva. Ennek megfelelően az implementálónak módot kell találnia rá, hogy az algoritmus különböző lefutásai során mindig a memóriában tarthassa az utolsó erode_width-1 inputként érkezett F vektort.

A lépés feladata az egyes rekeszekben időlegesen fellépő, rövid zajok kiszűrése, ezzel javítva a kimenet megbízhatóságát. Hátránya, hogy a TIME_EROSION eljárás csupán akkor adhat kimenetet, ha már sikeresen meghívták előtte erode_width-1 alkalommal (és memorizálta az akkor kapott bemeneteket). Ebből fakadóan az implementálónak kezelnie kell a feldolgozás elején azt a rövid, ám lényeges időtartamot, ameddig az eljárás csak bemenetet fogad, de kimenetet nem ad. Fontos megjegyezni, hogy ez a lépés a teljes feldolgozási időt megsokszorozza, ugyanis egy, az adott bemenethez tartozó kimenet csak erode_width-1 további bemenet betáplálása után hagyhatja el a rendszert.

MAX_AGGREGATE_TO_CRES(F, C, resolution). Az előfeldolgozó eljárás utolsó lépése, amely a CRES együtthatók végső kiszámításáért felelős. A CRES együtthatókat a korábbi transzformációkon átesett F vektor C rekeszszélességű darabokra bontásával, és a darabokon belüli maximumérték megkeresésével állítja elő. Minthogy C egy centben adott rekeszszélesség, figyelembe kell vennünk a korábbi WRAP_TO_CENTS hívásnál alkalmazott resolution értéket.

```

1 function MAX_AGGREGATE_TO_CRES(F, C, resolution)
2     w = C / resolution
3     i = 0
4     V = new_array(C)
5     u = 0
6     while(i < length(F)-w) do
7         local_max = 0
8         for j = 0 to j < w do
9             if(F[i+j] > local_max) then
10                 local_max = F[i+j]
11             end
12         end
13         V[u] = local_max
14         u = u + 1

```

```

15         i = i + w
16     end
17     return V
18 end

```

Ezzel elkészültek a CRES együttthatók.

CRES_READY (RESULT). Feladata az aktuálisan kiszámított CRES együttthatók továbbítása a szoftver többi része felé. (Ez történhet pl. szálak közötti küldési sorok segítségével.)

3.2. Tanítás

Az eljárás a tanítási szakasza az előfeldolgozó procedúra kimeneteként előálló CRES vektorok egy $\mathcal{Y}^{(t)}$ sorozatát, továbbá a kotta lapozásának ehhez viszonyított megfelelő időpontjait rögzíti. Ezen fázis kezdetét a felhasználó valamilyen megfelelő kezelőfelületi elem használatával jelezheti. A jelzés után az algoritmus azt feltételezi, hogy a felhasználó játszik, így megkezdődik a CRES vektorok rögzítése. A felvétel során az idő 1-gyel növekszik az előfeldolgozó eljárás minden újabb kimenetének létrejöttével. Kezdeti értéke 0, a teljes felvétel hossza pedig N .

Feltételezem, hogy az algoritmus implementációja a tanítás során éppúgy megjeleníti a kottát a zenész számára, mint a későbbi lapozási fázisban. A tanítás alkalmával a felhasználó hangszerez játéka során jelezheti a program számára (vagy megkérhet egy segítőt), hogy mely pontokon szükséges a kotta lapozása. A lapozási kérelmek felvétel kezdetéhez viszonyított időpontjainak sorozatát a T vektor rögzíti, az első lapozási idő T_1 . A felhasználó a tanítás során összesen l lapozási kérelmet nyújt be, amelyek hatására az implementáció a kérelem időpontjának rögzítése mellett a megjelenített kottát is lapozza.

A tanítás eredményeképp előálló $\mathcal{Y}$ és T vektorok együttesen elegendő információt hordoznak a következő fázis számára a csupán akusztikus jeleken alapuló, automatikus lapozás megvalósításához. Ekképpen a tanítás eredménye egy adott zeneműhöz, adott hangszerhez, adott kottához és részben adott akusztikus körülményekhez kötött $(\mathcal{Y}, T)$ rendszer. Jelen algoritmus nem támogatja az egyazon műhöz vagy hangszerhez tartozó, különböző $(\mathcal{Y}, T)$ és $(\mathcal{Y}', T')$ modellek egyeztetését, összehangolását. Az eddigi fejlesztés során nem történtek erőfeszítések ezen probléma feltárásának, megoldásának érdekében.

3.3. Lapozás

A lapozási fázis a tanítást követően léphet életbe: előfeltétele, hogy rendelkezésünkre áll egy $(\mathcal{Y}, T)$ modell valamilyen zenemű, kotta és hangszer együtteséhez kötve.

Az algoritmus a lapozási szakaszban megkísérel az aktuálisan az előfeldolgozási eljárás kimeneteként előálló CRES jelsorozatot az $\mathcal{Y}$ tanítósorozathoz illeszteni. Ennek érdekében a módszer a CRES vektorok aktuálisan előálló $\hat{\mathcal{Y}}$ sorozatából M hosszú szakaszokat képez. A

szakaszok képzése során ameddig nem érkezett be M darab új CRES vektor, addig az algoritmus várakozik. Az M -edik vektor t időpontban történő beérkezése után az eljárás megkísérli a megelőző $\hat{Y}^{(t-M+1)} \dots \hat{Y}^{(t)}$ vektort az $\mathcal{Y}$ tanítóvektor egy megfelelő részéhez optimális módon illeszteni. Az illesztéshez a procedúra a *dinamikus idővetemítés* algoritmusát használja. A dinamikus idővetemítés lehetővé teszi, hogy az egyes illesztések után meghatározzuk, hogy az illesztett M hosszúságú bemeneti szakaszhoz a tanítóadatok milyen K hosszúságú sorozata tartozik. A két szám arányából következtethetünk arra, hogy a kérdéses M hosszú szakaszon rögzített zene tempója milyen mértékben térhet el a tanítóanyagban található, neki megfelelő zenei minta tempójától: M/K ideális esetben épp ezen két tempó hányadosa. Az utolsó néhány ilyen hányados számontartásával megfelelő becslés adható arra, hogy a bemenet számára egységnyi idő eltelte (azaz egy újabb CRES vektor megérkezése) a tanítóanyag szempontjából mennyi idő elmúlását eredményezi – így hitelesen nyomonkövethetővé válik, hogy hol tart az aktuálisan játszott zene a tanítóanyaghoz képest. Ezen információ birtokában pedig megállapítható, hogy mikor érjük el az i -edik ($1 \leq i \leq l$) lapozási esemény tanítóanyag időkeretében rögzített T_i időpontját, tehát az algoritmus a megfelelő pillanatban lapozhat.

Összességében a lapozási szakasz a következő részekre bontható:

- Bemenet M hosszú szakaszolása.
- Idővetemítés végrehajtása az aktuális M hosszú szakaszon.
- Zenei tempók arányának közelítése az idővetemítések eredményei alapján.
- A tanítóanyag időkeretében mérő ω számláló növelése az aktuálisan számított tempóaránnal.
- A j -edik kottalap mutatása, ahol $j = \arg \min_{1 \leq j \leq l+1} \omega < T_j$ ($T_{l+1} = \infty$ megkötéssel).

A bemenet szakaszolása könnyen megvalósítható feladat, így az nem képezi ismertetés tárgyát. A következő részben az idővetemítés részletei olvashatóak.

3.3.1. Dinamikus idővetemítés

A dinamikus idővetemítés^[23] egy dinamikus programozást felhasználó eljárás, amelynek feladata két, akár eltérő hosszúsággal rendelkező jelsorozat optimális illesztésének meghatározása. Az illesztés egy, a jelsorozatok elemei között fennálló megfeleltetés megállapítását jelenti. Az algoritmus olyan jelsorozatok felett értelmezhető, amelyek elemei között definiált valamilyen távolságfüggvény. Az eredményként előálló megfeleltetés jellemzője, hogy az egymáshoz rendelt elempárok távolságainak összege minimális.

A kottalapozó alkalmazás esetében a jelsorozatok a CRES vektorok szériájából álló $\mathcal{Y}$ és $\hat{\mathcal{Y}}$ sorozatok megfelelő hosszúságú részei. Szükséges, hogy meghatározzassuk két CRES együtthatók sorozatából álló vektor távolságát. Egy lehetséges, naív választás az euklideszi távolság.

A további fejtegetés ezen távolságfüggvényt alkalmazza. Megjegyzendő, hogy számos további ismert függvény használata, továbbá új, a CRES együtthatók értelmezését figyelembe vevő távolságfüggvények definíciója szintén elképzelhető.

Tárgyalásunk szempontjából az idővetemítés által meghatározott konkrét megfeleltetés lényegtelen, csupán az számít, hogy az M hosszú bemenethez milyen hosszúságú szakaszt rendel az algoritmus a tanítóanyagból. A legutóbbi, CRES vektorok M hosszú sorozatából álló bemenetet a továbbiakban $\tilde{\mathcal{Y}}$ jelöli. Egy $\mathcal{V}$ CRES vektorsorozat w hosszú suffixe legyen a következő:

$$\mathcal{V}_{w\dots} = \mathcal{V}_w \mathcal{V}_{w+1} \dots \mathcal{V}_{|\mathcal{V}|}.$$

A $\mathcal{V}_{u,w}$ jelzés pedig a következő vektorsorozatot határozza meg:

$$\mathcal{V}_{u,w} = \mathcal{V}_u \mathcal{V}_{u+1} \dots \mathcal{V}_w \quad (u \leq w).$$

Legyen τ egy számláló, amely megmutatja, hogy a tanítóanyag $\mathcal{Y}$ vektorában melyik legmagasabb indexig sikerült ezidáig idővetemítéssel valamely $\tilde{\mathcal{Y}}$ bemenetet illeszteni. Kezdetben $\tau = 0$.

Amennyiben élünk a megszorítással, miszerint a játékos nem ugrik vissza a tanítóanyag jel-sorozatának valamely korábbi indexére, úgy feltételezhetjük, hogy dinamikus idővetemítéseink során a következő $\tilde{\mathcal{Y}}$ vektort elegendő csupán az $\mathcal{Y}_{\tau\dots}$ vektorhoz illeszteni. Természetesen ez a megszorítás nem jelentheti, hogy a muzsikás nem használhat például ismétlődőjeleket, ugyanis amennyiben az ismétlés a tanítás során jelen volt, úgy annak második eljátszása az első eljátszásánál mindenképpen valamilyen magasabb számokból álló indexsorozathoz köthető. A továbbiakban élni fogunk ezzel a megkötéssel.

Figyelembe véve, hogy a dinamikus idővetemítés számításigényes feladat, nem célszerű azt az $\mathcal{Y}$ tanítóanyag valamely teljes $\mathcal{Y}_{\tau\dots}$ suffixén futtatni. Ehelyett elegendő felülről becsülni a relatív tempót, és ennek fényében a tanítóanyag csupán valamely $\mathcal{Y}_{\tau,pM}$ darabját vizsgálni egy $p > 1$ egész paraméterre nézve.

Ekképpen idővetemítéseink sorozata az aktuális $\tilde{\mathcal{Y}}$ bemenet és a tanítóanyag egy $\mathcal{Y}_{\tau,pM}$ részének összevetéséből áll, ahol τ minden vetemítés után az optimálisan illesztett tanítóanyag-részlet méretével növekszik. Az algoritmus jelenlegi változata a dinamikus idővetemítés legegyszerűbb, lokális és globális megkötéseket, büntetéseket nélkülöző naív változatát alkalmazza. Az itt ismertetett változat $\tilde{\mathcal{Y}}$ bemenetét Y jelöli, hossza pedig M . A $\mathcal{Y}_{\tau,pM}$ bemenetet Z jelöli, hossza pedig $p \cdot M$. Az eljárás visszatérési értéke az $Y[M-1]$ tömbelemhez legnagyobb mértékben illeszkedő Z -beli elem indexe, tehát az M hosszúságú $\tilde{\mathcal{Y}}$ -hoz a legjobban illeszkedő szakasz hossza.

```

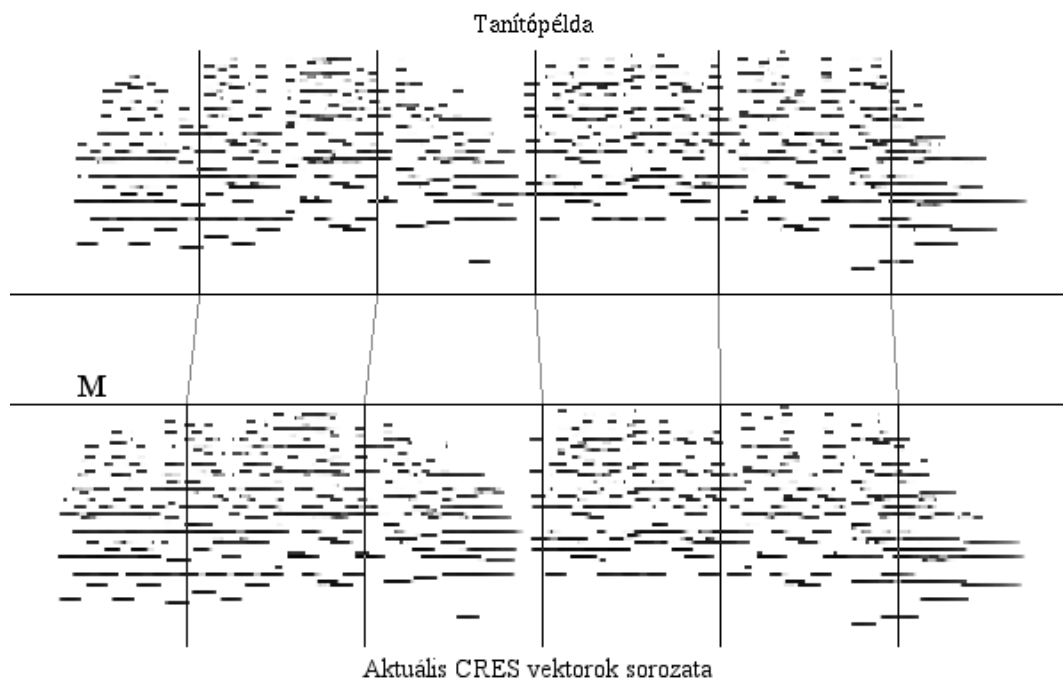
1 function DYNAMIC_TYME_WARP (Y, Z)
2     DTW = new_matrix(M, p*M)
3     for i = 0 to M do
4         DTW[i][0] = infinity
```

```

5      end
6      for i = 0 to p*M do
7          DTW[0][i] = infinity
8      end
9      DTW[0][0] = 0.0
10
11     for i = 1 to M-1 do
12         for j = 1 to p*M-1 do
13             cost = distance(Y[i], Z[j])
14             DTW[i][j] = cost + min(DTW[i-1][j],
15                                     DTW[i-1][j-1],
16                                     DTW[i][j-1])
17         end
18     end
19
20     minval = infinity
21     minpos = 0
22     for j = 1 to p*M-1 do
23         if (DTW[M-1][j] < minval) then
24             minval = DTW[M-1][j]
25             minpos = j
26         end
27     end
28     return j
29 end

```

A dinamikus idővetemítés CRES együtthatókból álló vektorok sorozatán történő futtatásának egy eredménye látható az 5. ábrán: Az aktuális CRES vektorok sorozata M hosszúságú cellákra bomlik, amelyeket halványszürke vonalak kötnek össze a tanítóanyag nekik megfelelő, hosszaikban eltérő celláival. Az idővetemítések az aktuálisan rögzített M hosszú cellák és a tanítóanyag még nem illesztett részei között történtek. Egy cellán belül egy oszlop egyetlen CRES vektornak felel meg. Az ábrán látható vektorok számításához használt rekeszméret 100 cent volt – így minden együttható pontosan egy félhang energiaszintjéről tanúskodik. Az oszlopok felsőbb részein található pontok a magasabb félhangok energiaszintjeit, a lentebbiek pedig a mélyebbekét jelölik. A pontok, vonalak sötétsége a nekik megfelelő félhang intenzitását jelölik – a sötétebbek erősebb hangoknak felelnek meg.



5. ábra: A CRES együttthatók közötti dinamikus idővetítés szemléltetése Muzio Clementi egy Adagio-ján, zongorán.

3.3.2. A megfelelő kottalap mutatása

Az algoritmus működtet egy számlálót (órát), amely segítségével megpróbálja megbecsülni, hogy hol tarthat a muzsikusi játékában a tanítóanyag időbeliségéhez mérten. Ez a számláló legyen ω , kezdeti értéke pedig 0. Ezen túlmenően vegyünk egy δ változót, amely az ω óra múlásának aktuális ütemét szabályozza. $\delta = 1$ megfelel az előfeldolgozó eljárás két, egymást követő kimenete között eltelt időnek. Kezdetben $\delta = 1$. A futás során minden újabb M darab CRES vektor beérkezésével lefut egy dinamikus idővetítés, amelynek q kimenete megadja, hogy a tanítóanyag milyen hosszú része illeszkedett optimálisan az M hosszú bemenethez. Bármely időpontban a legfrissebb q érték alapján számítsunk ki egy arányt: $\delta_1 = q/M$. δ_1 meghatározza, hogy a legutóbb illesztett input részlet tempója milyen arányban áll a tanítóanyagban neki megfeleltetett rész tempójával. A korábbi számítások eredményei tárolódnak rendre a $\delta_2, \delta_3 \dots$ változóknak, egészen valamilyen b indexig. Az algoritmus menete során δ értéke ezen változók alapján alakul, a következő formula szerint:

$$\delta = \sum_{i=1}^b \frac{2(b-i+1)}{b(b+1)} \delta_i.$$

A képlet a mindenkor legfrissebb b darab δ_i együtttható súlyozott mozgó átlagát adja meg. A δ_i együttthatóiként megjelenő súlyok kialakítása a következő módon történt: Az i -edik elem relatív fontossága legyen $(b-i+1)$. Tehát a legfontosabb elem az első, ennek relatív fontossága b , az utolsó (δ_b) elemé pedig 1, ez számít a legkevésbé δ kialakítása során. Az elemek között szétosztott súlyok az egész számok 1-től b -ig, így a kiosztott összsúly $\frac{b(b+1)}{2}$. Annak érdekében,

hogy az elemekhez rendelt súlyok összege 1 legyen, azok relatív fontosságát normalizálnunk kell azok összegével – így az egyes elemekhez rendelendő súlyok a $\frac{2}{b(b+1)} \cdot (b - i + 1)$ formula szerint alakulnak. A gyakorlatban ameddig nem áll rendelkezésre b darab δ_i érték, addig δ számításához felhasználható az összes addig rendelkezésre álló érték, a fenti képlet szerint. (Ekkor a súlyozás nem lesz ugyanaz, mint b érték esetén, azonban ez a helyzet csupán rövid ideig, az első b részlet megérkezéséig áll fenn.)

Az algoritmus feltételezi, hogy a felhasználó a program rendelkezésére bocsátotta a tanítóanyagnak megfelelően kottalapok azonosítóinak egy legalább $l + 1$ hosszúságú, akár ismétlődést is tartalmazó felsorolását. (l a lapozási kérelmek száma a tanítóanyagban.) Ezen felsorolás határozza meg az eljárás számára a kottalapok egymásutániságát. A kottában található ismétlődő jelek, és hasonló, nemszekvenciális lépéseket előíró utasítások kezelhetők az ismétlődő lapok újbóli felsorolásával az előírt játék szerinti megfelelő helyeken. (Példa: Egy ismétlődő jel az előző oldalra való lépésre utasítja a zenészt. Ekkor a lapozó program számára a kottalapok listájában a következő elemnek az előző oldalt választjuk.)

Az ω számláló δ -val történő sorozatos növekedésével szimulálható az idő múlása a tanítóanyag időkeretében. Minthogy δ frissíthető minden egyes végrehajtott idővetemítés után, így az idő múlásának mértéke igazodni fog az utolsó néhány idővetemítés tempóarányához. Probléma lehet azonban, hogy ezen eljárás csupán a közeli múlt adatai alapján becsüli meg a szimulált idő múlásának ütemét. Amennyiben például közel 1-es tempóarányú vetemítések egy sorozatát egy hirtelen lassulás követi a tanítóanyaghoz képest, úgy ω a lassított rész alatt is közelítőleg 1-es ütemben fog növekedni, noha ekkor már a bemenetként kapott minta lényegesen lassabb a tanítóanyagnál. Ekképpen ω sietni fog. Hasonló jelenség produkálható hirtelen gyorsulással a tanítóanyaghoz képest – ekkor ω késni fog. Minthogy a feldolgozás M együtthatóvektoronként történik, így teljesen online módon igazodó δ ütem megvalósítása ezen algoritmusban nem lehetséges. Nagyban javíthat azonban a megoldáson az ω korrigálása minden idővetemítés után, ugyanis ekkor pontosan tudható, hogy a tanítóanyag mely pozíciójához illeszkedett a bemenet vége. Ezt a pozíciót a már korábban definiált τ számláló adja meg.

Tehát az idővetemítések végén ω -t τ -hoz igazítjuk. Feltételezve, hogy az ω egy idővetemítést követően éppen sietett, a τ -val történő korrekció az óra visszaállítását eredményezné, amely megtörténe az idő múlásának monotonitását. Egy lehetséges megoldás ekkor a régi ω érték befagyasztása $(\omega - \tau)$ időre.

A továbbiakban feltételezzük, hogy az előfeldolgozás és az idővetemítések a háttérben történnek, a $\delta_1, \dots, \delta_b, \tau$ értékek maguktól frissülnek. Adottnak veszünk egy $(\mathcal{Y}, T)$ tanítóanyagot. A megfelelő lap mutatását szavatoló algoritmus felépítése a következő:

1. $\omega = 0, \delta = 0$.
2. $\omega := \omega + \delta$.
3. Történt idővetemítés?

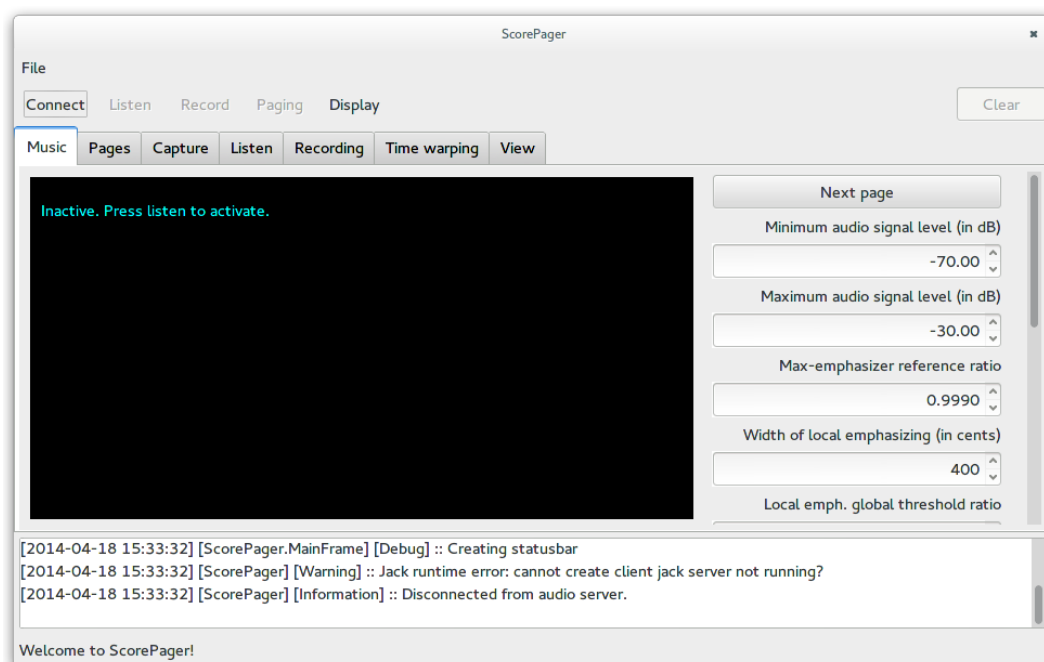
- (a) *Igen*: $\delta := \sum_{i=1}^b \frac{2(b-i+1)}{b(b+1)} \delta_i$. $\omega := \tau$. Mostantól δ_1 a legfrissebb, δ_2 a közvetlenül megelőző iterációban legfrissebb, δ_b a korábban utolsó előtti relatív tempót tárolja.
4. $k := \arg \min_{1 \leq j \leq l+1} \omega < T_j$ ($T_{l+1} = \infty$ megkötéssel). Ekkor k megadja a kottalapok előzetesen megadott listájában az aktuálisan a zenész számára prezentálandó kottalap indexét.
5. Ellenőrizzük, hogy a k indexű kottalapot mutatja-e a program. Ha nem, úgy váltunk a k . lapra.
6. Folytassuk az algoritmust a 2. lépésnél.

A megvalósítás

Az algoritmus egy prototípusát C++11-ben, Linux platform alá megvalósítottam. Ezen szoftver nagy segítséggel volt az algoritmus fejlesztésére nézve: lehetővé tette a gyors prototipizálást, a különböző belső, elsősorban az előfeldolgozással kapcsolatos részlépések során előálló eredmények vizualizációját. A program az alábbi külső komponensek, szoftver-erőforrások felhasználásával készült:

- *Standard C++ library*. Szálkezelés, I/O, algoritmusok és adatszerkezetek, kivétel osztályok, atomikus műveletek felhasználása.
- *Boost libraries* – Számos különböző területen, problémakörben kínál tökéletes kidolgozottságú C++ könyvtárakat. Szignálok (*publish & subscribe* funkcionalitás szoftverkomponensek között), és fájlkezelés céljából alkalmaztam.
- *Poco C++ libraries* – Modern, nyílt forráskódú függvénykönyvtár, amelynek célja elsősorban nagy teljesítményű hálózati alkalmazások fejlesztésének megkönnyítése. Naplózás kiépítésére használtam.
- *FFTW (Fastest Fourier-Transform in the West)* – A Fourier-transzformációk gyors végrehajtása érdekében alkalmaztam az előfeldolgozó algoritmus megvalósítása során.
- *JACK (Jack Audio Connection Kit)* – Kliens-szerver architektúrájú alkalmazásdémon és függvénykönyvtár, amely segítségével a program valós idejű hozzáférést szerez a rendszer (és a többi JACK-et támogató processzus) audió erőforrásaihoz.
- *wxWidgets* – Grafikus kezelőfelület megvalósítása.
- *Proton* – Saját fejlesztésű függvénykönyvtáram, amelynek célja számos gyakorta előforduló, a standard library és a boost által mégis lefedetlen probléma gyors kezelhetőségének biztosítása. A könyvtár fejlesztés alatt áll. Szálak közötti adattovábbításra (termelőfogyasztó mintára), adatpufferek egységesített kezelésére, kezelőfelület dinamikus felépítésére használtam.
- *Subspace* – Saját fejlesztésű függvénykönyvtár. Célja különféle multimédiás tartalmakkal történő munkák segítése. Számos digitális jelfeldolgozó algoritmust ezen függvénykönyvtárral valósítottam meg.

A szoftver indítás utáni képernyőképét szemlélteti a 6. ábra. A program felső eszköztárának segítségével vezérelhető a teljes lapozási folyamat. A *Connect* nyomógomb segítségével kapcsolódhatunk a számítógépen futó *Jack Audio Server*hez, amely a továbbiakban elláthatja a programot audio bemenettel. Az audió szerver előzetes elindítása nem szükséges – a gomb



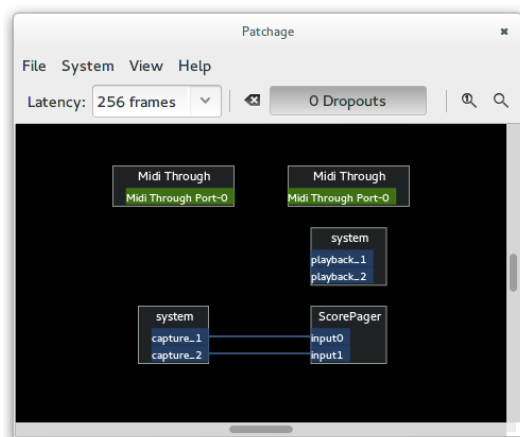
6. ábra: A prototípus (ScorePager) képe közvetlenül indítása után.

aktiválásával az automatikusan elindul. A szoftver egy virtuális sztereó bemenetet hoz létre a JACK audio démon segítségével. A 7. ábra ezt hivatott szemléltetni.

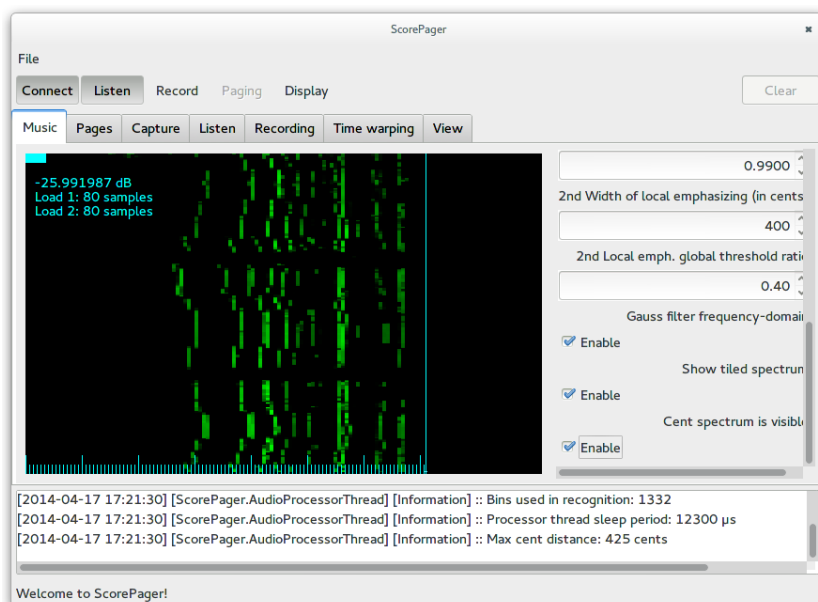
A *Connect* mellett található *Listen* gomb segítségével elindítható az előfeldolgozó eljárás. Ennek aktiválása után számos paraméter módosítása (egészen a deaktiválásig) lehetetlenné válik. A 8. ábrán az látható, ahogyan az alkalmazás megjeleníti az előfeldolgozási eljárás eredményeként előálló CRES együtthetők sorozatát.

A felső eszköztáron található további gombok (*Record*, *Paging*, *Display*) segítségével vezérelhető a tanítás és a lapozás folyamata, valamint kérhető a kotta megjelenítése a felhasználó számára.

Az eszköztár alatt „fülek” sorozata található. Az első fül a *Music*. Ezen fülön látható az előfeldolgozás kimenete, valamint a lapozás során a dinamikus idővetemítések eredménye is megjeleníthető itt. (Az 5. ábrán láthatóhoz hasonló módon.) A következő, *Pages* fülön megadható kottalapok képfájljainak egy sorozata. A lapozás során a program ezen kottalapokat jeleníti meg a felhasználó számára. A további fülek lehetőséget nyújtanak az algoritmus korábban megismert paramétereinek, illetve számos egyéb tényező beállítására.



7. ábra: A ScorePager program sztereó bemenetei, továbbá a számítógép további audio perifériái a Patchage elnevezésű programból megtekintve.



8. ábra: A ScorePager alkalmazás előfeldolgozó eljárásának kimenete.

Eredmények, további lehetőségek

Az itt bemutatott algoritmus prototípuszerű megvalósításával kitűnő eredményeket értünk el klasszikus művek kottáinak lapozása esetén. Az eddigi kísérletek zongorára, orgonára, illetve néhány fúvós hangszer, így például kürt, klarinét és trombita hangjának elemzésére terjedtek ki. Az utóbbi három hangszer hangja pusztán szintetizált volt, azonban ezek a hangok is erősen hordozták az eredeti hangszerek hangjainak spektrális jellegzetességeit. A szoftver bemutatásra került Dr. Csendes Tibor 2014. április 2-i szabadegyetemi előadásának^[24] keretei között, továbbá egy rövid bemutató videó is elérhető róla az interneten^[25].

A program ezidáig nem járt sikerrel emberi énekhang elemzésében, illesztésében. Ez vélhetően az énekbe ágyazott szöveg kiejtése által keltett spektrális zavaroknak tudható be, amelyek nem tették lehetővé a CRES együtthatók megbízható számítását azonos dal két egymás utáni

eléneklésének viszonylatában.

A jelenlegi megvalósítás korlátja, hogy ha valamely idővetemítés egy adott bemenethez nem a neki megfelelő részt illeszti a tanítóanyagban, akkor az eljárás jó eséllyel eltéved a hallott zenében, és képtelen korrigálni elcsúszását. Ennek kiküszöbölésében segítséget nyújthat az idővetemítés által illesztett részek optimumértékeinek nyomonkövetése. (Tehát az M hosszú rész illesztéséhez tartozó megfeleltetés össztávolság-értékének figyelembe vétele.) A kirívó értékek esetén lehetséges lenne életbe léptetni egy kereső funkcionalitást, amely akár a teljes korábban hallott hangmintát próbálja meg a tanítóanyaghoz illeszteni. Az eljárás tovább javítható, ha az elcsúszás érzékelése után további CRES vektorokat gyűjtünk még, és azokat is figyelembe vesszük az imént említett kereső funkció algoritmusában.

A kifejlesztett procedúra toleranciája a zenész hibáival szemben egyaránt múlik az eljárás paraméterezésén, és azon, ahogyan a muzsikusz megpróbálja kijavítani tévesztését. Abban az esetben, ha a zenész egyszerűen átlépi a hibás részt és nem ugrik vissza kijavítani azt, növeli az esélyét annak, hogy az algoritmus nem téved el a hallottak tanítóanyaghoz való viszonyításában. Ekkor a bemeneten nem jelenik meg további felesleges hanganyag, amely esetleges extra távolságértékeket vonna be az idővetemítés teljes távolságösszegébe.

Kiemelendő, hogy az itt bemutatott eljárás nem modellezi explicit módon a zene ritmikai tartalmát. Ebből fakadóan a hallott zenei anyagok tetszőleges ritmikai felbontással, struktúrával rendelkezhetnek, megfelelően annak, ahogyan előadások alkalmával a művészek szabadon értelmezik a kottában lejegyzett ritmikai elemeket. Az eljárás tervezése során felmerült, hogy éppen a ritmika, az ütemszerkezet figyelembevételével történjen a lapozás, ezt azonban elvetettem, ugyanis a ritmika önmagában túl kevés információt hordoz ahhoz, hogy például a fent felvázolt keresőfunkciót megvalósítsam, ráadásul modellezése, élő hangból történő kinyerése nehéznek bizonyul.

Kottareprezentáció tekintetében az algoritmus által tett egyetlen megkötés azon előfeltevésben merül ki, miszerint a kotta lapokra tagolt. A nyomtatott kották döntően megfelelnek ennek az előfeltevésnek. Elképzelhető olyan digitális reprezentáció, amely a kottát folyamszerűen, oldalakra való tagolás nélkül jeleníti meg. (Jellemzően ezek kottaszerkesztő programok szoktak lenni.) A digitális kotta efféle esetleges tagolatlansága azonban egyszerű módszerekkel szintén lapokra bontható. Az előbbi előfeltevést meghaladóan nincsen kifejezett követelmény a kotta formájával kapcsolatban, így az előállhat MIDI szekvenciák utasításaként, képfájlok sorozataként vagy például egy *.pdf* dokumentum oldalaként.

A dinamikus idővetemítés számos finomhangolási lehetőséget ad. Ezek segítségével elképzelhető, hogy javítható volna az illesztés megbízhatósága. Ilyenek hangolási lehetőségek, technikák a következők:

- Globális megkötések az illesztett minták arányára vonatkozóan.
- Lokális megkötések, amelyek bizonyos, például túl nagy távolságérték-növekedést eredményező lépéseket tiltanak.

- A következő lépés irányának megtalálásában az egyes irányok súlyozása. A súlyozás történhet úgy például, hogy preferálja az azonos tempót feltételező lépéseket a dinamikus programozási összefüggésben.

Az eltévedés esetén életbe lépő kereső üzemmód mellett elképzelhető volna a dinamikus idővetemítések átfedő módon történő alkalmazása, amikor is az egymást követő idővetemítések bemenetei egy bizonyos hosszúságú szakaszon megosztoznak. Az átfedő illesztések közül a legvalószínűbbnek választhatjuk például a minimális osztávolság-értéket adót. Ezzel valószínűleg növekedhetne az eljárás tévesztésekkel szembeni toleranciája.

Az előfeldolgozási eljárás kimeneteként létrejött CRES együtthatók egyéb problémák megoldásában is felhasználásra lelhetnek: Elképzelhető alkalmazásuk összhangzattani funkciók detektálására, illetve például annak eldöntésére, hogy egy harmónia dúr, moll, szűkített vagy bővített jellegű-e. Ekképpen létrehozható volna többek között olyan program, amely összhangzattani diktálási feladatok megoldását hallás alapján megadja. A CRES együtthatók segítséget nyújthatnak olyan szoftverek létrehozásában is, amelyek célja a hang alapján történő kottázás, tehát akusztikus információk alapján megvalósuló kottairás.

A fejlesztés további tervei a következők:

- Kísérletek folytatása fúvós- vagy szimfónikus zenekari művek partitúrájának lapozására.
- Professzionális zenészek bevonása a projektbe, a fejlesztés irányának szabályozása a visszajelzéseik alapján.
- Robusztus játékkövetés megvalósítása – az algoritmus hibázással szembeni toleranciájának javítása átfedő idővetemítések alkalmazásával.
- Súlyos tévesztés, az eljárás eltévedése esetén életbe lépő pozíciókereső funkcionalitás ki-fejlesztése. Az algoritmus képes lenne a hallottak alapján előre nem várt, a tanítóanyagban nem szereplő ugrásokat is végrehajtani a kottában, így akár korábban már eljátszott részekhez is visszalapozni. Ennek következményeként akár szükségtelenné is válhat az eljárás előzetes felkészítése a műben található ismétlésekre. (A kottalapok különleges, ismétléses sorrendjének megadásával.)
- Ütemmutató detektor^[26] megvalósítása, amely segítségével metronóm jeleníthető meg a zenész számára.

Hivatkozások

- [1] A. D. Kilmer, M. Civil, *Old Babylonian Musical Instructions Relating to Hymnody*.
Journal of Cuneiform Studies, Vol. 38, No. 1, Spring, 1986
- [2] BHZL hangjegyzírás, II. 108
- [3] *PageFlip lapozó rendszer*, <http://www.pageflip.com>
- [4] *Tonara automatizált kottalapozó rendszer*, <http://tonara.com>
- [5] *AutoFlip automatizált kottalapozó rendszer*, <http://www.velvetmatter.com>
- [6] BHZL kottajelek, II. 344
- [7] BHZL előadási jelek, I. 507
- [8] BHZL frazeálás, I. 610
- [9] BHZL ismétlés, II. 224
- [10] Dr. Keszler Lőrinc, *Összhangzattan*. Editio Musica, Budapest, 1952, 1. oldal
- [11] DS pp. 23
- [12] DS pp. 89-91
- [13] TCMT pp. 1049-1069
- [14] DS pp. 77-92
- [15] BHZL hangmagasság, II. 112
- [16] APB pp. 432-436
- [17] Almási Dénes Attila, *Zenei beviteli eszközök kezelése*, BSc szakdolgozat, Szegedi Tudományegyetem, 2012, 13. oldal
- [18] S. B. Davis, P. Mermelstein, *Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences*, IEEE Transactions on Acoustic, Speech and Signal Processing, 28(4), pp. 357-366
- [19] APB pp. 261
- [20] TCMT pp. 30
- [21] E. C. Poulton, *A linear relation between loudness and decibels*, Perception & Psychophysics, Vol. 36, Issue 4, pp. 338-342, August, 1984

- [22] APB pp. 210
- [23] T. K. Vintsyuk, *Speech Discrimination by Dynamic Programming*, Kibernetika, Vol. 4, No. 1, pp. 81-88, 1968
- [24] <http://www.u-szeged.hu/szabadegyetem/program-szabadegyetem/erdekes-optimalizalasi>
- [25] <http://rudanium.org/~denes/pub/scorepager.mp4>
- [26] TCMT Rhythm Recognition, pp. 520

Bibliográfia

- APB *The Audio Programming Book*, Richard Boulanger, Victor Lazarini; The MIT Press, Cambridge, Massachusetts, London, England; 2011. ISBN: 978-0-262-01446-5
- DS *Designing Sound*, Andy Farnell; The MIT Press, Cambridge, Massachusetts, London, England; 2010. ISBN: 978-0-262-01441
- TCMT *The Computer Music Tutorial*, Curtis Roads; The MIT Press, Cambridge, Massachusetts, London, England; 1996. ISBN: 978-0-262-18158-7
- BHZL *Brockhaus Riemann Zenei Lexikon I-III*. Szerk. Carl Dahlhaus, Hans Heinrich Eggebrecht. Budapest, Zeneműkiadó, 1983-1985. ISBN: 963-330-540-3