



Eötvös Loránd Tudományegyetem  
Informatikai Kar  
Numerikus Analízis Tanszék

---

## Szakdolgozat

# Zenei információk kinyerése hangfájlokból

**Dr. Németh Zsolt**  
Egyetemi adjunktus

**Nagy Rajmund**  
Programtervező Informatikus BSc

Budapest, 2019

# Tartalomjegyzék

<b>1. Bevezetés</b>	<b>2</b>
<b>2. Felhasználói dokumentáció</b>	<b>3</b>
2.1. Követelmények, a program telepítése . . . . .	3
2.2. A program használata . . . . .	4
2.3. A zenelejátszó . . . . .	5
2.4. Az adatok ábrázolása . . . . .	6
2.5. Navigáció . . . . .	8
2.6. Az elemzési paraméterek kiválasztása . . . . .	9
2.7. Az eredmény tárolása és betöltése . . . . .	10
<b>3. Fejlesztői dokumentáció</b>	<b>12</b>
3.1. Előismeretek . . . . .	12
3.1.1. Hangok reprezentációja . . . . .	12
3.1.2. Az elemzés során használt módszerek . . . . .	14
3.2. Megoldási terv . . . . .	16
3.3. Megvalósítás . . . . .	18
3.3.1. A GUI komponens . . . . .	19
3.3.2. A <code>state</code> komponens . . . . .	23
3.3.3. Az <code>algo</code> komponens . . . . .	24
3.4. Tesztelés . . . . .	31
3.5. Továbbfejlesztési lehetőségek . . . . .	32
<b>4. Hivatkozások</b>	<b>33</b>
<b>5. Függelék</b>	<b>35</b>
5.1. A magyarra fordított szakszavak . . . . .	35

# 1. Bevezetés

A *Music Information Retrieval* egy olyan interdiszciplináris tudományterület, melyben a kutatók zenei felvételekből olyan adatokat nyernek ki, melyekkel a műveket kategorizálni, minősíteni, vagy valamilyen új módon reprezentálni lehet. Ehhez mindenképpen szükséges az alkalmazott módszerek matematikai hátterének megértése, a jelfeldolgozásban való jártasság, illetve a probléma megközelítésétől függően a gépi tanulás [4], [5], a pszichoakusztika [3] és a mesterséges intelligencia [2] ismerete.

Rengetegféle információt lehet a zenéből kinyerni; elemzésünk céljából választhatunk olyan hagyományos zenei jellemzőket, mint a hangnem [6] vagy a tempó [9], megpróbálkozhatunk az éppen megszólaló hangszerek beazonosításával [7], [8] esetleg törekedhetünk a darab hangulatának a meghatározására [10].

A szakdolgozatomban egy olyan grafikus felületű programot mutatok be, ami zongoradarabokban hangkezdeteket, azaz a billentyűleütések időpillanatait keresi meg, ezzel lefektetve egy későbbi kottát követő vagy készítő alkalmazás alapjait.

## 2. Felhasználói dokumentáció

A Music Feature Extractor egy olyan program, ami a hangkezdetek automatikus észlelésével jelentősen felgyorsítja zeneművek ritmikai elemzését, illetve egy megbízható támpontot nyújt, ha kottát szeretnénk készíteni egy zenei felvétel alapján. Ennek megfelelően a bemenet egy hangfájl, amit az alkalmazás *rövid idejű Fourier-transzformációval* átalakít, majd a frekvencia tartományban (a *spektrum-ban*) megkeresi a hangkezdetekre jellemző mintákat. Az így kapott eredményt a program grafikus felületén kirajzolhatjuk, majd leellenőrizhetjük a beépített zene-lejátszó segítségével.

### 2.1. Követelmények, a program telepítése

Az alkalmazás használatának előfeltétele a Windows 7/8.1/10-es operációs rendszer és a Matlab Runtime 9.2-es, vagy annál újabb verziójának megléte a számítógépen. Amennyiben az utóbbi hiányzik, a telepítő a telepítés során a programmal együtt kicsomagolja, vagy – amennyiben a webes változatot futtatjuk – letölti az internetről azt.

A szoftver hardverigénye nem túl nagy, 3-4 perces hangfájlok elemzésénél is 400-550 MB között marad a memóriahasználat. Régi integrált videokártyák esetén némi késés tapasztalható a kirajzolásnál.

A telepítés után a futtatható bináris a következő útvonalon érhető el:  
...\\MusicFeatureExtractor\\application\\MusicFeatureExtractor.exe.

Név	Kiterjesztés
WAVE	.wav
OGG	.ogg
FLAC	.flac
AU	.au
AIFF	.aiff, .aif
AIFC	.aifc
MP3	.mp3
MPEG-4 AAC	.m4a, .mp4

**1. táblázat.** A támogatott fájlformátumok listája

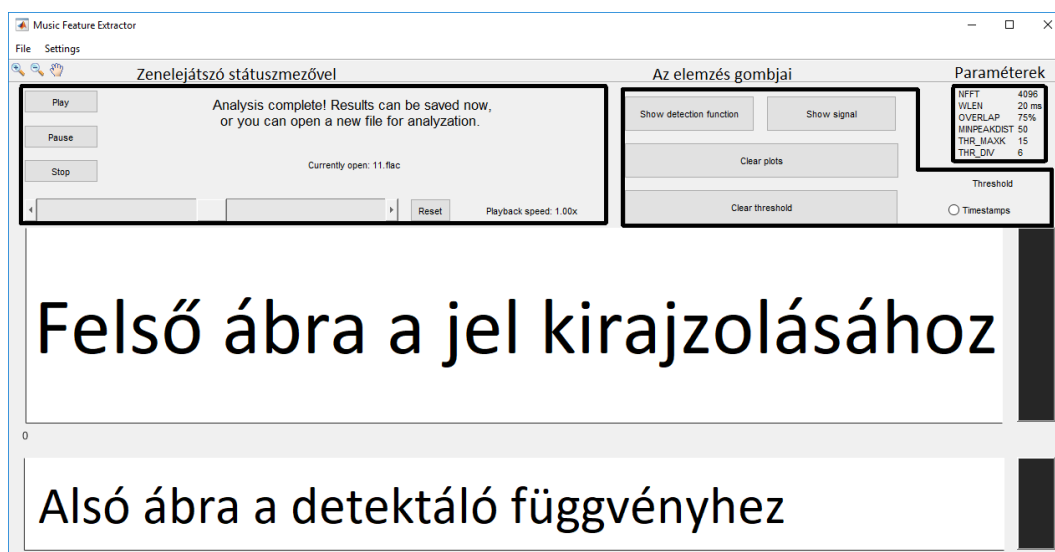
## 2.2. A program használata

A program elindítása után egy üdvözlőképernyő fogad minket, ami az elemzendő zenefájl megnyitására szólítja fel a felhasználót. Erre két módunk van: a *File* menüből (1. ábra) vagy kézzel választjuk ki a fájlt az *Open music file* opció választásával, vagy pedig egy korábbi elemzés eredményét tölthetjük be a *Load analysis results* menüponttal. Amennyiben változtatni szeretnénk az analízis paraméterein, a *Settings* menüben állíthatjuk be őket.

Open music file	Ctrl+O
Save analysis results	Ctrl+S
Load analysis results	Ctrl+L

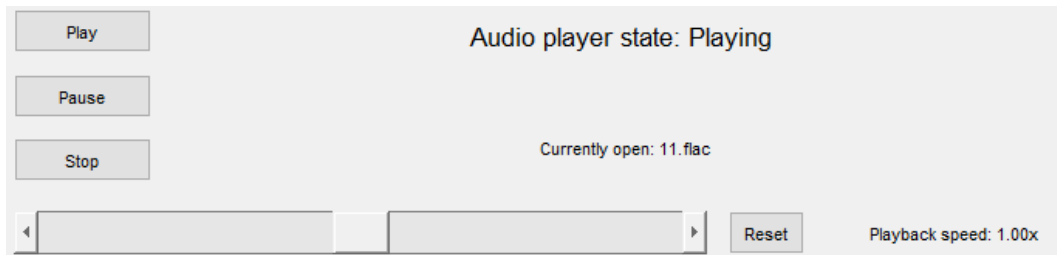
1. ábra. A *File* menü

Új fájl megnyitása esetén automatikusan elindul a bemenet feldolgozása, aminek az állapotáról a folyamatosan frissülő státuszmezőből tájékozódhatunk. Meglévő eredményeket betöltve, vagy a friss elemzés végeztével megjelennek a rajzolásért felelős gombok, illetve a lejátszó vezérlőgombjai.



2. ábra. A program teljes kezelőfelülete

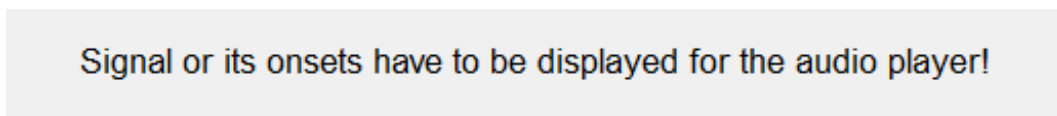
## 2.3. A zenelejátszó



3. ábra. A lejátszó vezérlőgombjai

Miután befejeződött a betöltött fájl feldolgozása, elérhetővé válik a zenelejátszó (3. ábra). A megnyitott fájl neve a státuszmező alatt látható. A *Play/Pause/Stop* gombok funkciója magától értetődő, a csúszka a lejátszási sebességet változtatja, aminek az aktuális értékét a *Playback speed: ...* felirat mutatja; a *Reset* gomb pedig ezt az értéket állítja vissza az alapértelmezettre. Fontos megjegyezni, hogy a lejátszási sebesség változtatása hangszínelváltozást idéz elő, ami az elemzést nem befolyásolja; ennek a feature-nek az elsődleges haszna a zeneműben való navigálás lehetővé tétele, illetve a tempó lassításával a sűrű részekben is könnyen ellenőrizhetjük az algoritmus hatékonyságát.

A lejátszás megkezdéséhez szükséges a zenefájl hullámformájának vagy a hangkezdeteinek a megjelenítése a felső rajzoló felületen - enélkül hibaüzenetet kapunk (4. ábra). A lejátszás során egy zöld vonal jelzi hogy mennyi van hátra a darabból, ami *Stop* gomb hatására visszaugrik a felület elejére.

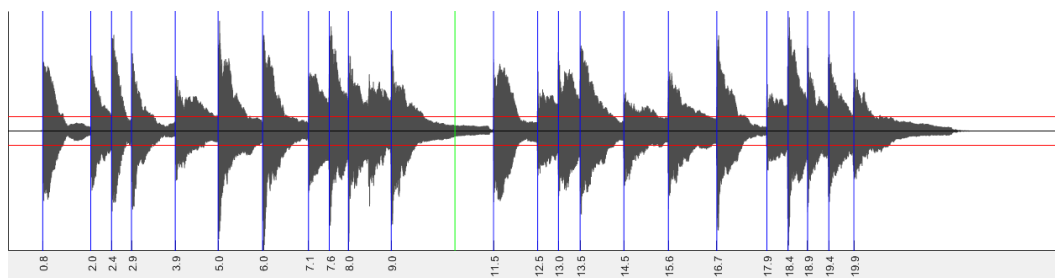


4. ábra. Lehetséges hibaüzenet

## 2.4. Az adatok ábrázolása

Az alkalmazásban megjeleníthetjük a megnyitott hangfájl hullámformáját (*Show signal*) a felső részen, illetve az alsó részen azt a detektáló függvényt, ami az elemzés során generálódott (*Show detection function*).

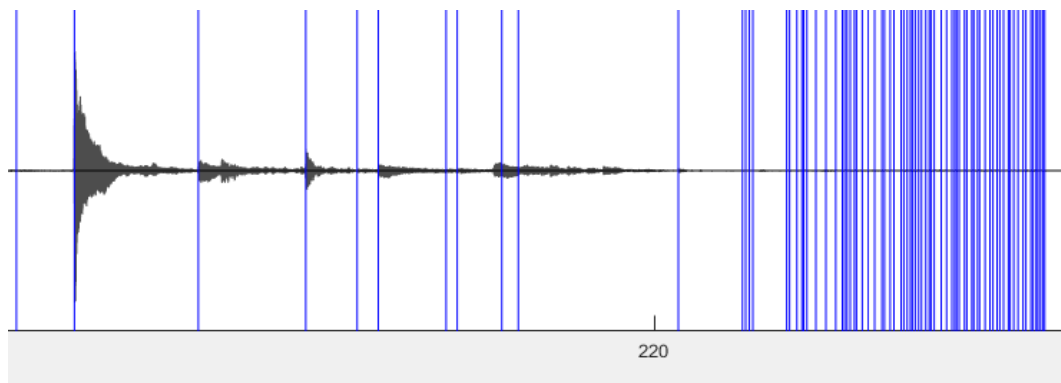
Az ábrák mellett jobbra található egy-egy fekete gomb, amiknek a megnyomására megjelennek az elemző algoritmus által detektált hangkezdetek. A *Timestamps* választógomb azt szabályozza, hogy a felső ábra vízszintes tengelyén a beosztások a hangkezdetekhez illeszkedjenek-e, vagy pedig a darab hosszától függő, egyenlő távolságközönként (például egy 3 perces darab esetében 20 másodpercenként lesz egy vonás, míg egy fél perces mű esetén ez a távolság 5 másodperc) helyezkedjenek el.



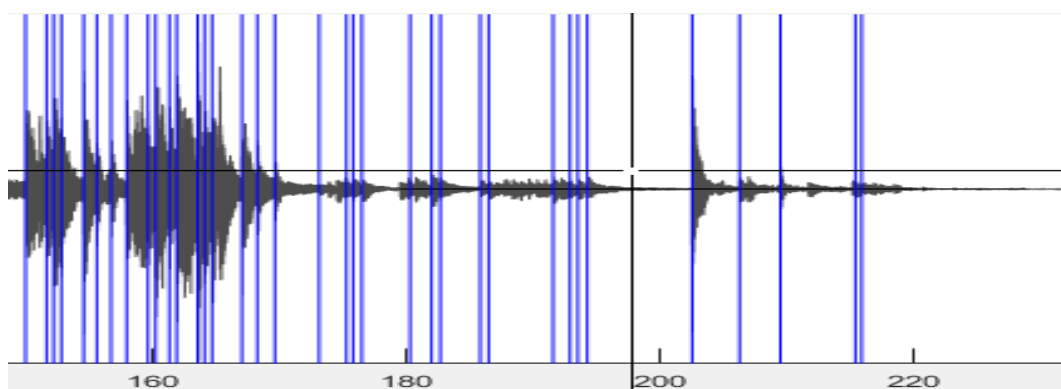
5. ábra. A jel hullámformája, a hangkezdetek és a küszöb

A 5. ábrán szürkével látható a hullámforma, kékkel a hangkezdetek, illetve piros vonallal a felhasználó által beállított küszöb - azok az észlelt hangkezdetek, amelyeknél a jel amplitúdója ezen érték alatt van, törlésre kerülnek.

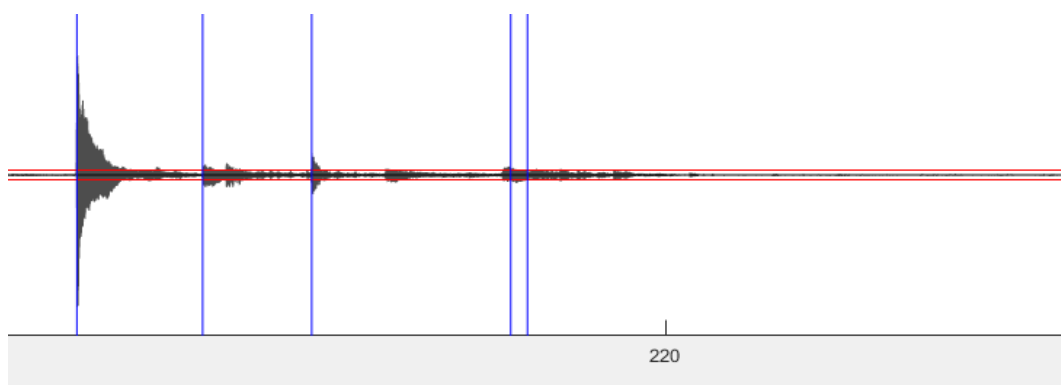
Az algoritmus olyan pontokat keres, ahol a jel a pillanatnyi állapotához viszonyítva hirtelen sokat változik. Ez azt eredményezi, hogy nem megfelelő paraméterezés esetén kifejezetten érzékeny lesz az elemzés a halk részeknél (6. ábra), de ezt egy kellően alacsony küszöbértékkel a helyes hangkezdetek elvesztése nélkül kijavíthatjuk (8. ábra). A küszöbérték beállításához kattintsunk a felső ábra hátterén, és használjunk az így megjelenő célkeresztet. (7. ábra)



6. ábra. Túlérzékenységet okozó paraméterek, alul a detektáló függvény látható



7. ábra. A küszöbértéket egy célkereszt segítségével állíthatjuk be

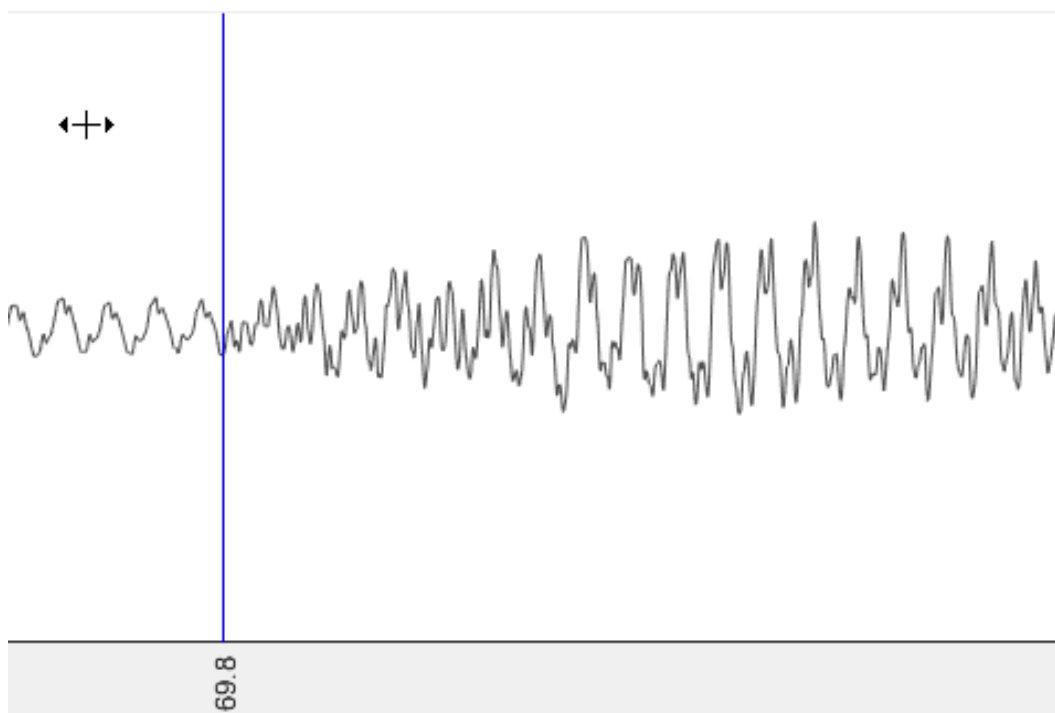


8. ábra. A javított eredmény. Látható, hogy a detektáló függvény nem változott



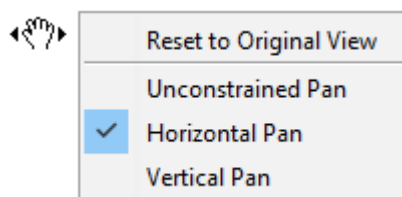
## 2.5. Navigáció

Az ábrákon navigálni a bal felső sarokban található eszköztár segítségével lehet - a használatuk a Matlabban jártas olvasónak már ismerős lesz. A nagyítók hatására a kurzor átváltozik (9. ábra), és a vízszintes tengely mentén közelíthetjük vagy távolíthatjuk az ábrákon lévő képet.



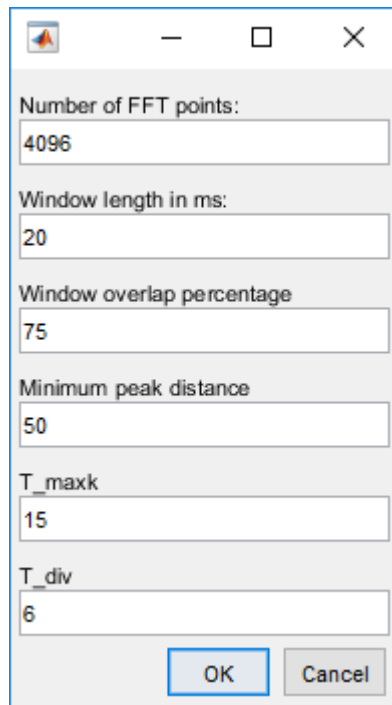
**9. ábra.** Megfelelő nagyítást alkalmazva láthatjuk, hogy miért is hívjuk hullámformának ezt az ábrázolásmódot

A jobb egérgomb hatására előjön egy menü, aminek segítségével visszaállíthatjuk az ábra eredeti kinézetét, illetve módosíthatjuk a navigációs módot. A pásztázó használatakor érdemes beállítani, hogy csak vízszintes irányban tudjunk vele mozgatni, így nem fog elcsúszni kép. (10. ábra)



**10. ábra.** A pásztázó ikon és a menüje

## 2.6. Az elemzési paraméterek kiválasztása



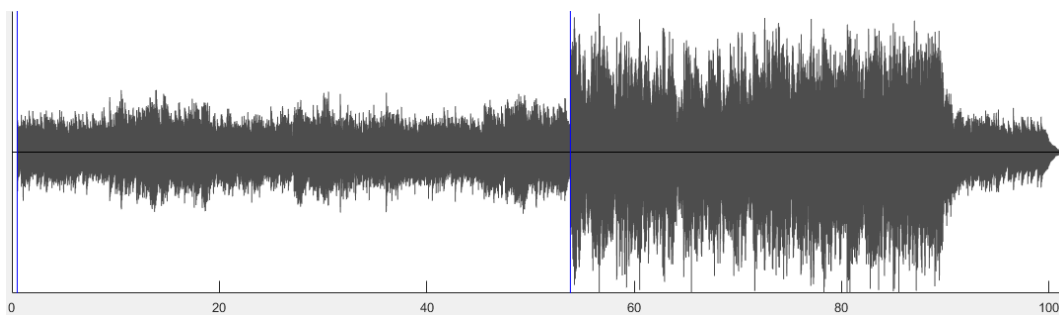
A screenshot of a software dialog box for selecting analysis parameters. The dialog has a title bar with standard window controls. It contains several input fields with the following labels and values:

- Number of FFT points: 4096
- Window length in ms: 20
- Window overlap percentage: 75
- Minimum peak distance: 50
- T\_maxk: 15
- T\_div: 6

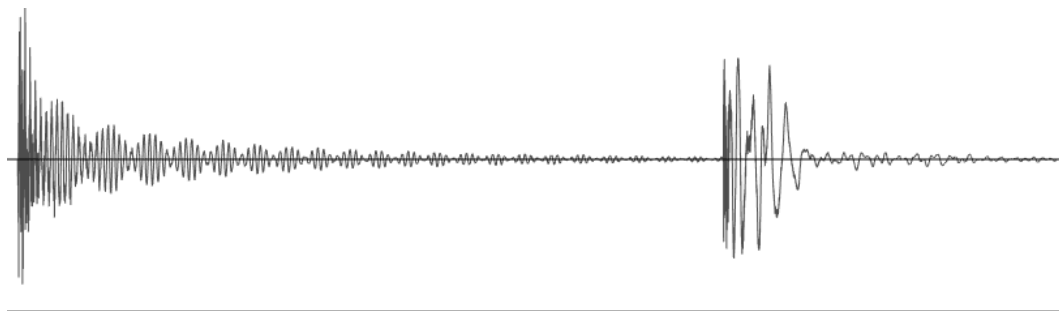
At the bottom right, there are two buttons: "OK" and "Cancel".

11. ábra. A módosítható paraméterek

Bár nincs olyan recept, ami minden esetben működik, mégis kijelölhetünk néhány irányvonalat, melyeket követve felgyorsíthatjuk az elemzés folyamatát. Először is kénytelenek vagyunk alávetni magunkat néhány limitációnak: az algoritmus akkor teljesít jól, ha a hangkezdetek jelentős energiaváltozással járnak (például zongora vagy gitár esetében), vagy ha *transziens* (rövid lecsengési idejű) hangok szólalnak meg (az ütős hangszerek esetében tipikusan ez történik). (13. ábra)



12. ábra. Kousaki Satori - Hyouri: Ennek a jelnek az energiája csupán kétszer változik jelentősen, így az algoritmus is csak a jel elejét és az 55. másodperc körüli váltást érzékelt



**13. ábra.** Tom-tom illetve lábdob hullámformája

Ezen felül az optimális eredmény érdekében kikötjük, hogy egy hangszeres, lehetőleg egyszólamú műveket elemzünk, és elsősorban zongorával foglalkozunk. Ekkor 20-30 milliszekundumos ablakméretet és 50-80%-os átfedést érdemes használnunk a rövid idejű Fourier-transzformáció során [1], illetve a legtöbb műnél feltehetjük, hogy legalább 50 milliszekundum távolság van két hangkezdet között. A THR\_MAXK és a THR\_DIV paraméterek segítségével az analízis érzékenységét szabályozhatjuk: az előbbi egyenesen, az utóbbi fordítottan arányos az észlelt hangkezdetek számával. Az ezekhez kapcsolódó pontos képlet a fejlesztői dokumentációban található.

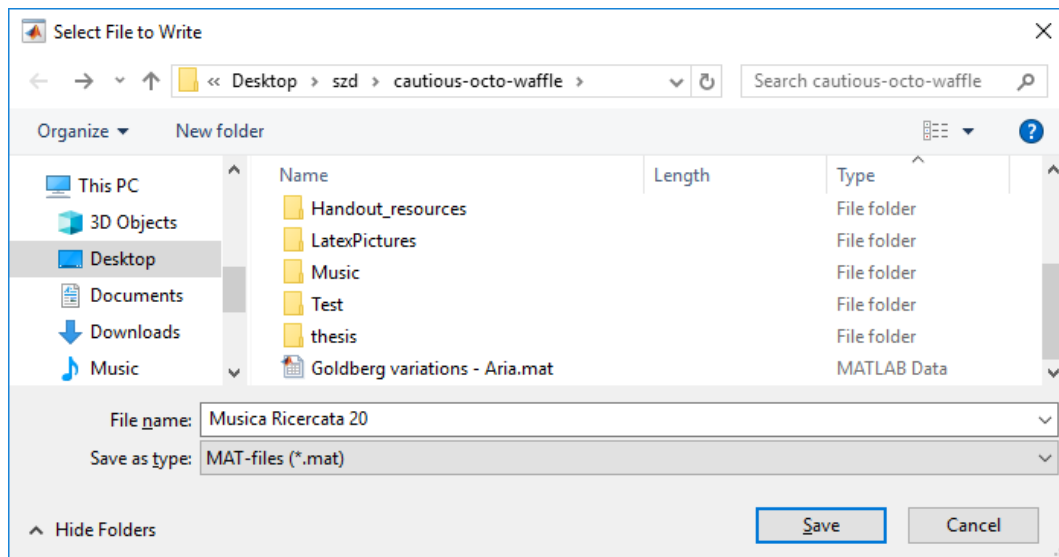
NFFT	4096
WLEN	20 ms
OVERLAP	75%
MINPEAKDIST	50
THR_MAXK	15
THR_DIV	6

**14. ábra.** A paraméterek alapértelmezett értéke

## 2.7. Az eredmény tárolása és betöltése

Az elemzés végeztével a *File* menüben *.mat* kiterjesztésű fájlokba menthetjük az eredményt, vagy betölthetünk már eltárolt fájlokat a *Load analysis results* menüpont segítségével. Az így létrehozott adatok tartalmazzák magát a hangfájlt, a feldolgozás során kiszámolt segédváltozókat (például a detektáló vagy az energiaburok-függvény) és a kinyert hangkezdeteket.

Ebből adódóan egy régi eredmény betöltése után egyből elkezdhetünk dolgozni, hiszen ehhez minden szükséges információ a program rendelkezésére áll. Az elemzési paraméterek automatikusan elmentődnek minden változtatáskor, így a felhasználónak nem kell ezzel foglalkoznia.



**15. ábra.** A mentés során a felhasználó kiválaszthatja a fájlnevet, így egyszerre több eredményt is tárolhatunk

## 3. Fejlesztői dokumentáció

### A megoldandó feladat

A hangkezdet-észlelés problémája a következő: adott egy zeneműről készült hangfelvétel - határozzuk meg, hogy mely időpillanatokban képződött zenei hang! A megoldást elsősorban szóló, lehetőleg egyszólamú zongoraművekre készítettem. A tervezés során szempont volt, hogy a program ne legyen feleslegesen nagy hardverigényű, illetve hogy a felhasználó, aki kottakészítés közben megerősítésként használja a szoftvert, különösebb előismeretek nélkül is hatékonyan tudja használni azt.

### 3.1. Előismeretek

#### 3.1.1. Hangok reprezentációja

**Hang:** Az, amit hangnak nevezünk, nem más, mint egy rugalmas közegben terjedő mechanikai rezgéshullám. Ez a közeg általában a levegő szokott lenni, de lehet víz, vagy példának okáért vas is.

**Zenei hang:** Periodikus hanghullám. (16. ábra). Négy fő jellemzője van: hangmagasság, hangerő, időtartam és hangszín.



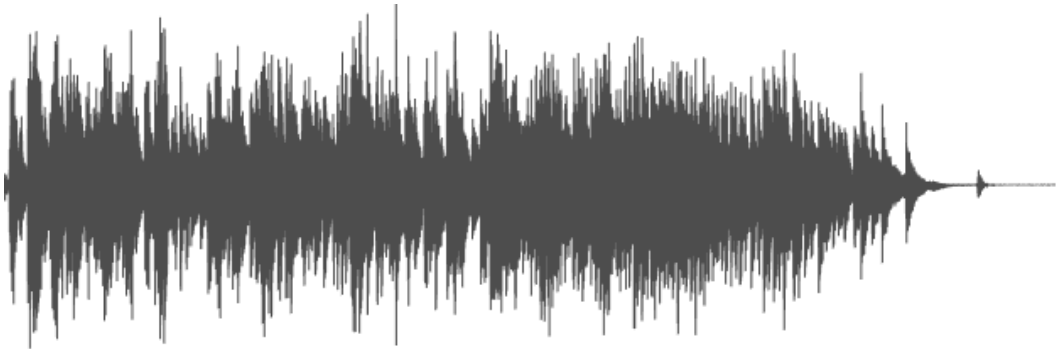
16. ábra. Szinuszhullám

A zenei hangokra tekinthetünk úgy, mint folytonos és periodikus függvényekre. Ezek rendelkeznek frekvenciával és amplitúdóval - az előbbiből a hang magasságát határozhatjuk meg (minél nagyobb a frekvencia, azaz minél gyorsabban rezeg a közeg, annál magasabb), az utóbbiból pedig a hangerőt (minél nagyobb az amplitúdó, annál hangosabb a jel).

**Mintavételezés:** Azt a folyamatot, melynek során az analóg jelből egy diszkrét értékek sorozatát állítjuk elő, mintavételezésnek hívjuk. Hangok esetében a jel pillanatnyi amplitúdóját rögzítjük valamilyen gyakorisággal (mintavételezési ráta), azonos időközönként.

**Membrán:** Vékony, rugalmas lemez, amely rezgésével hangokat kelt. Hangrögzítő eszközökben és hangszórókban használatos.

A digitálisan tárolt hangfájlok a mintavételezés során generált,  $-1$  és  $1$  közötti értékeket tartalmazzák a mintavételezési gyakorisággal együtt, és ez pontosan elég információ ahhoz, hogy a hangszórók membránját rezegtetve reprodukáljuk az eredeti jelet. Adja magát tehát az ötlet, hogy a hangot az eltelt idő függvényében reprezentáljuk (17-18. ábra).



**17. ábra.** Részlet Bach Goldberg-variációiból



**18. ábra.** Az előző ábra felnagyítva

### 3.1.2. Az elemzés során használt módszerek

Az előbb felvetett időtartománybeli reprezentáció nem alkalmas arra, hogy zenei információt nyerhessünk ki egy hangfájlból [1]. A hangokat elsősorban a hangmagasságuk alapján tudjuk megkülönböztetni, amit az (alap)frekvenciájuk határoz meg, és erről ez az ábrázolásmód semmilyen információt nem tartalmaz. Ahhoz, hogy az elemzésünket megkezdhesük, az adatokat át kell vinnünk a *frekvenciatarományba*.

**Fourier-transzformáció:** Legyen az  $f: \mathbb{R} \mapsto \mathbb{C}$  függvény négyzetesen Lebesgue-integrálható a  $[-\infty, +\infty]$  intervallumon. Ekkor az  $f$  Fourier-transzformáltja:

$$\mathcal{F}[f](x) = F(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t) e^{-2\pi i x t} dt.$$

A Fourier-transzformáció segítségével meghatározhatjuk, hogy egy függvényt milyen frekvenciakomponensek alkotnak. A probléma az, hogy jelfeldolgozásban mintavételezett jelekkel dolgozunk, így nem rendelkezünk az eredeti folytonos függvénnyel, csak a kvantált értékekkel. Ezeknek az átalakításához a transzformáció diszkrét változatára van szükségünk.

**Diszkrét Fourier-transzformáció:** Legyen  $x \in \mathbb{C}^N$  egy tetszőleges vektor. Ekkor az

$$X[k] = \frac{1}{\sqrt{N}} \sum_{n=0}^N x[n] e^{\frac{-2\pi i n k}{N}} \quad (k = 1..N)$$

vektor az  $x$  diszkrét Fourier-transzformáltja.

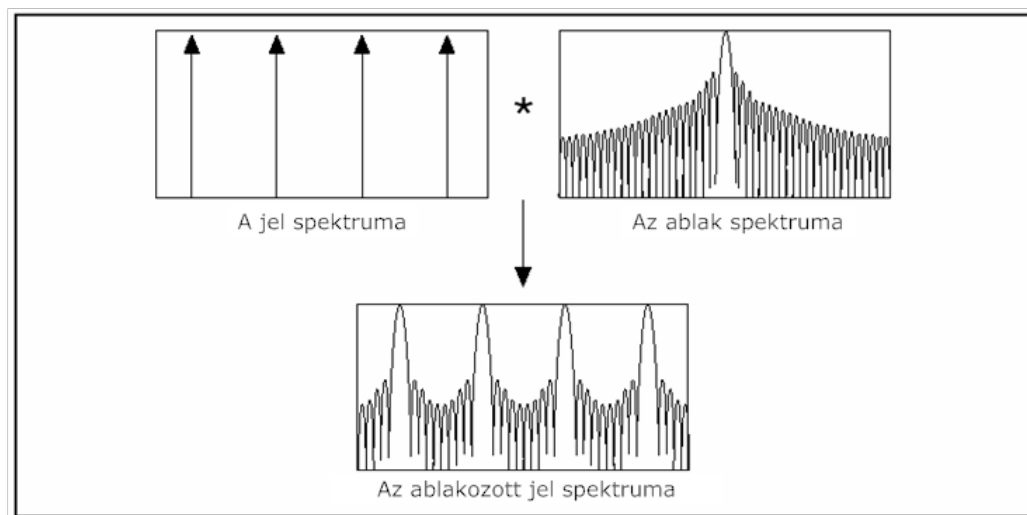
**Shannon mintavételi tétele:** Legyen a jelben szereplő legnagyobb frekvencia  $f_{max}$ . Ekkor a jel tökéletesen rekonstruálható, ha a mintavételezési frekvencia legalább  $2 \cdot f_{max}$ .

**Nyquist-frekvencia** A mintavételezési ráta fele a legnagyobb frekvencia, amit a mintavételezett jelből elő tudunk állítani. Ezt az értéket hívjuk Nyquist-frekvenciának.

Shannon mintavételi törvénye az oka annak, hogy a CD-lemezek mintavételezési frekvenciája 44100 Hz, ugyanis az emberi hallástartomány körülbelül 20 Hz-től 20 000 Hz-ig terjed, így a felvételekből elvileg tökéletesen előállíthatóak a zeneművek.

A diszkrét Fourier-transzformáltból ugyan meghatározhatjuk a jel frekvencia-komponenseit, de a zenei elemzéshez ez még mindig használhatatlan, hiszen minden időre vonatkozó információt elvesztettünk a transzformáció során. Ez azt jelenti, hogy kaptunk egy, a mű egészére vonatkozó leírást a benne megszólaló hangokról, de a hangkezdet-észleléshez ennél többre van szükségünk; muszáj megtartanunk elegendő temporális adatot ahhoz, hogy a billentyűleütéseket egyenként detektálni tudjuk.

**Ablakfüggvény:** Ha az STFT során közvetlenül a szegmensekre bontás után elvégeznénk a Fourier-transzformációt, egy torzult, hamis spektrumot kapnánk (19. ábra). Ennek kiküszöbölésére használunk ablakfüggvényeket, melyek jellemzően pozitív, szimmetrikus és véges tartójú függvények (sőt, az értékük szegmens határain kívül 0). Az ablakfüggvény típusa függ az elvégzendő feladattól, a szakdolgozat által tárgyalt alkalmazásban Hamming-ablakot használtam.



**19. ábra.** Az ablakfüggvények a spektrum helyreállítására törekednek (Forrás: [13])

A megoldást a rövid idejű Fourier-transzformáció (STFT) fogja nyújtani, melynek során a jelet felosztjuk szegmensekre (hangkezdet-észlelés esetében egy szegmens 20-50 msec-nyi adatot tartalmaz), majd minden egyes ilyen szakaszra alkalmazzuk a diszkrét Fourier-transzformációt valamilyen ablakfüggvénnyel (19. ábra).

A fejezet megírásához a [14] és a [13] oldalak nyújtottak segítséget.



## 3.2. Megoldási terv

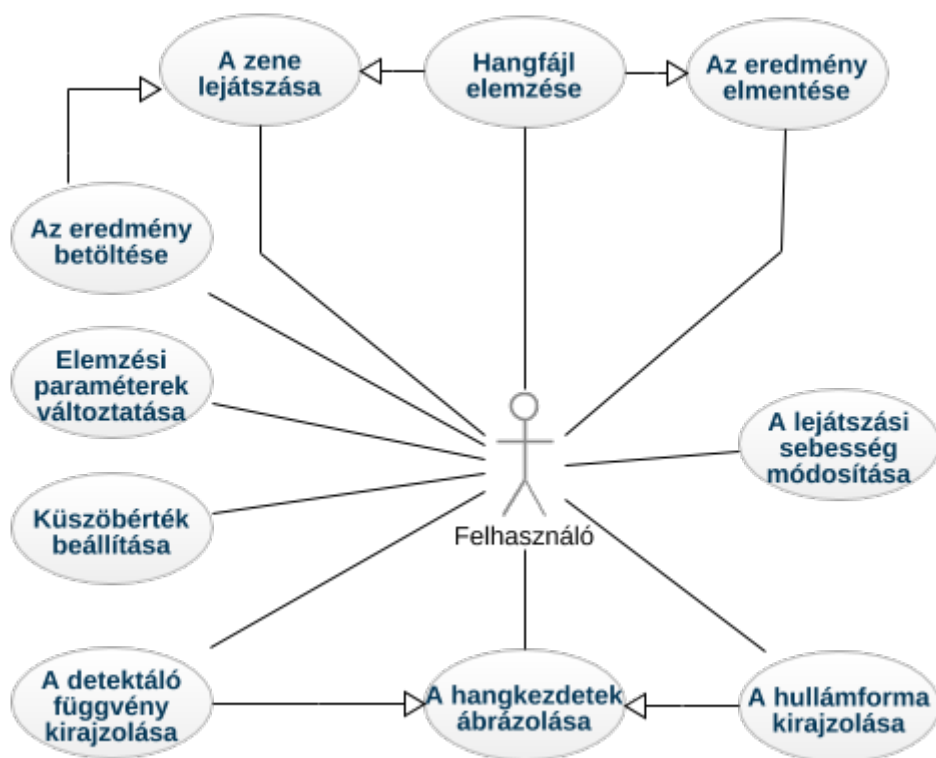
A feladat tehát a zenei hangok kezdeteinek megkeresése egy zongoradarabban. A megoldáshoz szerettem volna egy olyan grafikus felületet is készíteni, ami az észlelt hangkezdetek megjelenítésével jó támpontot ad egy zenei elemzéshez. Az adatok ábrázolása után az alkalmazás lehetőséget nyújt küszöb beállítására, mellyel a fals pozitívok drasztikusan csökkenthetőek optimalizálatlan elemzési paraméterek esetén.

Az implementációhoz a *MATLAB* programozási nyelvet, a felhasználói interfészhez a hozzá tartozó *GUIDE* eszközt választottam. Futási idő szempontjából a gyors vektorműveletek és a jól optimalizált *MATLAB* könyvtárak kulcsfontosságúnak bizonyultak, a fejlesztési idő pedig a magas színvonalú fejlesztőkörnyezetnek és a gyors prototípus-készítés lehetőségének köszönhetően könnyen irányítható volt.

### Követelmény-specifikáció

Az alkalmazás...

1. támogassa a felhasználói esetekként definiált funkciókat (20. ábra).
2. legyen könnyen használható az elemzés működésének megértése nélkül is.
3. ne használjon feleslegesen sok erőforrást: normál használat mellett a memóriaigény maradjon 1 GB alatt, és egy fél perces fájl elemzése ne tartson tovább pár másodpercnél egy alsó kategóriás üzleti laptopon.
4. nyújtson módot arra, hogy az eredményeket könnyen ellenőrizhessük kézzel; az ábrázolás legyen egyértelmű.
5. tárolja el az elemzési paramétereket, ne kelljen őket minden újraindításkor beállítani.
6. rendelkezzen olyan dokumentációval, amiben könnyen megtalálhatóak az egyes funkciók, illetve a mögöttük lévő elméleti háttér magyarázata.



20. ábra. Felhasználói esetek



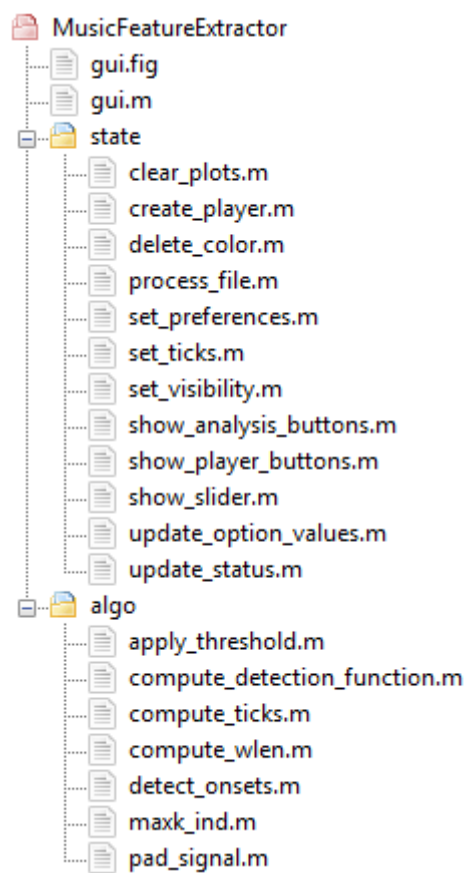
21. ábra. Drótvázterv

### 3.3. Megvalósítás

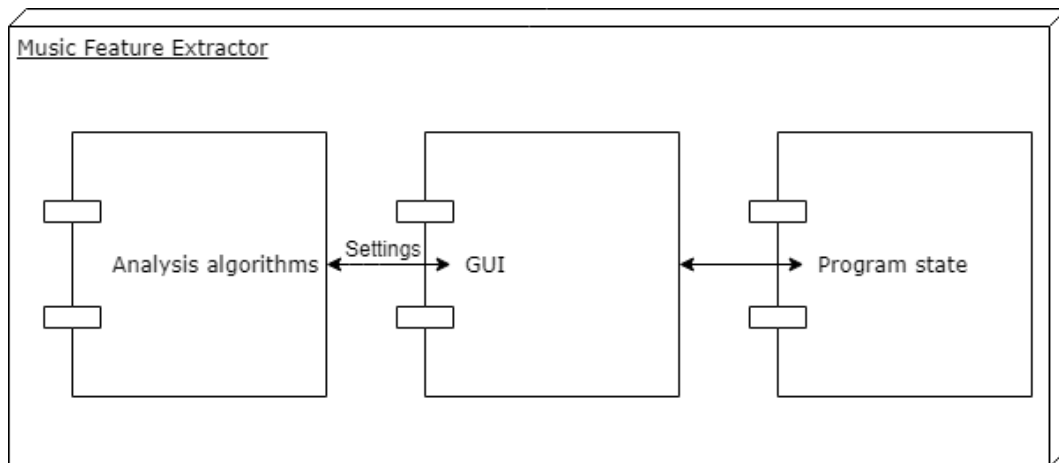
A szoftver három fő komponensre oszlik (22., 23. ábra).

1. A GUIDE segítségével készített *grafikus interfész* struktúráját a **gui.fig**, a callbackjeit a **gui.m** fájl tartalmazza.
2. Az *elemzési algoritmus* és a segédfüggvényei az **algo** könyvtárban találhatóak.
3. A program *állapotkezelése*, illetve a GUI-n található információk frissítése a **state** könyvtárban lévő függvényekkel történik.

A programban az egyes részek felelősségei jól elkülönülnek, így könnyen bővíthető új funkciókkal, illetve az algoritmusokat is egyszerűen javíthatjuk, cserélhetjük a felépítésnek köszönhetően. A MATLAB által támogatott, más programnyelveken implementált függvényekhez (a sebesség miatt elsősorban C-ről lehet szó) érdemes lehet egy új, **lib** nevű könyvtárral bővíteni a projektet.



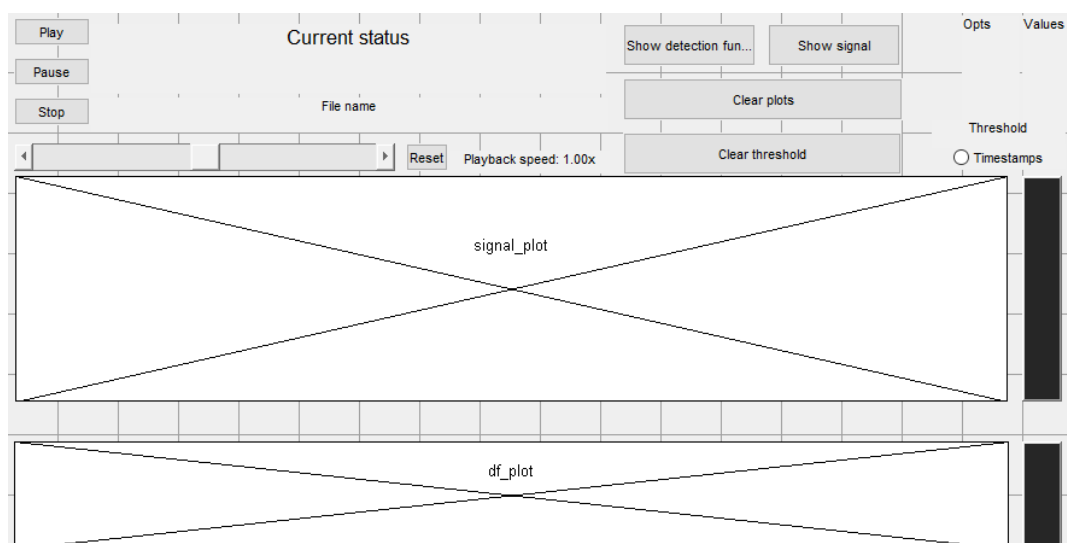
22. ábra. A program mappaszerkezete



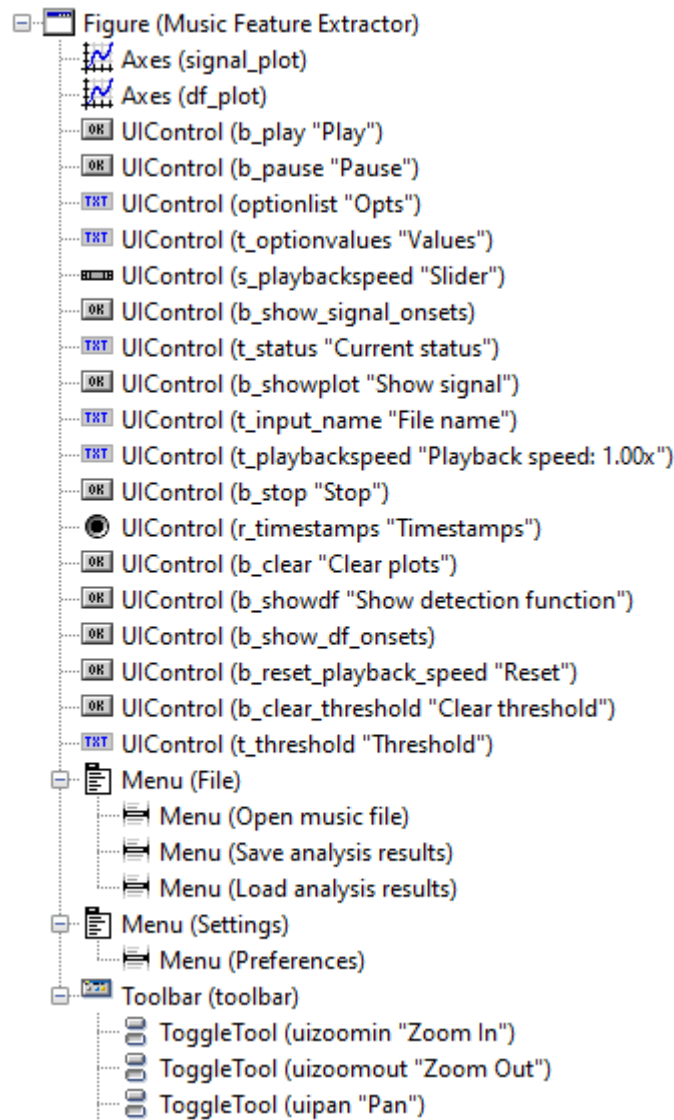
23. ábra. Komponensdiagram

### 3.3.1. A GUI komponens

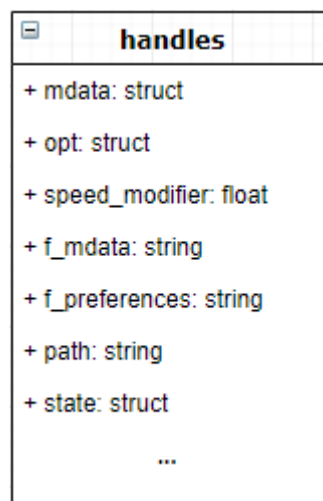
A felhasználói felület objektumait a GUIDE által generált *gui.fig* fájl tartalmazza, amit induláskor tölt be a szoftver. Ennek a kiterjesztésnek a felépítése megegyezik a *\*.mat* fájlokéval. A program által definiált objektumok a 24. és 25. ábrán láthatóak.



24. ábra. A GUI szerkezete



25. ábra. A GUI objektumai



26. ábra. A handles struktúra

A MATLAB-bal készült grafikus alkalmazások jellemzője, hogy tartalmaznak egy *handles* nevű struktúrát (26. ábra), aminek a segítségével az egyes függvények és komponensek között át lehet adni a program perzisztens változóit, illetve ez tárolja a felület objektumainak a memóriacímét is. A továbbiakban ennek a struktúrának a szerkezetéről lesz szó.

- Az *mdata* és a *state* változók a másik két komponens adatait tárolják.
- A *speed\_modifier* a lejátszási sebességet állító csúszka értéke – ezt a zenelejátszó létrehozásakor használjuk a mintavételi ráta módosítására.
- Az *f\_mdata* és az *f\_preferences* szöveges változók a legutóbb tárolt adatok elérési útvonalát tartalmazzák.
- A *path* mindig az aktuálisan betöltött hangfájl vagy elemzési eredmény útvonala.
- Az *opt* struktúra (27. ábra) pedig a felhasználó által beállított paramétereket tartalmazza.

## Ábrák és a hozzátartozó gombok

- *signal\_plot*: a jel és a hangkezdetei kirajzolására szolgáló ábra
- *df\_plot*: a detektáló függvény és az eredeti hangkezdetek kirajzolására szolgáló ábra
- *b\_showplot*: a jelet kirajzoló gomb
- *b\_showdf*: a detektáló függvényt kirajzoló gomb
- *b\_show\_signal\_onsets*: a hangkezdeteket felül kirajzoló gomb
- *b\_show\_df\_onsets*: a hangkezdeteket alul kirajzoló gomb

## A zenelejátszó gombjai

- *b\_play*: a lejátszás megkezdése
- *b\_pause*: a lejátszás szüneteltetése
- *b\_stop*: a lejátszás leállítása
- *b\_reset\_playback\_speed*: a lejátszási sebesség visszaállítása az alapértelmezettre

Az *s\_playbackspeed* változó a lejátszási sebességet módosító csúszka *handlerje*.

## Egyebek

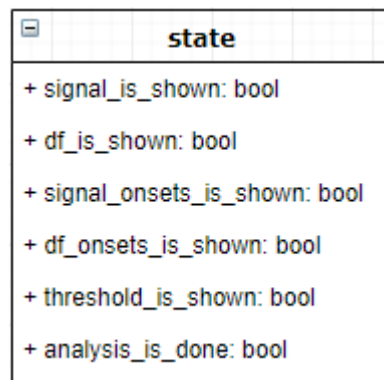
- *t\_optionvalues*: az elemzési paraméterek értékei szöveges formában
- *t\_optionlist*: az elemzési paraméterek nevei
- *t\_status*: a státuszmező szövege
- *t\_input\_name*: az aktuálisan megnyitott fájl neve
- *t\_playback\_speed*: a lejátszási sebesség szorzója
- *r\_timestamps*: az egyes hangkezdetek idejeinek a kijelzését kapcsolja ki/be
- *b\_clear\_threshold*: a felhasználó által beállított küszöbértéket törli
- *b\_threshold*: a küszöbérték szövegesen

opt	
+ nfft:	integer
+ wlen_ms:	integer
+ overlap_pct:	integer
+ minpeakdist_ms:	integer
+ thr_maxk:	integer
+ thr_div:	integer
+ threshold:	double

**27. ábra.** Az opt struktúra - a paraméterek jelentése a felhasználó dokumentációban található

### 3.3.2. A state komponens

Ennek a komponensnek a feladata a felhasználói felület és a tárolt adatok frissítése. Elsősorban úgynevezett *wrapper* függvényeket tartalmaz, melyeknek köszönhetően kifejezőbbé válik a kód többi része, és nem kell több helyen módosítani a programot, ha a megjelenítés módján változtatni akarunk. A GUI komponens callbackjei rendszeresen lekérlik ezeket az értékeket annak érdekében, hogy kiválasszák a végrehajtandó műveletet (például nem kezdünk el újra kirajzolni egy ábrát, ha az már a képernyőn van).



28. ábra. A state struktúra

### A program logikai állapotváltozói

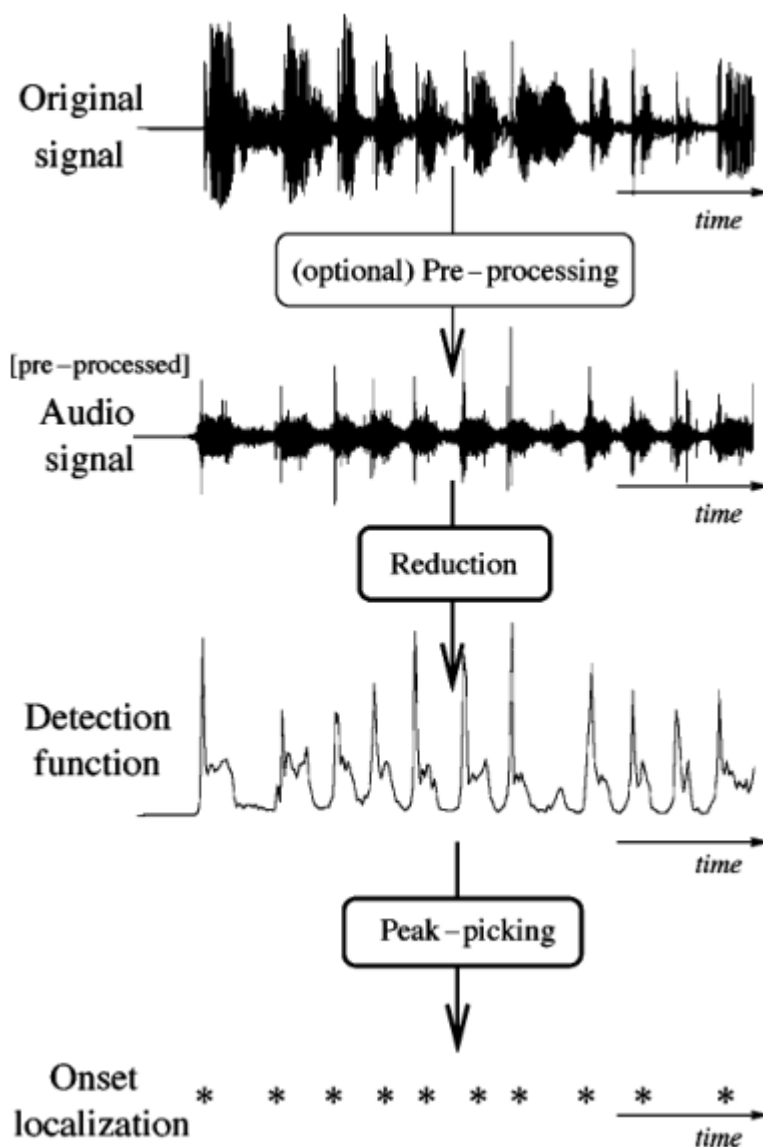
A *state* struktúra egyes elemei azt a kérdést válaszolják meg, hogy...

- *signal\_is\_shown*: a jel maga jelen van-e a képernyőn
- *signal\_onsets\_is\_shown*: a jel hangkezdetei jelen vannak-e a képernyőn
- *df\_is\_shown*: a detektáló függvény jelen van-e a képernyőn
- *df\_onsets\_is\_shown*: az eredeti hangkezdetek jelen vannak-e a képernyőn
- *threshold\_is\_shown*: a küszöb jelen van-e a képernyőn
- *analysis\_is\_done*: lefutott-e már legalább egyszer az elemzés



### 3.3.3. Az algo komponens

Az alkalmazás középpontjában a hangkezdet-észlelő algoritmus áll, melyet a [11], [15] publikációk és az [1] könyv alapján készítettem. Ezek jellemzően három részre oszlanak (29. ábra). Először egy előfeldolgozó szakaszban felkészítjük a bemenetet az elemzéshez (például zajszűrés alkalmazásával) - erre a programom esetében nem volt szükség, hiszen hanglemezekről származnak a felvételeink. Utána a hangjelet valamilyen módszerrel olyan függvénné alakítjuk át, ami az időtartománybeli reprezentációhoz képest sokkal jobban tükrözi a mű során zajló *zenei eseményeket* a változások kiemelésével. Az így kapott *detektáló függvény* hibáit *küszöböléssel* javítjuk, végül a csúcsok kiválasztásával megkapjuk a hangkezdeteknek megfelelő időpillanatokat.



29. ábra. A hangkezdet-detektálás lépései (Forrás: [11])

```

1 % set user-given variables
2 NFFT          = handles.opt.nfft;
3 WLEN_MS       = handles.opt.wlen_ms;
4 OVERLAP       = (0.01 * double(handles.opt.overlap_pct));
5 MINPEAKDIST_MS = handles.opt.minpeakdist_ms;
6
7 % sampling rate
8 fs           = handles.mdata.fs;
9 % compute the window length and pad the signal accordingly
10 [wlen, hop]   = compute_wlen(length(handles.mdata.x), ↵
    fs, WLEN_MS, OVERLAP);
11
12 % we only store the length after padding - this will not ↵
    be the
13 % original length of the audio file but the difference ↵
    is negligible
14 x            = pad_signal(handles.mdata.x, hop);
15 xlen         = length(x);
16 xlen_sec     = length(x) / fs;
17
18 update_status(handles, 'Transforming the signal...');
19
20 [detfunc, time, energy, loc_x] = ↵
    compute_detection_function(handles, x, wlen, hop, ↵
    NFFT, fs);
21
22 update_status(handles, 'Thresholding...');
23
24 detfunc       = apply_threshold(detfunc, hop, ↵
    handles.opt.thr_maxk, handles.opt.thr_div);
25
26 update_status(handles, 'Picking the peaks...');
27 drawnow();
28
29 minpeakdist   = MINPEAKDIST_MS * length(detfunc) / ↵
    (xlen_sec * 1000);
30 [peaks, loc]  = findpeaks(detfunc, ↵
    'MinPeakDistance', minpeakdist);

```

detect\_onsets.m (részlet)

**A detektáló függvény kiszámítása:** A detektáló függvény kiszámítása a következő módon történik:

1. Az előre megadott ablakmérettel alkalmazzunk Hamming-ablakozást a jelre [16].
2. Képezzük így a jel rövid idejű Fourier-transzformáltját a *gyors Fourier-transzformáció* algoritmus segítségével
3. Számoljuk ki a jel energiaburok-függvényét:

$$E(n) = \sum_{k \in bins} |\mathbf{STFT}_x^w(n, k)|^2$$

(ahol *bins* a frekvenciabineket, az  $\mathbf{STFT}_x^w(n, k)$  pedig az  $x$  jel  $w$  ablakozású rövid idejű Fourier-transzformáltjában az  $n$ . oszlop  $k$ . frekvenciabinjéhez tartozó értéket jelöli)

4. Vegyük az így kapott függvény három ponton átmenő lineáris regresszióját [1]:

$$D(n) = \frac{E(n+1) - E(n-1)}{3}$$

5. És végül vegyük a jel logaritmusának az idő szerinti deriváltját [11]:

$$\frac{d(\log(E))}{dt} = \frac{\frac{dE}{dt}}{E} \approx \frac{D}{E}$$

Vegyük észre, hogy a negyedik lépésben tulajdonképpen a jel energiafüggvényének a meredekségét számoljuk ki. Az  $E(n)$  függvény szoros összefüggésben van a zene hangerejének a változásával, amit az emberi fül *logaritmikusan* érzékel [11] (ezt szemlélteti a decibel skála is). Ez a magyarázata annak, hogy az energiaburok logaritmusának a deriváltját használjuk detektáló függvényként.

```

1 function [difflog, time, energy, loc_x] = ↵
    compute_detection_function(handles, x, wlen, hop, ↵
        nfft, fs)
2 %Calculate onset detection function and its time vector
3 %using linear regression.
4 %
5 %Input parameters:      x      - the music signal
6 %                      wlen    - window length in frames
7 %                      hop     - hop size in frames
8 %                      nfft    - number of fft points
9 %                      fs     - file sampling rate in Hz
10 %
11 %Output parameters:    difflog - the detection function
12 %                      time     - the corresponding time
13 %                      vector
14 %                      energy   - the energy envelope
15 %                      values
16 %                      loc_x    - the maximum signal
17 %                      value in the windows
18 %                      of the peak locations
19
20 xlen      = length(x);
21 win       = hamming(wlen, 'periodic');
22 rown      = ceil(nfft/2);
23 coln      = floor((xlen-wlen)/hop);
24 energy    = zeros(1, coln);
25 index     = uint32(1);
26 %the maximum signal values in the peak location windows
27 loc_x     = zeros(1, coln);
28
29 % We will update the user four times to give a sense of ↵
    progress
30 % The loop is broken up this way because updating the ↵
    status in every
31 % iteration significantly slows the process down
32 F = floor(coln/4);
33
34 update_status(handles, 'Transforming the signal... ([ ] ↵
    [ ] [ ] [ ] )');
35
36

```

```

37 for col = 1:F
38     xw = x(index:index+wlen-1).*win;
39     X = fft(xw, nfft);
40     energy(col) = sum(abs(X(1:rown)));
41     loc_x(col) = max(abs(x(index:index+wlen-1)));
42     index = index + hop;
43 end
44 update_status(handles, 'Transforming the signal... ([X] ←
    [ ] [ ] [ ] )');
45
46 <...>
47
48 update_status(handles, 'Transforming the signal... ([X] ←
    [X] [X] [X] )');
49
50 diff = zeros(1, coln);
51 difflog = zeros(1, coln);
52 %square the energy in order to gain the envelope
53 energy = energy.*energy;
54
55 energy(energy<0.1) = 0.1; %avoid division by zero
56 for i = 2:coln-1
57     diff(i) = (energy(i+1) - energy(i-1)) / 3;
58     difflog(i) = diff(i)/energy(i);
59 end
60
61 %the time of each value will be the middle of its window
62 time = (double(wlen)/2:double(hop):double(wlen)/2 + ←
    double(coln)*double(hop)-1) / double(fs);
63
64 end

```

**compute\_detection\_function.m** (a feldarabolt ciklusnak csak az egyik része látszódik a helytakarékoság érdekében)

**A küszöbölés módja:** A detektáló függvényből lokális maximumokat kiválasztva kapjuk a hangkezdet-jelölteket. Ezek közül azokat fogadjuk el, amikre a alábbi függvény értéke 1:

$$Thr(x) = \begin{cases} 0, & \text{ha } Det(x) < movmedian(x, hop) + \frac{\delta}{THR_{div}}, \\ 1 & \text{egyébként.} \end{cases}$$

ahol  $Det(x)$  a detektáló függvény,  $movmedian(x, hop)$  a detektáló függvény értékeinek a mediánja egy  $hop$  nagyságú ablakban az  $x$  körül, a  $THR_{div}$  egy felhasználó által beállított paraméter, és amennyiben a  $maxk(Det, n)$  a legnagyobb  $n$  érték indexe a  $Det$  függvényben, akkor a  $\delta$  az alábbi:

$$\delta = avg(Det(i) - movmedian(i, hop)) \quad (i \in maxk(Det, THR_{maxk}))$$

```

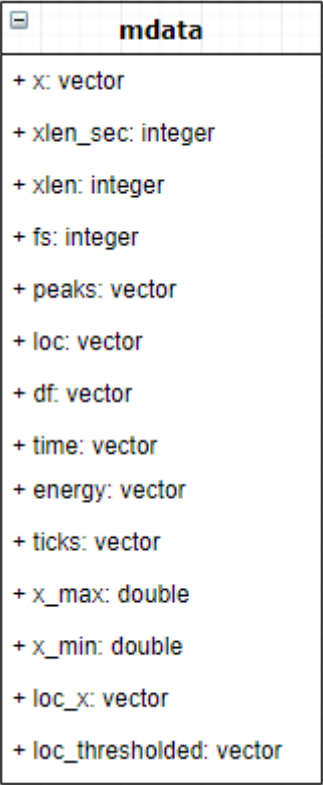
1 function f = apply_threshold(f, hop, k, div)
2 %Apply a threshold to the function in order to
3 %remove insignificant peaks
4 %
5 %Input parameters:      f      - the function
6 %                      hop    - hop size in frames
7 %                      k      - the amount of peaks to
8 %                          select for reference
9 %                      div    - the divisor in the algorithm
10 %Output parameters:    f      - the thresholded function
11     ind    = maxk_ind(f,k);
12     avg    = movmedian(f,hop);
13     delta  = mean(f(ind) - avg(ind));
14
15     f(f < avg + delta/double(div)) = 0;
16 end

```

apply\_threshold.m

## Az adatok tárolása

Az elemzés után az eredmény a handles struktúra mdata változójában található meg.



mdata	
+	x: vector
+	xlen_sec: integer
+	xlen: integer
+	fs: integer
+	peaks: vector
+	loc: vector
+	df: vector
+	time: vector
+	energy: vector
+	ticks: vector
+	x_max: double
+	x_min: double
+	loc_x: vector
+	loc_thresholded: vector

30. ábra. Az mdata struktúra

Az egyes változókban a következő információk találhatóak:

- $x$ : a jel
- $xlen\_sec$ : a jel hossza másodpercben
- $xlen$ : a jel hossza *frame*-ekben
- $fs$ : a mintavételezési ráta
- $peaks$ : a detektáló függvény értékei a csúcsokban
- $loc$ : a hangkezdetek indexei
- $df$ : a detektáló függvény
- $time$ : a detektáló függvényhez tartozó idővektor
- $energy$ : az energiaburok
- $ticks$ : a hangkezdetek idejei másodpercben

- *x\_max*: a jel legnagyobb amplitúdója
- *x\_min*: a jel legkisebb amplitúdója
- *loc\_x*: a hangkezdetek időpontjaiban a jel amplitúdói - a küszöböléskor van szerepük
- *loc\_thresholded*: a felhasználó küszöbölése után megmaradt hangkezdetek

Ezeket a változókat az összes többi komponens is használja.

### 3.4. Tesztelés

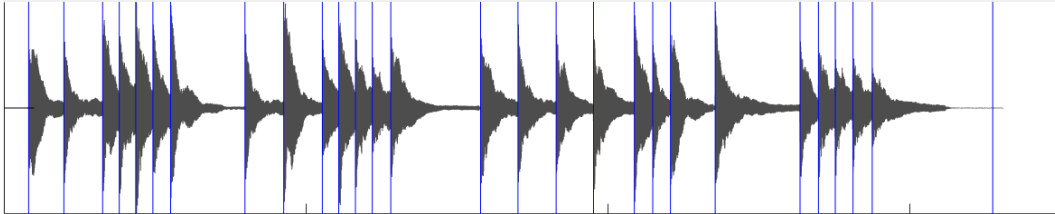
A felhasználói felület tesztje során a GUI összes objektumát ellenőriztem, elváltozást nem tapasztaltam. Minden elvárt viselkedés a felhasználói dokumentációban megtalálható, és ezek a tesztesetek maguktól értetődöek, így csak néhány példát mutatok be a 2. táblázatban.

Tesztelt eset	Tapasztalt viselkedés
A program indítása	Megjelenik az eszköztár, a menüsáv, a két ábra, illetve a tájékozódást segítő feliratok is
Fájl megnyitása menüből vagy billentyűkombinációval	Felugrik a fájlválasztó ablak, zeneszám kiválasztása esetén elindul az elemzés és frissül a felületen lévő cím, egyébként nem történik semmi
Betöltött hangfájl elemzése	A státuszmező folyamatosan tájékoztat az eredményekről, a végén megjelenik a teljes kezelőfelület
A <i>Play</i> , <i>Pause</i> , <i>Stop</i> gombok megnyomása kirajzolt jel és hangkezdetek nélkül	A státuszmező hibát jelez
A <i>Play</i> gomb megnyomása kirajzolt jel vagy hangkezdetek mellett	A lejátszás elindul
A <i>Pause</i> gomb megnyomása kirajzolt jel vagy hangkezdetek mellett	Amennyiben a zenelejátszás folyamatban van, akkor az leáll, újraindításkor ugyanonnan folytatódik tovább
A <i>Stop</i> gomb megnyomás kirajzolt jel vagy hangkezdetek mellett	Amennyiben a zenelejátszás folyamatban van, akkor az leáll, és legközelebb az elejéről indul újra

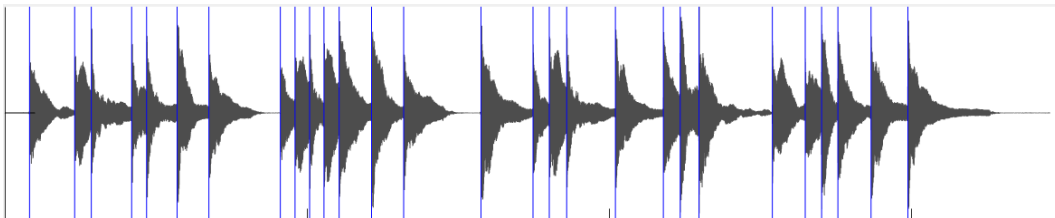
**2. táblázat.** Néhány példa a tesztelt funkciókra



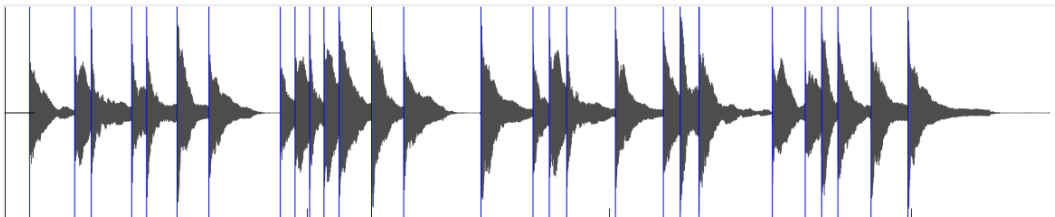
**Elemzési hatékonyság** Az algoritmust Bartók Béla Mikrokosmos 1. című kötete alapján teszteltem. A mérések során az eredmények rendkívül jók voltak, megfelelő paraméterezés mellett az összes hangkezdetet észlelte a program hiba nélkül, a néha előforduló fals pozitívokat pedig nagyon egyszerűen ki lehetett szűrni.



**31. ábra.** Mikrokosmos 1.: 3 - Reflection



**32. ábra.** Mikrokosmos 1.: 7 - Dotted notes



**33. ábra.** Mikrokosmos 1.: 17 - Contrary motion

### 3.5. Továbbfejlesztési lehetőségek

A program több irányban is továbbfejleszthető:

1. A detektáló függvényt lehetne javítani a szakirodalom alapján.
2. Mivel a zenei hangok helyét már megtaláltuk, sokkal egyszerűbbé vált a magasságuk meghatározása – ez lehetne egy új funkció.
3. A szoftver sebessége a GPU használatával és többszálú programozással drasztikusan nőhetne.

## 4. Hivatkozások

- [1] Klapuri, A., Davy, M.: *Signal Processing Methods for Music Transcription*, Springer, 2006.
- [2] Sung Bae-Cho: *Emotional image and musical information retrieval with interactive genetic algorithm*, Proceedings of the IEEE, Vol. 92, 2004.
- [3] Klapuri, A.: *Sound onset detection by applying psychoacoustic knowledge*, IEEE International Conference on Acoustics, Speech, and Signal Processing, 1999.
- [4] Salamon, J., Gómez E.: *Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics*, IEEE Transactions on Audio, Speech and Language Processing, Vol. 20, 2012.
- [5] Herrera, P., Peeters G., Dubnov S.: *Automatic Classification of Musical Instrument Sounds*, Journal of New Music Research, Vol. 32, 2003.
- [6] Zhu, Y., Kankanhalli, M.S., Gao, S.: *Music Key Detection for Musical Audio*, 11th International Multimedia Modelling Conference, 2005.
- [7] Eggink, J., Brown, G.J.: *Application of missing feature theory to the recognition of musical instruments in polyphonic audio*, International Conference on Music Information Retrieval, 2003.
- [8] Nagawade, M. S., Ratnaparkhe, V. R.: *Musical instrument identification using MFCC*, 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2017.
- [9] Eronen, J., Klapuri, A.: *Music Tempo Estimation With k-NN Regression*, IEEE Transactions on Audio, Speech, and Language Processing, Vol. 18, 2010.
- [10] Lu, L., Zhang, H.: *Automatic mood detection and tracking of music audio signals*, IEEE Transactions on Audio, Speech, and Language Processing, Vol. 14, 2006.
- [11] Bello, J.P. et al.: *A Tutorial on Onset Detection in Music Signals*, IEEE Transactions on Speech and Audio Processing, Vol. 13, 2005.
- [12] Shannon, C.E. *Communication in the presence of noise*, Proceedings of the IRE, Vol. 37, 1949.
- [13] *Méréselmélet jegyzet*, 12.fejezet, <http://www.mogi.bme.hu/TAMOP/mereselmélet/ch12.html> (2018. december 5.)

- [14] *A jelfeldolgozás alapjai* <https://gyires.inf.unideb.hu/GyBITT/02/> (2018. december 10.)
- [15] Rosao, C., Ribeiro, R, Martins de Matos, D.: *Influence of Peak Selection methods on onset detection*, 13th International Society for Music Information Retrieval Conference, 2012.
- [16] Dixon, S.: *Onset detection revisited*, Proc. of the 9th Int. Conference on Digital Audio Effects (DAFx-06), 2006.

## 5. Függelék

### 5.1. A magyarra fordított szakszavak

hullámforma	waveform
hangkezdet	onset
státuszmező	status bar
detektáló függvény	detection function
pásztázó	pan tool
rövid lecsengési idejű, tranziens	transient
rövid idejű Fourier-transzformáció	short time Fourier-transform
energiaburok-függvény	energy envelope function
mintavételezési ráta	sampling frequency