SABLON

TARTALOM

Ritmusérzék-fejlesztő alkalmazás fejlesztése Dobosok és ütősök esetén elengedhetetlen a tempótartás és az időbeli pontosság fejlesztése, és ez – ha kisebb mértékben is – más hangszeren játszó zenészek esetén is így van. Hagyományos módon ez metronómmal együtt történő gyakorlással tehető meg. A ritmusérzék további fejlesztésében nagy segítséget jelentene egy olyan alkalmazás, amely a mikrofonjel elemzése alapján a pontosságról képi visszacsatolást adna. Ritmusjáték esetén alapvetően kétféle pontosságról beszélhetünk: az egyik az ütemen belüli ütések pontossága, itt annak a vizsgálatára van szükség, hogy az elvi (metronóm által jelzett) ütésekhez képest siet-e vagy késik-e a zenész. Legalább ennyire fontos, hogy a zenész ne csak metronóm vagy más hangszerek játéka mellett legyen képes a zene tempójának stabil tartására: ebben az esetben a zene tempóját valós időben kell elemezni a lehető legkisebb késleltetés mellett. Könnyebbséget jelenthet, hogy a bemenőjel körülbelüli (megkívánt) tempóját ismertnek tekinthetjük. A hallgató feladata egy olyan szoftveres alkalmazás megvalósítása, amely mind az ütemen belüli pontosság, mind a tempótartás tekintetében segíti a gyakorlót. A hallgató munkájának a következőkre kell kiterjednie: • Tekintse át az ütem- és tempódetektálás módszereinek irodalmát!

• Matlabban alkosson a jel burkolóján alapuló, egyszerű ütésdetektáló algoritmust, ill. valósítson meg több különböző tempódetektáló módszert és hasonlítsa őket össze a pontosság, beállási idő (késleltetés), ill. a számításigény tekintetében!

• A JUCE fejlesztőkörnyezet segítségével C++ nyelven fejlesszen az audio bemenőjelet on-line feldolgozó programot a következő fő funkciókkal:

1) Ütésdetektálás: a program állítható tempójú és ütemszámú metronómjelet ad ki az audio kimeneten, a bemenőjel ütéseit detektálja, és mind a metronóm, mind a zenész ütéseinek helyét a képernyőn grafikusan megjeleníti.

2) Tempódetektálás: a program az audio bemenőjel alapján megállapítja a zene tempóját és azt megjeleníti. Az algoritmus késleltetése/pontossága legyen állítható.

• A C++ nyelven beprogramozott algoritmusok működését a Matlab eredményeivel történő összehasonlítással tesztelje!

• Tesztelje és értékelje a teljes megvalósított programot a pontosság és kezelhetőség tekintetében, ill. hasonlítsa össze a piacon elérhető megoldásokkal!

1. Bevezetés

A feladat egy olyan ritmusérzék-fejlesztő alkalmazás megvalósítása, amely futtatható hétköznapi személyi számítógépen és az eszköz mikrofonjának felhasználásával valósítja meg a bemenő jel előállítását. Ezen alkalmazás nagy segítséget nyújthat a zenészeknek a tempótartás, illetve váltás gyakorlásában és elérhetővé teszi ezt a funkciót az akusztikus eszközöket használók számára is.

1.1 Tervezés célja

A tervezés célja egy olyan alkalmazás elkészítése, amely Windows operációs rendszeren futtatható és a számítógép mikrofonjának bemenetét használja a szükséges bemeneti jel kinyerésére. Kimenetként a vett jel alapján, az adott hang tempóját jelzi ki. Két használati lehetőséget biztosít, az egyik a tempó automatikus felismerése és kijelzése, a másik az aktuálisan felismert tempó összehasonlítása egy előre beállított referenciával.

A használó valamilyen jellegű vizuális visszacsatolást nyerhet, amelyet a zenélés során közvetlenül értelmezhet. Az fejlesztés célja a lehető legpontosabb visszajelzés biztosítása.

A piacon jelen pillanatban már megtalálhatóak hasonló megoldások, de 1 évvel ezelőtt ez a megoldás még elérhetetlen volt a zenészek számára. Ahhoz, hogy hasonló funkciók előnyeit élvezhessék digitális eszközöket kellett vásárolniuk. Vannak olyan elektromos dobok a piacon, amelyek tartalmaznak ritmusérzék felismerő funkciót. A jelen megoldás ezzel szemben kiterjeszti ezt e lehetőséget hagyományos akusztikus hangszerek esetére.

1.2 Megvalósítás menete

A megvalósítás során elemzésre és összehasonlításra kerülnek különböző tempódetektáló algoritmusok. Az irodalomkutatás során cél, minél hasznosabb képet kapni már meglévő megoldásokról, és ezeket amennyiben lehetséges felhasználni. A tervezés során MatLab segítségével kerülnek megvalósításra az algoritmusok. Ezen program segítségével a módszerek könnyen összehasonlíthatóak, és meghozható a döntés, hogy melyik kerüljön végleges implementálásra. A megvalósítás során, cél az adott algoritmusok javítása különböző módszerek bevezetésével, kipróbálásával. A végleges implementáció C++-ban történik a JUCE fejlesztő környezet segítségével. Ezen eszköz lehetőséget biztosít a grafikus visszacsatolás megvalósításához és a bementi jel könnyű kezeléséhez. A megvalósítás végső lépése az elkészült program tesztelése és összehasonlítása.

Az összehasonlítás során figyelembe vett szempontok:

I; Futási idő, számításigény

II; Pontosság

III; Tempó változások követésének képessége

I: Futási idő, számításigény: A program működése során az egyik legfontosabb paraméter. A MatLab-ban való tervezés során rendelkezésre áll előre a teljes feldolgozandó jel, azonban a végleges implementáció során a feldolgozást élő hangon kell elvégezni. Ez azt jelenti, hogy egy adott időszelet feldolgozási ideje nem lehet hosszabb az adott időszelet hosszánál.

II: Pontosság: A cél minél pontosabb tempódetektálás. Ez azt jelenti, ha valaki 60BPM tempóval játszik, ne kapjon 64BPM-es visszajelzést. Különböző feldolgozási eljárások a detektálás folyamatába bizonyos mértékű pontatlanságot vihetnek. A cél ezek kiszűrése.

III: Tempó változások követésének képessége: Egy adott zenélés során nem ritka, hogy szándékos tempó emelése vagy csökkentése. A leg ideálisabb, ha ezeket a változásokat a program tudja kezelni. Továbbá, előnyt jelent, ha nem csak a fokozatos, apró változásokat tudja követni, hanem a nagyobb tempóugrásokat is. A nagy tempóváltások összességében ritkán fordulnak elő, de a legjobb erre is felkészíteni az alkalmazást.

2. Feladat pontosítása

A pontos feladat a jelfeldolgozás megvalósítása, és egy olyan optimálisnak mondható algoritmus kifejlesztése, amellyel élő hang elemzésére alkalmassá tehető az alkalmazás. A cél, hogy az algoritmus a vett jel alapján, minél biztosabban adjon visszacsatolást annak tempójáról. Fontos szempont, hogy minél kevesebb bemeneti paraméter legyen szükséges a használathoz, a felhasználó egyszerűen tudja használni, anélkül, hogy bármilyen szakmai háttérrel rendelkezne az alkalmazás műszaki háttere kapcsán.

2.1 Tempódetektálás

Ezen funkció célja, hogy egy szabadon játszott zenéről képes legyen megmondani a tempóját, és ezt a tempót élőben jelezze vissza a program a felhasználója felé. Az algoritmusnak, minél rugalmasabbnak kell lennie, adott esetben nem csak egy hangszer használata esetén kell, hogy képes legyen a tempó megállapítására, hanem akár komplexebb esetekre is. Nagy segítség, ha detektálás, akár teljes többhangszeres játék esetén is működik, ezzel akár egy koncert próbán is használható.

Az feldolgozás sebessége rövidebb idő alatt kell, hogy megtörténjen, mint a feldolgozott hang hossza, ahhoz hogy a tempó élőben visszajelezhető legyen. A kijelzés frissítési ideje fejlesztői döntés, a tervezés során tapasztaltak alapján. Feltételezhető, hogy a tempó nem változik rövid időn belül nagy mértékben.

A megvalósítás történhet, bármilyen számítógépen implementálható megoldással, de a cél, hogy minél stabilabb visszajelzést kapjon a használó és egyszerűen tudja használni a programot.

Ideális esetben a megoldás képes a tempó változások követésére vagy abbahagyott játék után, azt folytatva újra visszatalálni az aktuális tempóhoz.

2.2 Tempókövetés

Ezen funkció célja, hogy a felhasználó egy előre beállított tempóhoz képest kapjon visszajelzést arról, hogy éppen siet vagy késik. Ez nagy segítséget nyújt, akár színpadi játék során, ahol egyértelmű és gyors visszacsatolást képes adni a program.

A referencia tempót előre egy bemeneti paraméterként kell beállítani. A használat során az alkalmazás visszajelzést ad, a referencia tempótól való eltérés irányáról és mértékéről.

A fejlesztés során feltételezhető, hogy a zenész egységnyi idő alatt nem változtat a tempóján nagymértékben. A visszajelzés frissítési ideje a fejlesztés során tapasztaltaknak megfelelően előre beállított.

3. Irodalomkutatás

Az irodalomkutatás az elérhető tudásanyag tanulmányozása. Jelen esetben főként az interneten történő keresés. Egyetemek oldalainak felkeresése, tudományos cikkek keresése és a használt eszközök dokumentációjának tanulmányozása.

3.1 Általános cél

Az irodalomkutatás elsődleges célja, hogy kontextusba kerüljek a megvalósítandó feladattal, megszerezzem a sikeres fejlesztéshez szükséges tudást. Az érintett témák a hang digitalizálása, feldolgozás, valamint a már meglévő tempó megállapítására szolgáló eljárások megismerése és összegyűjtése.

3.2.1 Hang energia alapján működő algoritmus

Forrás: <http://mziccard.me/2015/05/28/beats-detection-algorithms-1/>

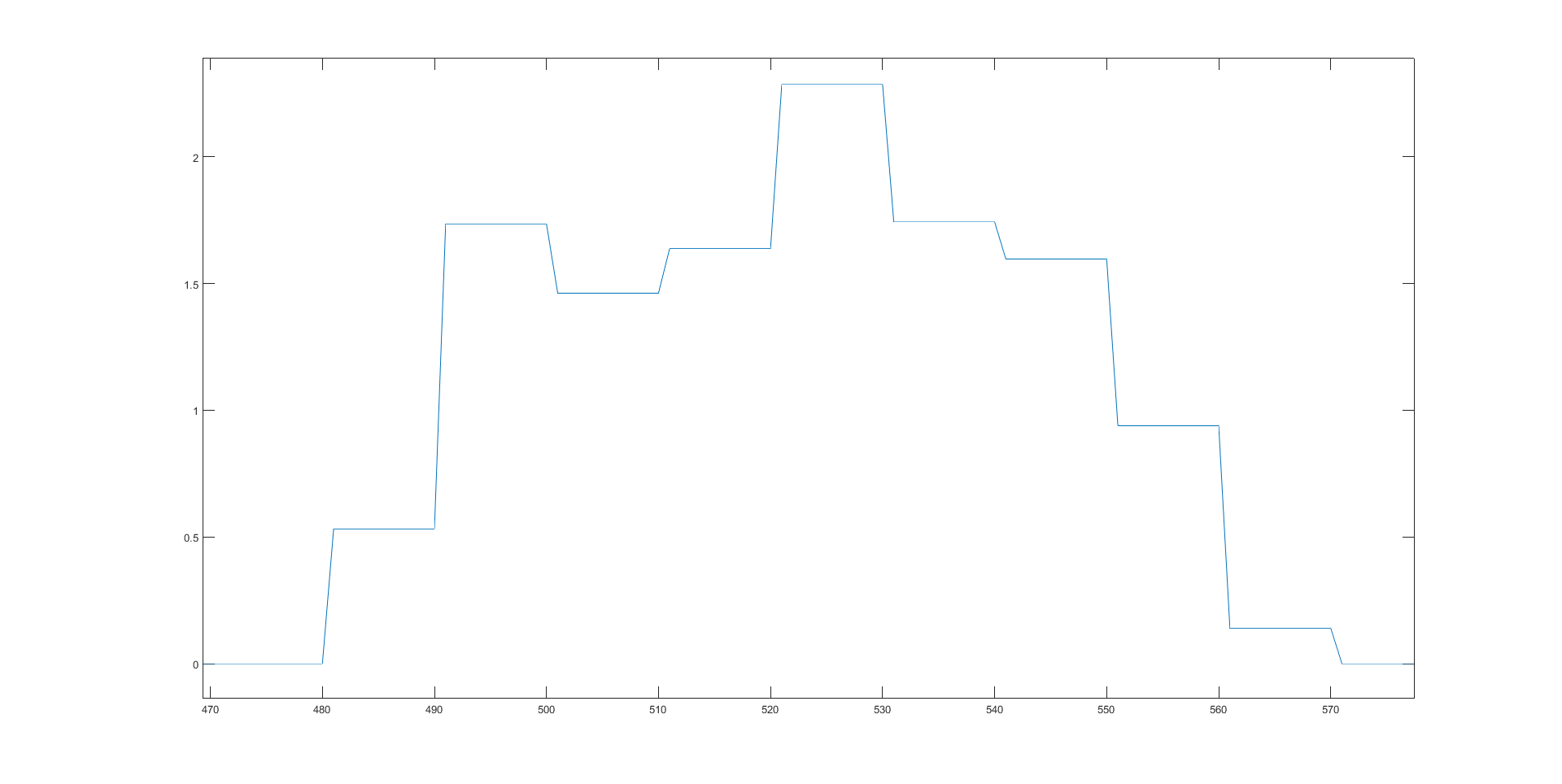
<http://archive.gamedev.net/archive/reference/programming/features/beatdetection/index.html>

<http://www.flipcode.com/misc/BeatDetectionAlgorithms.pdf>

Ezen ötlet a hang energiájának (amplitúdójának) megfigyelésén alapul. Az algoritmus alapötlete, egy küszöbként szolgáló energiaszint meghatározása és az adott energiaszinteknek az összehasonlítása a küszöb értékkel.

Az algoritmus a következőképp néz ki. Adott egy bemenő sztereo jel ***e*** , ennek csatornáit összegezzük:

A következő lépésben egy adott idősávra meghatározzuk az átlagos energia szintet:



A képen a hang 10es ablakos energia szintjei vannak…

Kiszámoljuk, az átlagos energia szintet az adott index környezetében ez legyen „C”:

Ha az adott ***avg(e)*** érték nagyobb az aktuális ***C*** értéknél, azt mondhatjuk, hogy ütést találtunk. Ezután az értékeit feltöltjük a soron következő értékekkel és újra ugyan ezekez a lépéseket hajtjuk végre.

Az ütések helyeit letároljuk és bizonyos frissítési idő által meghatározott periódusonként kiszámítjuk a jel tempóját.

Ha az adott hang energia szintje nem változik, elég lehet a küszöb értéket egyszer kiszámolni, de a pontosabb mérés érdekében érdemes ezt folyamatosan korrigálni.

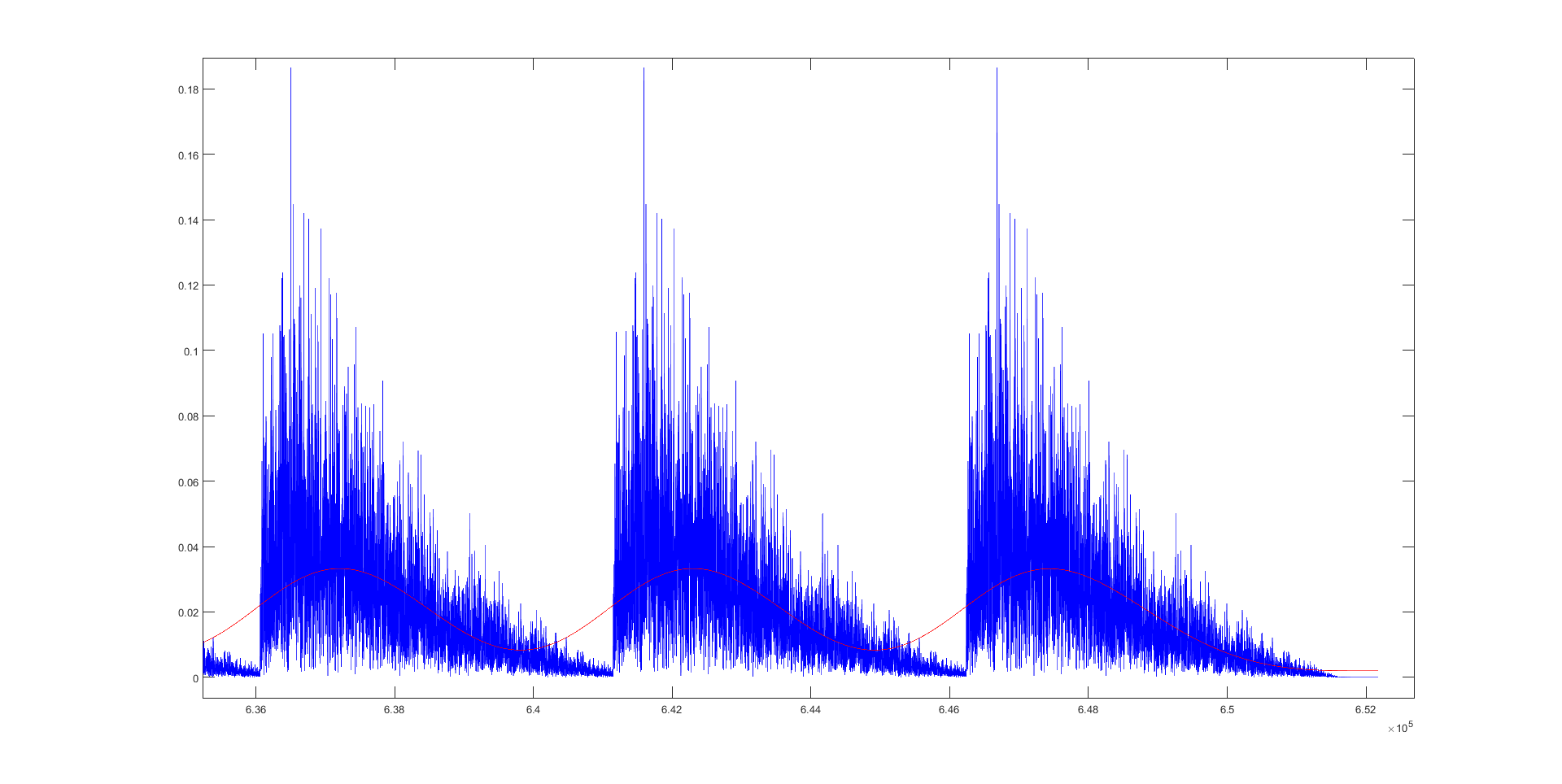
3.2.2 Előfeldolgozás

Forrás: <http://moodle.autolab.uni-pannon.hu/Mecha_tananyag/digitalis_jelfeldolgozas/ch07.html>

<http://joesul.li/van/beat-detection-using-web-audio/>

A bemenő jel egy sztereó jel. Ennek csatornáit összegezni kell, ügyelve arra, hogy ne oltsák ki egymást a két csatornai jelei, ha ellentétes fázisban érkeznének be. Ez az előző 3.2.1-es módszernél is megfigyelhető volt.

Ezek után a jel a következőképpen néz ki:



A kék az eredeti összegzett hang a piros az aluláteresztő szűrő alkalmazása utáni jel…

A fenti képen kékkel ábrázolva három ütés mintája látható. Megfigyelhető egy ütés menete sok amplitúdó változást tartalmaz mind felfutás, mind a lecsengés során. Az alapötlet az, hogy mivel a gyors változásokkal teli jelet nehéz feldolgozni, célszerű ebből egy simább jelet kinyerni.

Széles körben elterjedt és jól alkalmazható erre az aluláteresztő szűrő.

//TODO aluláteresztő szűrő miért, hogy stb.

A jelen egy megfelelően paraméterezett aluláteresztő szűrést alkalmazva a fenti képen a pirossal jelölt jel áll elő. Megfigyelhető, hogy ezen jel már sokkal simább és nem tartalmaz hirtelen változásokat, ezáltal könnyebben feldolgozható.

Az aluláteresztő szűrés miatt fázis eltolás lép fel, azonban ez az ütemek egymáshoz képesti viszonyán nem változtat, de adott esetben ezen fáziseltolás korrigálható.

Ezek után mondhatjuk azt, hogy az ütések helyei a jel lokális maximumai.

3.2.3 Ablak illesztéses módszer

<http://joesul.li/van/beat-detection-using-web-audio/>

Feltesszük, hogy a bemenő jelből már rendelkezésre állnak az ütések helyei. A következő lépésben a cél az aktuális tempó megállapítása.

Az ötlet a cikk alapján a következő, egy adott tempó kirajzol egy úgynevezett konvolúciós ablakot. Ezt az ablakot lehet ráilleszteni az adott ütemekre az eredeti jelen.

Ezt az illesztést több tempó által előállított ablakkal elvégezve a legjobbat választva megállapítható a tényleges tempó.

3.2.4 Ütem távolság gyakoriság

Forrás: <http://joesul.li/van/beat-detection-using-web-audio/>

A következő ötlet, azon alapul, hogy rendelkezésre állnak az eredi jelben az ütések helyei és valahogyan ebből kell kinyerni a tempóra vonatkozó információt.

A módszer lényege a következő, egy adott ütést összemérünk az összes többi ütéssel és ezek távolságát egy tárba eltároljuk. Ezzel megkapjuk, hogy melyik a leggyakoribb távolság.

Például:

|  |  |
| --- | --- |
| Távolság(ms) | Gyakoriság |
| 1.6 | 21 |
| 1.1 | 12 |
| 0.8 | 2 |

Ezen információ alapján kiszámolható a hangra jellemző tempó.

További javítás, hogy az adott távolságok egymás többszörösei lehetnek, ami annyit jelent hogy csak mondjuk minden második ütést talált meg az eljárás. Ha ezen információt is figyelembe vesszük és ezen eseteket csoportosítjuk pontosabb eredményt kaphatunk.

4. Tervezés

A tervezés arra szolgál, hogy az ismertetett feladathoz a lehető legjobb megoldást találjam, és a tényleges implementáció előtt, ezeket kipróbálhassam és összehasonlíthassam. A tervezéshez használt eszköz a MatLab, amely lehetőséget nyújt az egyszerű összehasonlításra és az előre megírt eszközei segítségével, gyorsan implementálhatóvá teszi a kívánt eljárást.

4.1 Analóg és Digitális hang

Megkülönböztethetünk analóg, illetve digitális hangot, melynek fontos szerepe van a feladat szempontjából. Az analóg hang időben folytonos, míg a digitális hang diszkrét. A feladat során a bemeneti jel analóg, amit a számítógép digitális jellé alakít és a tényleges feldolgozás már ezen a jelen történik.

4.1.1 Hangtechnikai alapfogalmak

Forrás: <http://magasztos.hu/tananyag/MaczikM_Hangtech.pdf>

Hallható hang: Rugalmas közegben fellépő mechanikus rezgéseket 20Hz és 20kHz közti frekvenciatartományban hallható hangnak nevezzük.

Frekvencia: A hanghullám az egyik hangnyomás maximumától a másik hanghullám maximumáig tart, amelyek sűrűsége megadja a másodpercenkénti rezgésszámot, azaz a frekvenciát. A periódus idő reciproka. f = 1/T

Amplitúdó: A rezgés amplitúdója határozza meg a hangerőt.

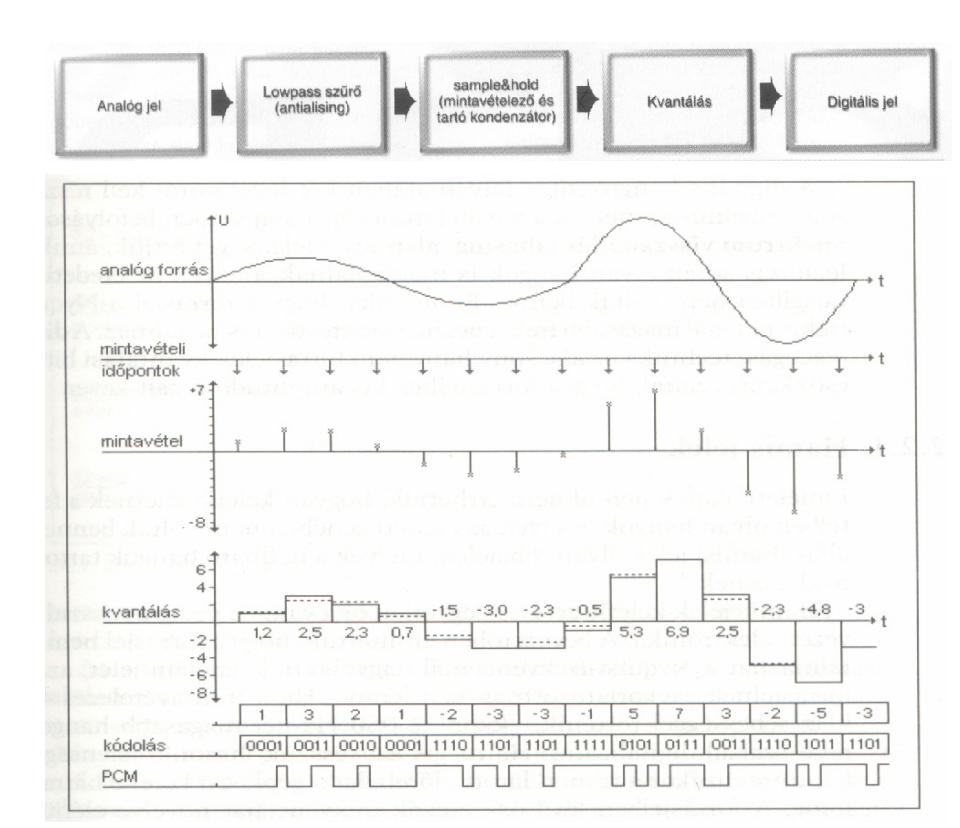
Mikrofon: A levegő mechanikai rezgéseit, sebesség, illetve nyomásváltozásait alakítja elektromos energiává.

CD minőség: (16 bit, 44,1 kHz) Az egyik legelterjedtebben alkalmazott felbontás.

AD konverzió(Analóg jel digitalizásála):

A felvétel során a keletkezett hangnyomás a mikrofon membránját rezgeti. A rezgéseket a mikrofon feszültség értékekké alakítja, amelyek már könnyen kezelhetőek. Ez a jel még analóg folytonos jel, azonban a számítógép számára digitális jelekre van szükség.

A jel digitalizálása során a Nyquis-Shannon tétel alapján a hallható hang alapján ,amely 20Hz -20kHz közötti, a mintavételi frekvencia el kell érje a 40kHz-et. A mintavétel során az eredeti hangban nem jelenlévő összetevők is belekerülnek a vett jelbe(aliasing hatás). Ennek kiküszöbölésére aluláteresztő szűrőt alkalmaznak, amely a 20kHz feletti frekvenciakomponenseket csillapítja(antialiasing). Ezután következik a mintavételezés, majd ahhoz, hogy az értékeket egész számként rögzíthessék, kvantálást hajtanak végre. A folyamat legvégén előáll a digitális jel, amely számítógép segítségével is könnyen kezelhető.



forrás: <http://magasztos.hu/tananyag/MaczikM_Hangtech.pdf>

Fourier tétel: Minden folytonosan változó periodikus jel felbontható állandó frekvenciás szinuszhullámok összegére.

Nyquis-Shannon tétel: Egy jelsorozat szinuszos összetevői közül a legnagyobb előforduló frekvencia kétszeresével mintavételezett jelsorozat egyértelműen reprezentálja az eredeti jelet.

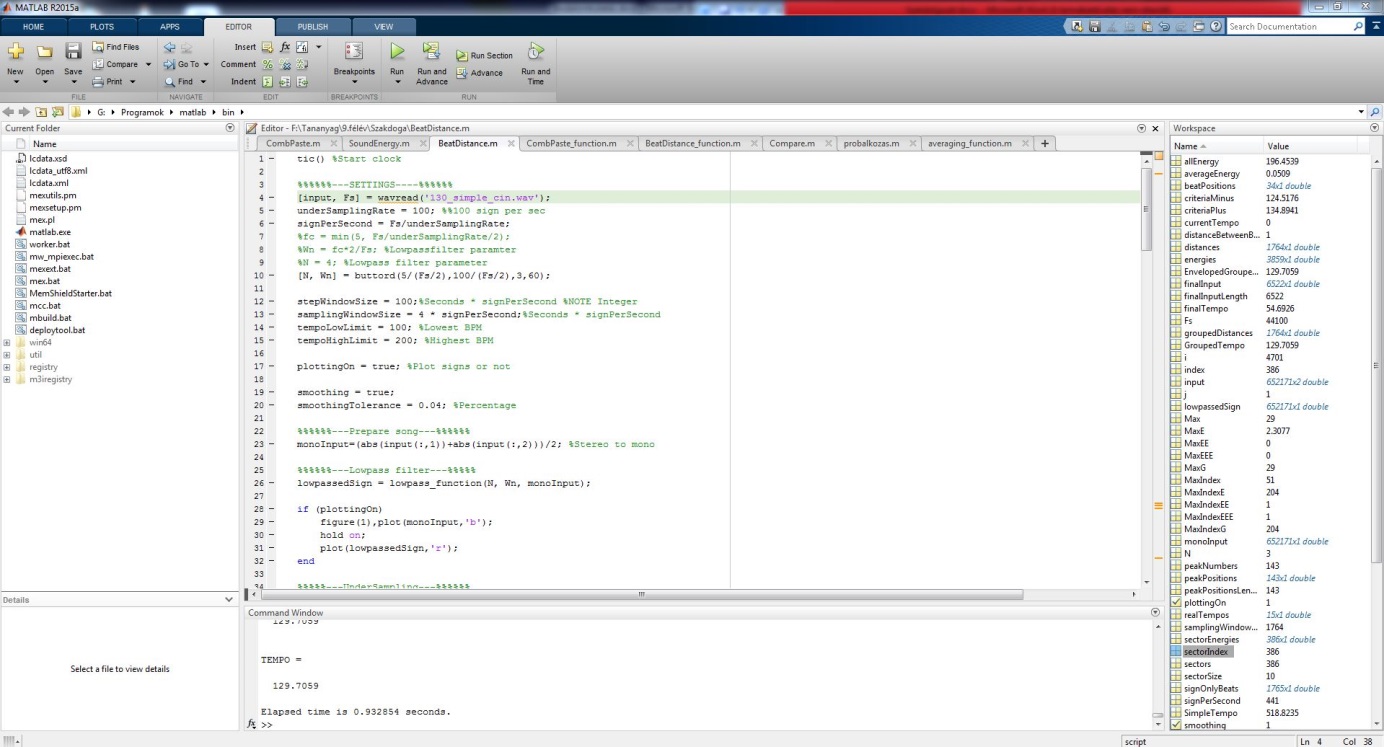
4.2 Választott eszközök

4.2.1 MatLab

Az eljárás megtervezéséhez használt eszköz a MatLAb, amely egy asztali számítógépen használható alkalmazás, ami lehetőséget biztosít különböző analízisek, matematikai műveletek elvégzésére és tervezések megvalósítására egy saját programozási nyelv használatával.

Előnyei, hogy rengeteg eszköz áll rendelkezésre, amely meggyorsítja a tervezést, ilyenek a különböző szűrő tervezők vagy az előre implementált matematikai függvények. Tökéletes eszköz egy adott eljárás eredményének szemléltetésére, rendkívül rugalmas és könnyen használható ábrázolási eljárásokat tartalmaz.

Nagy segítség a program használata során a körülötte kialakult nagy létszámú és tapasztalatú közösség, ezáltal könnyű egy-egy adott problémára megoldást találni és alkalmazás fejlődése is folyamatos.



Az általam használt verzió: R2015a

A program könnyen telepíthető és használható. A dolgozatban található grafikonok nagy része ezen eszköz segítségével készült.

Megjegyzés: Ügyelni kell arra, hogy a számítógép hardvereinek driverei megfelelően működjenek és telepítve legyenek. Előfordulhat, hogy a program hibaüzenet nélkül állítja le a működését, ha valamilyen illesztő program működése nem megfelelő.

Rendszer követelmények: …

4.2.2 JUCE

A JUCE egy zenei alkalmazások fejlesztését támogató eszköz, amely egy nagy teljesítményű professzionális eszköz. Elérhető a legnépszerűbb platformokra, mint Windows, Mac, Linux.

Segítségével összetett alkalmazások fejleszthetőek egyedi igények szerint, továbbá támogatja VST, AU és AAX pluginok fejlesztését is.

Amiket rendelkezésre bocsát:

1. Könnyen szerkeszthető felhasználói felület

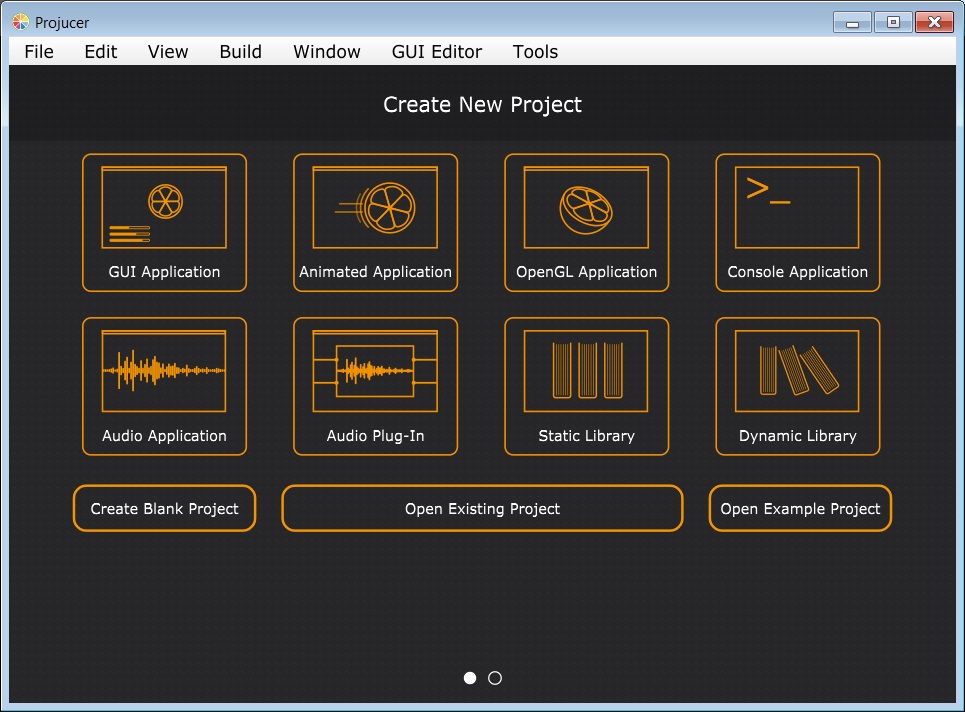
2. Egyedi fordító és futtató környezet

2. Plugin fejlesztési lehetőség

3. Széles körű formátum támogatás

A fejlesztés nyelve C++, melyhez bármilyen IDE felhasználható.

Az újabb verzió lehetőséget teremt mobil platformokra való fejlesztések számára is.



Az általam használt verzió: JUCE 5.2

Megjegyzés: Az alkalmazás telepítése során ügyelni kell az elvárt szoftverek megfelelő állapotára és verziójára. A tapasztalataim szerint, ezt a verziót célszerű a Visual Studio 2017es változatával használni, amely a legújabb C++ könyvtárakat tartalmazza. További hibalehetőség, ha Windowson nincs feltelepítve az InternetExplorer, ugyanis az alkalmazás a bejelentkezéshez fixen ezt a böngészőt használná, de ha nem áll rendelkezésére nem ad vissza hibaüzenetet.

Rendszer követelmények:…

4.2.3 Git

A GIT egy verzió kezelő szoftver, amely lehetőséget biztosít a munka verziózására, illetve egy távoli szerveren való tárolására, ha a helyi mentés esetlegesen megsérülne.

4.2.4 FL Studio 12

Az FL Studio egy digitális audió munkaálomás. 18 éves fejlesztési múlttal és rengeteg hasznos funkcióval. Jelen téma esetében a referencia jelek előállítására használtam. Segítségével könnyen generálható bármilyen ütemmenet és beállítható annak tempója.

Redszer követelmények:…

4.2.5 Nero WaveEditor

A Nero által biztosított alap funkciókat ellátó hangszerkesztő szoftver. Lehetőséget biztosít a hangformátumok könnyű konvertálására és vágására.

Használt verzió:

Rendszer követelmények:

4.3 Algoritmusok

4.3.1 Jel előkészítés

2oldal

4.3.2.1 Módszer 1

3oldal

4.3.2.2 Módszer 2

3oldal

4.3.2.3 Módszer 3

3olda

4.3.3 Összehasonlítás

4oldal

4.3.4 Kiválasztás

1oldal

Indoklás

5. Implementáció

10sor

5.1 Környezet

fél oldal

5.2 Követelmények a programtól(TODO)

fél oldal

5.3 Program működése

1oldal

5.4 Bemutatás tesztelés

3oldal

5.5 Értékelés

1 oldal

5.5.1 Összehasonlítás MatLAb

2oldal

5.5.2 Összehasonlítás más programmal

2oldal

5.6 Továbbfejlesztési lehetőségek

fél oldal

6. Rövidítések