Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Méréstechnika és Információs Rendszerek Tanszék

Akusztikai sugárkövetés GPU-n

SZAKDOLGOZAT

Készítette

Gajdács András

Konzulens

dr. Bank Balázs

2013. december 20.

SZAKDOLGOZAT-FELADAT

Gajdács András (AZZO9U)
szigorló villamosmérnök hallgató részére

Akusztikus sugárkövetés GPU-n

A koncerttermek akusztikai viselkedésének szimulációja nagyban megkönnyíti a tervező munkáját, hiszen így az elvárt akusztikai paraméterek (pl. hangtisztaság, beszédérthetőség) előre becsülhetők, és az előrejelzett problémák még a terem megépítése előtt korrigálhatóak. A legtöbb teremszimulációs módszer alkalmas arra is, hogy a terem impulzusválaszát adott pozícióban levő hangforrás (virtuális hangszer vagy beszélő) és adott pontban lévő hallgató esetére meghatározza. Az előálló impulzusválasz segítségével lehetőségünk nyílik az auralizációra is, azaz meghallgatjuk, hogy az adott felvétel hogyan hangozna a modellezett teremben.

A jelenleg PC-n elérhető számítási kapacitás azonban még mindig csak egyszerűbb modellek esetén teszi lehetővé a teremmodellek valós időben történő futtatását. Ez esetben tipikusan valamilyen geometriai módszert, pl. a sugárkövetés valamelyik változatát alkalmazzák, mert a hangzás szempontjából legfontosabb középfrekvenciás tartományban adott számításigény mellett ez vezet a legpontosabb eredményhez. Az utóbbi években ezen algoritmusok grafikus processzort (GPU-t) alkalmazó változatai is megjelentek, abból a megfontolásból kiindulva, hogy az akusztikus sugárkövetés alaplépései nagyon hasonlatosak a GPU-n hatékonyan megvalósítható grafikus sugárkövetés lépéseihez. A hallgató feladata egy ilyen, GPU-t alkalmazó teremszimulációs program megvalósítása, mely alkalmas a virtuális terem impulzusválaszának számítására, valamint konvolúció segítségével offline auralizációra is.

A hallgató munkájának a következőkre kell kiterjednie:

- Tekintse át a termek akusztikai modellezésének lehetőségeit, különös tekintettel a geometriai módszerekre!
- Elemezze az irodalomból megismert sugárkövetési módszereket az algoritmusok párhuzamosíthatósága, valamint a GPU-n történő alkalmazhatósága tekintetében!
- A fenti elemzés alapján kiválasztott egy vagy két algoritmust implementálja MATLAB környezetben, és vizsgálja meg annak pontosságát téglatest alakú termek modellezésének esetére! Referenciaként a tükörforrások módszerét használja!
- A fentiek alapján kiválasztott, azaz megfelelően párhuzamosítható, ugyanakkor kellően pontos algoritmust implementálja PC-n futó alkalmazásként, a párhuzamosítható részeket pedig mind CPU-n, mind GPU-n futó verzióban készítse el!
- Hasonlítsa össze az elkészült teremszimulációs program futási idejét a csak CPU-n futó, ill. a GPU-t is kihasználó verziók esetére!

Tanszéki konzulens: Dr. Bank Balázs, docens

Budapest, 2013. október 11.

.....
Dr. Jobbágy Ákos
tanszékvezető

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

1117 Budapest, Magyar Tudósok krt. 2. I. ép. I.E.444.
Telefon: 463-2057, Fax: 463-4112
http://www.mit.bme.hu • e-mail: mitadm@mit.bme.hu

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott *Gajdács András*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2013. december 20.

Gajdács András
hallgató

Kivonat

Az akusztikai sugárkövetés termek impulzusválaszának meghatározására szolgáló sztochasztikus eljárás. Nagy számításigénye eddig kizárta valós idejű alkalmazhatóságát. A GP-GPU-k térhódításával azonban új, párhuzamos sugárkövető algoritmusok is megjelentek, amelyek képesek kihasználni a GPU-kban rejlő hatalmas számítási kapacitást.

Szakdolgozatom fő célja az akusztikai sugárkövetés és a HRTF-fel történő auralizáció megvalósítása GPU-n, és a valós idejű futtathatóságuk vizsgálata. Tanulmányoztam a termek impulzusválaszának felépítését a számításigény csökkentése érdekében, amely zárt terekben lehetséges a zengés újrahasznosításával.

Áttekintettem a Möller-Trombore metszésteszt-algoritmust, illetve kidolgoztam egy saját, a feladat igényeihez igazított, gyors és kis tárhelyigényű eljárást. Elemeztem a sugárkövetés során az elfogáshoz használt véges térfogatból adódó problémákat, mint a pontatlan távolság- és iránymeghatározás, és ezekre megoldási javaslatot tettem. Az impulzusválasz előállítását ismertettem frekvenciafüggő elnyelésű anyagok jelenléte esetén is.

Megismerkedtem a tükörforrások módszerével, amely referenciaként szolgált a sugárkövetéssel előállított impulzusválasz ellenőrzéséhez. Összefüggést állítottam fel az impulzusválasz statisztikai tulajdonságai és a követett sugarak száma között.

Implementáltam az általam kidolgozott metszésteszt-algoritmust CUDA nyelven, ezt felhasználva sugárkövető alkalmazást fejlesztettem. A mérések alapján ez bizonyos megkötések mellett valós időben is alkalmazható.

Bemutattam, hogyan lehetséges a sugárkövetés eredményének felhasználásával HRTF-en alapuló auralizációt megvalósítani frekvenciafüggő abszorpciójú anyagok egyidejű jelenléte esetén, majd az algoritmust megvalósítottam GPU-n.

Abstract

Acoustic ray tracing is a stochastic method for calculating the impulse response of a room. Until recently it couldn't be used in real-time applications due to its huge computational cost. With the advent of GP-GPUs new algorithms have appeared, which can profit from the vast computing capacity of GPUs.

The main goal of my thesis is implementing ray-tracing and HRTF-based auralization with GPUs with real-time applicability. I studied the structure of room's impulse responses to reduce computational costs and this is possible by reusing the reverberant part of the impulse responses.

I have overviewed the Möller-Trombore intersection algorithm and proposed a a new method with less computational and memory requirements. I analyzed the problems that are caused by the usage of a finite volume around the listener, like inaccurate distance and direction calculations, and suggested solutions for them. I showed the assembling procedure of an impulse response with frequency dependent and independent materials, too.

I studied the image source method, which was used as reference in the verification of the impulse response obtained by ray tracing. I stood up a link between the statistical attributes of the impulse response and the number of rays.

I implemented the proposed intersection algorithm in CUDA, and using this I created a ray-tracer application. Based on the tests it can be stated that this is appropriate for real-time usage.

I showed a way of realizing HRTF-based auralization using the results of ray-tracing when frequency dependent materials are also present, then I implemented the algorithm in CUDA.

Bevezető

A környezetünkből minket érő hanghatások percepcióját nagymértékben képesek befolyásolni a befoglaló tér akusztikai tulajdonságai. Egy koncert meghallgatásakor jelentkező szubjektív élményérzet, illetve egy konferenciateremben, vagy tanteremben elhangzó előadás beszédérthetősége, és így a kommunikáció sikeressége mind a terem fizikai kiterjedésének, alakjának és a felhasznált anyagoknak is függvénye. Ennek jelentőségét már az ókori építészek is felismerték, és eredményeiket színházak, amfiteátrumok tervezésekor alkalmazták. Tehát az akusztika mint tudományág gyökerei idáig nyúlnak vissza. Napjainkban az épülettervezés folyamatában egyre inkább használatosak a termek akusztikai szimulációjára szánt szoftverek, mint például az ODEON (Odeon, 2013), vagy a CATT (CATT, 2013), továbbá igény mutatkozik ezen szimulátorok valós idejű változatára is virtuálisvalóság-rendszerek, játékszoftverek élethűbb hangzásvilágának fejlesztése során.

A különböző akusztikai folyamatok egzakt leírásának matematikai és fizikai alapjait, a hanghullámok terjedésének egyenleteit a 19. század óta ismerjük. Ezek analitikus megoldása általában csak egyszerűbb, speciális esetekben lehetséges, leginkább elméleti jelentőségű problémák vizsgálatakor és új modellek felállításakor célravezető. Az utóbbi évtizedek rohamos számítástechnikai fejlődése azonban teret adott a numerikus számítási módszerek elterjedésének. Ezek kulcsmozzanata a probléma meghatározott módon történő diszkretizációja, amely így algoritmikusan kezelhetővé válik.

A numerikus módszerek a hullámegyenlet közelítő megoldását állítják elő a tér diszkrét pontjaiban. Ide sorolható többek között a végeselem-módszer (FEM), illetve az akusztikában viszonylag újkeletűnek számító, de egyre inkább elterjedő időtartománybeli véges differenciák módszere (FDTD) (Svensson and Kristiansen, 2002). Ez utóbbinál a hullámegyenletben szereplő parciális deriváltakat véges differenciás közelítéseikkel helyettesítve válik algoritmikusan számíthatóvá a feladat. Ezen módszerek vitathatatlan előnye az egzaktságuk olyan értelemben, hogy a térben felvett háló finomításával a megoldás

az analitikus eredményhez konvergál. Így tehát az ismert hullámtani jelenségek, mint a szóródás és a diffrakció, eredendően megjelennek a megoldásban, lévén, hogy ezek a hullámegyenletből levezethetők. Jelen dolgozat célja a valós idejű auralizáció, vagyis a virtuálisan felépített teremben elhelyezett (mozgó) hangforrás(ok) által a terem egy (mozgó) pontjában keltett hangfolyam előállítása. A terem átvitelét lineárisnak feltételezve ez adott konfiguráció mellett ekvivalens az idővariáns impulzusválasz bizonyos időközönkénti meghatározásával. Az előbb említett módszerek a térben kifeszített hálónak gyakorlatilag minden pontjában megadják az impulzusválaszt a számítások során, holott csupán a tér egyetlen pontjában lenne rá szükség. Pontosan emiatt ezek számításigénye nagy, ami alkalmatlanná teszi őket az itt kitűzött cél megvalósítására.

A numerikus módszereken kívül azonban más eljárások is rendelkezésünkre állnak, amelyek nem a hullámegyenlet megoldásán, hanem az az alapján alkotott, egyszerűsített modelljén alapulnak. Ilyenek a geometriai akusztika eszközei, mint a tükörforrások módszere, illetve az akusztikai sugárkövetés, amelyekről külön fejezetek szólnak részletesen. Az előző két eljárás közös tulajdonsága, hogy a pontszerű hangforrás egy kiválasztott hullámfrontjának terjedését sugarakkal reprezentálja, amelyek kizárólag egyenes vonalban haladnak, a határfelületeken visszaverődnek, de nem szóródnak. Ebből ered azon hátrányuk, hogy a hullámtani jelenségek egy részét, mint a diffrakció, vagy a szóródás, nem képesek figyelembe venni. Problémát jelent továbbá, hogy az impulzusválasznak csak egy időben korlátozott részét állítják elő kellő pontossággal, ezért meg kell vizsgálni azt, hogy a hangérzet szempontjából melyek az impulzusválasz lényeges paraméterei.

A feladat megoldása során törekedni kell arra, hogy az elkészült alkalmazás valós idejű szimulációt is lehetővé tegyen. Ehhez tisztában kell lennünk a PC-k számítási teljesítményével, és olyan algoritmust kell választani, amelyik valós időben elfogadható minőségű hangfolyamot állít elő. A számítógépek számítási teljesítményén általában az általános processzor (CPU) számítási teljesítményét értik, amely egy modern, Intel Core i7-4770K processzor esetében 100 GFLOPS ($\times 10^9$ FLoating point Operations Per Second), amennyiben a maximális, 8 darab szál fut egyidejűleg, és a feladat jellege olyan, hogy használhatók a SIMD vektorutasítások, egyéb esetben a fenti teljesítmény negyede az irányadó.

Nem szabad megfeledkezni azonban arról, hogy a jelenleg használt PC-k zömében megtalálható valamilyen dedikált grafikus processzor (GPU) is, amelyek általános cé-

lú programozása lehetővé vált néhány éve. Ezek számítási kapacitása messze felülmúlja az általános processzorokét: egy modern, nVidia GTX 780Ti típusú videokártya akár 5 TFLOPS-ra is képes. A memória-sávszélességek közti különbség is egy nagyságrendnyi: az említett CPU esetében 25 GB/s körüli, GPU esetében pedig 336 GB/s. Sajnos azonban a gyakorlatban előforduló problémák megoldására használt algoritmusok csak egy kis töredéke az, amely hatékonyan futtatható GPU-n, történetesen azok, amelyekben ugyanazon műveletsort kell különböző adatokon végrehajtani, és emellett minél kevesebb feltételes elágazás, illetve szálak közti adatdependencia van. A sugárkövetés esetében ez a feltétel teljesül, hiszen a sugarak egymástól függetlenül haladnak, és minden sugárnál a visszaverődés irányát és helyét kell meghatározni, így ezen eljárás nagy valószínűséggel hatékonyan implementálható GPU-n, ez pedig nagy előnyt jelent a megoldás valós időben történő meghatározásánál. A megvalósításhoz azonban szükséges a GPU architektúrájának megismerése, illetve az alapvető programozási paradigmák elsajátítása.

A dolgozat első fejezetében röviden ismertetek néhány, a feladat megoldása során nélkülözhetetlen, és/vagy hasznos akusztikai és pszichoakusztikai fogalmat. Ezután a második fejezetben áttekintem a geometriai akusztika által kínált két megoldást, a tükörforrások módszerét, amelyet referenciaként fogok használni a későbbiekben, illetve az akusztikai sugárkövetést. Ez utóbbinál kitérek a felmerülő problémákra és megoldási javaslatokat teszek rájuk. A harmadik fejezetben a két módszer által szolgáltatott impulzusválaszokat egyeztetem, és ismertetem, hogyan lehet garantálni a sugárkövetéssel előállított impulzusválasz bizonyos statisztikai tulajdonságait. Ezután ismertetem, hogyan lehet HRTF-en alapuló auralizációt megvalósítani, miközben frekvenciafüggő elnyelésű anyagok is jelen vannak a virtuális térben. Az ötödik fejezet a grafikus processzorok programozásába kíván rövid bevezetőt nyújtani. Ezen tudás birtokában a hatodik fejezetben sorra veszem az akusztikai sugárkövetés GPU-n való implementációja során felmerülő kérdéseket és értékelem az algoritmus teljesítményét, valamint a hetedik fejezetben ismertetem az impulzusválasz összeállításának menetét. Végül a nyolcadik fejezetben értékelem az eddigi munkát, és összegzem azokat gondolatokat, amelyek megvalósítása már nem fért bele a dolgozat kereteibe.

1. fejezet

Akusztikai és pszichoakusztikai fogalmak

1.1. HRTF (Head-Related Transfer Function)

Két darab egy síkban elhelyezett, irányfüggetlen mikrofon segítségével meghatározható egy tetszőleges, ugyanazon síkban fekvő hangforrás iránya a mikrofonokhoz képest, amennyiben ismerjük a hang terjedési sebességét, illetve hogy a mikrofonok által vett jelek között mekkora időbeli késleltetés lépett fel. Azonban már ebben az esetben sem lesz egyértelmű a megoldás, mivel a mikrofonokat összekötő egyenes a síkot két félsíkra osztja, és adott késleltetésnél mindkettőben kijelölhető egy annak megfelelő irány.

Tehát ha az emberi fülek irányfüggetlen mikrofonokként működnének, még egyértelmű síkbeli lokalizációra sem lennének alkalmasak. A valóságban azonban a külső fülek és a fej által alkotott struktúrán a hanghullámok szóródnak, visszaverődnek, illetve elhajlanak, a tér különböző pontjaiból érkező hanghullámok ily módon más és más transzformáción esnek át, amely transzformációknak azonban megfeleltethető egy-egy lineáris átviteli függvény. Ha kellően a távoltérből, vagyis néhány méternél távolabbról vizsgáljuk az említett átviteli függvényeket, akkor azt találjuk, hogy azok már csak az iránytól függenek. Az agy ezen átviteli függvények ismeretében meg tudja határozni a hangforrás irányát.

Minden embernek mindkét füléhez definiálható tehát a fejhez rögzített gömbi koordinátarendszerben egy távolságtól függetlennek tekinthető $H(f,\theta,\varphi)$ átvitelifüggvény-halmaz, amely kapcsolatot teremt a frekvenciamenet és beérkezés iránya között. Ezt nevezzük a fej és a külső fülek komplex átviteli függvényének, angol terminológiában *Head-Related*

Transfer Function, röviden **HRTF**-ként szerepel (J. Blauert, 1999).

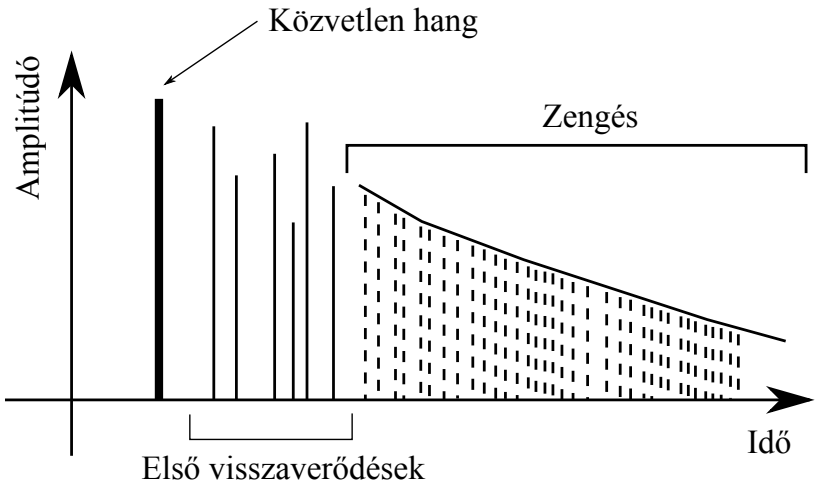
A HRTF-et általában süketszobában végzett mérésekkel lehet meghatározni különböző diszkrét irányokban úgy, hogy az alany hallójárataiba kifelé néző mikrofonokat, az adott irányban pedig egy pontszerűnek tekinthető hangforrást (távol lévő hangfalat) elhelyezve mérjük az így kialakult rendszer átvitelét. A mérések során a hangforrás és az alany távolsága állandó, tehát egy fej körüli gömbfelület mentén kell változtatni a hangforrás helyzetét. A tanulmányok (Simon Skluzacek, 2012) szerint az emberi hallás iránybeli felbontóképessége 15°-nál nem jobb, így ennél sűrűbben nem érdemes mérési pontokat felvenni. Az átviteli függvény helyett általában az annak megfeleltetett lineáris FIR szűrő impulzusválaszát szokás tárolni, amelyet **HRIR**-nek neveznek (*Head-Related Impulse Response*).

A HRTF-et felhasználva virtuális térben auralizációt valósíthatunk meg, vagyis egy hang irányának ismertében kiszámíthatók a binaurális (a két hallójárat bejáratához érkező) jelek, amelyeket fej-, vagy fülhallgatóval hallgatva hang alapján is tájékozódni tudunk a virtuális térben. Sajnos a HRTF nagyban függ a külső fül és a fej alakjától és méretétől, ezért minden egyén esetén csak méréssel lehet pontosan meghatározni azt. Amennyiben nincs lehetőségünk mérésre, úgy egy HRTF adatbázisból egy olyan alany mérési eredményeit kell kiválasztanunk, akinek a fejszerkezete leginkább közelíti az alkalmazást felhasználni kívánó személyét, az auralizáció minősége azonban ilyenkor természetesen romlik.

Amennyiben HRTF-fel szeretnénk térhangzást előállítani, úgy számolnunk kell azzal, hogy a számításigény jelentősen növekedni fog. Részben emiatt is fontos megismerni, hogy hogyan épül fel egy terem impulzusválasza, és ki lehet-e használni valamelyik tulajdonságát arra, hogy a számításigényt csökkentsük.

1.2. Termek impulzusválaszának tagolása

A termek impulzusválaszát három, az impulzusválasz grafikonján is többé-kevésbé jól elkülöníthető részre bonthatjuk, amit (Howard and Angus, 2009) szerint ismertetek.



1.1. ábra. *Egy terem impulzusválaszának tipikus felépítése*

1.2.1. Közvetlen hang

Egy teremben elhelyezett hangforrásból a hallgatóhoz először a legrövidebb úton haladó hanghullám fog megérkezni. Amennyiben a hallgató rálát a hangforrásra (nincs közöttük akadály), úgy ez a közvetlen hangot és a légvonalbeli utat jelenti, az impulzusválaszban pedig ez adja az első és – hacsak nem egy speciális, szimmetrikus elrendezésről van szó – a legnagyobb amplitúdójú csúcsot. Ekkor az első csúcs megjelenésének ideje az impulzusválaszban a hangforrás és a hallgató távolságának és a hangsebességnek a hányadosaként számítható.

A közvetlen hangnak fontos szerepe van az impulzusválaszban, mivel ez a forrás által kibocsátott hangot jelenti, amely kizárólag frekvenciafüggetlen csillapítást, és késleltetést szenved, tehát a szoba határfelületeinek tulajdonságai ebben még nem jelennek meg. Emiatt például a beszédérthetőség szempontjából előnyös, hogy a közvetlen hang minél nagyobb súllyal kerüljön az impulzusválaszba. A hallgató a relatív intenzitása alapján tudja megítélni a hangforrás távolságát, továbbá az iránymeghatározás is részben ez alapján történik.

1.2.2. Korai visszaverődések

Az impulzusválasz azon csúcsait, amelyek a hangforrástól maximum néhány (3-4) határfelületen történő visszaverődés után jutnak a hallgatóig, korai visszaverődéseknek nevezzük (angol terminológiában: early reflections). Ezek intenzitásukban elmaradnak a közvetlen hangtól, és attól nagyobb késéssel érkeznek meg a hallgatóhoz. Fontos meg-

jegyezni, hogy ezek iránya, száma és intenzitása nagyban függ a hallgató és a forrás terembeli elhelyezkedésétől. Éppen ezért képes az agyunk felhasználni ezeket a forrás irányának és távolságának meghatározásánál, kiegészítve a közvetlen hangból nyert információkat. Emellett a terem méretét is ez alapján tudjuk becsülni, ez felelős a térérzetért.

1.2.3. Zengés

Sokszori visszaverődés után a hullámtér egyre inkább diffúzzá válik, és a hullámok minden irányból egyformán, egyre gyakrabban érik a hallgatót. Ezt a jelenséget nevezzük zengésnek (reverberant sound), amely tehát a korai visszaverődésekkel ellentétben már nem hordoz információt a forrás irányáról, csupán a teremérzethez járul hozzá. Ha a zengés gyorsan lecseng, akkor kényelmetlenül érezhetjük magunkat egy adott teremben. Egy koncerten a zengés teszi lehetővé, hogy a hallgató füle összegezze a különböző visszaverődési utakon érkező hullámok energiáját, amitől például egy koncert esetében a hangszerek hangját gazdagabbnak, erőteljesebbnek érezzük. Ugyanakkor egy előadás esetében, ha a terem zengésének lecsengése túl lassú, az rontja a beszéd érhetőségét.

A zengés lecsengése egy közelítőleg exponenciális folyamat, amennyiben a terem határfelületeinek van csillapítása. Ha a felületek csillapítását jellemző abszorpciós együttható kicsi, akkor viszonylag sok visszaverődés után is jól hallható zengés van jelen, vagyis csak lassan fog leépülni a diffúz hullámtér. Az impulzusválasz nagy részét ilyenkor a zengés fogja kitenni, és akár több másodperc is kellhet ahhoz, hogy már nem hallható szintre essen vissza a hangintenzitás. Csillapítás nélküli esetben pedig nincs lecsengés, a zengés állandósul.

Nagy könnyebbséget jelent a numerikus számítások során, ha kihasználjuk a zengést keltő hullámtér diffúz mivoltát, vagyis azt, hogy a teremben a különböző forrás- és hallgatópozíciók mellett nyert impulzusválaszok zengés része ugyanazt az érzetet kelti, nem lehet köztük különbséget tenni. Így egy teremben elegendő egyszer kiszámítani a teljes impulzusválaszt egy tetszőleges – de általános, nem szimmetrikus – elrendezés mellett, majd a továbbiakban, amikor más helyre kerül a forrás és/vagy a hallgató, csupán a korai visszaverődésig határozzuk meg az új impulzusválaszt, a zengést pedig utólag keverjük hozzá.

1.3. Abszorpciós együttható

Határfelületen visszaverődve egy hanghullám intenzitása csökkenhet, vagyis az energiájából valamennyi elnyelődhet. Előfordulhat, hogy az abszorpció függ a beesési iránytól is, egyszerű modellek esetében azonban egyetlen skalárral leírható a jelenség. Azt a $[0,1]$ intervallumba eső számot, amely kifejezi, hogy a beeső hullám energiájának hányadrésze nyelődik el visszaverődés közben, abszorpciós együtthatónak nevezzük. Jele: α. Az abszorpciós együttható függ a mérési frekvenciától, anyagi minőségtől, a felfogatástól, valamint a felület kialakításától is.

1.4. Átlagos szabad úthossz

Egy teremben az átlagos szabad úthossz (*Mean Free Path – MFP*) az a távolság, amelyet a benne elhelyezett pontszerű részecske megtesz két ütközés között átlagosan. Levezethető, hogy ez egyenes arányban van a terem térfogatával (V), illetve fordítottan aránylik annak belső felszínéhez (S) (Howard and Angus, 2009):

$$MFP = \frac{4V}{S} \tag{1.1}$$

1.5. RT60

Az RT_{60} a termet jellemző paraméter; azt az időtartamot adja meg, amely alatt a hangforrás által kibocsátott hanghullám egységnyi felületre vett teljesítménye 60 dB-lel csökken, mialatt a határfelületeken visszaverődéseket szenved (Howard and Angus, 2009). A 19. században *Wallace Clement Sabine* vezette le a következő összefüggést, amelyet az átlagos szabad úthosszból kiindulva nyert (c_{air} a hangsebesség, a az átlagos abszorpciós együttható a terem felületein):

$$RT_{60} = \frac{24 \cdot \ln(10)}{c_{air}} \frac{V}{Sa} \tag{1.2}$$

1.6. EDC

Az EDC (Energy Decay Curve) egy görbe, amely az impulzusválasz energiájának még hátralévő részét az összenergiájához hasonlítja, és azt logaritmikusan ábrázolja. Előállítá-

sa diszkrét idejű jeleknél (N a minták száma):

$$EDC[k] = 10\log_{10}\left(\frac{\sum_{i=k}^{N-1} h^2[i]}{\sum_{j=0}^{N-1} h^2[j]}\right) \tag{1.3}$$

Jelen dolgozat szempontjából azért releváns, mert egy zárt terem impulzusválaszának EDC görbéjén a zengés alatt egy egyenes látható annak exponenciális lecsengése miatt, amely lehetőséget nyújt a sugárkövetéssel előállított impulzusválasz ellenőrzésére.

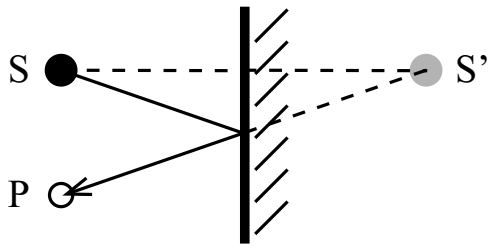
2. fejezet

Geometriai akusztikai módszerek

2.1. Tükörforrások módszere

2.1.1. Az eljárás ismertetése

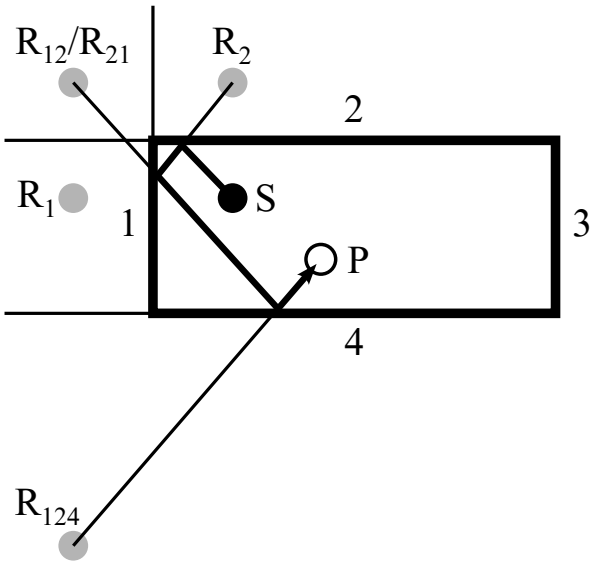
A tükörforrások módszere (Allen and Berkley, 1979) kis, téglatest alakú szobák akusztikai szimulációjára, adott pozícióban lévő hangforrás és hallgató mellett pedig az így kialakuló lineáris rendszer impulzusválaszának véges ideig történő meghatározására szolgáló determinisztikus eljárás. A hangforrást pontszerűnek feltételezi, az abból egy adott időpontban kiinduló hullámfrontot, illetve annak időbeli terjedését pedig sugarak reprezentálják. Ha egy sugár falnak ütközik, akkor az szóródás nélkül tükrös visszaverődést szenved (csillapítás figyelembevétele lehetséges, akár frekvenciafüggően is). A tükörforrások módszerének alapállítása, hogy egy sugár visszaverődése modellezhető úgy, mintha egy virtuális forrás lenne az eredeti sugárforrás adott falon vett tükörképének helyén.



2.1. ábra. *Tükörforrás szerkesztése*

Nemcsak az elsőrendű, vagyis közvetlenül a hangforrásból érkező sugarak visszaverődését modellező virtuális források, hanem a magasabb rendűek is képezhetők az előző rendben képződött virtuális forrásoknak a terem falaira történő ismételt tükrözésével (ki-

véve arra a falra, amelyikre történő tükrözéssel az adott forrás létrejött).



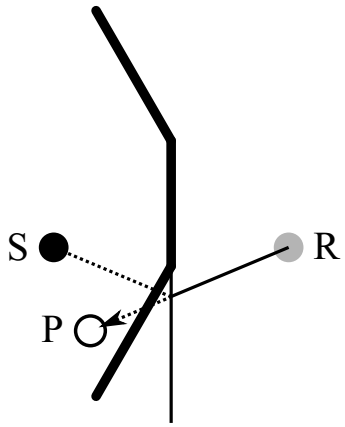
2.2. ábra. *Útrekonstrukció*

A 2.2 ábrán látható R_{12}/R_{21} forrás mind R_1, mind R_2 tükrözésével létrejönne, ilyen esetekben is csak egyszeres súllyal kell figyelembe venni az impulzusválasz számítása során, mivel minden virtuális forráshoz csak egyetlen út szerkeszthető a teremben. Jelen esetben ennek menete a következő: az R_{124} forrást összekötjük a hallgatóval, ez az szakasz pedig megad egy metszéspontot a 4-es oldalon. A metszéspontot összekötjük a R_{124} forrás ősével, vagyis R_{12}/R_{21}-gyel, ez ismét egy metszéspontot ad, ezúttal az 1-es számmal jelölt oldalon. Ezt a metszéspontot R_1-gyel és R_2-vel is össze kell kötni, de ezen két szakasz közül csak az lesz az érvényes, amelyik metsz egy másik oldalt. R_2-vel összekötve a 2-es oldalon adódik egy metszéspont, melynek a forrással történő összekötése után a terem belsejébe eső szakaszok összessége alkotja a teljes utat.

Komplexebb geometriájú termeknél szintén használható a tükörforrások módszere, azonban ilyenkor az előbb említett útvonal rekonstruálása minden létrejövő forrás esetében elengedhetetlen az úgynevezett rejtett források jelenléte miatt.

A 2.3 ábrán látható konstrukció esetében P hallgató számára az R virtuális forrás egy rejtett forrás, ugyanis a terembeli út rekonstruálása során a PR szakasz a 2-es oldallal nem, csak annak síkjával ad egy metszéspontot.

Egy általános geometriájú, n oldalból álló teremben minden, legfeljebb r-ed rendű visszaverődés kiszámításához $O(n^r)$ művelet szükséges, beleértve a számítástechnikailag igencsak költséges útrekonstrukciót. Ez az exponenciális komplexitás az algoritmust



2.3. ábra. *Rejtett forrás*

használhatatlanná teszi bonyolultabb elrendezések szimulációjánál. Amiért azonban mégis hasznos, hogy a módszer determinisztikus, és az ennek segítségével számított impulzusválasz a fejezet elején megadott feltételek és téglatest alakú terem mellett megegyezik a hullámegyenlet által szolgáltatott analitikus megoldással (Allen and Berkley, 1979) (Stephenson, 1990). Az impulzusválasz birtokában lehetőség nyílik más módszerek által szolgáltatott megoldások verifikációjára. Nem téglatest alakú termek esetében a fellépő diffrakciót a módszer képtelen figyelembe venni, ezért kisfrekvencián nem megbízható az eredmény.

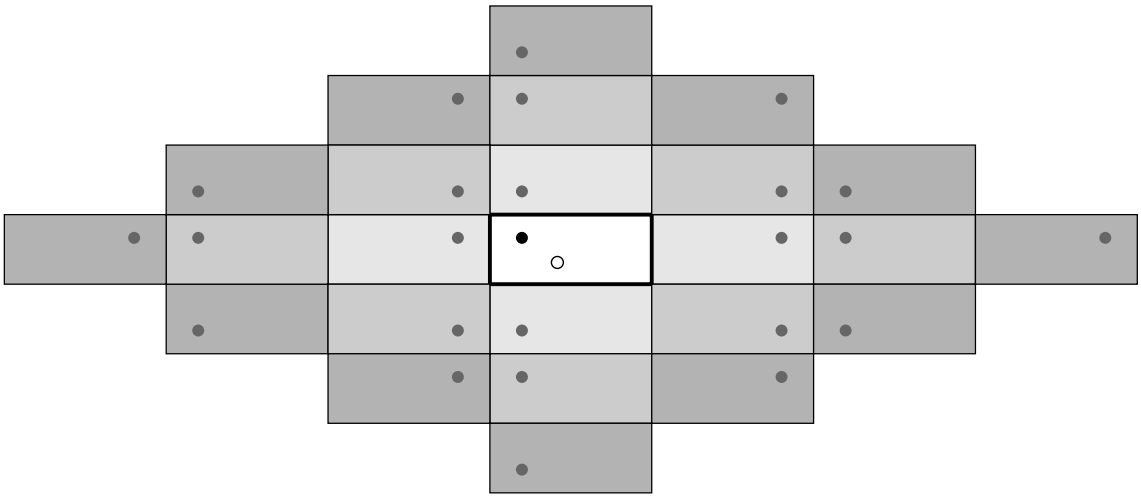
2.1.2. Impulzusválasz számítása

A virtuális források koordinátáinak birtokában számítható azok távolsága a hallgatótól. Pontforrás esetén a hullámfront gömb alakú, a kisugárzott hangenergia pedig egyenletesen oszlik el ezen gömbfelület mentén. Ebből következik, hogy az egységnyi felületre eső hangteljesítmény csökken a forrástól távolodva, mégpedig r^{-2}-nal arányosan, ahol r a forrástól vett távolság. Az impulzusválasz előállításához szükség van egy mintavételi frekvenciára. A választás a dolgozat egészében egységesen a hangfeldolgozásban elterjedt, $f_s = 44.1\,kHz$-es mintavételi frekvenciára esett. Ismervén a hang terjedési sebességét levegőben $\left(c_{air} = 340\,\frac{m}{s}\right)$ számítható a hang késése, amelyből a következő összefüggés alapján megkapható, hogy az impulzusválasz hányadik mintájához, és mekkora súllyal járul hozzá az adott forrás:

$$n = \left\lfloor \frac{r f_s}{c_{air}} \right\rfloor \qquad w = \frac{(1-\alpha)^k}{r^2} \tag{2.1}$$

Itt n a minta tömbbeli indexe, w a súlya, k a forrás rendje, az α pedig a falak (frekvenciafüggetlen) abszorpciós tényezője.

Lévén, hogy az itt kapott impulzusválaszt referenciaként használjuk a továbbiakban, elengedhetetlen annak biztosítása, hogy meghatározott ideig egzakt legyen. Ehhez érdemes alaposabban megvizsgálni, hogy mi történik a források tükrözésekor.



2.4. ábra. *A tükörforrások előállítása a terem tükrözésével*

Amint a 2.4 ábrán is látható, a források tükrözése ekvivalens az azt körülvevő terem tükrözésével. A termek sötétedő háttere a tükrözés növekvő rendjére utal. Ha a hallgató köré egy R sugarú kört rajzolunk, akkor a körön belül találjuk azokat a forrásokat, amelyeknek R/c időn belül van járulékuk az impulzusválaszban. Adott maximális rendű tükörforrások mellett tehát azt a maximális sugarú kört kell felvenni, amelynek még teljes területe fedve van a tükrözött termek által. Ha fedetlen területet is megengednénk, akkor elképzelhető, hogy az eggyel nagyobb maximális rendű tükörforrások mellett a körön belülre kerülne egy újabb forrás, ami pedig az impulzusválasz módosulását vonná maga után. Úgy lehet tehát ezen kör sugarát algoritmikusan meghatározni, hogy megkeressük az eggyel nagyobb rendű tükörforrások közül a hallgatóhoz legközelebbit. Mivel az ábrán látható módon a termek tükörképei rétegesen épülnek egymásra, így biztosan nem fognak a magasabb rendű tükörképek sem járulékot adni adott sugár mellett.

2.2. Akusztikai sugárkövetés

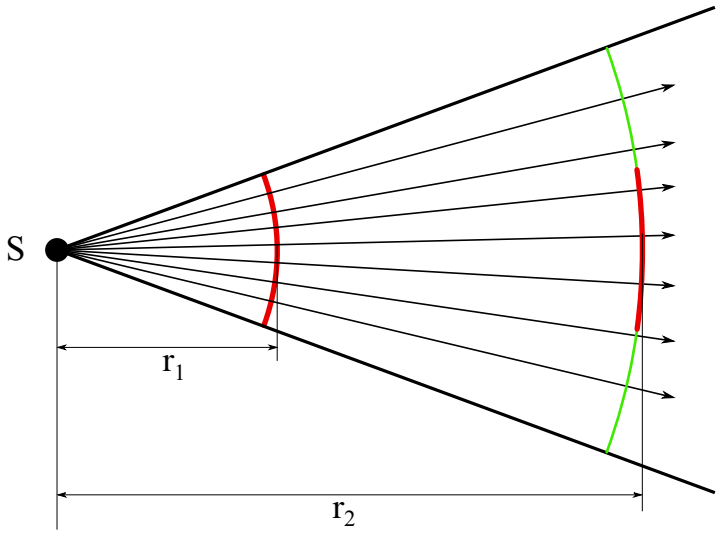
Az akusztikai sugárkövetés a tükörforrások módszeréhez hasonlóan a geometriai akusztika egyik módszere, amely segítségével sztochasztikusan közelíthető egy terem impulzusválasza adott hangforrás- és hallgatópozíció mellett (Krokstad et al., 1968)]. Feltételezi, hogy a hangforrásból az energia kisugárzása diszkrét kvantumokban, sugarak formájában történik, amelyek egyben a hang terjedését is hivatottak reprezentálni. Minden sugár kezdeti energiája megegyezik az összenergia és az összes kibocsátott sugár számának hányadosával. A sugarak a hangforrást a tér minden irányába véletlenszerűen, egy gömbfelület mentén egyenletes eloszlásban hagyják el, bár az eloszlás módosításával lehetőség van a forrás irányfüggő sugárzásának figyelembevételére is. A sugarak a közegben érvényes hangterjedési sebességgel haladnak, a határfelületeken pedig tükrösen visszaverődnek, illetve (frekvenciafüggő) csillapítást szenvednek.

Alapesetben a modell az olyan hullámtani jelenségekkel, mint a diffrakció, vagy a szóródás, nem számol, azonban történtek kísérletek ezek hiányának pótlására (éldiffrakció: (Lokki et al., 2002)(Taylor et al., 2009), szóródás: (Rindel, 1995). Emiatt a módszer a terem alacsony frekvenciás átvitelének szimulációjára nem alkalmas. Gömbhullámok terjedésekor a hullám intenzitása fordítottan arányos a távolság négyzetével, ezzel azonban külön nem kell számolnunk a módszer használata során, ugyanis adott térszögön belül elindított sugarak elé egy adott nagyságú, terjedési irányra merőleges felületet felvéve, majd a felületet a sugárforrástól távolítva az azt metsző sugarak száma, vagyis a fluxus fordítottan arányos lesz a távolság négyzetével.

Ez könnyen belátható a 2.5 ábra tanulmányozásával. A sugarak egy adott Ω térszögben terjedve egy egyre növekvő területű gömbhéjon oszlanak el, amelynek felszíne az r távolság függvényében:

$$T = \Omega r^2 \tag{2.2}$$

Emellett egy adott nagyságú gömbfelületet felvéve az azt metsző sugarak átlagos száma, amennyiben a sugarak iránybeli eloszlása egyenletes, a két felület hányadosával lesz



2.5. ábra. *A sugárfluxus egységnyi felületen, különböző távolságoknál*

arányos.

$$n_{avg} = \frac{n_{all} T_0}{\Omega r^2} \sim \frac{1}{r^2} \tag{2.3}$$

2.2.1. A sugárkövetésnél felmerülő problémák és megoldásuk

Ha egy sugár eléri a hallgatót, akkor az az impulzusválaszhoz az addig megtett út és a közegben érvényes terjedési sebességből számítható időpillanatban (illetve diszkrét időben ezzel analóg módon, a k-adik mintánál) az út közben elszenvedett csillapításoknak megfelelő nagyságú w járulékot ad, majd továbbhalad (vagyis modellünkben nem vesszük számításba a hallgatót). Formálisan, ha $y[k]$ a diszkrét idejű impulzusválasz, d a megtett út, w a járulék nagysága, M a sugár ütközéseinek száma és α egységesen az összes fal abszorpciós tényezője:

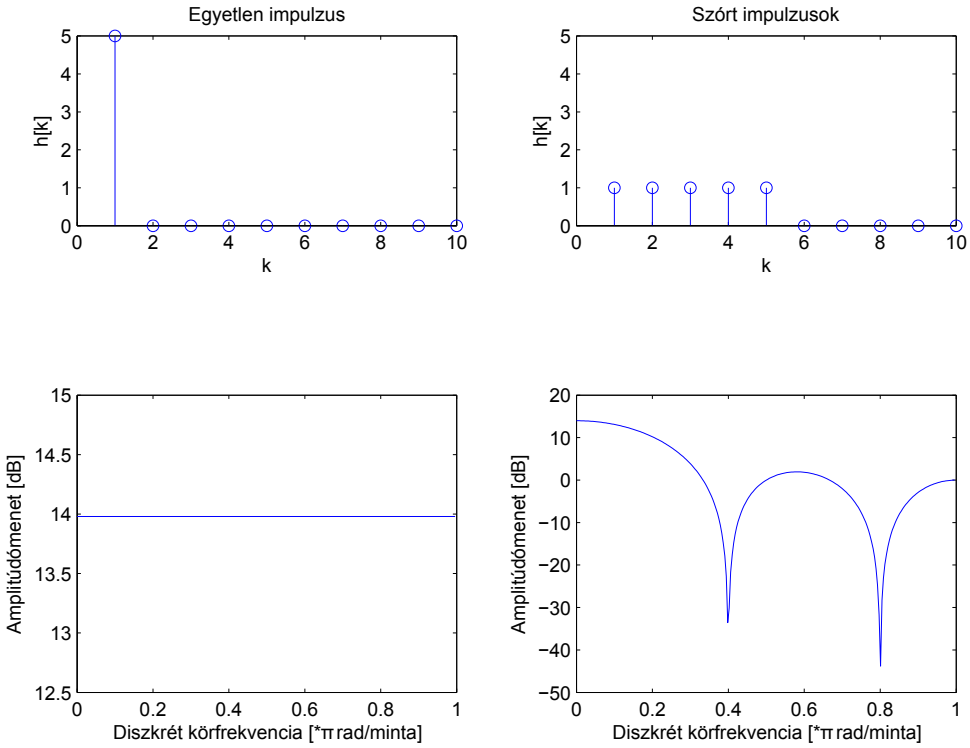
$$k = \frac{d}{c_{air}} \cdot f_s \tag{2.4}$$

$$w = (1-\alpha)^M \tag{2.5}$$

Az azonban, hogy egy sugár egy pontszerű hallgatót eltalál, nulla valószínűségű esemény. Ahhoz tehát, hogy a sugarak elfoghatók legyenek, definiálni kell a hallgatók köré egy véges térfogatot, amelyet ha egy sugár érint, akkor azt érvényes találatnak könyvelünk el. Célszerű egy gömböt választani a hallgató köré, mert így minden irányban egyformán

kerül kiterjesztésre a metszéshez szükséges minimális távolság. Elővigyázatosnak kell lennünk azonban ezen gömb sugarának megválasztásakor, mivel minél nagyobb sugarat választunk, annál nagyobb a valószínűsége, hogy az impulzusválaszban fals extra pulzusok jelennek meg, túl kicsiny átmérő esetén pedig több sugarat kell indítani az ugyanolyan statisztikai tulajdonságok eléréséhez, tehát nő a számításigény.

Ennek kapcsán azonban felmerül egy másik probléma is: az ugyanazon forrásokból (vagy virtuális forrásokból) érkező sugarak azon térszögön belül bárhol eltalálhatják a hallgatót körülölelő gömböt, amelyből az látszik. Akár hozzávesszük a gömbön kialakított metszéspont és a gömb középpontjának távolságát a sugár által megtett úthoz, akár sem, ez azt fogja eredményezni, hogy az ugyanazon forrásból indított sugarak által megtett utak a hallgató tényleges távolságától eltérhetnek, amely eltérés nagysága legfeljebb a gömb sugara lehet. Ez azzal a következménnyel jár, hogy a diszkrét idejű impulzusválaszban nem egyetlen csúcs fog megjelenni, hanem az akörül lévő $R \cdot \frac{f_s}{c_{air}}$ darab pozíció között fognak eloszlani a járulékok.

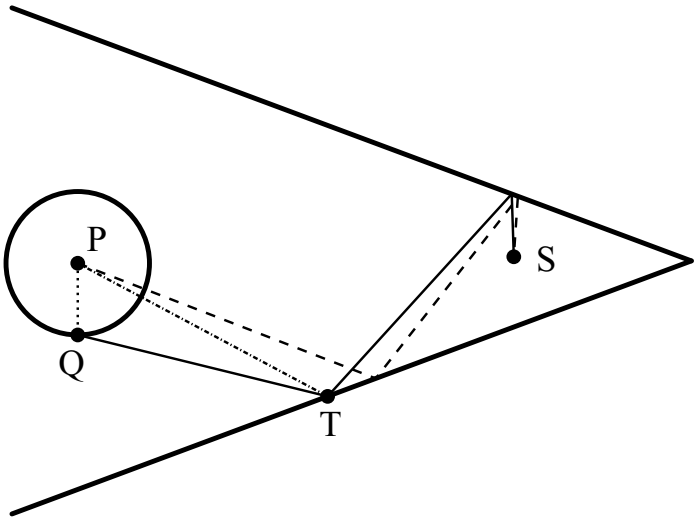


2.6. ábra. *A szórtan megjelenő impulzusok hatása az amplitúdómenetre*

Érdemes megvizsgálni az így létrejövő impulzusválasz által reprezentált rendszer amp-

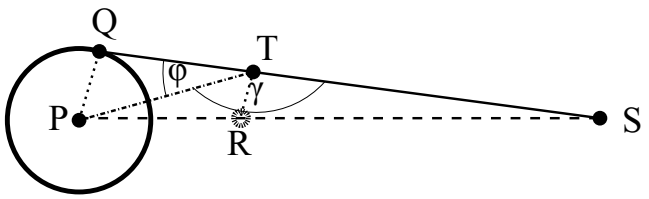
litúdómenetét, és összehasonlítani az ideális esettel. Ha csak egyetlen csúcs van jelen, akkor csak frekvenciafüggetlen amplitúdómódosítást és késleltetést végez a rendszer, viszont ha egymást követő minták között oszlik el ugyanez a csúcs, akkor a 2.6 ábrán látható aluláteresztő-lyukszűrő hatás jelentkezik. Ez jelentős minőségromláshoz vezet az audioalkalmazásokban, ezért valamilyen módon korrigálni szükséges. A szakirodalmat kutatva a problémáról semmilyen érdemi információt nem találtam.

Ha az $R \cdot \frac{f_s}{c_{air}}$ értéke egynél kisebb, és egészrész-képzéssel határozzuk meg a impulzusválaszbeli indexeket, akkor sem biztos, hogy egyetlen csúcs lesz, mert elképzelhető, hogy pont egy egész érték körül szóródik az értéke, így a legkisebb negatív irányú eltérés már az eggyel kisebb indexű pozícióra hivatkozik. A gömb sugara és a mintavételi frekvencia közötti megkötés létesítése tehát nem vezet eredményre.



2.7. ábra. *A sugarak visszaverődésének és elfogásának modellje*

A 2.7 ábrán egy modell látható arról, hogy hogyan terjednek a teremben a sugarak a forrás és a hallgató között, az egyik jelölt esetben pontosan eltalálva a hallgatót, a másik esetben pedig csak a körülötte lévő gömböt metszve. A két útvonalat kiegyenesítve is láthatjuk a 2.8 ábrán.



2.8. ábra. *A 2.7 ábrán látható utak kiegyenesítve*

A $\triangle TPS$-re felírható egy korrekciós összefüggés a koszinusztétel segítségével, amely

így megadja a hallgató és a forrás közötti pontos távolságot, vagyis az SP szakasz hosszát, függetlenül attól, hogy ez az útvonal a sugarak számára bejárható-e, vagy sem:

$$\begin{aligned} \cos(\gamma) &= \cos(\pi - \varphi) = -\cos(\varphi) \\ SP &= \sqrt{TP^2 + ST^2 + 2 \cdot ST \cdot TP \cdot \cos(\varphi)} \end{aligned} \tag{2.6}$$

Az utolsó falon történő visszaverődés helyét a gömb metszépontjával, valamint a gömb középpontjával összekötő szakaszok közötti szög ismeretében tehát elvégezhető a távolságkorrekció, elejét véve a káros szűrőhatásnak.

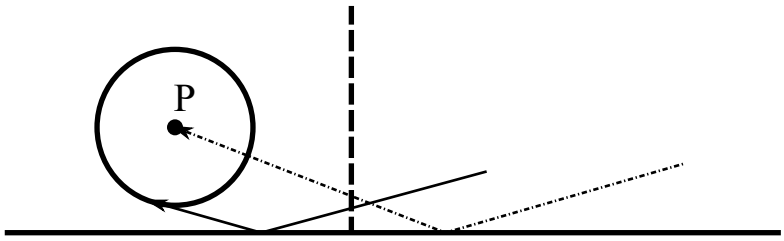
A későbbiekben szükségünk lesz a pontos irány meghatározására is, vagyis a 2.7 ábrán az $\overrightarrow{SP}$ irányú vektorra. Ezt megkaphatjuk úgy, hogy a $\overrightarrow{PT}$ vektorhoz hozzáadjuk a $\overrightarrow{TR}$-t, majd a kapott vektort megfordítjuk (vagyis összességében $\overrightarrow{TP}$-hez adtuk $\overrightarrow{RT}$-t). Az $\overrightarrow{RT}$ a $\triangle QPS \sim \triangle TRS$ hasonló háromszögek felismerésével:

$$\overrightarrow{RT} = \overrightarrow{PQ} \cdot \frac{TS}{QS}$$

Így tehát egy $\overrightarrow{SP}$-ral egyirányú vektor:

$$\mathbf{v}_{SP} = \overrightarrow{TP} + \overrightarrow{RT} = \overrightarrow{TP} + \overrightarrow{PQ} \cdot \frac{TS}{QS} \tag{2.7}$$

A fals extra pulzusokat azonban továbbra sem tudjuk elkülöníteni. A jelenséget megvizsgálva azt találjuk, hogy ennek akkor a legnagyobb a valószínűsége, ha a gömb közel kerül a határfelületekhez, ugyanis ilyenkor olyan sugár is eltalálhatja, amelyik közel párhuzamosan halad a fallal. A 2.9 ábrán látható egy olyan eset, amikor a visszavert sugár



2.9. ábra. *Fals extra pulzust generáló találat*

épphogy érinti a gömböt, és a feltételezett valódi sugár (pontvonallal) a határfelületen jóval távolabbi pontról verődik vissza. Ilyenkor könnyen meglehet, hogy ez nem is létező sugárutat jelent, amennyiben a pontozott vonalnál egy másik határfelület van. Az

ilyen esetek számának csökkentése érdekében ökölszabályként megfogalmazható, hogy a gömb felszínét nem szabad a sugaránál közelebb engedni a határfelületekhez.

2.2.2. A sugárkövetésnél használt metszésteszt-algoritmusok

A terem virtuális felületeinek kialakítása a számítógépes grafikai alkalmazásokban használt háromszöghálós eljárást lesz célszerű használni azon oknál fogva, hogy háromszögek esetében viszonylag egyszerű egy egyenessel vett metszéspont meghatározása. A szakirodalomban számos metszéspontszámításra szánt algoritmus fellelhető, amelyeken belül két nagy kategóriát szokás elkülöníteni: a háromszögek pontjainak koordinátáival közvetlenül számoló (Möller and Trumbore, 1997), valamint az abból valamilyen előfeldolgozott adathalmazt előállító, és azt felhasználó algoritmusokat (Snyder and Barr, 1987) (Badouel, 1990). Ezek közül jelen dolgozat keretein belül csak a Möller-Trombore algoritmussal foglalkozom, mert a GPU-knál a gyors memória szűkös erőforrás, és ez kevés tárhelyet használ, továbbá a kiindulópontja megegyezik az általam kidolgozott algoritmuséval.

Möller-Trombore algoritmus

A Möller-Trombore algoritmus kizárólag a háromszög pontjainak koordinátáit használja fel bemeneti adatként. Az **O** ponttal és **D** irányvektorral adott, a sugarat reprezentáló egyenes egyenlete:

$$\mathbf{R}(t) = \mathbf{O} + t \cdot \mathbf{D} \tag{2.8}$$

Az algoritmus baricentrikus koordináta-rendszert használ a megoldáshoz, tehát ha a metszéspont a háromszög területére esik, akkor annak koordinátái a $\mathbf{V_0}, \mathbf{V_1}, \mathbf{V_2}$ csúcspontok koordinátáinak súlyozott összegeként előáll oly módon, hogy a súlyok összege 1, és egyik sem negatív:

$$\mathbf{O} + t \cdot \mathbf{D} = (1 - u - v)\mathbf{V_0} + u\mathbf{V_1} + v\mathbf{V_2} \tag{2.9}$$

Ezt átrendezve adódik a következő lineáris egyenletrendszer:

$$\begin{bmatrix} -\mathbf{D} & \mathbf{V_1}-\mathbf{V_0} & \mathbf{V_2}-\mathbf{V_0} \end{bmatrix} \cdot \begin{bmatrix} t \\ u \\ v \end{bmatrix} = \mathbf{O}-\mathbf{V_0} \tag{2.10}$$

Ennek megoldása számításigényes, még optimalizált esetben is egy mátrix invertálásának komplexitásával összemérhető. Memóriaigénye viszont szerény, amely ismervén a GPU-k szűkös belső memóriáinak méreteit akár kedvezővé is teheti az algoritmust.

Saját algoritmus

Az itt ismertetésre kerülő eljárás az algoritmusok másik nagy csoportjába tartozik, vagyis előfeldolgozott adatokat használ a számítások során. Tulajdonképpen ugyanabból a megoldási elvből indul ki, mint a Müller-Trombore algoritmus. A sugár $\mathbf{r_0}$ kezdőpontjának és $\mathbf{t}$ irányvektorának, valamint a háromszög $\mathbf{p_1}$, $\mathbf{p_2}$, $\mathbf{p_3}$ csúcspontjainak birtokában felírható a következő összefüggés a metszéspontra:

$$\mathbf{P}\cdot\mathbf{a} = \begin{bmatrix} \mathbf{p_1} & \mathbf{p_2} & \mathbf{p_3} \end{bmatrix} \cdot \mathbf{a} = \mathbf{r_0} + k\cdot\mathbf{t} \tag{2.11}$$

Itt $\mathbf{a}$ a háromszög csúcspontjainak súlyait tartalmazó vektor, amelynek minden komponense pozitív kell, hogy legyen. A negatív súly azt jelenti, hogy a metszéspont a háromszög területén kívülre esik, vagyis nincs valódi metszéspont. A súlyok összegének pedig egynek kell lennie, amit felhasználva megoldható k-ra az egyenletrendszer:

$$k = \frac{1-\sum\{\mathbf{P^{-1}}\cdot\mathbf{r_0}\}}{\sum\{\mathbf{P^{-1}}\cdot\mathbf{t}\}} \tag{2.12}$$

A $\sum\{\}$ a vektor komponenseit összegző operátort jelent. Ha k negatív, akkor a sugár a háromszögtől távolodik, ilyen esetben tehát nincs metszéspont. A k ismeretében ellenőrizni kell még továbbá, hogy valóban nemnegatív-e $\mathbf{a}$ minden komponense, ellenkező esetben fals metszéspontot kaptunk. Az eljárás vitathatatlan előnye abból adódik, hogy a P^{-1} mátrix minden mátrixhoz előre kiszámítható, és a megoldás műveletigénye közelítőleg két mátrixszorzásával egyezik meg. Tárhelyigénye szintén kedvező, mivel az inverz mátrix 9 eleme pontosan ugyanannyi memóriát foglal, mint amennyit a három pont 3-3

koordinátája összesen. Ez egyértelműen azt jelenti, hogy GPU-n legalább olyan gyorsan, vagy gyorsabban fog futni, mint a Möller-Trombore algoritmus, azzal tehát nem is érdemes a továbbiakban foglalkozni.

Felmerül azonban egy hátrány is, ugyanis amennyiben a háromszög egyik csúcsa az origóban van, vagy a háromszög valamelyik koordinátasíkban fekszik, akkor a mátrix nem invertálható. Ezen felül lehet kerekedni oly módon, hogy a termet felépítő háromszöghálót eltoljuk például az $x, y, z > 0$ térnyolcadba.

Metszésteszt a hallgatót körülvevő gömbbel

Egy sugár és a hallgatót körülvevő gömb háromféle viszonyban lehetnek: a sugár elhaladhat mellette, ekkor nincs metszéspont, érinti, amikor pontosan egy metszéspont van, vagy átdöfheti két metszéspontot adva. Ha $\mathbf{o}$ a hallgató helye a térben, az R sugarú gömb egyenlete $|\mathbf{x}-\mathbf{o}|^2 = R^2$, a követett sugár egyenlete pedig $\mathbf{x}(k) = \mathbf{r}_0 + k\mathbf{t}$, ahol $\mathbf{t}$ egységvektor, akkor a sugár egyenletét a kör egyenletébe beírva rendezhető k-ra az összefüggés (Eberly, 2006):

$$k = -(\mathbf{t}\cdot(\mathbf{r}_0-\mathbf{o})) \pm \sqrt{(\mathbf{t}\cdot(\mathbf{r}_0-\mathbf{o}))^2 - (\mathbf{r}_0-\mathbf{o})^2 + R^2} \tag{2.13}$$

Két metszéspont esetén a közelebbivel kell számolnunk, hogy a korrekciós algoritmus megfelelően működjön.

2.2.3. Frekvenciafüggés figyelembevétele

A sugárkövetés alkalmas akár a termet felépítő anyagok frekvenciafüggő elnyelésének figyelembevételére is. Ez történhet például úgy, hogy amennyiben bizonyos frekvenciaközönként, például oktávonként ismerjük az alkalmazott anyagok elnyelését, úgy minden frekvenciasávnál nyilvántartjuk adott sugár esetében, hogy addig az mekkora csillapítást szenvedett. Amikor egy újabb háromszögről verődik vissza, a háromszög anyagát ismerve (k az anyag indexe), és annak az i-edik frekvenciasávhoz tartozó α_{ki} abszorpciós tényezőit lekérdezve, majd az addigi csillapításokat $(1-\alpha_{ki})$-val szorozva megkapjuk a visszave-

rődés utáni w_i súlytényezőket, amely M féle anyag esetén a következő alakot ölti:

$$w_i = \prod_{k=1}^{M} (1 - \alpha_{ki})^{h_i} \tag{2.14}$$

Az impulzusválasz összeállítása előtt szükségünk van a teljes frekvenciasáv felosztására, vagyis egy (lehetőleg logaritmikus) szűrőbank létrehozására. Ennek birtokában az impulzusválasz összeállítása úgy történik, hogy a szűrők $g_i[k]$ impulzusválaszait a frekvenciasávban elszenvedett összcsillapításnak megfelelően súlyozva a terem impulzusválaszába másoljuk a beérkezés idejének megfelelő indexpozíciótól kezdődően. Egzakt formában ez N darab sugár, L darab frekvenciasáv esetén, az m-edik sugárnak a hangterjedés véges sebessége miatt elszenvedett késleltetését mintaszámban kifejező d_m jelölést bevezetve a következőképp írható le:

$$h[k] = \sum_{m=1}^{N} \sum_{i=1}^{L} w_i \cdot g_i[k - d_m] \tag{2.15}$$

Csak megjegyzésképp megemlítenék egy másik megoldási lehetőséget is, miszerint frekvenciasávonként építjük fel a terem impulzusválaszát, és ezekben egy sugár beérkezésekor csak egy, a csillapításnak megfelelő nagyságú impulzust adunk hozzá a megfelelő indexpozíciónál. A sugárkövetés végeztével ezen impulzusválaszokat és a hozzájuk tartozó szűrőket pedig FFT konvolválva, majd a kimeneteket összegezve kapnánk meg a teljes impulzusválaszt. Ez a megoldás akkor előnyösebb az előző megoldásnál, ha sok nem nulla elem van az egyes frekvenciasávokban összeállított impulzusválaszokban, különben FFT konvolúció helyett ennél a megoldásnál is érdemesebb időtartománybeli konvolúciót alkalmazni.

3. fejezet

Az impulzusválasz ellenőrzése

3.1. Az impulzusválaszok skálázása

Ahhoz, hogy a két módszer által szolgáltatott impulzusválaszt össze tudjuk hasonlítani, azokat valamilyen módon egyeztetni szükséges. A sugárkövetéssel nyert impulzusválasz fizikai jelentéssel bíró mennyiséggé alakítható, hiszen ha azt normáljuk az összes indított sugár számával, akkor az a hallgató köré írt gömbre jutó teljesítményarány időfüggvényét fogja megadni. A tükörforrások módszerével előállított impulzusválaszból pedig úgy kaphatjuk meg ugyanezt az időfüggvényt, ha az impulzusokat nem csak szimplán r^{-2}-nal súlyozzuk, hanem a következő faktorral:

$$f_P = \frac{R^2\pi}{4\pi r^2} = \frac{R^2}{4r^2} \tag{3.1}$$

Ez a kifejezés megadja a forrás által kisugárzott egységteljesítménynek a hallgató köré írt R sugarú gömb hatásos felületére (vagyis egy R sugarú kör területére) eső teljesítményhányadát (feltéve, hogy $r \gg R$ fennáll).

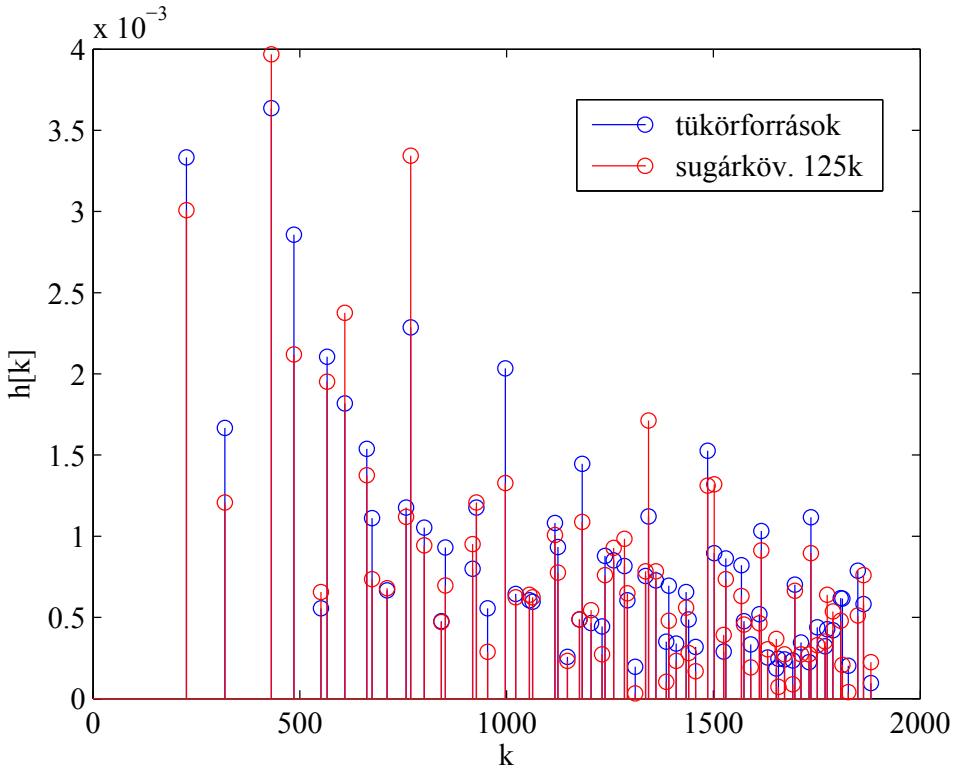
Lévén, hogy csak adott maximális rendű visszaverődésekkel számolunk, megkötést kell még tennünk az előállított impulzusválaszok vizsgálandó hosszát illetően, hiszen módosulhatnak, amennyiben magasabb rendű visszaverődéseket is számításba veszünk. Az összehasonlítható tartomány hosszát a tükörforrások módszerénél ismertetett eljárással kell meghatározni, majd a sugárkövetéssel nyert impulzusválaszt is ekkora méretűre kell csonkolni.

Rendkívül fontosnak tartom itt megemlíteni, hogy az impulzusválaszok tehát energia-

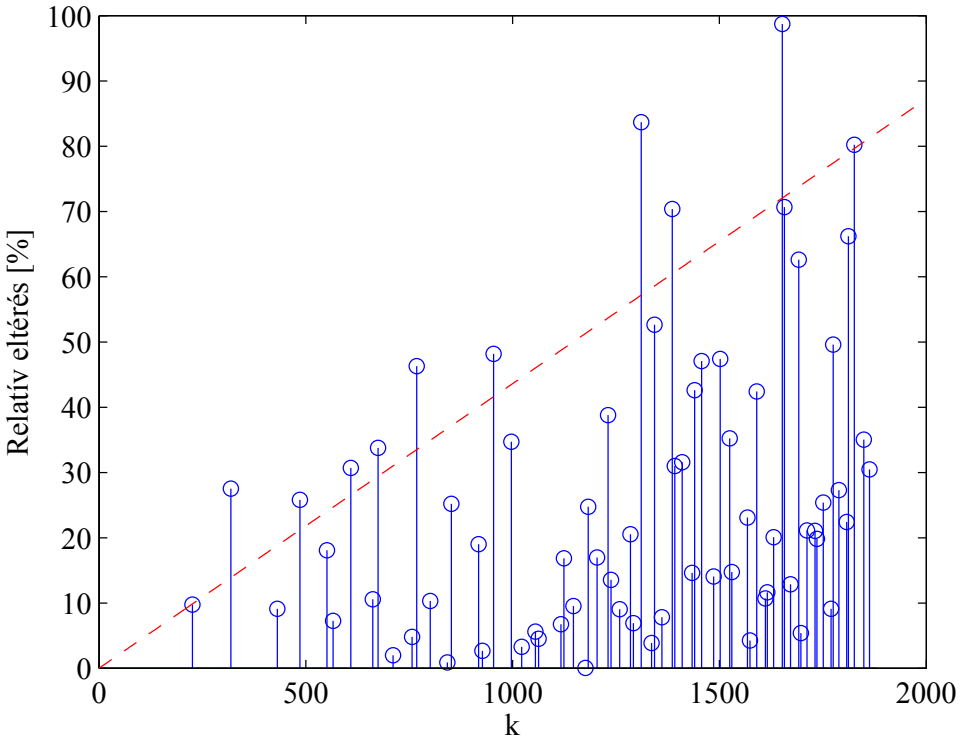
normában vannak kezelve, vagyis az egyes csúcsok a beérkező hanghullámok intenzitásával arányosak. Tehát mielőtt felhasználnánk ezeket bárhol, intenzitásról vissza kell térnünk nyomás jellegű mennyiségre, hiszen egy terem impulzusválaszának mérésekor a mikrofonok szintén a nyomásváltozással arányos jelet adnak. Lévén, hogy $I \sim p^2$, ezt mintánkénti gyökvonással tehetjük meg.

3.2. Az impulzusválaszok összevetése

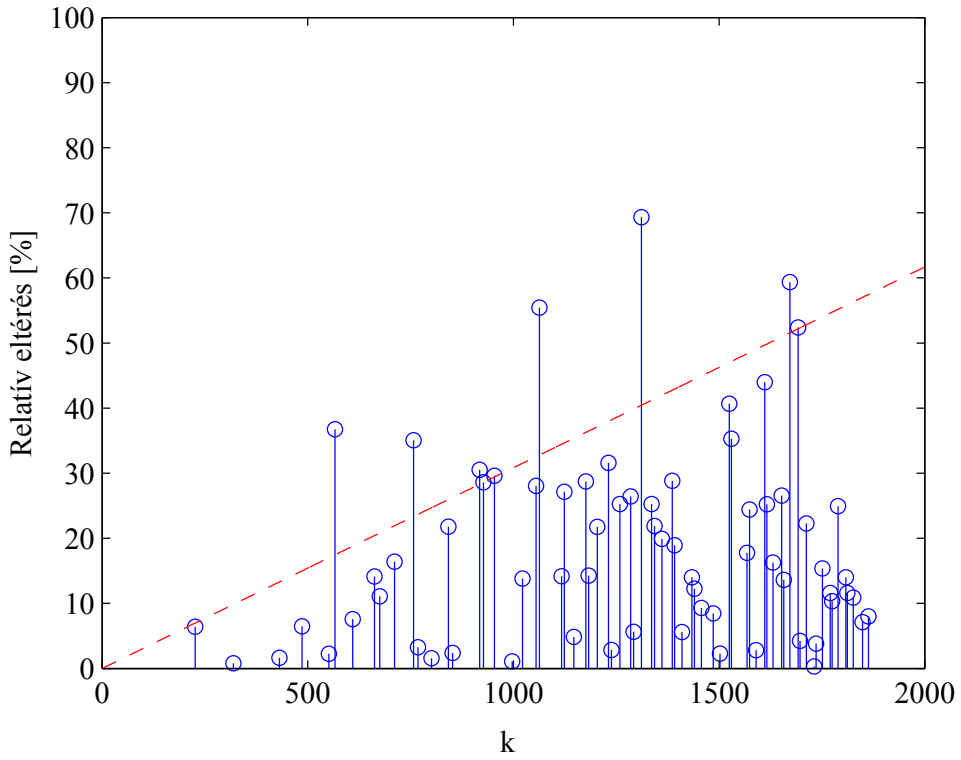
Az eljárásokat a leírt módosításokkal *MATLAB*-ban leprogramozva egy $3 \times 3 \times 3$ méteres, 0 abszorpciós tényezőjű határfelületekkel rendelkező szoba impulzusválaszának meghatározását választottam a módszerek összehasonlításának alapjául, $S(1,1,1.5)$ és $P(2,2,2.5)$ konfiguráció mellett, a hallgató gömbjének sugara $R = 0.2\,m$. Mindkét módszer legfeljebb nyolcadrendű visszaverődésekkel számolt, ugyanis a tükörforrások módszerének futásideje és memóriaigénye miatt ennél magasabb rendek kiszámítása nehézségekbe ütközött. A sugárkövetést az indított sugarak számának változtatásával többször is lefuttattam, rendre 125 ezer, 250 ezer és 500 ezer sugárral.



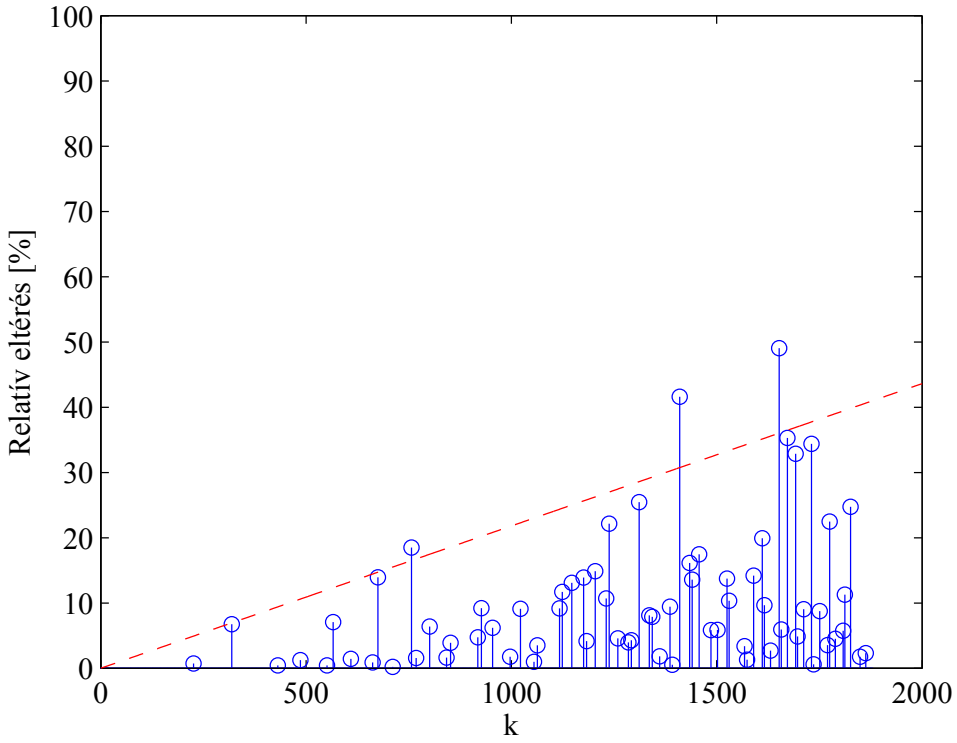
3.1. ábra. *125 ezer sugárral számított impulzusválasz és a referencia*



3.2. ábra. *Relatív eltérés 125 ezer sugár mellett*



3.3. ábra. *Relatív eltérés 250 ezer sugár mellett*



3.4. ábra. *Relatív eltérés 500 ezer sugár mellett*

Amint a 3.2, 3.3, 3.2 ábrákon is látszik, az indított sugarak számának növelésével a tükörforrások módszere által szolgáltatott referenciához képest egyre csökken a relatív eltérés. Ahhoz, hogy a későbbiekben korlátot tudjunk felállítani adott konfidenciaszint és maximális relatív eltérés mellett a sugarak minimális indítandó számára, érdemes megvizsgálni a jelenség elméleti hátterét is a valószínűségszámítás eszközeivel.

3.3. Minimális sugárszám adott maximális relatív eltéréshez

Tekintsünk a hallgatótól egy adott r távolságban lévő forrást, vagy virtuális forrást, amely minden irányban egyenletesen emittál sugarakat. Annak valószínűsége, hogy a hallgatót körülvevő R sugarú gömböt eltalálja egyetlen véletlenszerűen indított sugár, megegyezik azzal, hogy a hallgató körüli gömb mekkora térszögben látszik a teljes 4π szteradián térszöghöz képest. A gömb legnagyobb keresztmetszete egy R sugarú kör alapú kúpot képez a forrással, legyen ψ ennek a félnyílásszöge! Ezzel már kifejezhető a

térszögek aránya:

$$p = \frac{2\pi(1-\cos(\psi))}{4\pi} = \frac{1}{2}\cdot\left(1-\frac{r}{\sqrt{r^2+R^2}}\right) \approx \left.\frac{R^2}{4r^2}\right|_{r\gg R} \tag{3.2}$$

Annak valószínűsége, hogy n kilőtt sugár x találatot eredményez, egy p paraméterű binomiális eloszlással írható le:

$$\mathbf{P}(X=x) = \binom{n}{x} p^x (1-p)^{n-x}, \quad x = 0,1,2,...,n \tag{3.3}$$

A p paraméter értéke meglehetősen kicsi, a fent ismertetett szobában egy negyedrendű visszaverődés például 3×4, vagyis 12 méterre helyezhet egy virtuális forrást a hallgatótól, ebből a távolságból a találat valószínűsége $0,00694\,\%$.

Sajnos a fent említett binomiális eloszlás matematikailag nehezen kezelhető. Tegyük fel most azonban, hogy az np szorzat, vagyis a beérkező sugarak várható értéke legalább 10! Ekkor a binomiális eloszlás jól közelíthető normális eloszlással:

$$\mathcal{B}(n,p) \approx \mathcal{N}(np, np(1-p)), \quad np \geq 10 \tag{3.4}$$

Innen azonnal ki is olvasható a szórásnégyzet, és megadható például egy 2σ-s konfidenciaintervallum, amelybe így 95%-os valószínűséggel beleesik a ténylegesen beérkezett sugarak száma. Mivel p kicsi, ezért

$$p \approx p\cdot(1-p), \tag{3.5}$$

a konfidenciaintervallum tehát:

$$X|_{k=2} = np \pm 2\sqrt{np} = np \pm \frac{R\sqrt{n}}{r}, \tag{3.6}$$

a relatív eltérés pedig:

$$\frac{\Delta X}{X} = \frac{2}{\sqrt{np}} = \frac{4r}{R\sqrt{n}} \tag{3.7}$$

Ezzel tehát adott időtartamig garantálni tudjuk az impulzusválasz előírt minőségét. A 3.2, 3.3 és 3.4 ábrákon szaggatott piros vonal jelöli a 2σ-s határt. Szemmel láthatóan az e

fölé eső relatív eltérések száma 5 %-on belül van.

Amint azt az 1. fejezetben ismertettem, csillapítás nélküli esetben az impulzusválasz farokrészének burkolója konstans, szemben a korai reflexiók r^{-2}-os lecsengésével. Ez úgy lehetséges, hogy a virtuális források száma a távolság növekedésével négyzetesen növekszik, amíg az egyes források intenzitása természetesen továbbra is r^{-2}-nal csökken. Egy csúcs az impulzusválaszban ekkor $\xi \cdot r^2$ valószínűségi változó összege (ξ egy általunk nem ismert konstans), az így létrejövő valószínűségi változó szórása pedig a centrális határeloszlás-tétel értelmében $r\sqrt{\xi}$-ad részére csökken. Amint láttuk, a p találat valószínűsége szintén r^{-2}-nal arányos, vagyis a sugarak számának relatív eltérése az impulzusválasz farokrészében:

$$\frac{\Delta X}{X} = \frac{4}{R\sqrt{\xi}\sqrt{n}} \tag{3.8}$$

Adott sugárszám mellett tehát a relatív eltérés állandó a zengésnél. A korai reflexióknál pedig 3dB-es fluktuáció még nem, vagy alig hallható, tehát ha úgy állítjuk be az indított sugarak számát, hogy a zengés kezdetekor legfeljebb ekkora ingadozás legyen megengedve, akkor az impulzusválasz minősége végig kielégítő marad.

4. fejezet

Auralizáció megvalósítása HRTF-fel

A 2.2.3 pontban bemutattam, hogyan lehet figyelembe venni a határfelületek frekvenciafüggő elnyelését, ezáltal realisztikusabbá tenni modellünket. Ahhoz azonban, hogy térhatást kelthessünk, auralizációt kell megvalósítanunk, vagyis mindkét fülre az adott irányban érvényes átviteli függvény szerint egy lineáris transzformációt kell végeznünk a beérkező hanghullámokon. Ebben a fejezetben bemutatom, hogyan lehetséges auralizációt úgy megvalósítani, hogy közben frekvenciafüggő anyagok is jelen vannak a virtuális teremben, nem nélkülözve a konkrét realizációt.

4.1. A HRIR adatbázis felbontása szűrőbank segítségével

Először is szükségünk lesz a szimulálandó anyagok különböző frekvenciákon mért abszorpciós együtthatóira, valamint egy HRIR adatbázisra. Ha ezek rendelkezésünkre állnak, a HRIR adatbázisban szereplő összes impulzusválaszt át kell váltanunk energianormába (mintánként négyzetre kell emelni), hogy dolgozhassunk velük, majd fel kell bontanunk frekvenciasávokra. Ezt egy szűrőbank segítségével tehetjük meg, ahol jelen esetben az abszorpciós tényező mérőfrekvenciái lesznek a frekvenciasávok középfrekvenciái. Ezen áteresztve tehát a HRIR adatbázis impulzusválaszait minden irány minden frekvenciasávjához kapunk két elemi impulzusválaszt (bal és jobb csatorna). Ezeket egy adott irányú sugár beérkezésekor az adott frekvenciasávban elszenvedett összcsillapításnak megfelelő súllyal megszorozva, a sugár által megtett útnak megfelelő késleltetéssel a teljes, csatornánkénti impulzusválaszhoz adva, majd ezt minden sugár minden frekvenciasávjára elvégezve kapjuk a teljes impulzusválaszt. Vagyis az impulzusválasz kifejezése,

ha az m-edik sugár a φ_m, θ_m által meghatározott irányból érkezett, és $g_{[\theta_m,\varphi_m,i]}$ jelöli az ezen irányhoz tartozó i-edik frekvencia-részsávbeli elemi impulzusválaszt:

$$h[k] = \sum_{m=1}^{N} \sum_{i=1}^{L} w_i \cdot g_{[\theta_m,\varphi_m,i]}[k - d_m] \tag{4.1}$$

4.2. Frekvenciasávok száma, töréspontí frekvenciák

A SAE Institute (2013) weboldalán található táblázatban számos anyag megtalálható különböző összeállítások mellett, ezért a feladat megoldása során az itt szereplő együtthatókkal dolgoztam. Az abszorpciós együtthatót 6 diszkrét frekvencián, 125 Hz-től kezdve oktávonként adták meg, ez látható a 4.1 táblázatban néhány gyakran előforduló anyagra.

4.1. táblázat. *Néhány anyag abszorpciós együtthatója a frekvencia függvényében*

Anyag / konfiguráció	125 Hz	250 Hz	500 Hz	1 kHz	2 kHz	4 kHz
Fa padló	0.15	0.11	0.10	0.07	0.06	0.07
Festett betonfal (sima)	0.10	0.05	0.06	0.07	0.09	0.08
Szőnyeg betonon	0.02	0.06	0.14	0.37	0.60	0.65

A szűrőbankot tehát úgy kell megtervezni, hogy az egyes frekvenciasávok között legyen átfedés, a töréspontí frekvenciákat pedig célszerű logaritmikus skálán félúton elhelyezni a mérési pontok között. Ez az az alsó frekvencia $\sqrt{2}$-szörösét jelenti, például a 125 és 250 Hz-es mérési pontokhoz tartozó szűrőknél ez a töréspontí frekvencia $125\sqrt{2} = 176\,Hz$ lesz. A logaritmikus skálát az a tény indokolja, hogy hallásunk is logaritmikus, vagyis egy nagy frekvenciájú hang adott nagyságú megváltozása kevésbé hallható, mint ugyanekkora megváltozás egy kisebb frekvencián (Stevens et al., 1937). Emiatt a szűrők letörési meredekségét is célszerű kisebbnek választani nagyobb frekvenciákon, vagy eleve olyan szűrőstruktúrát alkalmazni, amelynek letörése logaritmikus skálán lineáris (pl. Butterworth-szűrő).

4.3. Szűrőválasztásnál felmerülő fogalmak

A szűrők típusának kiválasztása előtt előbb ismertetek két jelfeldolgozással kapcsolatos fogalmat, a csoportkésleltetést és az előhullámzást (pre-ringing).

4.3.1. Csoportkésleltetés

Egy lineáris, időinvariáns rendszer csoportkésleltetése (angolul: group delay) alatt a fázismenetének ($\phi(\omega)$) frekvencia szerinti deriváltját értjük:

$$\tau_g(\omega) = -\frac{d\phi(\omega)}{d\omega} \tag{4.2}$$

Ennek haszna az, hogy amennyiben a bemeneti jelünk kvázi szinuszos, vagyis

$$x(t) = a(t) \cdot \cos(\omega_0 t + \theta) \tag{4.3}$$

úgy a kimenet jól közelíthető az alábbi kifejezéssel:

$$y(t) = |H(j\omega_0)|\, a(t - \tau_g) \cdot \cos(\omega_0 t + \theta - \tau_\phi) \tag{4.4}$$

feltéve, hogy:

$$\left| \frac{d\log(a(t))}{dt} \right| \ll \omega_0 \tag{4.5}$$

vagyis az $a(t)$ jel kellően lassan változik. A csoportkésleltetést tehát felfoghatjuk úgy, mint a bemeneti jel burkolójának késleltetését. A τ_ϕ a fáziskésleltetés, amely a következőképp számítható:

$$\tau_\phi(\omega) = -\frac{\phi(\omega)}{\omega} \tag{4.6}$$

Audiojelek feldolgozására szánt szűrők tervezésekor figyelnünk kell azok csoportkésleltetésének frekvenciafüggésére, ugyanis ha az túl nagy varianciát mutat, akkor az észrevehető minőségromláshoz vezethet (Blauert and Laws, 1978).

4.3.2. Pre-ringing

Amennyiben lineáris fázismenetű FIR szűrőt tervezünk, úgy a kapott impulzusválasz szimmetrikus lesz a $k = 0$ időtengelyre, az akauzalitást pedig úgy lehet feloldani, hogy a szűrőt időben eltoljuk annyi mintával, amennyi ahhoz szükséges, hogy az összes minta a $k \geq 0$ félsíkra essen. Ez a mintaszám a szimmetria miatt a szűrő fokszámának a fele lesz.

A főcsúcsot tehát egy előhullámzás (pre-ringing) fogja megelőzni, ez pedig a kimeneti jelben is intenzíven jelen lehet, ha a szűrőt egy meredek felfutású hangfelvétellel (például lábdob, cintányér hangja) konvolváljuk.

Ilyen szűrő tervezése esetén ezért érdemes használat előtt megvizsgálni, hogy okoz-e ez az előhullámzás valamilyen hallható, nem tolerálható elváltozást. Ha igen, akkor érdemes a szűrő specifikációján lazítani, illetve más szűrőstruktúra alkalmazását megfontolni.

4.4. A szűrők típusa, tulajdonságai

A szűrőbank tervezésekor felmerült IIR szűrő használatának lehetősége is, azonban később ezt elvetettem a csoportkésleltetés frekvenciafüggésének hangminőségre gyakorolt negatív hatásától tartva. A döntés végül a lineáris fázismenetű, szimmetrikus FIR szűrőkre esett, ezek csoportkésleltetése a szűrő rendjének a felével megegyező számú mintányi, függetlenül a frekvenciától. A szűrőket *MATLAB*-ban, a `fir1()` függvény segítségével generáltam, amely egy ideális szűrő végtelen impulzusválaszának ablakozásával állítja elő a szűrőegyütthatókat. Ablakként Kaiser-ablakot használtam, amelynek az α paraméterét növelve az egyre magasabb törésponti frekvenciájú szűrők letörési meredekségét csökkenteni tudtam, így egy kvázi logaritmikus szűrőbankhoz jutottam.

4.2. táblázat. *A szűrőbank törésponti frekvenciái és a használt ablakparaméterek*

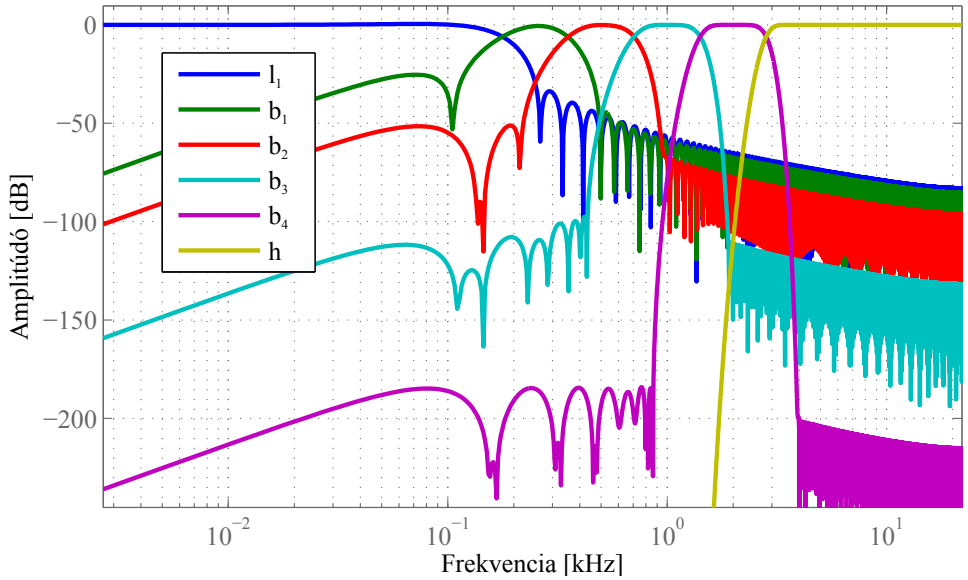
f_c	Kaiser α
177 Hz	2.5
354 Hz	5
707 Hz	10
1,414 kHz	20
2,828 kHz	50

Ahhoz, hogy a szűrőbank átvitele egységnyi legyen, rekurzívan készítettem el az egyes frekvenciasávokra a szűrőket. Ez azt takarja, hogy előbb minden f_c törésponti frekvenciához előbb készítettem egy aluláteresztő szűrőt (nevezzük ezeket l_i-nek, $i = 1,...,5$), ezekből pedig a b_j ($j = 1,...,4$) sáváteresztő szűrőket az egymást utáni aluláteresztő szűrők különbségeként állítottam elő:

$$b_j = l_{j+1} - l_j, \quad j = 1,...,4 \tag{4.7}$$

Az utolsó, felülátereszt ő szűrőt (h) pedig egy egységnyi átvitelű, a többi szűrő csoportkésleltetésével megegyező késleltetésű szűrőből az l_5-öt kivonva kaptam meg. A szűrőbankot tehát az $l_1, b_1, b_2, b_3, b_4, h$ szűrők alkotják.

A szűrők rendjét 512-nek választva már megfelelő zárótartománybeli elnyomást és tolerálható áteresztőtartománybeli hullámzást lehetett elérni, a csoportkésleltetés pedig így 256 mintányira adódott. A szűrők amplitúdómenete a 4.1 ábrán látható.

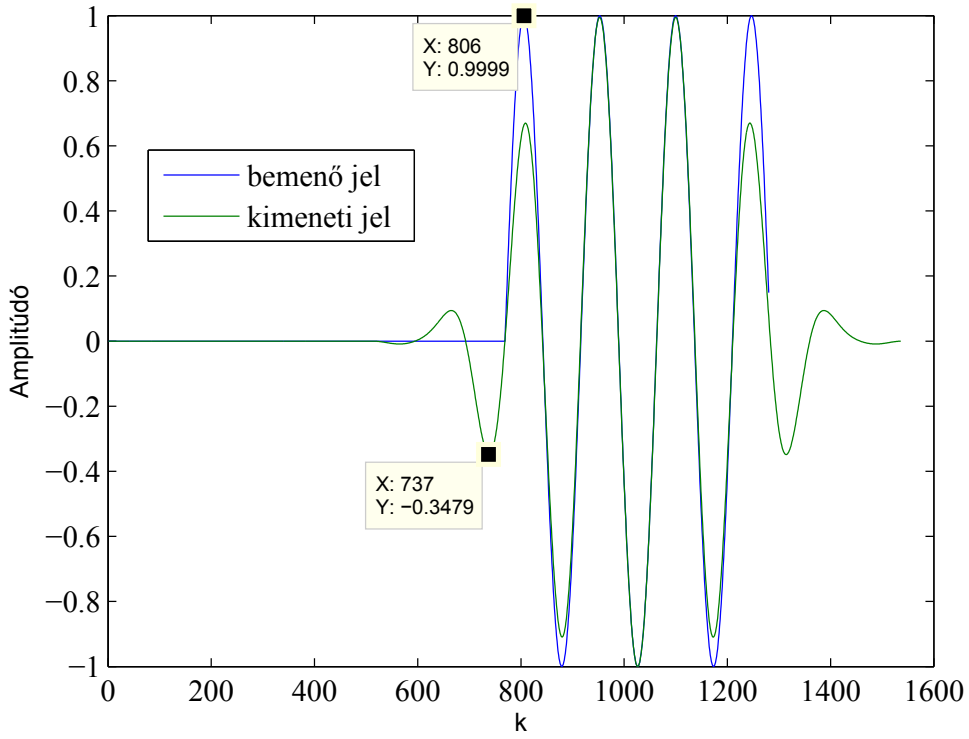


4.1. ábra. *A szűrőbankot alkotó szűrők amplitúdómenete*

4.5. A választott szűrők feladatra való alkalmasságának vizsgálata

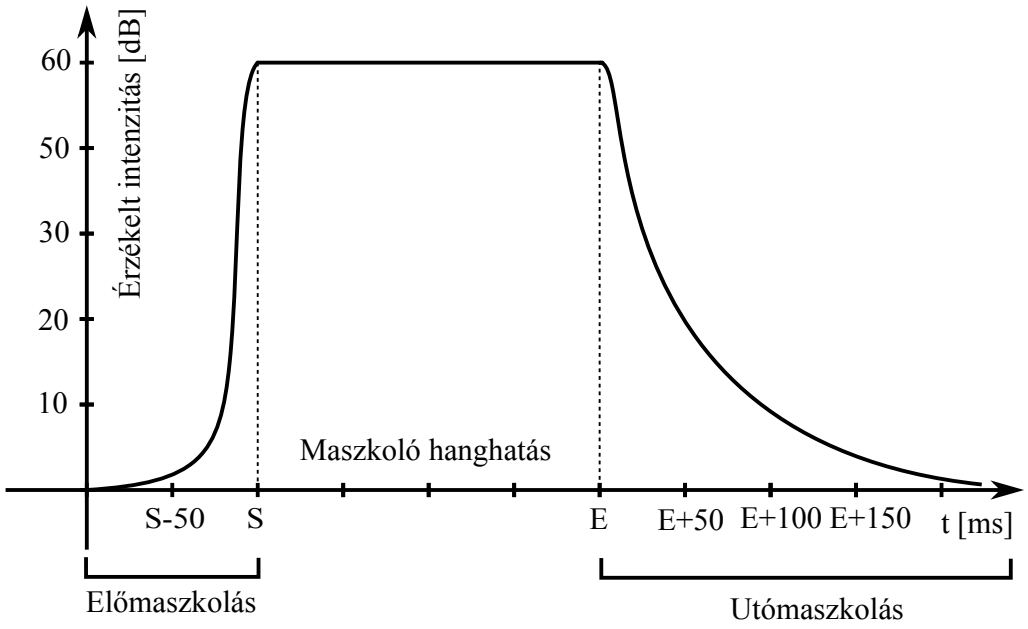
Lévén, hogy szimmetrikus FIR szűrőkből épül fel a szűrőbank, meg kell vizsgálnunk, hogy a kialakuló pre-ringing okoz-e valamilyen hallható nemkívánatos mellékhatást. 4.2 ábrán látható jelet figyelhetjük meg a kimenetén. A jelenség akkor lesz a legszembetűnőbb, ha a szűrők bemenetére egyetlen, az áteresztőtartományukba eső frekvenciájú, négyszögablakkal kivágott, szinuszos jelet vezetünk. A b_1 jelű szűrőnél volt legkifejezettebb az előhullámzás, 300 Hz-es szinuszjelet adva hozzávetőlegesen csak 9dB-lel maradt el az intenzitása a hasznos kimeneti jelhez képest. Ez azonban mégsem fog hallható elváltozást okozni a kimeneti jelben.

Az emberi hallás egyik tulajdonsága ugyanis, hogy egy adott hanghatás beérkezése előtt, illetve után érkező hangokat figyelmen kívül hagyja bizonyos időtartamig, amennyi-



4.2. ábra. *Pre-ringing szimmetrikus FIR szűrőnél*

ben azok intenzitása nem ér el egy bizonyos küszöbszintet. Ezt a jelenséget nevezzük **időbeli maszkolás**nak, avagy **temporal masking**nak (Fastl and Zwicker, 2007). A maszko-



4.3. ábra. *Időbeli maszkolás*

lás időtartamát a maszkoló hangnak a maszkolt hanghoz viszonyított intenzitása határozza

meg, az 4.3 ábráról leolvashatók az egyes időtartamokhoz tartozó relatív küszöbértékek. Az 4.2 ábrára visszatekintve látható, hogy alig több mint 1 *ms*-ig tart csak a pre-ringing esetünkben, vagyis a maszkolás működni fog. A szűrőbank tehát ebben a formájában alkalmas a feladatra.

4.6. HRIR adatbázis választása, szűrés

A HRIR adatbázis, amellyel a feladat megoldása során dolgoztam, az IRCAM (2013) oldalán megtalálható adatbázisok egyike. Ebben $44.1kHz$-en mintavételezett, 512 minta hosszú impulzusválaszok találhatók, az azimut szög $0-345°$-ig változik $15°$-os lépésekben, mialatt a másik szögkoordináta a horizonthoz képest $-45-45°$-ig változik. $60°$-nál az azimut szög már csak $30°$-onként van léptetve, $75°$-nál $60°$-onként, $90°$-nál pedig csak egyetlen mérési pont van. Ez összesen 187 irányt jelent, de érdemes interpolálni ezeket úgy, hogy mindenhol egységesen $15°$-onként legyen a lépésköz, hiszen így a későbbiekben könnyebb lesz indexelni a két szögkoordináta alapján az impulzusválaszokat tartalmazó adatstruktúrát. A sugár beérkezésének tényleges irányához a legközelebbi mérési irányt választva a kvantálási hiba legfeljebb $7,5°$ lesz, ami nem okoz érzékelhető eltérést (Simon Skluzacek, 2012).

A frekvenciasávokra történő felbontás úgy történik adott irányban, hogy a hozzá tartozó impulzusválaszt konvolváljuk a szűrőbank egyes szűrőivel. Így tehát 6 darab 1024 minta hosszú elemi impulzusválaszhoz jutunk irányonként, illetve hangcsatornánként.

5. fejezet

GP-GPU programozás

5.1. Lehetséges programozási nyelvek

A GPU-k általános célú programozásához (GPGPU, General-purpose computing on graphics processing units) választanunk kell egy programozási nyelvet és API-t (Application Programming Interface). Jelenleg erre két alternatíva kínálkozik: az egyik az nVidia cég által fejlesztett, és kizárólag a saját termékeiken működő CUDA nyelv, a másik pedig az OpenCL, amely a Khronos Group nevű konzorcium által karbantartott, nyílt szabvány, tehát hardverfüggetlen kód írható segítségével. Tudni kell azonban, hogy az nVidia nem támogatja az OpenCL-t az 1.2-es verziójától kezdve, de korábbi eszközillesztői sem optimalizáltak. A grafikus kártyák piacát pedig két nagy gyártó, nevezetesen az nVidia és az AMD uralja, vagyis a programozási nyelv választásakor tulajdonképp célhardvert is választunk.

CUDA nyelven, mivel a kód nem hardverfüggetlen, általában egyszerűbb ugyanazon feladatot megvalósítani, mint OpenCL-ben: egyszerűbb felállítani a futtatási kontextust (kevesebb API-függvényhívás) és rendelkezésre áll számos elemi utasítás, amelyet a GPU hatékonyan tud végrehajtani (pl. reciprokgyökfüggvény). A CUDA mellett szól továbbá, hogy az OpenCL-nél hamarabb vált elérhetővé, ezért használata jobban elterjedt, így végül a választás erre a nyelvre esett.

5.2. Az architektúra bemutatása

A GPU-ra történő fejlesztés teljesen más paradigmát kíván meg a szekvenciális programozásnál megszokotthoz képest, amely a célhardver eltérő felépítéséből adódik. Hatékony algoritmusok fejlesztéséhez, és programkódok írásához tehát elengedhetetlen a GPU architektúrájának bizonyos szintű ismerete.

A GPGPU programozás megjelenése óta az nVidia GPU-in többször is architekturális változásokat eszközölt. Ez egyrészt új CUDA függvények megjelenéséhez vezetett, melyek használatánál a kompatibilitás régebbi architektúrákkal nem biztosított, másfelől pedig a GPU korlátai (futtatható szálak, on-chip memóriák mérete) is módosultak, ezekhez pedig lehetséges, hogy más optimális kód tartozik. Érdemes tehát megállapodni egy minimális GPU-architektúra-verzió mellett a minél hatékonyabb fejlesztés érdekében. Talán a legfontosabb mérföldkövet a Fermi architekúránál megjelenő cache-elt globálismemória-hozzáférés jelentette, amely a szoftveres oldalon használt verziószám szerint Compute Capability 2.0-t jelent (a különbségeket részletesen is tárgyalja a programozási referencia: (nVidia Corporation, 2013)), ezért az ennél régebbi GPU-kon való (hatékony) futtathatóság a fejlesztés során nem volt szempont.

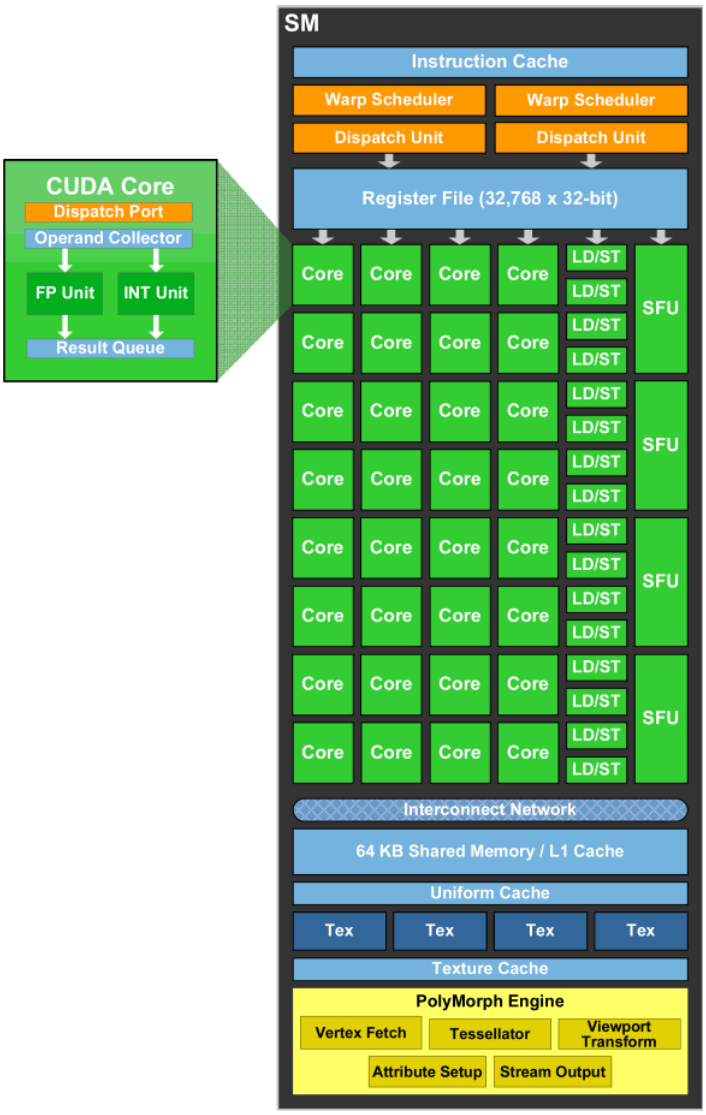
5.2.1. A hardver felépítésének rövid ismertetése

Az 5.1 ábrán a Fermi architektúrán alapuló GPU-k blokkvázlata látható. A grafikus kártyán lévő külső memóriához a szélessávú hozzáférést a 6 darab 64 bites memóriavezérlő biztosítja. Ezen hozzáférések cache-eltek (L2 cache az ábrán), tehát nem szükséges a szálaknak sorfolytonosan olvasni a globális memóriát, illetve a véletlenszerű hozzáférési minták sem vezetnek drasztikus teljesítménycsökkenéshez. A GF100-as magon 16 darab Streaming Multiproccessort (SM) helyeztek el, de a videokártya típusától függ, hogy ezek közül éppen mennyi van engedélyezve. Egy ilyen SM kinagyított képe látható az 5.2 ábrán.

A 32k darab 32 bites regisztert tartalmazó regisztertömb dinamikusan, és automatikusan (a fordító által) kerül elosztásra a szálak között, de minden szálhoz ugyanannyi regiszter rendelődik. A "Core" felirattal ellátott blokkok egyszeres pontosságú lebegőpontos, valamint 32 bites egész számokon tudnak műveleteket végrehajtani, de egy utasításciklusban mindegyiknél csak ugyanaz lehet a művelet. Az SFU egységek (Special Function

5.1. ábra. *Fermi architektúra (nVidia Corporation, 2013a)*

Unit) különböző függvények (pl. szinusz, koszinusz) gyors, hardveres kiértékelését teszik lehetővé, illetve a LD/ST (Load/Store) egységek a memóriaműveletekhez szükségesek. A műveletvégző egységek alatt látható egy közös shared memory/L1 cache blokk, amelynek felosztását szoftverből, API függvények segítségével lehet konfigurálni 48/16, 32/32, 16/48 kB arányokra. A shared memory sebességét tekintve egy szabadon felhasználható L1 cache, amelynek tartalmát a programozó határozhatja meg. Fontos megemlíteni, hogy ez bankos szervezésű, 32 bank van, és a bankok 32 bites szavanként váltják egymást. A többi blokk jelen dolgozat szempontjából irreleváns, főképp grafikus alkalmazások során használatosak, azonban a warp ütemezőkről (warp scheduler) a következő pontban még szó esik.



5.2. ábra. *Egy SM blokkvázlata (nVidia Corporation, 2013a)*

5.2.2. Az architektúra osztályzása, műveletvégzés

Az nVidia az architektúra típusát SIMT-nek (Single Instruction Multiple Thread) nevezte el. Ennek tulajdonságai részben a SIMD-re (Singe Instruction Multiple Data), részben pedig a hagyományos értelemben vett többszálú programozásra (SMT - Simultaneous Multithreading) hasonlítanak, így e kettő közé helyezhető el. Ha a fizikai utasításvégrehajtást tekintjük, mindenképp a SIMD jelleg érvényesül, vagyis ugyanazon utasítás kerül végrehajtásra a GPU-ban egyszerre több adaton, vagy adatpáron. Ehhez azonban nincs szükség vektorregiszterek használatára, ugyanis programozói szemszögből ez úgy tükröződik vissza, hogy olyan programszálakat (thread) futtatunk, amelyek ugyanazt az utasítássorozatot hajtják végre. Lehetőség van elágazásokra (`if-else`), azonban egyszerre

csak azok a szálak tudnak futni, amelyek ugyanazon műveletet hajtják végre. Ennek az a következménye, hogy elágazásnál a szálak egyik része az egyik ágat végrehajtja, miközben a szálak másik fele várakozik, majd a másik ágra fordítva. Tehát ilyen esetben mindkét ág végrehajtási idejét ki kell várni, ez pedig lassítja a futást.

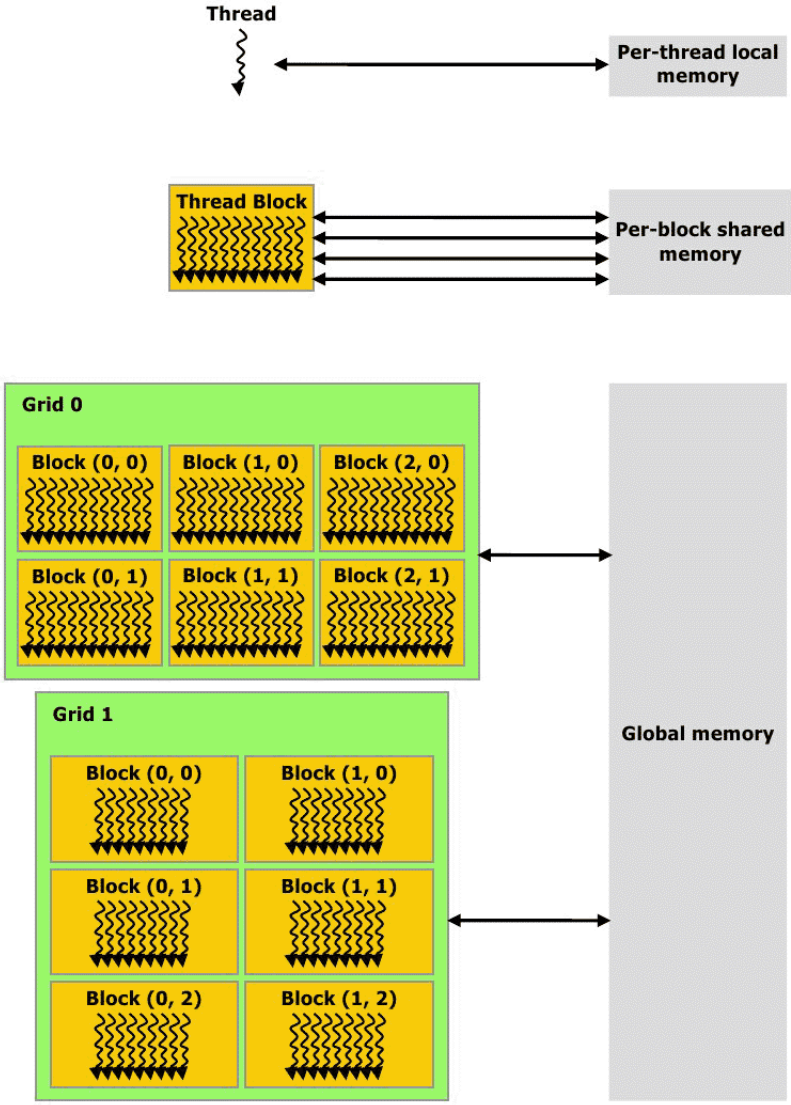
A tényleges végrehajtás 32-es szálkötegekben, ún. warpokban történik, amelyen belül minden szinkron zajlik. Általában egy SM-en célszerű több warpnyi szálat elindítani, mint ahány fizikailag képes egyidejűleg futni rajta. Ennek célja az, hogy a lassú külső memóriák késleltetését számításokkal fedjük el: amíg egy warp külső memóriaolvasás miatt várakozni kényszerül, addig egy másik warp kerül ütemezésre. Ez pedig úgy lehetséges, hogy minden szálhoz külön regiszterterület van rendelve a regiszterfájlból, amelyhez kizárólagos hozzáférése van, tehát ily módon lehetséges a regiszterek mentését nélkülöző, gyors kontextusváltás.

5.2.3. Memóriaterületek

A memóriamodell heterogén, vagyis dedikált grafikus kártya esetén fizikailag, integrált esetben pedig csak logikailag ugyan, de a CPU és a GPU melletti memória elkülönül, ezek között az átjárás API függvényekkel lehetséges. Azokat az adatokat tehát, amelyeket fel kívánunk használni a számítások során, a GPU mellett lévő memóriába kell másolni, majd a kimenet is innen érhető el.

A grafikus kártyán belüli memóriahierarchia az 5.3 ábrán látható. A megoldandó feladatot első lépésben logikailag független egységekre, úgynevezett blokkokra kell osztani. Azért fontos, hogy ezek logikailag függetlenek legyenek, mert a tényleges futás során is blokkokba vannak csoportosítva a szálak, a blokkok között pedig kizárólag a globális memórián (GPU melletti külső memórián) keresztül valósulhat meg bárminemű kommunikáció, amely rendkívül lassú a belső tárolóelemekhez képest. A blokkok teljes halmaza a grid. Fizikailag garantált, hogy egy blokk egy SM-en fog futni, azonban egy SM-en akár több blokk is futhat időmultiplexelve, ha az erőforrásigény azt megengedi.

Az egy blokkban lévő szálak látnak egy közös memóriaterületet, ez a shared memory. Ezen keresztül lehetőség van a szálak közötti kommunikációra, ekkor viszont szükségessé válhat szinkronizációs pontok elhelyezése a kernelen belül (`__syncthreads()`), amely szintén lassítja a futást, így célszerű ezek számát is minimálisra szorítani. A szálak pedig



5.3. ábra. *Memóriahierarchia (nVidia Corporation, 2013)*

külön regiszterterülettel rendelkeznek, amelyekhez kizárólagos a hozzáférésük.

5.3. A CUDA programozási nyelv

5.3.1. Nyelvi alapok

A CUDA heterogén programozási nyelv, ami azt takarja, hogy a megírt programkód két részre tagolható: a CPU-n futó host, valamint a GPU-n futó device kódra. Mindkettő közös tulajdonsága, hogy szabványos C/C++ nyelven írható. A host kódban van lehetőségünk az API függvényeinek használatára, amellyel többek közt konfigurálhatjuk a grafikus kártyát, és manipulálhatjuk a videomemória tartalmát. A GPU-n hatékonyan

nem implementálható algoritmusokat szintén host kódként célszerű megvalósítani. Device kód kétféle függvénytípus lehet: az egyik a kernelfüggvény, amelynek deklarációjában a `__global__` kulcsszó szerepel a (kötelezően void) visszatérési értéke előtt, a másik pedig a `__device__` típusmódosítóval ellátott device függvény. A kernelfüggvény az a függvénytörzs, amelyet minden egyes GPU-n futó programszál végrehajt. Ezt a függvényt lehet meghívni a host kódból az alábbi szintaktika segítségével:

```
myKernel<<<blockDim, gridDim>>>(a, b, c);
```

A blockDim és a gridDim változók int3 struktúrák (3 darab int taggal rendelkeznek: x, y, z), amelyek a programozót a futtatott szálak virtuális felosztásában segítik. Egy blokkon belül a blockDim változóban szereplő `x*y*z` darab szál fog futni, és ez a blokk a gridDim változóban meghatározott módon `x, y, z` irányban több példányban is létrejöhet. Futáskor minden szál hozzáfér egy threadIdx, blockIdx és egy blockDim nevű, int3 típusú változóh, amelyek segítségével a szálak egyértelműen azonosíthatók. Így határozható meg például az, hogy adott szál az adathalmaz mely elemét dolgozza fel. A következő ábrán ez a felosztás nyomon követhető `blockDim(4, 3, 0)` és `gridDim(3, 2, 0)` változóértékek mellett:

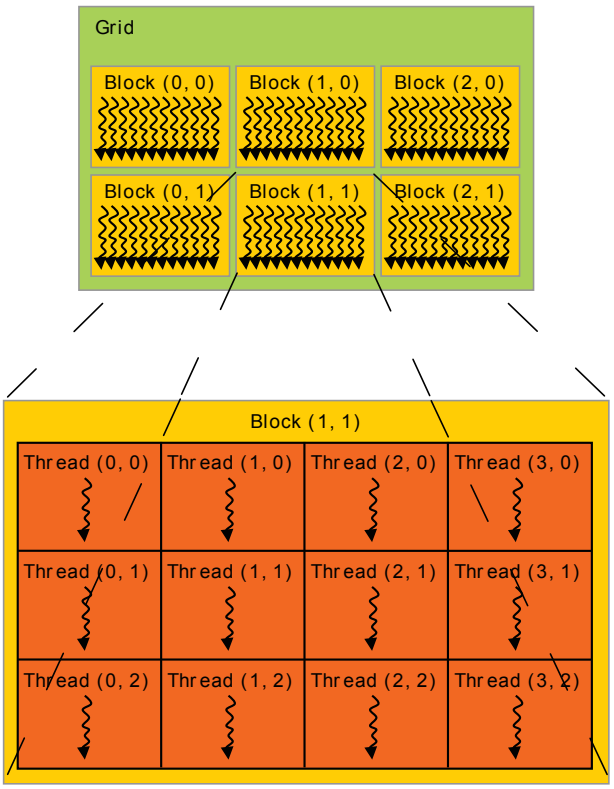
A device függvények kizárólag a szálak által a kernelfüggvényből, vagy másik device függvényből hívhatók meg, segítségükkel olvashatóbbá tehetők a kernelfüggvények.

5.3.2. Általános irányelvek programozás során

Érdemes a kernelfüggvény írása során az architektúrából adódó programozási alapelveket betartani a hatékonyság érdekében. Lehetőség szerint kerülni kell az elágazásokat (`if-else`, `switch-case`), mert ilyenkor a futásidő közel annyi lesz, mintha az összes ág lefutott volna.

A shared memoryhoz történő hozzáféréseknél ügyelni kell arra, hogy a warpon belüli összes szál különböző memóriabankhoz tartozó memóriacímre hivatkozzon, mert különben csak több olvasási ciklusban szolgálható ki a kérés. Egyetlen kivétel az, amikor több szál ugyanazt a memóriacímet olvassa, ilyenkor egyetlen olvasási ciklusban az összes érintett szálhoz eljut az adat (32-bit broadcast access).

A blokkok erőforrásigényének (pl. sharedmemory-felhasználás) kialakításánál figyelembe kell venni, hogy legalább 8 warpnak el kell férnie egy SM-en ahhoz, hogy a memó-



5.4. ábra. *A szálak blokkokba, a blokkok pedig gridbe vannak rendezve (nVidia Corporation, 2013)*

riák késleltetését hatékonyan el lehessen fedni számításokkal. A blokkonként felhasznált erőforrásokról a fordítás során a "Verbose PTXAS Output" opció bekapcsolásával kaphatunk tájékoztatást, de az nVidia által biztosított profiler segítségével is meghatározható a szűk keresztmetszet.

6. fejezet

Sugárkövetés megvalósítása GPU-n

6.1. A terem geometriájának bevitele, segédmátrixok előállítása

A feladat megoldása során egy téglatest alakú szobával dolgoztam, amelynek oldalhosszúságai a kódban paraméterként állíthatók. Ez összesen 8 csúcsot (*vertex*et) és 12 háromszöget (*face*t) jelent, hiszen minden oldallapot két háromszögre kell bontanunk. Ha bonyolultabb geometriát szeretnénk bevinni, akkor azt megtehetjük úgy, hogy egy 3D modellező szoftverben (pl. a *blender* programban) felépítjük, majd a csúcsok és az abból felépülő háromszögek listáját átültetjük valamilyen módon (pl. fájlba exportáljuk, és onnan olvassuk be). Természetesen a háromszögekhez az anyagukat is meg kell adni, amelyek egyelőre csak a kódban lettek definiálva. Elképzelhető az is, hogy a későbbiek során egy külső fájlban lesznek nyilvántartva az anyagok, és a különböző frekvenciákon mért abszorpciós tényezőik, így azok a felhasználók által is módosíthatóak lennének.

A háromszögek csúcspontjainak koordinátáiból a 2.11 egyenletben szereplő módon a **P** mátrixot összeállítva, majd azt invertálva állnak elő a segédmátrixok. Mivel ez egyszeri művelet adott geometria esetén, így feleslegesnek találtam GPU-n implementálni, a háromszögek száma ugyanis általában kicsi (egyszerűsített modellekkel dolgozunk), és CPU-n is kellően gyorsan elvégezhető a műveletsor.

6.2. Az impulzusválasszal kapcsolatos számítások

Az impulzusválasz közelítése kapcsán számos, számítástechnikai szempontból fontos kérdés felmerül. Ezekre a kérdésekre az 1. fejezetben tárgyalt akusztikai fogalmak segít-

ségével adhatunk választ.

6.2.1. A teljes impulzusválaszra vonatkozó paraméterek

Kérdés például, hogy adott teremben milyen hosszan kell az impulzusválaszt meghatározni ahhoz, hogy az elhagyott rész hiánya ne okozzon hallható változást végeredményben. Gyakorlati tapasztalatok alapján azt mondhatjuk, hogy miután az impulzusválasz teljesítménye 60 dB-nyit esett a kezdeti értékéhez képest, akkor a hátralévő rész már jelentéktelen. Az ehhez szükséges idő pedig pontosan az RT_{60}.

A sugárkövetés szempontjából fontos, hogy ismerve az impulzusválasz hosszát, hány visszaverődésen keresztül kell követni egy sugarat. Ennek kiszámításához felhasználtam az átlagos szabad úthossz (MFP) fogalmát, amely megmondja, hogy két visszaverődés között átlagosan mekkora utat tett meg a sugár. Ezt a hangsebességgel elosztva az ütközések közötti átlagos időtartam adódik, amely időtartammal az impulzusválasz teljes hosszát elosztva megkapjuk, hogy hány visszaverődés után (r_{max}) kell leállítani a sugárkövetést:

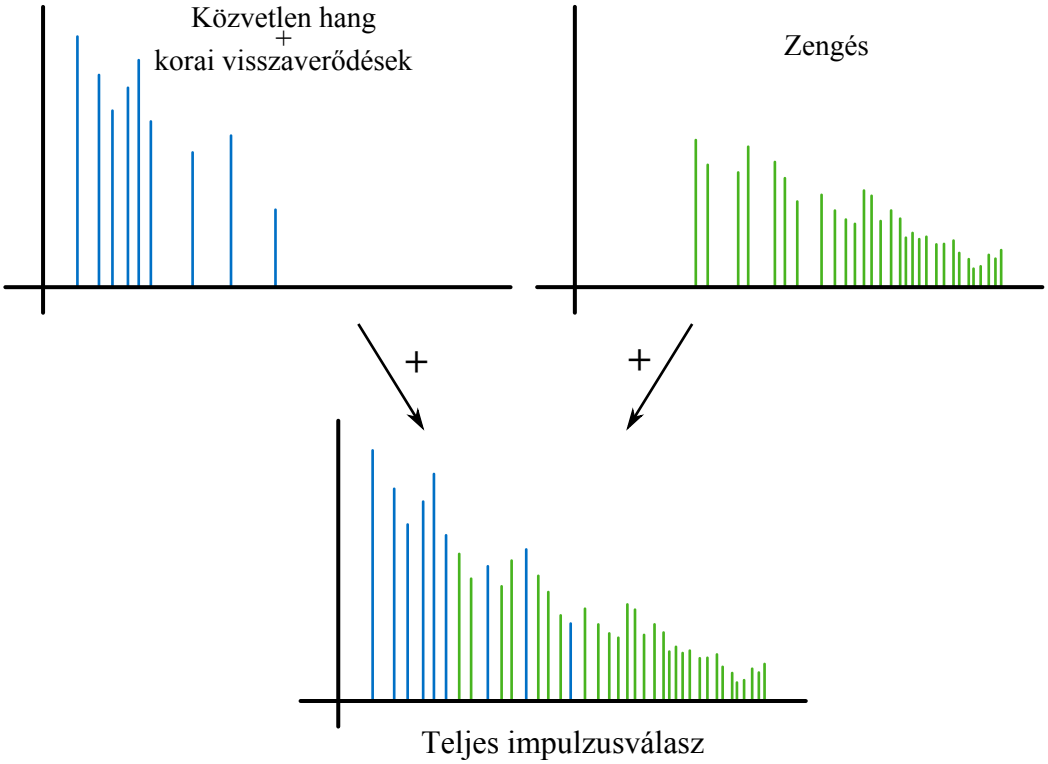
$$r_{max} = \frac{RT_{60} \cdot c_{air}}{MFP} \tag{6.1}$$

Sajnos nem ismerjük a sugarak által adott visszaverődési rend elérése alatt megtett út hosszának eloszlását, mivel az erősen függ a terem alakjától (Kuttruff, 2000). Az impulzusválasz végén emiatt biztosan lesznek hiányzó, vagy nem elegendő súllyal jelenlévő impulzusok, hiszen a soron következő visszaverődési rendig követett sugarak által megtett út csak egy MFP-nyivel lenne nagyobb átlagosan, vagyis feltehetőleg még jelentős hányada esne az impulzusválasz számunkra hasznos részébe. Ezt némiképp korrigálhatjuk úgy, hogy egy bizonyos biztonsági faktorral (pl. 1,2-del) megszorozzuk az így kapott (r_{max}) értéket.

6.2.2. A közvetlen hang és a korai reflexiók elkülönítése a zengéstől

A teljes impulzusválaszban a zengést csak egyszer szeretnénk kiszámítani, mivel az a forrás és a hallgató teremben elfoglalt pozíciójától függetlennek tekinthető. Ezért fontos annak meghatározása, hogy hol kezdenek elkülöníthetetlenné válni a hallgatót érő sugarak, vagyis mikortól válik dominánssá a zengés. Szimulációk alapján azt mondhatjuk,

hogy 4 visszaverődést elszenvedve a sugarak már nagyjából minden irányból egyenletesen érkeznek a hallgatóhoz, és nem rendelkeznek az irány- és helymeghatározáshoz szükséges információtartalommal, vagyis ezt tekinthetjük a zengés kezdetének. A zengés kiszámításánál ezért az első 5 visszaverődés után vesszük csak figyelembe a hallgatót érő találatokat. A teljes impulzusválaszt pedig a folyamatosan változó közvetlen hangot és korai reflexiókat tartalmazó impulzusválasz és ezen zengés összefésülésével nyerjük.



6.1. ábra. *A teljes impulzusválasz felépítése*

6.2.3. A hallgatót érintő sugarak száma

Számítástechnikai szempontból fontos ismernünk, hogy az összes indított sugár közül várhatóan hány fogja elérni a hallgatót, mivel találat esetén el kell tárolnunk a sugár irányvektorát, az addig elszenvedett csillapításait a különböző frekvenciákon, valamint a megtett út hosszát. Néhány visszaverődés után a sugarak a teremben véletlenszerű helyen tartanak és véletlenszerű irányban haladnak, és mindenképpen valamilyen felületnek fognak ütközni. Szimulációk alapján az az eredmény adódott, hogy a sugárkövetés minden periódusában a hallgatónak ütköző sugarak aránya az összes indított sugárhoz képest közelít a hallgató köré vont R sugarú gömb, valamint a gömb és a terem S összfelületének

arányához:

$$p_{hit} = \frac{4R^2\pi}{S + 4R^2\pi} \tag{6.2}$$

Ez azt jelenti, hogy ha legfeljebb ötödrendű visszaverődéseket engedünk meg, akkor a találatok átlagos száma az összes indított sugár számának $(5+1) \cdot p_{hit}$-szerese lesz (a $+1$ azért kell, mert visszaverődés nélkül is érheti találat a hallgatót). A találatok számának eloszlása binomiális, a várható érték pedig jócskán meghaladja a 10-et, emiatt élhetünk normális közelítéssel:

$$\mathcal{B}(n, p_{hit}) \approx \mathcal{N}(np_{hit}, np_{hit}(1 - p_{hit})) \tag{6.3}$$

Itt n az összes indított sugár számát jelöli. A szórás tehát $\sigma = \sqrt{np_{hit}(1 - p_{hit})}$, aminek ismeretében megadható különböző konfidenciaszintek mellett, hogy a várható érték hányszorosának megfelelő memóriaméretet szükséges lefoglalni ahhoz, hogy el tudjuk tárolni a találatokat. A gyakorlatban előforduló esetekben csillagászati annak valószínűsége, hogy a várható érték másfélszeresénél több memóriára legyen szükség, ezért a feladat megoldása során végig ekkora memóriaterületet foglaltam le a találatok nyilvántartására. Például egy $3 \times 4 \times 5$-ös szobában $20\,cm$ sugarú gömbbel, már 10 ezer sugár esetén is majdnem 4σ-s konfidenciaszintnek, holott ennél jóval több sugár szükséges általában.

6.3. Véletlenszám-generálás GPU-n

Sajnos a grafikus processzor nem tartalmaz hardveres véletlenszám-generátort, így ezt csak szoftveres úton tudjuk pótolni, és természetesen csak pszeudorandom sorozatot lehet ily módon előállítani. Számos megoldás közül választhatunk, a legcélszerűbb viszont, ha egy lineáriskongruencia-generátort implementálunk, ugyanis az összes közül ez a leggyorsabb algoritmus, hátránya viszont, hogy nem a legjobb minőségű véletlenszám-sorozatot adja. A sugárkövetésnél szerencsére a nagyfokú véletlenszerűség nem követelmény, hiszen akár szabályos irányrendszer szerint is indíthatnánk a sugarakat, csupán az egyenletes eloszlás számít.

Egy lineáriskongruencia-generátorral a következő számsorozatot nyerhetjük:

$$X_{n+1} \equiv (aX_n + c) \pmod{m} \tag{6.4}$$

Ebben a kifejezésben a a szorzó, c az inkremens, m pedig a modulus, amelyek értékét gondosan kell megválasztani ahhoz, hogy az értékkészlet minden eleme előforduljon a számsorozatban. Célszerű valamilyen jól bevált kombinációt keresni egy táblázatból; a dolgozat megoldása során a *Borland Delphi* fordítóban implementált lineáriskongruencia-generátornál felhasznált számhármast használtam:

$$a = 134775813, \quad c = 1, \quad m = 2^{32}$$

Ez azért is előnyös, mert 32 bites számokkal dolgozva a modulo művelet a csonkolás miatt automatikusan megtörténik. Szükségünk van emellett egy X_0 kezdeti értékre, amit például a beépített `clock()` függvénnyel biztosíthatunk, ez ugyanis egy olyan regiszter értékét kérdezi le, amely a GPU órajelciklusának ütemében folyamatosan növekszik, időnként pedig túlcsordul, tehát értéke véletlenszerű lesz.

6.4. A sugarak egyenletes elosztása egy gömbfelület mentén

Rendelkezünk tehát egy véletlenszám-generátorral, továbbra is kérdés azonban, hogy hogyan hozhatunk létre egyenletes irány menti eloszlást egy gömbfelületen. Azt gondolhatnánk naivan, hogy gömbi koordinátákban $r = 1$ sugár mellett a két irányszöget értékkészletükön belül egyenletes eloszlással kiválasztva egy gömb felületén egyenletes lesz az eloszlás, azonban könnyen belátható, hogy a gömb két pólusához közelítve sűrűsödik az egységnyi felületre eső sugarak száma. Egy nagyon elegáns módszert mutatott be viszont Marsaglia (1972), amely két, a $(-1,1)$ intervallumon egyenletes eloszlású véletlen számból indul ki, nevezzük ezeket x_1, x_2-nek! Azokat a párosokat elvetjük, amelyekre $x_1^2 + x_2^2 \geq 1$, a többiből pedig a következőképp kapjuk meg egy egységnyi hosszú irány-

vektor Descartes-koordinátáit:

$$x = 2x_1\sqrt{1-x_1^2-x_2^2}$$
$$y = 2x_2\sqrt{1-x_1^2-x_2^2}$$
$$z = 1-2(x_1^2+x_2^2)$$

Az így kapott egységvektorok iránybeli eloszlása egyenletes egy gömbfelület mentén.

6.5. A sugárkövetés kernelfüggvénye

A GPU-s megvalósítás alapkoncepciója az, hogy minden programszál egyetlen sugarat követ végig az előírt rendig. A blokkok és a grid egydimenziósak, jelen esetben nincs jelentősége, hogy egy blokkban hány warpnyi (hányszor 32) szál fut, ezért 256-ra választottam, és ebből a blokkból annyit hozok létre a gridben, hogy meglegyen az előírt sugárszám, ami így értelemszerűen csak 256 egész számú többszöröse lehet.

A sugárkövetés menete a következő:

1. Generálunk egy véletlenszerű irányba mutató egységvektort.
2. A forrásból kiindulva az adott irányban ellenőrizzük, hogy metsszük-e a hallgató köré vont gömböt (az általam kidolgozott algoritmussal). Ha igen, akkor elvégezzük az 2.2.1 pontban ismertetett távolságkorrekciót, valamint megadjuk a helyes irányt.
3. Tároljuk a korrigált távolságot, irányt, és az eddig elszenvedett csillapításokat.
4. Ezután megkeressük azt a háromszöget az összes közül, amelyet a sugár metsz, és emellett a kiindulási ponthoz a legközelebb van.
5. A sugár irányvektorát tükrözzük a felület normálvektorára, irányát pedig megfordítjuk.
6. Az új kiindulási pont a metszéspont.
7. A felület csillapításait hozzávesszük az eddig elszenvedett csillapításokhoz.

8. Ismételjük a folyamatot a 2. ponttól kezdve a megadott rend eléréséig, annyi különbséggel, hogy az új kiindulási pontból és az újonnan számított irányvektor mentén haladunk tovább.

A háromszögek normálvektora

Minden háromszög normálvektorát meg lehet kapni oly módon, hogy a 2.11 egyenletben szereplő **P** segédmátrix három sorvektorát összeadjuk. Formálisan:

$$\mathbf{P} = \begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{pmatrix}, \quad \mathbf{c}^T = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}, \quad \mathbf{n} = \mathbf{c}^T\mathbf{P} \tag{6.5}$$

Ez nem triviális, azonban szimbolikusan levezettem, hogy ezzel valóban a normálvektort kapjuk. A bizonyítást terjedelme miatt itt nem ismertetem.

A sugarak visszaverődése

Az egységnyi hosszú **n** normálvektorra a szintén egységnyi hosszú **t** irányvektort a következő formula segítségvel tudjuk tükrözni és irányát megfordítani:

$$\mathbf{t}_{refl} = -2(\mathbf{t}\cdot\mathbf{n})\mathbf{n} + \mathbf{t} \tag{6.6}$$

A metszéspont, és egyben az új kiindulási pont **r** helyvektorát pedig a 2.12 egyenlettel definiált k változó, és a régi kiindulási pont $\mathbf{r}_0$ helyvektora segítségével tudjuk kifejezni:

$$\mathbf{r} = \mathbf{r}_0 + k\mathbf{t} \tag{6.7}$$

Kimeneti adatok tárolása

Felmerül a kérdés, hogy hogyan és hol tároljuk a találatok adathalmazát. A shared memory mérete maximum $32-48\,kB$, és könnyen előfordulhat, hogy zengés számításánál akár 400-500 visszaverődésen át követni kell a sugarakat. A 6.2.3 pontban szó volt arról, hogy a találati arány átlagosan a gömb és a terem felületének a hányadosa. Így például egy $40\,cm$ sugarú gömb, és $3\times3\times3\,m$-es szoba esetén akár 5000 találat is lehet. Minden találatnál 11 darab `float` értéket kell tárolni (egy távolság, hat abszorpciós együttható

és az irányvektor három komponense), amelyek egyenként 4 byte-osak, tehát könnyen kifuthatunk a memóriából.

Emiatt zengés számításánál mindenképpen a globális memóriába célszerű írni a találatok adatait. A jelenlegi megoldás kezdetleges, mert egyetlen tömbbe kerül az összes találat, az aktuális indexpozíciót pedig egyetlen globális memóriában tárolt változó tartja nyilván. Mivel egyszerre több szál is szándékozhat találatot kiírni a memóriába, ennek a változónak a lekérdezését és inkrementálását elemivé (atomikussá) kell tenni. Ez azt jelenti, hogy ebben az esetben szerializálódnak a memóriaműveletek, egyszerre mindig csak egy szál tudja lekérdezni és növelni a változó értékét. Ez elméletileg problémát is jelenthetne, de mérések alapján azt találtam, hogy alig egy százaléknyival lassabb így a kód egy átlagosnak mondható konfiguráció mellett ahhoz képest, mintha nem is tárolnám a találatokat.

6.6. A sugárkövető algoritmus áteresztőképessége, futásidők

A GPU-s alkalmazás teljesítményének méréséhez C nyelven is implementáltam az általam kidolgozott algoritmust, így alkalmam nyílt CPU-n is lefuttatni. Egy modernnek számító Intel Core 2 Duo E8400 processzor (@ 4.0 GHz) egyetlen magján futtatva 2^{22} darab sugárt követtem 12 visszaverődésig egy téglatest alakú szobában, ahol ez 35,8 másodpercig tartott. Ugyanez egy középkategóriásnak mondható nVidia GeForce GTX 260-as grafikus kártyán alig több, mint 0,125 másodperc alatt lezajlott. A sugárkövetés tehát meglehetősen jól képes kihasználni a GPU-kban rejlő számítási kapacitást, ez ugyanis 287-szeres sebességbeli különbséget jelent.

Tekintve, hogy a szoba 12 háromszögből épült fel, adódik, hogy a GPU így egy másodperc alatt hozzávetőlegesen

$$\frac{(12+1)\cdot 2^{22}\cdot (12)}{0.125} \approx 5,2\cdot 10^9$$

vagyis 5,2 milliárd metszéstesztet tud elvégezni. Egyetlen háromszög leírásához szükségünk van mindenképp a három csúcspontjának mindhárom koordinátájára, ami összesen 9 darab lebegőpontos érték. A segédmátrix szintén 9 darab lebegőpontos számot tartalmaz, tehát az általam kidolgozott algoritmus minimális mennyiségű adattal dolgozik. Az 5,2 milliárd háromszöghöz tartozó, 9 darab 4 byte-os `float` változó beolvasása $175,5\,GB/s$-

os adatforgalmat generál. A GTX 260 fedélzeti memóriájának elméleti sávszélessége $111,9\,GB/s$ (nVidia Corporation, 2013b), vagyis az algoritmus memóriasávszélesség-limitált, sok háromszög esetén ez fogja a szűk keresztmetszetet jelenteni.

A 6.1 táblázatban látható két szélsőséges példa (egy kis átlagos lecsengésű szoba, és egy nagy, templomszerű terem) akusztikai paraméterei, valamint a mért futásidők.

6.1. táblázat. *Futásidők kis, egyszerű, és egy nagy, komplex terem esetén*

Teremméret [m]	$3 \times 3 \times 3$	$8 \times 20 \times 5$
$1 - \alpha$	0.9	0.98
Háromszögek száma	12	100
R [m]	0.3	0.3
MFP [m]	2	5.33
RT_{60} [s]	0.81	10.84
r_{max}	166	830
n	$\sim$ 1 millió	$\sim$ 7 millió
Zengés számítása	380 ms	113.4 s
Számítás 4 visszaverődésig	11 ms	680 ms
Zengés számítása CPU-val	108,7 s	-
Számítás 4 visszaverődésig CPU-val	3,1 s	193,5 s

Amint a táblázatból az kitűnik, a CPU-n futó verzió még a kisebbik szobánál is alkalmatlan valós idejű feladatok megoldására. Ha viszont GPU-val számolunk, akkor a kis szoba esetében mind a zengés, mind pedig a csak közvetlen hang és korai visszaverődések számolásánál kedvező futásidőket kapunk. Ha egy virtuálisvalóság-szimulátorban, pl. PC-s játékban szeretnénk felhasználni a teremszimulátort, akkor elképzelhető, hogy néhányszor $10ms$-os nagyságrendű frissítési idő a követelmény. Ezalatt már a teljes impulzusválasz kiszámolása a zengéssel együtt nem lehetséges, viszont a közvetlen hangot és a korai visszaverődéseket ilyen gyakorisággal lehet frissíteni, utólag pedig hozzá lehet keverni a zengést. A nagyobbik szoba esetében más a helyzet, itt a zengés kiszámítása már perceket vesz igénybe, de az első 4 visszaverődés kiszámítása is fél másodperc, ami bizonyos felhasználási területekhez nem elegendő. Egy épülettervező rendszernél viszont, ahol csak több fix pozícióból szeretnénk szimulációkat végezni, hasznos lehet.

7. fejezet

Auralizáció és az impulzusválasz összeállítása GPU-n

Az auralizációhoz szükséges ismernünk a sugarak beérkezésének hallgatóhoz képesti irányát, ezért nyilván kell tartanunk azt, hogy a hallgató éppen merre néz arccal. A következő pontban erre ismertetek egy lehetséges megoldást.

7.1. A hallgató és a forrás helyzete

Ha felveszünk egy Descartes-féle koordináta-rendszert jobbkéz-szabály szerint úgy, hogy az x tengely a magasságot, az y tengely a szélességet, a z tengely pedig a mélységet jelentse, akkor semmilyen problémát nem jelent a hallgató és a forrás pozíciójának megadása. A forrás minden irányban egyenletesen sugároz, ezért nincs is szükség más térbeli információra, azonban a hallgató a HRTF miatt irányfüggő karakterisztikával rendelkezik, vagyis nem mindegy, hogy merre néz.

A HRIR adatbázisban a $(\theta = 0, \varphi = 0)$ referenciairányt a hallgató arcára merőlegesen érkező sugarak jelentik, a számítások során viszont a kifelé néző irányt fogom használni, hiszen így az irányvektor mutatja, hogy a hallgató arccal merre néz. Ezt az irányváltást persze az azimut szög számításakor majd figyelembe kell venni.

Legyen tehát $\mathbf{t} = (t_x, t_y, t_z)$ a normált irányvektor! Ha megállapodunk abban, hogy a fejet nem lehet jobbra és balra dönteni (angol terminológiában ez a *roll*, vagy *tilt*), úgy

két másik kísérővektor is definiálható e mellé:

$$\mathbf{u} = (0, t_z, -t_y) \tag{7.1}$$

$$\mathbf{v} = (t_z, 0, -t_x) \tag{7.2}$$

Az **u** mindig a jobb fül irányába, **v** pedig mindig a fejtető felé mutat, és a ezek páronként merőlegesek egymásra. Ezek birtokában hozzákezdhetünk a sugárkövetés által szolgáltatott találatokból az impulzusválaszok összeállításához.

7.2. Részleges impulzusválaszok összeállítása a találatokból

A találatok feldolgozását 64 darab, 256 szálat futtató blokk végzi úgy elosztva, hogy minden szálra egy-egy találat jusson, majd az egész grid továbblép a találatok tömbjében, amíg végig nem ér rajta. Minden blokk az általa feldolgozott találatokból egy részleges impulzusválaszt állít elő egy csak általa hozzáférhető memóriaterületen. Egy blokkon belül a szálak szimultán kiszámolják a találatokhoz tartozó gömbi koordináta-rendszerbeli szögeket, amikből aztán meg lehet határozni a hozzá tartozó HRIR-eket.

- az azimut szög (θ) az irányvektor **t** és **u** vektorok által kifeszített síkra vett vetületének a **t**-vel bezárt szöge adja (mivel ez csak 0 és 180° között lehet, a 360°-ra történő kiterjesztés a vetület **u**-val vett skaláris szorzatának előjele alapján lehetséges)
- a másik szög (φ) pedig az irányvektor **v**-vel bezárt szöge

Ezek formálisan, ha **d** a normált irányvektor, $\mathbf{k} = (\mathbf{d}\cdot\mathbf{t})\cdot\mathbf{t} + (\mathbf{d}\cdot\mathbf{u})\cdot\mathbf{u}$ a vetület és $\mathrm{sgn}()$ az előjelfüggvény:

$$\theta = \mathrm{sgn}(\mathbf{k}\cdot\mathbf{u}) \arccos\left(\frac{\mathbf{d}\cdot\mathbf{t}}{|\mathbf{k}|}\right) + (1 - \mathrm{sgn}(\mathbf{k}\cdot\mathbf{u}))\cdot 180^\circ \tag{7.3}$$

$$\varphi = 90^\circ - \arccos(\mathbf{d}\cdot\mathbf{v}) \tag{7.4}$$

Az irányinformációk birtokában ezután mind a 256 szál a találatokon egyesével halad végig. A HRIR adatbázis a globális memóriában van, tömbbe szervezve, frekvenciasávokra lebontva, mindkét hangcsatornára. A kiszámított irányhoz legközelebbi diszkrét irány mindkét hangcsatorna minden frekvenciasávjának impulzusválaszát az elszenvedett csillapításnak megfelelő súllyal megszorozva a részleges impulzusválaszhoz adjuk a sugár

által megtett útból kiszámított késleltetéssel eltolva. A másolás egy-egy elemi impulzusválasz esetében 4 ciklust vesz igénybe, mivel azok 1024 minta hosszúak és 256 szál van.

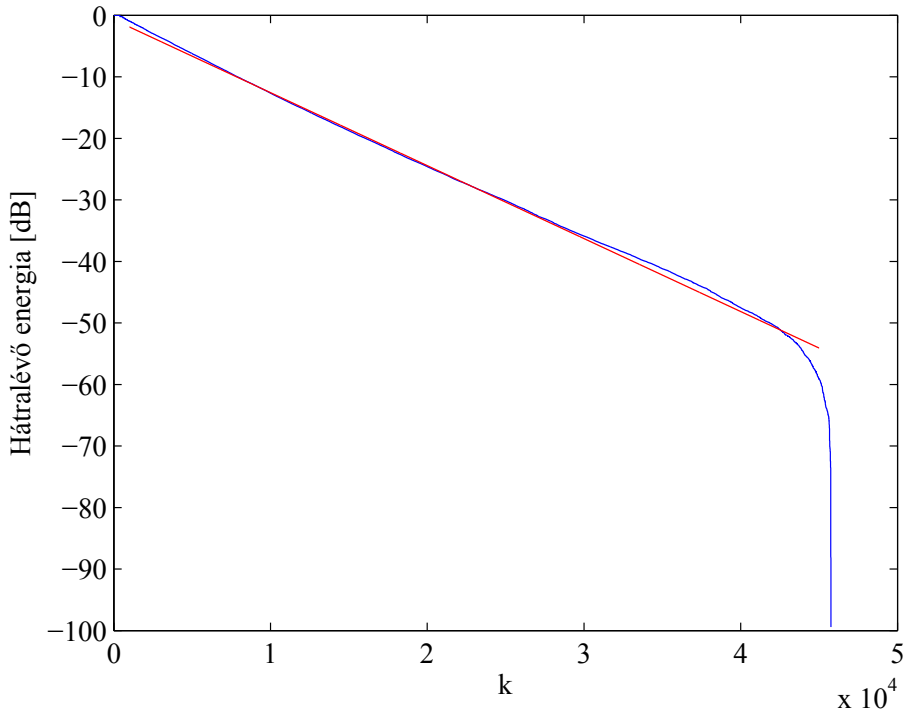
Itt említeném meg, hogy fontos tudatában lennünk a warp-szervezésnek, vagyis hogy a szálak 32-es kötegekben futnak fizikailag, azon belül minden szinkron történik, de a warpok aszinkron haladnak a kernelkódban. Megeshet, hogy egy blokkon belül az egyik warp még csak az első találathoz tartozó elemi impulzusválaszok másolását végzi, még egy másik már például az ötödikét. Mivel azonban ugyanarra a globálismemória-területre dolgoznak, és nem atomikus műveletekkel végzik a módosításokat, könnyen lehet, hogy egy warp által beolvasott adat a globális memóriában módosul a kiolvasás és a visszaírás között eltelt időben (egy másik warp hozzáadott egy újabb impulzusválasz-részletet). Amikor ez a warp visszaírja az eredményeit, ezek az időközi módosítások elvesznek. Emiatt a blokkon belül minden találat feldolgozása után minden warpot be kell várni, mielőtt egy újabb találatot kezdenének feldolgozni. Ezt szálszinkronizációnak nevezzük, és a `__syncthreads()` függvénnyel tehetjük meg.

7.3. A részleges impulzusválaszok összevonása

Mivel a 7.2 pontban leírt megoldás 64 blokkot használt, így 64 darab sztereó részleges impulzusválaszt kaptunk eredményül. Ezek összefésülését szintén 64 darab 256 szállal futó blokk végzi. Ez mindössze annyit takar, hogy minden szál a 64 részleges impulzusválasznak egy adott pozíciójában összegez, majd mindkét csatornában egy kimeneti adattömb ugyanazon pozíciójába írja az eredményt.

7.4. Az impulzusválasz ellenőrzése

Az ellenőrzést egy $3 \times 4 \times 5$ méteres szobával, $S(1,1,1)$ és $P(3,2,3)$ konstellációval, z irányba néző hallgatóval végeztem, a hallgató körüli gömb sugara $0,2\,m$ volt, a falak abszorpciós tényezője $0,1$. Valódi HRIR adatbázis helyett egységimpulzusokat használtam, így egyszerre tudtam ellenőrizni a szűrők együttes egységnyi átvitelét frekvenciafüggetlen abszorpciós együtthatók mellett, valamint az impulzusválasz elejét össze tudtam hasonlítani azzal, amit a tükörforrások módszerével kaptam. A sugárkövetés paramétereit a fejezetben tárgyaltak szerint állítottam be. A referenciával összehasonlítva a maximális eltérés a konfidenciaszintnek megfelelően 95 %-ban a beállított $1.5\,dB$-en belül volt.



7.1. ábra. *Az ellenőrzéshez használt impulzusválasz EDC görbéje*

Az impulzusválasz többi részén így csak a lecsengés exponenciális mivoltáról kellett meggyőződni, ami az impulzusválasz EDC görbéjének szemrevételezésével történt, ez látható a 7.1 ábrán kék színnel. A közvetlen hang és a korai visszaverődések után a lecsengés hosszan exponenciális volt, ez abból látható, hogy logaritmikus skálán a [-5;-35] dB-es szakaszra jól illeszthető egy egyenes, a végén lévő eltérés pedig betudható az impulzusválasz véges hosszának.

7.5. Áteresztőképesség

Az impulzusválaszokat összeállító és az auralizációt jelenleg megvalósító algoritmusok kezdetlegességük ellenére közelítőleg egymillió találatot tudnak feldolgozni másodpercenként, a 6.1 táblázatban lévő szobák esetében zengés számításánál rendre 3.5 és 10.9 másodperc, csak direkt hang és korai visszaverődések frissítésekor pedig rendre 164 ms és 470 ms futásidőket eredményezett, amely virtuálisvalóság-rendszereknél túlzottan nagy késleltetést okozna. Ezen lehet változtatni például úgy, hogy kihagyjuk a frekvenciasávokra történő bontást, és egyetlen abszorpciós tényezővel dolgozunk minden frekvencián. Ha nem kell szűrni, akkor a HRIR-ek hossza az eredeti 512 minta hosszú marad,

tehát 256 szál 2 ciklusban el tudja végezni az impulzusválaszhoz adást. Tehát így összesen 12-ed részére csökkenthető a számításigény, így már a kisebb szoba esetében 10 ms körüli értéket kapunk, ami már kellően gyors. Például virtuálisvalóság-rendszerek esetében, ahol a pontosság nem követelmény, ellenben a gyors frissítés igen, ez nem is igazán kompromisszum.

8. fejezet

Összefoglalás

8.1. Eredmények értékelése

Az elengedhetetlen akusztikai és pszichoakusztikai fogalmak bevezetése után részletesen ismertettem a geometriai akusztika két fontos eszközét, a tükörforrások módszerét és az akusztikai sugárkövetést.

Az akusztikai sugárkövetés elvének ismertetése után feltártam a hallgató köré vont véges térfogatból adódó problémákat, tanulmányoztam azok hatását az impulzusválaszra, illetve javasoltam egy megoldást a hallgató és a források közti út, valamint az irány korrekciójára. A sugárkövetés leggyakrabban ismételt részfeladata a metszéstesztelés, ezért tanulmányoztam az irodalomban fellelhető algoritmusokat, ezek közül kiválasztottam egyet, amely leginkább illik a GPU-k architektúrájához, illetve magam is kidolgoztam egy eljárást. Ismertettem továbbá, hogy hogyan lehet frekvenciafüggő abszorpciójú anyagokat bevonni a modellbe.

Mértékegység-egyeztetés után összevetettem a sugárkövetéssel előállított impulzusválaszt a tükörforrások módszere nyújtotta referenciával, ezután pedig elemeztem ez utóbbi statisztikai tulajdonságait. Megadtam, hogyan lehet adott hallgató körüli gömbméret esetén biztosítani az a megengedett relatív eltérést adott konfidenciaszint mellett az indított sugarak számának változtatásával.

Ezután bemutattam, hogyan lehet auralizációt megvalósítani a sugárkövetésből nyert adatokból, ha frekvenciafüggő abszorpciójú anyagok is jelen vannak. Ehhez frekvenciasávonként el kellett készíteni egy HRIR adatbázist, a frekvenciasávokra bontáshoz pedig szűrőket kellett tervezni. Erre a célra FIR szűrőket alkalmaztam, és megvizsgáltam, hogy

ez okozhat-e valamilyen hallható, nemkívánatos elváltozást a kimenetben.

Az 5. fejezetben röviden ismertettem a megértéshez szükséges, GP-GPU programozáshoz szükséges alapfogalmakat, illetve bemutattam az architektúrát, a főbb programozási irányelveket.

Ezek ismeretében tárgyaltam a sugárkövetés megvalósítását GPU-n, amely magában foglalta a terem geometriájának bevitelét háromszögháló formájában, a véletlen számok, és abból egyenletes iránybeli eloszlású vektorok generálását, megfontolásokat a memóriafoglalásokat illetően, majd pedig a konkrét kernelfüggvény lépéseinek ismertetését. Ezután teszteltem az algoritmust, és megállapítottam az áteresztőképességét, ezután példákon keresztül megmutattam, hogy ez teljesítmény valós idejű alkalmazásoknál is elegendő lehet, a felhasználói igényektől függően.

Végül a 7. fejezetben ismertettem egy lehetséges megoldást arra, hogy hogyan lehet az auralizációt elvégezni a sugárkövetéssel nyert találatok információinak felhasználásával, majd ellenőriztem egy teljes impulzusválasz EDC görbéjén, hogy megfelel-e a követelményeknek az impulzusválasz lecsengése.

8.2. Továbbfejlesztési lehetőségek

A terem belsejét felépítő háromszögek számának növekedésével a jelenleg használt metszéstesztelő algoritmus számításigénye lineárisan növekszik. Ezen lehetne javítani úgy, hogy a metszésteszteket nem mindig minden háromszögre végezzük el, hanem valamilyen módon felosztjuk a teret, csoportokat alkotunk a háromszögekből, és csak ezen csoportokon belül lévő háromszögekre végzünk metszéstesztet, amikor áthalad rajta a sugár. Ilyen, térfelosztásra alkalmas adatstruktúra például a *k-d fa*.

Mindenképpen érdemes továbbgondolni, hogy hogyan képezhető a találatokból impulzusválasz. A jelenlegi megvalósítás kifejezetten nem hatékony, ugyanis minden egyes találatnál az adott irány mind a 6 frekvenciasávjához tartozó elemi impulzusválasz csatornánként bemásolásra kerül. Ez azt jelenti, hogy ha például egy virtuális forrásból 1000 darab találat érkezik, akkor az elemi impulzusválaszok ezerszer lesznek másolva ahelyett, hogy mind az 1000 találat súlyait összevonnánk, és csak egy másolási ciklust végeznénk. A feladat tehát a találatok közül az egyediek megtalálása és összevonása. Ennek megoldására a hashelés tűnik a legkézenfekvőbb megoldásnak, a hashtáblákat akár a shared

memoryban is ki lehetne alakítani, a megfelelő eloszlást pedig kvadratikus próbálással, vagy kettős hasheléssel lehetne biztosítani.

Az egyedi találatok kiválogatása ihletett egy másik megközelítést is, amely szerint egy virtuális forrásból érkező egyetlen találatból meg lehet mondani a forrás impulzusválaszbeli pontos súlyát a hallgatótól való távolsága alapján, csakúgy mint a tükörforrások módszerénél. Ezzel elejét lehet venni az impulzusok bizonytalanságának, tehát egy kvázi determinisztikus eljárás lenne. Ugyan hiányzó impulzusok ekkor is lesznek, azonban ezek legnagyobb valószínűséggel az impulzusválasz végén fordulnak elő, és az indított sugarak számának szabályozásával megadható egy konfidenciaszint arra vonatkozólag, hogy adott távolságon belül lévő forrásokat mekkora valószínűséggel találjanak el legalább egyszer a sugarak. A frekvenciafüggő csillapítások kezelése sem jelent problémát, ugyanis egy virtuális forráshoz csak egyetlen terembeli út szerkeszthető, ahogyan a 2.1 pontban kifejtettem, tehát az ugyanazon forrásból érkező sugarak mindig ugyanazokat a csillapításokat szenvedik el.

Köszönetnyilvánítás

Ezúton szeretném megköszönni konzulensemnek, Dr. Bank Balázsnak, hogy a kezdeti nehézségek ellenére toleráns volt, bátorított a feladat megoldására, illetve a konzultációk során hasznos tanácsokkal látott el. Köszönöm szüleimnek, hogy támogattak a nehéz időkben, végül pedig szeretném megköszönni csodálatos menyasszonyomnak, hogy mindvégig mellettem állt, és lelkesített, amikor csak szükségem volt rá.

Irodalomjegyzék

Allen, J. B. and Berkley, D. A. (1979). Image method for efficiently simulating small?room acoustics. *The Journal of the Acoustical Society of America*, 65(4):943–950.

Badouel, D. (1990). Graphics gems. In Glassner, A. S., editor, *Graphics Gems*, chapter An Efficient Ray-polygon Intersection, pages 390–393. Academic Press Professional, Inc., San Diego, CA, USA.

Blauert, J. and Laws, P. (1978). Group delay distortions in electroacoustical systems. *Acoustical Society of America Journal*, 63:1478–1483.

CATT (2013). CATT-Acoustic. `http://www.catt.se`.

Eberly, D. (2006). *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*. Interactive 3D technology. Taylor & Francis.

Fastl, H. and Zwicker, E. (2007). *Psychoacoustics: Facts and Models*. Springer series in information sciences. Springer.

Howard, D. and Angus, J. (2009). *Acoustics and Psychoacoustics*. Taylor & Francis.

IRCAM (2013). Listen HRTF Database. `http://recherche.ircam.fr/equipes/salles/listen`.

J. Blauert (1999). *Spatial hearing – The psychophysics of human sound localization*. The MIT Press.

Krokstad, A., Strom, S., and S?rsdal, S. (1968). Calculating the acoustical room response by the use of a ray tracing technique. *Journal of Sound and Vibration*, 8(1):118 – 125.

Kuttruff, H. (2000). *Room Acoustics, Fourth Edition*. E-Libro. Taylor & Francis.

Lokki, T., Svensson, P., and Savioja, L. (2002). An efficient auralization of edge diffraction. In *Audio Engineering Society Conference: 21st International Conference: Architectural Acoustics and Sound Reinforcement*.

Marsaglia, G. (1972). Choosing a point from the surface of a sphere. *The Annals of Mathematical Statistics*, 43(2):645–646.

Möller, T. and Trumbore, B. (1997). Fast, minimum storage ray-triangle intersection. *J. Graph. Tools*, 2(1):21–28.

nVidia Corporation (2013). Cuda c programming guide. `http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf`.

nVidia Corporation (2013a). Fermi Compute Architecture Whitepaper. `http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf`.

nVidia Corporation (2013b). GTX 260 Specifications. `http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-260/specifications`.

Odeon (2013). Odeon Room Acoustics Software. `http://www.odeon.dk`.

Rindel, J. H. (1995). Computer simulation techniques for acoustical design of rooms. In *Proceedings*, pages 81–86.

SAE Institute (2013). Absorption Coefficient Chart. `http://www.sae.edu/reference_material/pages/Coefficient%20Chart.htm`.

Simon Skluzacek (2012). *Angular Resolution of Human Sound Localization*. PhD thesis, Carthage College, Kenosha, Wisconsin.

Snyder, J. M. and Barr, A. H. (1987). Ray tracing complex models containing surface tessellations. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 119–128, New York, NY, USA. ACM.

Stephenson, U. (1990). Comparison of the mirror image source method and the sound particle simulation method. *Applied Acoustics*, 29(1):39–40.

Stevens, S. S., Je, and Newman, E. B. (1937). A scale for the measurement of the psychological magnitude of pitch. *J. Acoust Soc Amer*, 8:185–190.

Svensson, U. P. and Kristiansen, U. R. (2002). Computational modelling and simulation of acoustic spaces. In *in Proc. of the 22nd Int. AES Conf., Helsinki, Finland, June 15-17, 2002*, pages 1–20.

Taylor, M., Chandak, A., Ren, Z., Lauterbach, C., and Manocha, D. (2009). Fast edge-diffraction for sound propagation in complex virtual environments. In *EAA auralization simposum*, pages 15–17, Espoo, Finland.