TIME STRETCH ÉS PITCH SHIFT ALGORITMUSOK - A KEZDETEKTŐL A WAVELET-TRANSZFORMÁCIÓIG

TIME STRETCH AND PITCH SHIFT ALGORITHMS - FROM THE BEGINNING TILL THE WAVELET-TRANSFORM

GALAMBOS RÓBERT
BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNY EGYETEM
VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
MÉRÉSTECHNIKA ÉS INFORMÁCIÓS RENDSZEREK TANSZÉK

KIVONAT

A cikk a time stretch és pitch shift algoritmusokat tárgyalja, részletezve e két fogalomkört, bemutatva az algoritmusok kialakulásának és fejlődésének útvonalát első megjelenésüktől napjainkig, kitérve hibáikra, és előnyeikre, valamint az esetleges speciális felhasználási lehetőségekre. Bemutatja a wavelet-transzformáció előnyeit, a hagyományos short time Fourier-transzformációval szemben. Rávilágít a megvalósítási problémákra, és tapasztalati eredményekkel támasztja alá az algoritmusok használhatóságát.

ABSTRACT

The article is about time stretch and pitch shift algorithms. It explains the concept of these two categories, shows the algorithms' way of evolution from their appearance to nowadays, demonstrating their advantage, disadvantage, and in some cases potential special usage. It shows the advantages of the wavelet transform compared to the short time Fourier transform, and reveals the implementations problems, by investigating the usage of the algorithms within practical cases.

BEVEZETÉS

A hallás az emberi érzékelés egyik legkifinomultabb formája, környezetünkről szerzett információink lényeges forrása. Képesek vagyunk időtartománybeli és frekvenciatartománybeli különbségeket érzékelni, és ebből kifolyólag az az igény is megfogalmazódhat bennünk, hogy a hallott jel ezen paramétereit meg is megváltoztassuk. Egy zenei darab esetében sok olyan szituációt lehet elképzelni, amelyben szükség van a darab időbeli lefutását vagy frekvenciamenetét módosítani. Például szükség lehet egy rosszul feljátszott hang vagy egy rossz belépés korrekciójára, vagy egy túl lassú darab felgyorsítására.

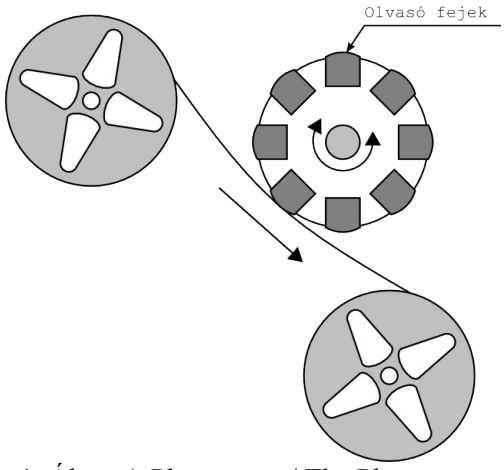
A SEBESSÉGÁLLÍTÁS TÖRTÉNETE

A múltban, amikor még nem volt sem digitális, sem analóg hangrögzítés, a sebességállítás egyetlen módja az volt, ha a zenészek a darabot újrajátszották, és ezáltal változtattak a mű tempóján vagy hangfekvésén.
Később megjelentek az analóg hanghordozók, mint például a hanglemez, vagy a magnószalag. A stúdiótechnika és a rádiós műsorszórás is elindult útján. Igazán ekkor merült fel először a sebességállítás és hangmagasság-állítás problémája. A felvételek utólagos módosítására különböző indokok voltak. A stúdiótechnikában a rosszul felvett sávok ütemeit és hangfekvését próbálták korrigálni, míg a rádióban értékes műsoridőt próbáltak megtakarítani. Ezt először analóg módon valósították meg, az adott lejátszók motorjának fordulatszámát változtatva, így lassítva, vagy gyorsítva a lemez forgását, illetve a magnószalag haladását.
Az ilyen fajta jelmódosítás a Fourier-transzformáció hasonlósági tételére épül (1).

$$\mathcal{F}\{f(t/\alpha)\} = |\alpha| F(\alpha\omega) \quad (1)$$

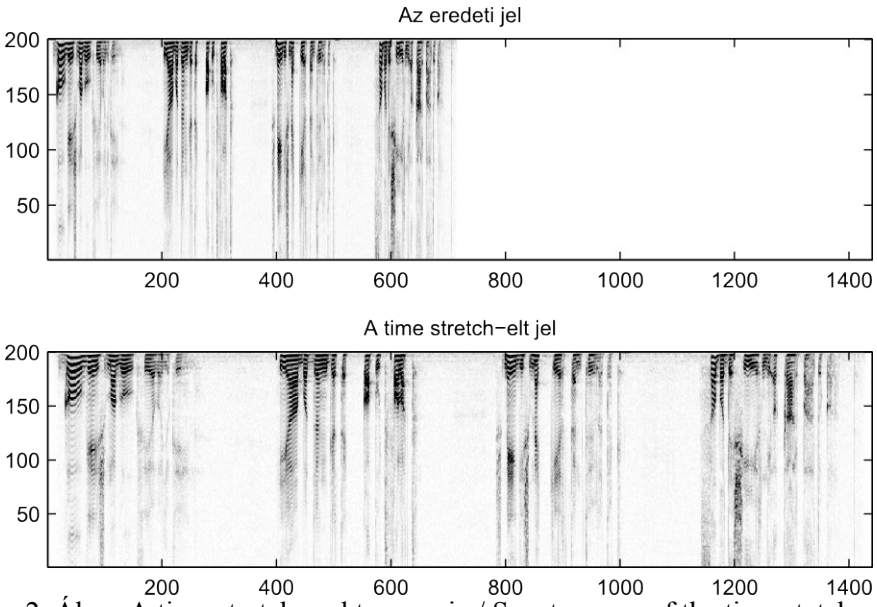
amiből jól látható, hogy az időtengely menti változtatás hatására a frekvenciakomponensek is megváltoznak. Ha tehát a jel gyorsul, akkor a frekvenciakomponensek a magasabb frekvenciák felé, ha lassul, akkor az alacsonyabb frekvenciák felé tolódnak el a frekvenciatengely mentén. Ez korlátozza a módszer használhatóságát, hiszen ha már egy fél hanggal eltolódnak a

frekvenciakomponensek, akkor az már hallható disszonanciát okozhat az egymást követő zeneművekben. Számszerűsítve ez azt jelenti, hogy 5%-nál nagyobb sebességállítás már fél hangnál nagyobb frekvenciakomponens-eltolást eredményez.
Az 1960-as években Pierre Schaeffer előállt egy megoldással a problémára. Phonogene-re keresztelt találmánya egy módosított szalagos lejátszó (Zölzer2002), ahol a szokványos egy olvasófej helyett többet helyezett el egy kör alakú forgórészen, melyet adott irányban és sebességgel lehetett forgatni. A gép vázlatos felépítése látható az 1. ábrán.



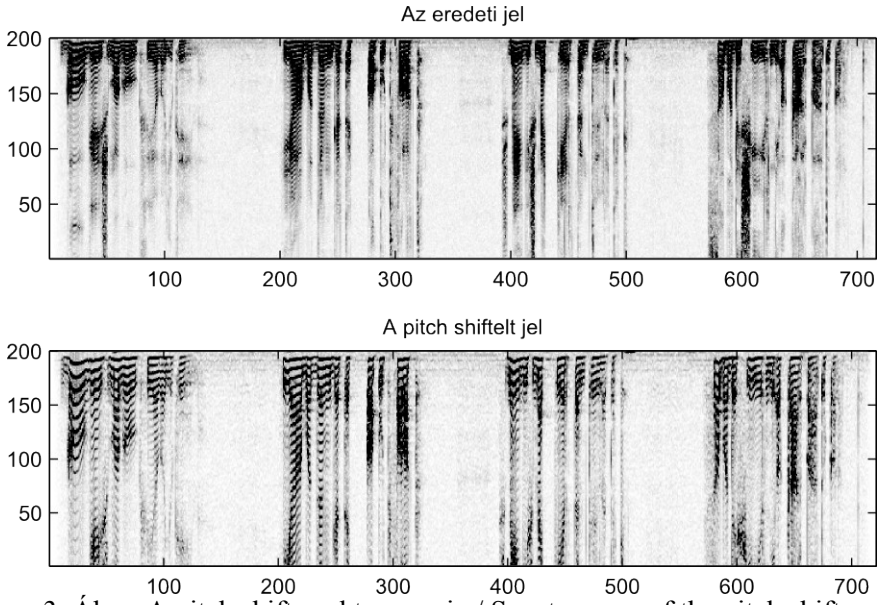
1. Ábra: A Phonogene / The Phonogene

A szalag és az olvasófejek relatív sebességével a hangmagasságokat lehet változtatni, míg a szalag sebességével a darab hossza befolyásolható.
A digitális technika megjelenésével a problémát új szemszögből lehetett megközelíteni. A jelfeldolgozás adta lehetőségek, a DSP-k és a számítógépek nagy számítási kapacitása, valamint a Fourier-transzformáció új lehetőségek előtt nyitották meg az utat. A jeleket mind az időtartományban, mind a frekvenciatartományban új módszerekkel lehetett befolyásolni.
Így alakultak ki a time stretch és pitch shift algoritmus csoportok is. Time stretch algoritmusoknak nevezzük azokat az algoritmusokat, melyek megváltoztatják egy jel hosszát, tehát befolyásolják a jel sebességét, de változatlanul hagyják a frekvenciakomponenseit, vagyis a hangmagasságait. Ezzel ellentétben a pitch shift algoritmusok a jel sebességét, tehát hosszát hagyják változatlanul, míg a frekvenciakomponenseit, hangmagasságait megváltoztatják. A time stretch-et szemléltető spektrogram a 2. ábrán, míg a pitch shiftet szemléltető spektrogram a 3. ábrán látható.



2. Ábra: A time stretch spektrogramja / Spectrogram of the time stetch

Mint az megfigyelhető, a harmonikusok frekvenciája nem változott meg, de a hosszuk kétszer akkora lett.



3. Ábra: A pitch shift spektrogramja / Spectrogram of the pitch shift

Ezzel ellentétben pitch shift esetén megfigyelhető a harmonikusok frekvenciaváltozása, miközben időtartamuk változatlan.

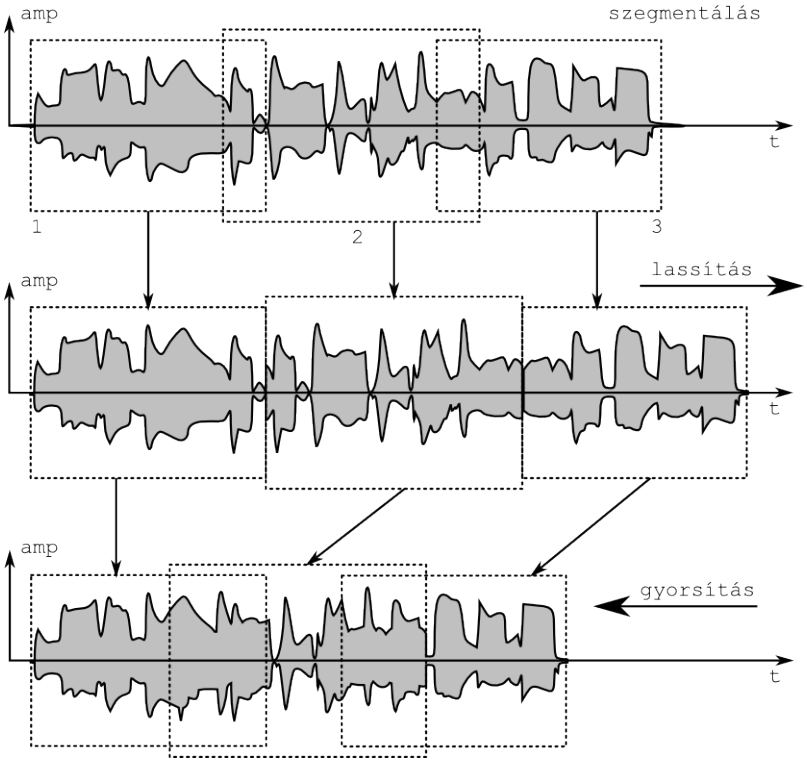
NAPJAINK SZOFTVEREI

Napjainkban több olyan szoftver van piacon, mely tartalmaz time stretch és pitch shift algoritmusokat. Ezek jelentős része valós idejű felhasználásra készült, vagy egy nagyobb szoftver része. Így nem megengedhető, hogy a teljes számítási kapacitás erre az egy feladatra fordítódjon. Leginkább ezeknél jelentkezik, hogy a felhasználó felületük egyszerű, és a hangminőségük is hagy kivetnivalót maga után. A kettő valamelyest összefügg, mert többnyire kevés paraméter áll a felhasználó rendelkezésére, hogy finomhangolja az amúgy is gyengébb minőséget produkáló algoritmust. Természetesen léteznek olyan szoftverek is melyek kizárólag erre a feladatra specializálódtak. Ezek hangminősége és hangolhatósága már lényegesen jobb lehet. Ilyen, szinte etalon algoritmusnak számít például az iZotope cég által kifejlesztett Radius technológia, melynek működéséről szinte semmit nem lehet tudni, és piaci értéke miatt nem is hagynak kiszivárogni információkat (IzotopeWeb). Az MPEX a másik algoritmus, mely elfogadható minőséget produkál, és erről az eljárásról még némi információt is ki lehet deríteni. Valószínű, hogy a jelet először a frekvenciatartományba transzformálja, majd egy neurális hálóval próbál alkalmazkodni a zenéhez, különböző paraméterekre figyelve (ütem, dallam, zörejek…), majd ezen paraméterek segítségével hozza létre a time stretch-elt jelet (MPEXWeb).

TIME STRETCH

IDŐTARTOMÁNYBELI SZEGMENTÁLÁS

Az első algoritmus, amely alkalmas volt a time stretch hatás létrehozására, az az időtartománybeli szegmentálás, vagy más néven OverLap and Add (OLA). Az eljárás a Phonogene működését másolja, és ahogy az a 4. ábrán is látható, a jelet időtartománybeli átlapolt szegmensekre bontja fel, majd ezeket a szegmenseket csúsztatja szét, illetve össze, végül újra összegzi a jelet (Zölzer2002).



4. Ábra: Az OverLap and Add szemléltetése / Demonstration of the OverLap and Add

Az időtartománybeli szegmensek kivágása felírható $w[n]$ ablakfüggvény segítségével, a (2) egyenlet szerint:

$$z[n] = \sum_{k=-\infty}^{\infty} x[n + (n_1 - n_2)k] \cdot w[n - n_2 k] \quad (2)$$

ahol $z[n]$ a kimeneti jel, $x[n]$ a bemeneti jel, $w[n]$ a szegmensek kivágására szolgáló ablakfüggvény, n_1 a szegmenskezdetek időbeli távolsága az eredeti jelben, n_2 pedig a szegmenskezdetek időbeli távolsága a feldolgozott jelben. Az ablakfüggvényekkel való szorzás miatt az összegzés után jelszint-ingadozás léphet fel. A probléma megoldható, ha az ablakfüggvények eltolt összegével korrigáljuk a jelszintet (3).

$$z'[n] = \sum_{k=-\infty}^{\infty} w[n - n_2 k] \quad (3)$$

ahol a $z'[n]$ a jelszint-ingadozás korrekciójához felhasználható sorozat. A k paraméter jelen esetben azért járja be a $]-\infty;\infty[$ intervallumot, mert jelfolyamról beszélünk, melynek nem definiált sem az eleje, sem a vége. A valóságban természetesen véges hosszúságú jelekre alkalmazzuk a képletet, és számolnunk kell az elején és végén keletkező torzulásokkal, de ezek nem számottevőek, és könnyen kiküszöbölhetőek.

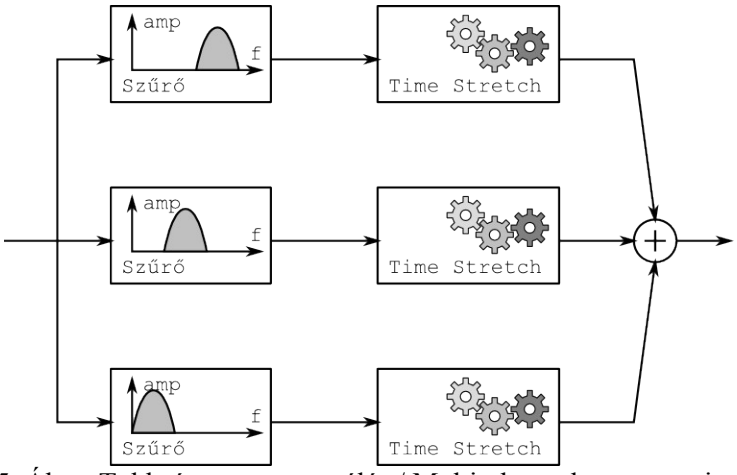
Mivel a $w[x]$ ablakfüggvény csak véges számú pontokban különbözik nullától (időtartományban korlátos), ezért a (2) egyenletből következik, hogy az eljárás kevés számítási erőforrást igényel, és viszonylag egyszerű. Sajnálatos módon a hangminőség is követi ezt a tendenciát. Többféle hibajelenség figyelhető meg az algoritmusnál, és teljesen kiküszöbölni egyiket sem lehet.

Az első ilyen hibajelenség a tempóingadozás. A jel szegmentálásakor az egyes szegmenseken belül megmarad az eredeti jelsebesség, hiszen ezen a részen nem módosít az algoritmus. Ebből az következik, hogy bár időben hosszabb részre nézve a sebességállítás bekövetkezik, ez mégis lokális tempóingadozásokkal fog járni az egyes szegmensek átlapolódása és a szegmensek közepe között. Ez az ingadozás annál nagyobb, minél nagyobb szegmensekkel dolgozunk.

Az átlapolódó szegmensek is hibajelenséget hordoznak magukkal. Két egymást követő szegmens átlapolódó részében lévő információ mindkét szegmensben jelen lesz, és a sebességállítás hatására, amikor a szegmenseket szét-, illetve összecsúsztatjuk, időben elcsúsznak egymáshoz képest. Emiatt megfigyelhető a tranziens jelek megduplázódása ezen a részen, ami a szubjektív megfigyelő számára már hallható, és zavaró elváltozás. Mivel az ablakfüggvényt úgy kell megválasztanunk, hogy az ne okozzon jelentős spektrális elváltozást a jelben, így az átlapolódás teljesen nem szüntethető meg, de az átlapolódó részek mérete, és az általuk keltett hatások viszonylag jól kézben tarthatóak.
Egy másik, az átlapolódásból származtatható hibajelenség a fésűszűrő hatás. A hiba következik az előzőekben leírt tranziens duplázódásból. Ha a szegmentálás után az eltolás mértéke kicsi (tipikusan néhány tíz minta), akkor az információduplázódásból származó elváltozás fésűszűrőként jellemezhető, hiszen egy adott pillanatban a jelet önmaga eltoltjával összegezzük. A leszívások száma a szűrőkarakterisztikában függ az eltolás mértékétől, a leszívások mértéke pedig a két szegmens ablakfüggvényének adott pontban vett értékétől. A leszívás akkor a legnagyobb, ha a két érték közel azonos, és akkor a legkisebb, ha az egyik érték közel nulla, tehát az egyik szegmensből származó információ elhanyagolható.
Mivel ez a szűrőhatás csak az átlapolódások területén van jelen, és ott is függ a két szegmens ablakfüggvényének az adott ponton vett értékétől, így ha egy adott frekvencián nézzük időtartományban a jelet, a sebességállítás után azt tapasztaljuk, hogy bizonyos frekvenciákon az átlapolódásoknál a jelszint a fésűszűrő hatás miatt lecsökken. Az így keletkezett hangerő-ingadozás miatt kialakul egy frekvenciaszelektív amplitúdómoduláció, mely szintén hallható elváltozásokat okoz.
Az algoritmus mégis közkedvelt, főleg a real-time megvalósítások terén, mert könnyű implementálni, és kevés erőforrást igényel. Hibáit és hiányosságait több változatban próbálták javítani. Mivel a legtöbb hibát a szegmensek közötti átmenet okozza, így a különbözőképpen megválasztott szegmentálási pontok az időtengely mentén, javíthatnak az algoritmus működésén. Ezeket a pontokat szinkronizálhatjuk a jel különböző paramétereihez, ezzel elérve, hogy bizonyos hibák kevésbé jelenjenek meg a végeredményben.
A Syncronized OverLap and Add (SOLA) a keresztkorreláció segítségével határozza meg azokat szegmentálási pontokat, ahol az egyes darabok egymásra csúsztatásánál azok a lehető legkevésbé korrelálnak, így csökkentve az információduplázódást, és az ezzel járó zavaró hatásokat. A keresztkorreláció kiszámítása azonban jóval több erőforrást igényel (Zölzer2002).
Ennek elkerülésére használják az Envelop-Matching for Time Scale Modification (EM-TSM) algoritmust, mely a keresztkorreláció kiszámolása helyett, a jelre illesztett burkológörbe alakulását használja, hogy meghatározza a szegmentálás paramétereit (Wong1998).
Mivel a szegmentálási algoritmusok hibáinak jelentős része a jelben lévő tranziens részek miatt keletkezik, ezért az is egy lehetőség, hogy a sebességállítást a tranziens jelek ideje alatt nem hajtjuk végre. Ez ugyan azt jelenti, hogy maga tranziens sebessége nem változik, és újabb tempóingadozási hibák kerülnek a kimeneti jelbe, de egy nagyobb időintervallumra nézve a sebességállítás elég pontos, és a többi hiba is csökkenthető (Wong1997).
Ha rendelkezésünkre áll plusz információ a jelről, például, hogy az monofonikus, vagy monofonikusként kezelhető, akkor ezen információ alapján is elvégezhetjük a szinkronizációt. A Pitch Syncronized OverLap and Add (PSOLA) algoritmus azt használja ki, hogy a jel rendelkezik alapharmonikussal, melynek periódusidejéhez illeszthetjük a szegmentálási pontokat (Zölzer2002, Schnell2002). A PSOLA másik tulajdonsága, hogy formánstartó, ezért használják előszeretettel a beszédfeldolgozásban, ahol főleg pitch shiftelést végeznek vele. Jól felparaméterezve lehetőség van csak a hang frekvenciáját megváltoztatni, és az emberi torok által formált szűrőt, mely kialakítja a formánsokat, változatlanul hagyni.
Ha a jelről nem mondhatjuk ki, hogy monofonikusként kezelhető, akkor lehetőségünk van Waveform similarity Syncronized OverLap and Add (WSOLA) algoritmust használni, mely a hullámforma hasonlóság alapján próbálja a szegmentálási pontokat megtalálni, valamivel általánosabbá téve a PSOLA algoritmust (Verhelst1993).

Többsávos szegmentálás

A hibák tovább csökkenthetők, ha figyelembe vesszük, hogy az egyes frekvenciasávok különböző paraméterezést, vagy akár különböző szegmentálási algoritmust is használhatnak, hogy minél kevesebb hibát okozzanak az eredményben. Mivel a jel magas és alacsony frekvenciás komponensei más sebességgel változnak, ezért ez a fajta a szétbontás ésszerű.
A többsávos szegmentálás első lépése, hogy a bejövő jelet egy szűrőbank segítségével szétbontjuk frekvenciasávokra, majd ezeket a sávokat külön feldolgozzuk valamelyik szegmentálás alapú algoritmussal, végül a sávok kimeneti jeleit összegezzük. Ezt szemlélteti az 5. ábra.

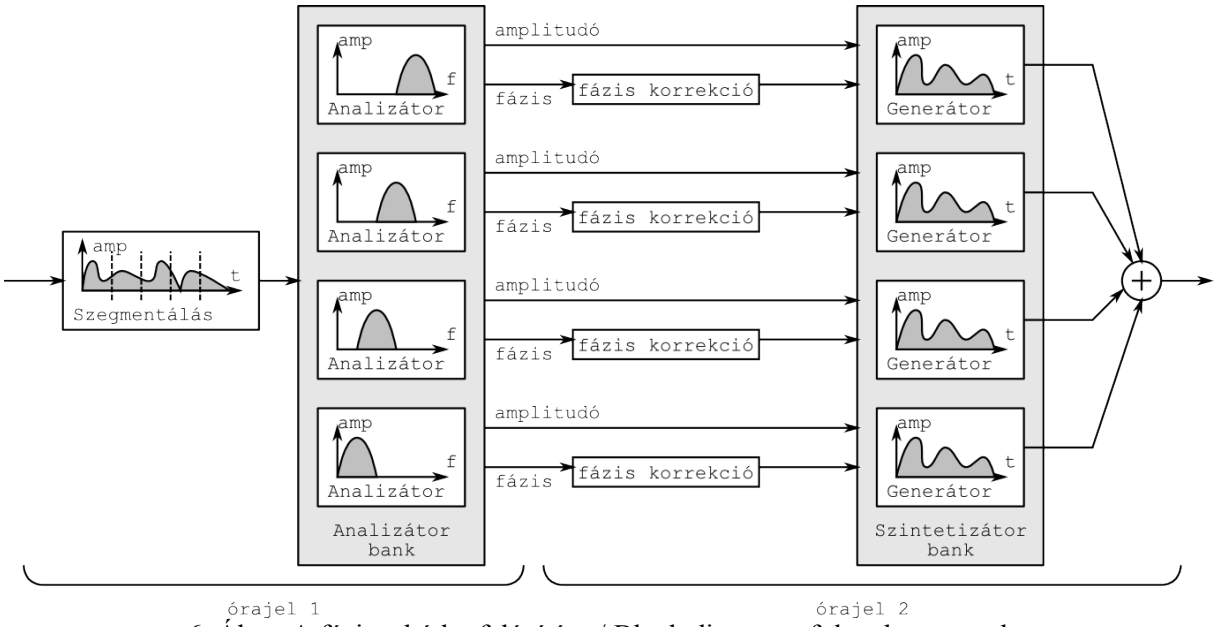


5. Ábra: Többsávos szegmentálás / Multi-channel segmentation

A többsávos szegmentálás, mivel több frekvenciasávon hajtja végre a time stretch algoritmust, ezért több erőforrást igényel, valamint minden sávon különböző paraméterei lehetnek az algoritmusnak, ami jelentősen megnöveli a paraméterek állapotterét, és nehézkessé teszi az algoritmus konfigurálását.

Fázis vokóder

Eddig a jelet főként időtartományban elemeztük, és nem használtuk ki a frekvenciatartomány nyújtotta tényleges lehetőségeket. A többsávos szegmentálás már egy lépés ez irányba, de valójában az időtartománybeli szegmentálásra épül, így nem sorolható teljes értékűen a frekvenciatartománybeli algoritmusok közé.
A fázis vokóder alapú algoritmus lényege, hogy a jelet az időtartományból áttranszformálja a frekvenciatartományba, ahol az idő- és frekvenciasíkon módosítja a jel különböző paramétereit, majd visszatranszformálva az időtartományba, előállítja a szintetizált, sebességállított jelet.
Ahhoz, hogy reprodukálhatóan le tudjunk írni egy időtartománybeli szegmenst a frekvenciatartományban, szükségesek az adott szegmens amplitúdó- és fázisinformációi. Így az algoritmus a jel időbeli szegmenseiből kinyeri a különböző frekvenciákhoz tartozó fázis- és amplitúdóértékeket, majd ezen információk felhasználásával felparaméterez egy szintetizátorbankot, amely generátorai segítségével előállítja a sebességállított jelet (Zölzer2002). Az analízis és szintézis más órajelen működik, ezzel biztosítva, hogy a kimeneti jelnek más legyen a sebessége, így abszolút fázisértékeket nem használhatunk a generátorok paraméterezésére, mert ekkor hiába a különböző órajelek, a fázisok kötöttsége miatt csak újramintavételezést végzünk a jelen. Ezért az amplitúdó mellett csak a fázisváltozást használjuk fel mint információt. Ezt szemlélteti a 6. ábra.



6. Ábra: A fázisvokóder felépítése / Block diagram of the phasevocoder

Azzal, hogy csak a fázis változását használjuk fel a szintézis során, információt vesztünk az adott szegmensről, és nem ismerjük az abszolút fázisértékeit. Ennek az a következménye, hogy az egyes frekvenciasávok egymáshoz képesti abszolút fázisai nem követik az eredeti jel frekvenciasávjainak egymáshoz képesti abszolút fázisait. Ez fázisinkoherenciát okoz a frekvenciasávok között, és olyan hatást kelt a szubjektív megfigyelőben, mintha a jelforrás egy zengő térben lenne elhelyezve. A hibajelenség csökkenthető, ha a fáziskorrekciókat több frekvenciasávot átfogva számoljuk ki (Laroche1999b).

Az analízis általában Short Time Fourier-Transzformáció (STFT) segítségével történik, amellyel, mint tudjuk, véges felbontás érhető el az idő-, és frekvenciasíkon, ezért kompromisszumot kell kötnünk az idő-, és frekvenciafelbontás terén. Minél nagyobb a felbontás az időtengely mentén, annál kisebb lesz a felbontás a frekvenciatengely mentén, és minél nagyobb a felbontást a frekvenciatengely mentén, annál kisebb lesz az időtengely mentén. Ebből kifolyólag ha a frekvenciatengely felbontását úgy választjuk meg, hogy az a szubjektív megfigyelőnek már ne legyen zavaró, akkor a tranziens jeleket nem tudjuk kellő időbeli felbontással reprodukálni, melynek következménye, hogy az ilyen jelrészletek „szétkenődnek” az időtengely mentén. A hibajelenség megszüntetésének egyik módszere lehet a sebességállítás kikapcsolása az ilyen jelrészletekre (Röbel2003). Ezzel elérhetjük, hogy mind az időtengely felbontásából, mind a fáziskoherenciából származó hibák hatásai csökkennek.

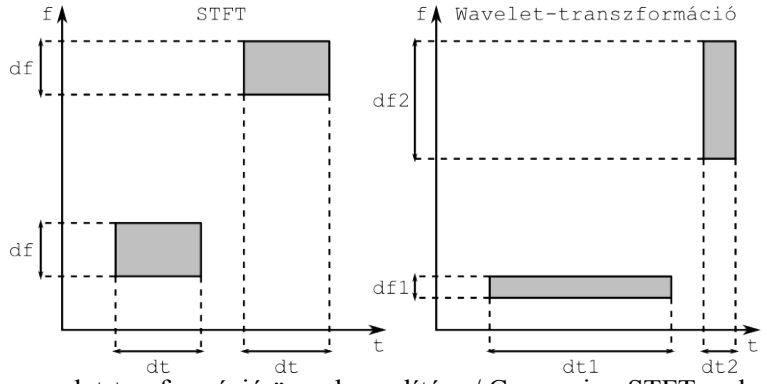
A szintézis általában inverz STFT-val, vagy szinuszgenerátorok segítségével történik. Mindkettőnek megvannak a maga előnyei és hátrányai. Míg a STFT a numerikus számolások szempontjából sokkal kedvezőbb, addig a szinuszgenerátorokból álló szintézisbank lehetőséget ad az egyes generátorok frekvenciáinak befolyásolására, így képesek vagyunk ennél az algoritmusnál rögtön végrehajtani a pitch shiftelést, valamint tudjuk a jel frekvenciakomponenseit harmonizálni, diszharmonizálni (Laroche1999a).

Mivel az analízist követően lehetőségünk van az egyes frekvenciakomponensek amplitúdóinak a módosítására, így a fázis vokóder algoritmus kis módosítással alkamazható mint dinamika-kompresszor, -expander, vagy többcsatornás zajszűrő.

WAVELET-TRANSZFORMÁCIÓ ALKALMAZÁSA

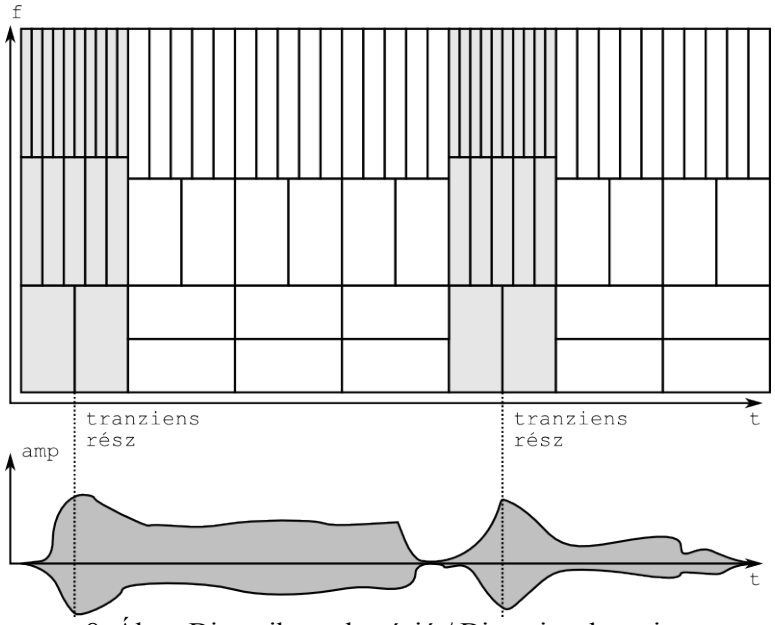
Mivel az STFT idő-, és frekvenciasík-felbontása egyenletes, ezért nem illeszkedik teljes mértékben a feldolgozandó jelhez, és az emberi halláshoz. Általában egy jel különböző frekvenciasávjait különböző módon érzékeljük. Az alacsonyfrekvenciás részek kevesebb időbeli felbontást és nagyobb frekvenciabeli felbontást igényelnek, míg a magasabb frekvenciás részeknél az időbeli felbontás a fontosabb, és a frekvenciabeli kevésbé, ezért ésszerű a STFT helyett olyan transzformációt alkalmazni, mely ezekhez a különbségekhez alkalmazkodik.

Megoldás lehet a több különböző felbontású párhuzamos STFT számítása is, melyekből fel lehet építeni egy olyan rendszert, amely a különböző felbontású STFT-ok közül mindig a jelnek az adott időpillanatban legalkalmasabb paraméterekkel számított STFT-jét alkalmazza (Bonada2000). Az eljárás problémája, hogy a több párhuzamosan számított STFT nagyobb számítási kapacitást igényel.
A wavelet-transzformáció is minden szempontból alkalmas erre a feladatra, hiszen magasabb frekvenciás komponenseknél nagyobb az időtengely felbontása, és kisebb a frekvenciatengely felbontása, míg az alacsonyabb frekvenciás komponenseknél a frekvenciatengely felbontása a nagyobb és az időtengely felbontása a kisebb (Mallat1999). Ezt szemlélteti a 7. ábra is.



7. Ábra: STFT és wavelet-tranformáció összehasonlítása / Comparing STFT and wavelet-transform

A wavelet-transzformációnál egy scale fügvénnyel, és egy mother wavelettel tudjuk leírni a teljes szűrőbankot. A scale függvény a 0 frekvencia körüli részét fedi le, míg a mother waveletből időtartománybeli eltolással, nyújtással, vagy zsugorítással származtatott daughter waveletek lefedik a frekvenciatengely többi részét.
Ha megfelelő a mother waveletet (például complex Morelet waveletet) használunk, akkor a szűrőbankunk kimenete ugyanúgy amplitúdó- és fázisinfomrációkat fog tartalmazni, mint a STFT-nál, tehát egyszerűen kicserélhető a fázis vokóder algoritmusnál az analízisbank wavelet-transzformációval megvalósítottra (Shew0000).
A párhuzamos STFT-ok módszeréhez hasonlóan (Bonada2000), a wavelet-transzormációnál is elképzelhető, hogy a transzformáció felbontását időben változtatjuk, ezzel létrehozva egy olyan rendszert, amely adaptálódik a jelhez. Ez azt jelenti, hogy a jel tranziens részeinél megnöveli a felbontást az időtengelyen, a periodikus jelrészeknél pedig a frekvenciatengelyen. Ezt szemlélteti a 8. ábra.

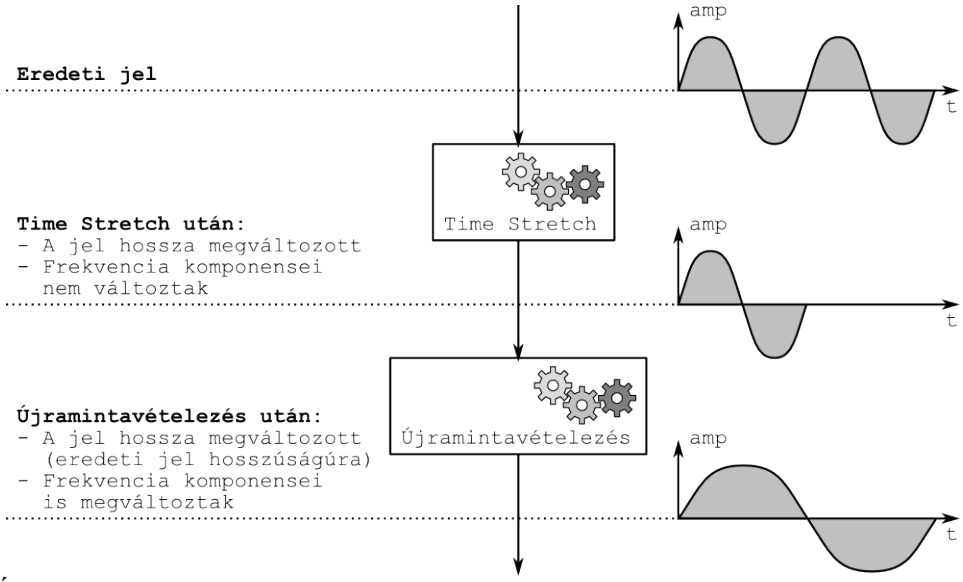


8. Ábra: Dinamikus adaptáció / Dinamic adaptation

Az adaptáció segítségével az egyes transzformációkat már nem kell párhuzamosan számolni, és így valamelyest csökkenthető az eljárás számításigénye. Természetesen a jelben detektálni a tranziens részeket, és ennek megfelelően beállítani az adaptációt nem egyszerű feladat, de nem is lehetetlen.

PITCH SHIFT

Minden time stretch algoritmusból egyszerűen származtatható pitch shift algoritmus, és a legtöbb szoftverben, amely a time stretch mellett pitch shiftre is képes, ez hasonlóan van megoldva. Nagy valószínűséggel még a korábban említett MPEX algoritmusnál is.
Hogy elérjük a pitch shift hatást, a time stretch után be kell iktatni egy újramintavételezést (Zölzer1997). Ezt szemlélteti a 9. ábra.



9. Ábra: Time stretch használata pitch shiftelésre / Using Time Stretch for Pitch Shifting

A time stretch-elés hatására megváltozik a jel hossza, és a frekvenciakomponensei változatlanok maradnak. Ezután, ha az újramintavételezést megfelelően felparaméterezve elvégezzük, elérhető, hogy a jel hossza újra megváltozzon, méghozzá úgy, hogy visszanyerje az eredeti hosszát. Mivel az újramintavételezés egyszerre változtatja meg a jel hosszát és frekvenciakomponenseit is, így végeredményként van egy feldolgozott jelünk, amely az eredeti jel hosszával rendelkezik, frekvenciakomponensei viszont módosítva lettek. Ez éppen a pitch shift definíciójának felel meg. Például ha egy 1 perc hosszú jelet 200%-ra nyújtunk time stretch algoritmussal, majd ezután újramintavételezzük az eredeti mintavételi frekvencia felével, akkor a mintaszámunk megegyezik a kiindulási mintáink számával, tehát a regisztrátum hossza nem változik. Mivel azonban újramintavételezést alkalmaztunk, ezért a jelet az eredeti mintavételi frekvenciával visszajátszva, a frekvenciakomponensek egy oktávval feljebb csúsznak.
Ezzel a módszerrel minden time stretch algoritmus használható pitch shiftelésre, de mint az korábban említve lett, a fázis vokóder alapú algoritmusoknál nincs szükség erre a plusz lépésre, mert azoknál az algoritmuskonál a generátorok frekvenciájának a módosításával is elérhető ugyanez a hatás.

MEGVALÓSÍTÁS ÉS SAJÁT EREDMÉNYEK

A megvalósítást alapvetően két nagy csoportra tudjuk bontani. Offline és online megvalósításra. Először offline megvalósításban vizsgáltuk a fent említett algoritmusok alap változatait. Mindezt MATLAB szkriptek segítségével oldottuk meg. Az offline megvalósítás előnye, hogy PC-n nagy számítási kapacitás, háttértár, és memória áll rendelkezésünkre, és így szabadabban próbálgathatjuk a különböző megvalósításokat anélkül, hogy a futásidő, vagy maga az eszköz korlátozna minket ebben.
Az első megvalósított algoritmus a szegmentálás alapú algoritmus volt. Futásidőben, mint az várható volt, nagyon jó eredményeket mutatott, hiszen az algoritmus egyszerűsége miatt igen kicsi a számítási

igénye. Az megvalósítás ennek megfelelően kellően egyszerű, stabil és robusztus. Ezért is szeretik ezt az algoritmusfajtát napjaink szoftvereiben. Manapság a legtöbb zeneszerkesztő és stúdiós programcsomag része valamilyen szegmentálás alapú time stretch és pitch shift algoritmus. A komplex szoftverekben és digital audio workstation-ökben, ahol a time stretch és pitch shift csak egy eleme a teljes jelfeldolgozó láncnak, szükséges, hogy time stretch és pitch shift mellett jusson még számítási kapacitás különböző visszhangokra, dinamika kompresszorokra, zengetőkre, és más, akár modellezésen alapuló eszközökre, miközben megkövetelt a real-time lejátszás és editálás.
A szegmentálás alapú algoritmus hangminőség/számítási kapacitás aránya nagyon jó. Nagyobb lassításokra kevésbé alkalmas, mert az időtartományban „széthúzott" jel által tartalmazott rövid tranziensek megsokszorozódnak, és visszhangszerű ismétlésként lesznek hallhatóak a feldogozott jelben. Továbbá a tapasztalatok alapján nagyon kis sebességállításnál a fésűszűrő hatás dominánsabbá válik, és a sebességállítás mértékéhez képest az állított jel minősége némileg romlott. Így a módszer a gyorsításra alkalmasabbnak bizonyul. A szegmentáláshoz használt ablakfüggvény formájával ezen némileg tudunk segíteni, és a hanganyaghoz igazított beállításokkal szélesebb körben használhatóvá válik az algoritmus.
A többsávos szegmentálás implementációja az előző algoritmus felhasználásával történt. A sávok száma tipikusan tíz körüli, ami azt jelenti, hogy a számítási kapacitás is egy nagyságrenddel nagyobb, hiszen a szegmentálást minden sávra végre kell hajtani, és ehhez hozzá jön még a szűrőbankkal való sávokra bontás is. A paraméterek beállítása nehézkes, mert minden frekvenciasávra külön be kell állítani a sávnak megfelelő értékeket, és ez nem triviális.
A hangminőség némileg javul, de mivel teljes egészében a szegmentálásra épül, így annak minden hibáját örökli, és ennek megfelelően a lassításnál jelentkezett a tranziens sokszorozódás, valamint nagyon kis sebességállításoknál a fésűszűrő hatás. Igaz, ezek nem a teljes frekvenciatartományon, hanem csak egy-két frekvenciasávon lesznek érzékelhetőek először. Továbbá rossz beállítások mellett a frekvenciasávok egymáshoz képesti késleltetései új hibaforrást hoztak a rendszerbe. Bár az algoritmus pár százalékkal nagyobb sebességállításra is alkalmas, a megnövekedett komplexitás, és konfigurációs nehézségek miatt nem igazán éri meg használni. Ezt bizonyítja az is, hogy ritkán található meg ez az algoritmus a nagyobb szoftverekben.
A következő algoritmus a fázis vokódoláson alapuló volt. Ennél az eljárásánál az implementációhoz szükséges FFT rutint használni, ami a számítási kapacitásban is meglátszik. Szerencsére az FFT-re rengeteg optimalizált algoritmus létezik, melyeket csak használni kell. A paraméterek száma jóval kevesebb, mint a többsávos szegmentálásnál, így az algoritmus konfigurációja is jóval könnyebb, és a paraméterek is egyszerűen kézben tarhatóak.
Az algoritmus felépítéséből kifolyólag a lassításnál nem jelentkeznek azok a problémák, amelyek a szegmentálásnál, így ez az eljárás jól használható lassításra. Akár többszörösére is nyújtható a jel anélkül, hogy jelentős zavaró hatások jelentkeznének. A gyorsításnál viszont az tapasztalható, hogy a tranziens részek „szétkenődnek", és például egy ütős hangszer tranziens kezdete elvész, vagy elmosódik. Ebből kifolyólag ez az algoritmus gyorsításra kevésbé alkalmas, mint a szegmentálás alapú.
A legtöbb szoftverben fázis vokóder alapú algoritmusok is vannak, melyek általában ki vannak egészítve valamilyen fajta adaptációval, hogy a jelhez igazodva jobb minőségű sebességállítást érjenek el. Tipikus, hogy ezek az algoritmusok a nagy számítási kapacitásuk miatt offline módon használhatóak csak, vagy célhardver szükségeltetik hozzájuk.
Az utolsó algoritmusfajta a wavelet alapú fázis vokódolás, ahol az FFT helyét a wavelet transzformáció veszi át, ebből kifolyólag ez némi számítási kapacitásigény-növekedéssel járhat. Tapasztalataink szerint ez azért számottevő volt, de ez valószínűleg a nem futásidőre optimált megvalósítás következménye.
A wavelet-transzformáció nyújtotta előnyök méréseim során is nagynak bizonyultak. A transzformáció használata miatt a gyorsításnál jelentkező „szétkenődés" jelentős mértékben csökkent, engedve ezzel, hogy a sebességállítás határait még jobban kitoljuk. Ez a konstrukció már felveszi a versenyt a szegmentálás nyújtotta gyorsítási lehetőségekkel, és mellette a fázis vokóder nyújtotta lassítási lehetőségekkel is rendelkezik.
Az online megvalósítás egy ADSP-BF537 EZ-KIT Lite kártyán történt, amely fixpontos 16 bites aritmetikával rendelkezik csak. Itt is a szegmentálás megvalósítása volt a legkisebb számításigényű feladat. A fázis vokódernél már a DSP határait feszegette a program. Mint említettük, a generátorbank

leggyakrabban vagy inverz STFT, vagy szinuszgenerátorok vektora. Az inverz STFT használatánál több százas nagyságrendű pontszámmal, az egyes frekvenciakomponensek kicsi értéke már olyan számolási és kvantálási hibát hozott a 16 bites rendszerben, hogy a szintetizált jel a szubjektív megfigyelőnek már túlzottan zajos volt. A szinuszgenerátoros megvalósításnál ez a probléma nem jelentkezett, de az egyes generátorok számolása annyi erőforrást igényelt, hogy csak erősen csökkentett sávszélesség mellett futott az algoritmus a DSP-n. A waveletes megvalósításnak már nem is volt létjogosultsága a processzoron. Valószínűleg egy gyorsabb, és 32 bites lebegőpontos aritmetikán ez az algoritmus is jobban szerepelt volna.

Összefoglalás

Az algoritmusok elméleti értékelése, implementáció utáni tapasztalataink, valamint futtatási eredményeink is egybevágóan tükrözték a következő eredményeket:

Algoritmus	Hosszváltoztatás	Megjegyzés
Időtartománybeli szegmentálás	70% - 110%	Gyorsításra alkalmasabb
Többsávos szegmentálás	65% - 115%	Gyorsításra alkamasabb
Fázis vokódolás	85% - 200% …	Nagyobb lassítás is elérhető
Wavelet alapú fázis vokódolás	65% - 200% …	Tranziensek részletei megmaradnak

Az időtartománybeli szegmentálás inkább gyorsításra alkalmas, mert a lassításnál a tranziensek duplázódása nagyon zavaró hatású.
A többsávos szegmentálás ezt a határt némileg kitolja, de még így is inkább gyorsításra alkalmas, és a rengeteg paraméter beállítása időigényesebb a felhasználó számára.
A fázis vokóder inkább lassításra alkalmas, mert gyorsításra használva a tranziens részeit a jelnek érzékelhetően „szétkeni”, ami főleg nagyobb mértékű gyorsításnál válik zavaróvá.
A wavelet alapú fázis vokóder az idő-, és frekvenciasík más jellegű felbontása miatt megtartja a tranziensek részleteit is, és nem „keni szét” őket, ezáltal nagyobb gyorsításokat is el lehet vele érni, mint az eredeti fázis vokóderrel.
Természetesen ezek a hosszváltoztatási határok függenek az éppen aktuális közönségtől, hiszen mindenki máshogy ítéli meg az egyes algoritmusok által okozott hibákat, és ezek súlyát, de általában, ha azt a számítási kapacitás engedi, érdemes a wavelet alapú fázis vokódert használni.

HIVATKOZÁSOK

[Zölzer2002]
Szerzők: Udo Zölzer
Könyvcím: Digital Audio Effects
Kiadó: JohnWiley&SonsLtd
év: 2002

[Zölzer1997]
Szerzők: Udo Zölzer
Könyvcím: Digital Audio Signal Processing
Kiadó: JohnWiley&SonsLtd
év: 1997

[Wong1998]
Szerzők: Wong, J.W.C.; Au, O.C.; Wong, P.H.W.
Cím: Fast time scale modification using envelope-matching technique(EM-TSM)
Kiadvány: ISCAS '98. Proceedings of the 1998 IEEE International Symposium of Circuits and Systems
oldalak: 550 - 553
év: 1998

[Wong1997]
Szerzők: Wong, P.H.W.; Au, O.C.; Wong, J.W.C.; Lau, W.H.B.
Cím: On improving the intelligibility of synchronized over-lap-and-add(SOLA) at low TSM factor",
Kiadvány: Speech and Image Technologies for Computing and Telecommunications., Proceedings of IEEE, volume 2
oldalak: 487 - 490
év: 1997

[Schnell2002]
Szerzők: Norbert Schnell; Geoffroy Peeters; Serge Lemouton; Philippe Manoury; Xavier Rodet
Cím: Synthesizing a choir in real-time using Pitch Synchronous Overlap Add (PSOLA)
Kiadvány: Proceedings of the IEEE 1st Benelux Workshop on Model based Processing and Coding of Audio
év: 2002

[Verhelst1993]
Szerzők: Verhelst, W.; Roelands, M.
Cím: An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech
Kiadvány: Acoustics, Speech, and Signal Processing 1993 IEEE International Conference, volume 2
oldalak: 554 - 557
év: 1993

[Laroche1999a]
Szerzők: Laroche J.; Dolson M.
Cím: New phase-vocoder techniques for pitch-shifting, harmonizing andother exotic effects
Kiadvány: Applications of Signal Processing to Audio and Acoustics, 1999 IEEE Workshop
oldalak: 91 - 94
év: 1999

[Röbel2003]
Szerzők: Axel Röbel

Cím: A new approach to transient processing in the phase vocoder
Kiadvány: Proc. of the 6th Int. Conference on Digital Audio Effects (DAFx-03)
év: 2003

[Laroche1999b]
Szerzők: Jean Laroche; Mark Dolson
Cím: Improved Phase Vocoder Time-Scale Modification of Audio
Kiadvány: Speech and Audio Processing, IEEE Transactions, volume 7
oldalak: 323 - 332
év: 1999

[Mallat1999]
Szerzők: Mallat, Stèphane
Könyvcím: A Wavelet Tour of Signal Processing, Second Edition (Wavelet Analysis & Its Applications)
Kiadó: Academic Press
év: 1999

[Shew0000]
Szerzők: Geoff Shew
Cím: A Continuous Wavelet Transform Based Pitch Shifting Method For Audio Signals
Kiadvány: ELEC 486 Term Project

[Bonada2000]
Szerzők: Bonada J.
Cím: Automatic Technique in Frequency Domain for Near-Lossless Time-Scale Modification of Audio
Kiadvány: International Computer Music Conference
oldalak: 396 - 399
év: 2000

[IzotopeWeb]
http://www.izotope.com/tech/radius/

[MPEXWeb]
http://mpex.prosoniq.com/