



MISKOLCI EGYETEM
GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

CT és MRI képsorozatok előfeldolgozása és valósidejű háromdimenziós vizualizációja

Barcsák Csaba
II. éves mérnök informatikus hallgató

Konzulens:
Dr. Mileff Péter
egyetemi adjunktus
Általános Informatikai Tanszék

Miskolc, 2009

Köszönetnyilvánítás

Hálás vagyok a témavezetőmnek, Dr. Mileff Péternek, aki figyelemmel kísérte munkámat, sokat segített a háromdimenziós számítógépes grafika problémáinak megértésében, és dolgozatom átolvasása után építő javaslatokat tett.

Köszönöm Prof. Dr. Fornet Béla radiológus orvosnak, és a Dolphio Consulting Kft. szakembereinek, hogy dolgozatom átolvasása után építő javaslatokat tettek.

Szeretném megköszönni Marc Levoy-nak, a Stanford University professzorának, hogy személyes megkeresésemre a rendelkezésemre bocsátotta az általa készített CT és MRI képsorozatokat.

Köszönettel tartozom az AIESEC-es mentoromnak, Dobronyi Anettnak, aki nélkül nem lettem volna eléggé motivált ahhoz, hogy ez a munka megszülessen.

Tartalomjegyzék

1. Bevezetés.....	1
2. Történelmi Áttekintés.....	3
2.1. Az orvosi képalkotás múltja.....	3
2.2. Az emberi test vizualizációjának eredményei.....	4
2.2.1. Kétdimenziós előfeldolgozás.....	5
2.2.2. Háromdimenziós képalkotás.....	7
2.3. A szakterület jövője.....	9
3. A MedVis térfogat-vizualizációs alkalmazásban használt módszerek.....	10
3.1. Felhasznált adathalmazok.....	11
3.2. Kétdimenziós feldolgozás.....	11
3.2.1. Kontraszt javítása.....	12
3.2.2. Zajcsökkentés.....	13
3.2.3. Szegmentálás.....	16
3.2.4. Multi-planáris rekonstrukció.....	17
3.3. Háromdimenziós képalkotás.....	19
3.3.1. Térfogat vizualizáció (Volume rendering).....	19
3.3.1.1. Térfogat metszés (Volume slicing).....	19
3.3.1.2. Voxel-ek reprezentációja téglatestekkel.....	20
3.3.1.3. Voxel-ek reprezentálása point-sprite-okkal.....	22
3.3.2. Poligon alapú megjelenítés.....	23
3.3.2.1. Masírozó kockák.....	24
3.4. A bemutatott módszerek összehasonlítása.....	26
4. Az alkalmazás bemutatása.....	28
5. Továbbfejlesztési lehetőségek.....	30
6. Konklúzió.....	31
Irodalomjegyzék.....	32
A. Függelék.....	34

1. Bevezetés

Az elmúlt években az orvosi képalkotó diagnosztika nagy fejlődésen ment keresztül. Ezzel egyidejűleg fejlődött az orvosi képek számítógépes feldolgozása. Az orvosi képfeldolgozás ma is dinamikusan fejlődő terület. A számítógépek hardver eszközeinek fejlődése miatt egyre nagyobb részletességű háromdimenziós modelleket állíthatunk elő a CT és MRI képsorozatokból, melyekről animációkat is készíthetünk. Ezek a modellek segítenek az orvosoknak a diagnózis felállításában, használhatjuk ezeket sebészi tervezésre, sebész szimulátor programokban, és háromdimenziós anatómiai atlasz készítéséhez amely megkönnyíti az anatómia tanulását.

A képek egyre tökéletesebb vizualizációjának témája napjainkban is időszerű, mert a betegségek gyors felismerése emberek életét mentheti meg, és az orvosi képfeldolgozás ebben nagy segítséget ad az orvosoknak. A hardver eszközök folyamatos fejlődése egyre újabb és újabb lehetőségeket nyit meg a terület kutatói előtt, akik arra törekednek, hogy a lehető leggyorsabban és legjobb minőségben állítsanak elő képeket.

Ebben a dolgozatban a CT és MRI képek kétdimenziós és háromdimenziós vizualizációjánál felmerülő problémák megoldására helyezem a fő hangsúlyt, amelyhez megvalósítás eszközeként a Java technológiát választottam. A vizualizációs problémák összetettsége miatt a terület érinti mind a kétdimenziós, mint a háromdimenziós képszintézist. A felmerülő problémák a képek előfeldolgozása (zajszűrés, kontraszt feljavítás, szegmentálás), kétdimenziós rekonstrukció, háromdimenziós megjelenítés. A háromdimenziós megjelenítés témakörében több megoldás is bemutatásra kerül.

Az orvosi képek vizualizációja bár nem oldja meg a klinikai eseteket, de sokat segít az orvosnak a döntéshozatalban, egy-egy diagnózis pontosabb felállításában.

Kutató munkám során tehát elsődleges célom az volt, hogy megismerkedjek az orvosi képalkotás legfontosabb problémáival, valamint olyan megoldásokat dolgozzak ki, amelyek hatékonyak és kellően rugalmasak. Munkám során további célként egy olyan alkalmazás készítését tűztem ki, amely jól összefoglalja az elért eredményeket. Az alkalmazás legfontosabb funkciói:

- CT és MRI képsorozat beolvasása
- a képsorozat előfeldolgozása
 - kontraszt feljavítás
 - zajszűrés
 - szegmentálás
- a képsorozatok kétdimenziós rekonstrukciója
- a képsorozatok háromdimenziós megjelenítése térfogat vizualizáció (volume rendering) segítségével

1. Bevezetés

- felhasználó által definiált síkmetszet létrehozása
- a térfogat vizualizáció többféle megoldással történő megvalósítása
- poligon alapú háromdimenziós modell előállítása és megjelenítése

A munkám első részében megismerhetik, hogy hogyan fejlődött ki ez a tudományterület, majd a napjainkban az orvosi képek feldolgozásához használt megoldások kerülnek bemutatásra. Először néhány kétdimenziós képfeldolgozó eljárást ismerhet meg az érdeklődő, majd a kidolgozott háromdimenziós megjelenítési módszerek kerülnek ismertetésre.

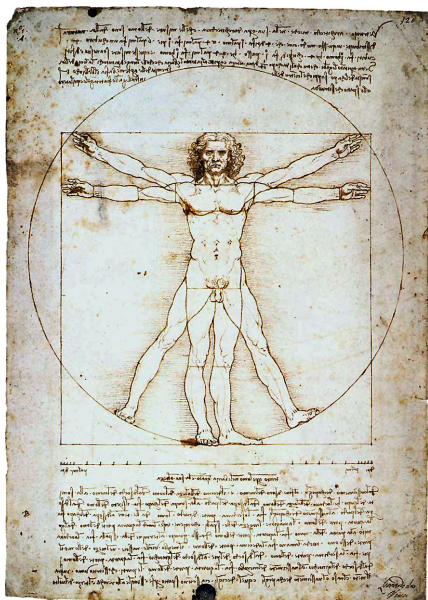
A második fejezetben az alkalmazás fejlesztéséhez használt megoldások és ötletek kerülnek részletes bemutatásra. Elsőnek a képek előfeldolgozási módszerei kerülnek ismertetésre. A képek előfeldolgozása fontos, mivel ennek segítségével ki tudjuk javítani a képeken a pixel és kontraszthibákat, ezáltal a háromdimenziós megjelenítésnél szebb eredményt kapunk. Ebben az alfejezetben szó lesz arról, hogy mi a hisztogram, hisztogram kiegyenlítés, továbbá több zajszűrő algoritmust is bemutatok, majd a képek szegmentálásáról olvashatnak, amit thresholding eljárás segítségével valósítottam meg, ezután a multi-planáris rekonstrukcióval ismerkedhet meg az olvasó. A második részfejezetben az alkalmazásomban használt háromdimenziós képalkotási módszerekről lesz szó. Az olvasó megismerkedhet a manapság egyre népszerűbb, de még kevesek által ismert térfogat vizualizációval (volume rendering) majd a poligon alapú megjelenítés kerül bemutatásra.

2. Történelmi Áttekintés

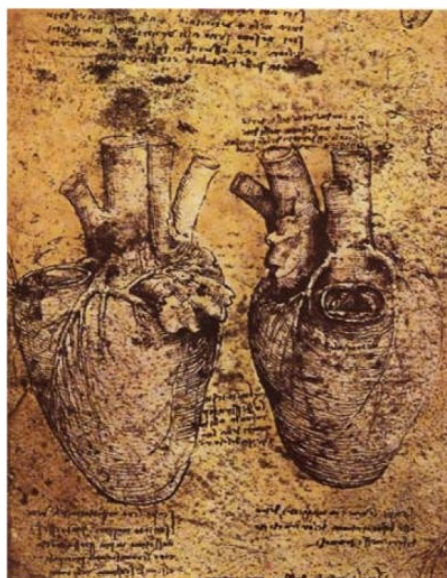
Az orvosi diagnosztikai képalkotás hosszú fejlődésen ment keresztül a történelem során. Mivel mindezek bemutatása nem férne el a dolgozat keretein belül, így ebben a fejezetben azon leglényegesebb pontokat emelem ki, amelyek a legfontosabb szerepet játszották abban, hogy az orvosi képalkotás és képfeldolgozás ma itt tart. Előbb meg kell ismernünk az anatómia történelmét (persze csak a fontosabb neveket és eseményeket kiragadva), majd rövid ismertetést adok a ma használt eljárásokról.

2.1. Az orvosi képalkotás múltja

Az emberi test működése már az ősidők óta érdekli az embereket. Huang Ti anatómia könyve úgy született (ie.2600 körül), hogy soha nem végzett boncolást, de már ekkor tudta, hogy a vér körforgásban van, és a körforgást a szív biztosítja. A görög Hippokratész (i.e.460-i.e.377) sem boncolt soha, emiatt anatómiai ismeretei tévedéseken alapultak. Az anatómia atyjának tekintett Herophilosz (ie.335- ie.280) volt az első olyan orvos, aki tudatosan boncolt emberi testeket. A mendemondák szerint annyira elhivatott volt, hogy 600 foglyot élve boncolt fel. Galénosz (129-199) gladiátorok orvosaként dolgozott, emiatt jól ismerte az ember belső szerveit [15]. Galénosz halála után kezdetét vette a középkori hanyatlás és közel 1000 évig senki sem végzett boncolást. A bolognai Mondino dei Luzzi jóvoltából 1315-ben ismét elkezdődött a boncolás. Boncolási tanulmányát széles körben terjesztették. Az évi egy-két tetem feltárása lassan teret nyert Európa valamennyi egyetemén. Leonardo Da Vinci (1452 április 15. - 1519 május 2.) olasz polihisztor is gyakran végzett boncolásokat. Az emberi test művészetekre tett hatását és arányait a **Vitruvius-tanulmány** munkájában foglalta össze.



Ábra 2.1.1: Leonardo Da Vinci:
Vitruvius-tanulmány



Ábra 2.1.2: Leonardo Da Vinci
szívről készült tanulmánya

2.1. Az orvosi képalkotás múltja

A német Vesalius (1514-1564) a reneszánsz anatómiaórák professzoraival ellentétben az aktív boncolás híve volt. A történelem legnagyobb boncnokaként tartják számon. Bolognában 1570-ben megalakult az első önálló bonctani tanszék. A XVIII-XIX. században Britanniában tömegesen nyíltak meg magán anatómiai iskolák, amik halálraítéltek tetemeit boncolták. Jó pénzt fizettek egy egy emberi testrészt, hulláért, emiatt megszorodtak a hullarablások. 1828-ra a londoni anatómiai iskolák főszezonban, több mint 200 hullarablót alkalmaztak. A XXI. században sem fejeződött be a holttestek tudományos vizsgálata [21].

Az anatómia története amiatt kapott helyet ebben a munkában, mivel nélkül nem létezne a modern orvostudomány, és a radiológia megszületése előtt csak ennek segítségével tudtak az ember belső szerveiről rajzokat készíteni, ismereteket szerezni. A továbbiakban a radiológia történelméből kerülnek ismertetésre fontosabb mozzanatok.

- 1895-ben Wilhelm Conrad Röntgen felfedezte az X sugarat forradalmasítva ezzel az orvostudományt. 1901-ben fizikai Nobel-díjat kapott.
- 1912 ben a tengeralattjárók detektálására kifejlesztették a radart. Az ultrahangot orvosi célra először Prof. Ian Donald használta Glasgow-ban.
- Az MRI első klinikai használatára a Nottingham University Kórházában került sor 1967-ben. A képek minősége ekkor még rossz volt, de azóta sokat fejlesztettek rajta.
- 1971-ben elkészül az első CT prototípus
- 1975-ben elkészül az első PET scanner.
- 1979-ben Allan M. Cormack és Godfrey N. Hounsfield orvosi Nobel-díjat kaptak a CT kifejlesztéséért.

2.2. Az emberi test vizualizációjának eredményei

A röntgensugár felfedezése óta több mint 100 év telt el. Ez alatt az orvosi képalkotó diagnosztika fejlődése töretlen volt, és a fejlődés még ma is tart. A képalkotó diagnosztikai eszközök fejlődését az informatika, biotechnológia, nanotechnológia és genetika eredményei támogatják [4]. Az így kinyert, „javított” adatok megjelenítése a háromdimenziós grafika és a számítógépes képfeldolgozás szerves részét képezik.

Sok kutató foglalkozik az orvosi képek előfeldolgozásával. Az előfeldolgozás amiatt fontos, mivel segítségükkel feljavíthatjuk a képeket, és kiválaszthatjuk a képekről azokat az információkat amiket meg szeretnénk jeleníteni, ezáltal pontosabb, illetve részletesebb háromdimenziós modellt tudunk majd készíteni az adatokból. A téma szorosan összefügg a háromdimenziós számítógépes képszintézissel. Eszközkét tehát az ott kidolgozott megközelítések alkalmazhatók. A nagy részletességgel bíró adatok, modellek megjelenítése napjainkban is probléma, mert mind a mai napig nem állnak rendelkezésünkre kellő számítási kapacitással bíró hardver eszközök.

2.2. Az emberi test vizualizációjának eredményei

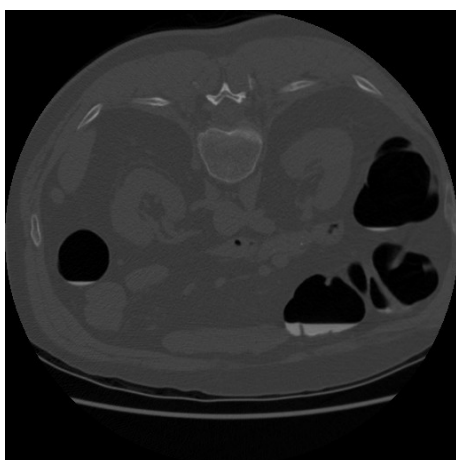
A kutatás fő iránya így tehát az, hogy minél nagyobb felbontású háromdimenziós modelleket legyenek képesek megjeleníteni valós időben. A hardverek folyamatos fejlődése miatt mindig új megoldásokra, ötletekre van szükség.

2.2.1. Kétdimenziós előfeldolgozás

Ebbe a csoportba azokat az eljárásokat soroltam, melyek tetszőleges kétdimenziós képsorozaton elvégezhetők. Az irodalomban számos eljárás létezik, amelyek közül csak néhány fontosabbat emelek ki. A képek kétdimenziós (elő)feldolgozása amiatt kap fontos szerepet, mert lehetséges, hogy az input adatként szolgáló képek rossz minőségűek. Fontos tehát a képeken jelentkező hibák javítása, enyhítése, továbbá az, hogy különféle képtranzformációkkal bizonyos részeket kiemelhessünk a képből, amely a későbbiekben segítséget nyújt az orvosnak a helyes diagnózis felállításában.

Kontraszthibák javítása

Előfordulhat, hogy a képeken a kontraszt beállítás nem a célnak megfelelő. A 2.2.1-es ábrán egy kontraszthibás CT képet láthatunk. A kontraszthiba miatt az orvos nem látja jól a képen levő információt, ennek következtében téves diagnózist is megállapíthat, emiatt ezeket a hibákat ki kell javítanunk. Ezt hisztogram alapú eljárásokkal tehetjük meg. A hisztogram alapú eljárásokat amiatt kedvelik a képfeldolgozásban, mivel matematikailag jól leírhatók.



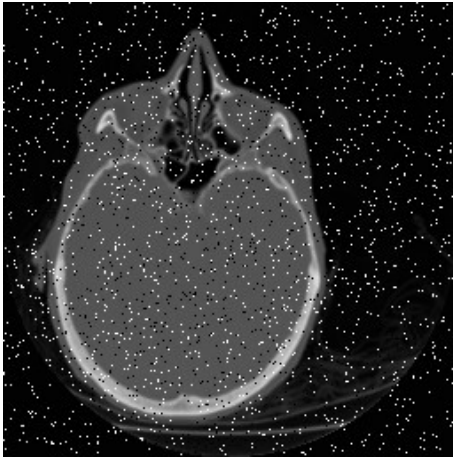
Ábra 2.2.1: Kontraszthibás kép



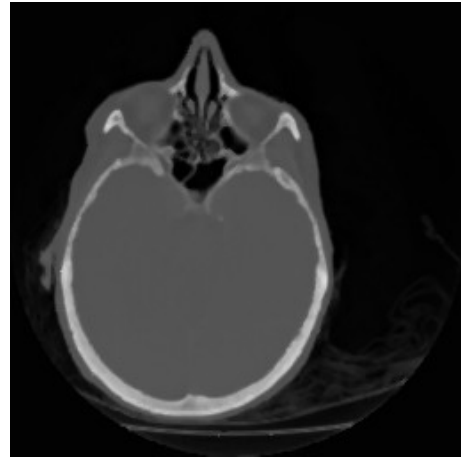
Ábra 2.2.2: Feljavított kép

Zajszűrés

A képeken a digitalizálás következtében zaj jelenhet meg, ez rontja a kép minőségét. A zaj sok problémát okozhat a képek szegmentálásánál. A képen található zaj csökkentésére több megoldás létezik. Különböző típusú zajok csökkentésére különböző megoldások léteznek. A 2.2.3-as ábrán egy intenzitás zajos koponya CT képet láthatunk. Akkor beszélünk intenzitás zajról, ha a képen a kiugró intenzitásértékű pixel-ek a fehér vagy fekete színhez közeliak. A 2.2.4-es ábrán a zajcsökkentett kép látható.



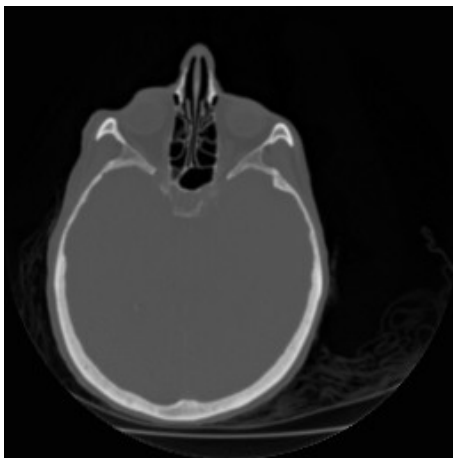
Ábra 2.2.3: Intenzitás zajos kép



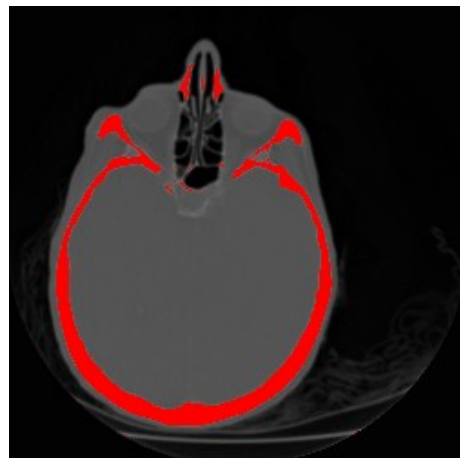
Ábra 2.2.4: Zajcsökkentett kép

Szegmentálás

A szegmentálás az a folyamat, melynek során a képen található adatokat olyan struktúrákba képezzük le, amelyek egy speciális feladathoz szükségesek [3]. A szegmentálásra amiatt van szükség, mivel a képen nem minden adatot szeretnénk vizualizálni (pl.: háttér). Többféle szegmentálási eljárás létezik. Léteznek threshold alapú, régió alapú, éldetektálás alapú és modell alapú eljárások.



Ábra 2.2.5: Szegmentálás előtti kép



Ábra 2.2.6 Szegmentálás utáni kép

A 2.2.5-ös ábrán a szegmentálás előtti, a 2.2.6-os ábrán pedig egy threshold alapú szegmentálás utáni CT kép látható. A képen a csont került külön struktúrába. A thresholding eljárás a 3.2.3-as fejezetben kerül bemutatásra.

2.2.2. Háromdimenziós képalkotás

Adódnak olyan problémák (pl.: sebészi tervezés), amelyek megoldásához a kétdimenziós képsorozatot vagy annak egy részét három dimenzióban kell megjeleníteni. A probléma nehézsége abból adódik, hogy a képeken nehéz elkülöníteni a három dimenzióban megjeleníteni kívánt részeket (pl.: csont), továbbá sok adatot kell egyszerre vizualizálnunk. A kétdimenziós képsorozat háromdimenziós formában történő megjelenítésére többféle megoldás fejlődött ki. Ezek közül ismertetek most néhányat.

Poligon alapú megjelenítés (Polygon rendering)

Ezen eljárás során a háromdimenziós grafikai objektumokat poligonokból (legtöbb esetben háromszögekből) építjük fel. Ez az eljárás nagyon elterjedt, mivel a grafikus gyorsítókártyák is ezt támogatják, ennek következtében gyors. Hátránya viszont, hogy nem tudjuk vele megjeleníteni az összes rendelkezésre álló adatot, csak a modellezni kívánt objektum határoló felületét, továbbá ha valamilyen transzformációt szeretnénk végrehajtani a modellen (pl.: vágás, nyújtás), akkor a modellt újra kell számolni, és ez sok időt vehet igénybe.

Térfogat vizualizáció (Volume rendering)

Ez az eljárás voxel-ekből (volumetric pixel) építi fel a háromdimenziós grafikai objektumokat. Az, hogy mi a voxel, nehéz definiálni. A szakirodalom sokszor háromdimenziós pixel-ként említi. Voxel-eket többféleképpen is reprezentálhatjuk. Megtehetjük gömbökkel, kockákkal, téglatestekkel de más testeket is használhatunk, például hatszög alapú hasábokat [16]. A voxel-eket reprezentálhatjuk négyzetekkel is, de itt figyelniük kell arra, hogy azok mindig a kamera felé nézzenek. A megközelítés rendkívül gyors megjelenítést tesz lehetővé kis adathalmazon.

A térfogat vizualizáció előnye a poligon alapú vizualizációval szemben, hogy az összes rendelkezésre álló adatot meg tudjuk jeleníteni vele, és transzformációnál a modellt nem kell újraszámolni. Hátránya viszont a nagy memóriaigénye, emiatt eddig csak kevés helyen alkalmazták. Alkalmazzák az orvosi képfeldolgozásban, geológiai adatok megjelenítéséhez, és néhány számítógépes játékban is (pl.: Delta Force, Crysis) a terep modellezésére.

Voxel-ekkel megadott grafikai objektumokból elő lehet állítani azok poligonhálós változatát. Ez úgy lehetséges, hogy azonosítjuk a háromdimenziós térfogat szintfelületeit. Ezen probléma megoldásához az egyik legnépszerűbb algoritmus a **masírozó kockák** [17], amely először eldönti minden voxel-re, hogy a voxel a grafikai objektum belsejében található vagy pedig kívül azon, és ha két szomszédos voxel eltérő típusú akkor közöttük határnak kell lennie [5], de léteznek más algoritmusok is mint pl.: a BPA (Ball-Pivoting Algorithm) [9] ami egy labda végiggörgetésének szimulációja segítségével oldja meg a problémát, vagy a Delaunay háromszögesítés.



Ábra 2.2.7: Diffúz gömb poligon alapú és voxel alapú alakban.

A 2.2.7-es ábra bal oldalán a poligon alapú gömböt látjuk, a jobb oldalon a voxel alapút. A képen a voxel-ek kockákból állnak. Az alacsony voxel felbontás miatt most könnyű különbséget tenni a voxel és poligon alapú gömbök között, de nagy felbontás esetén már nehéz észrevenni.

Sugárkövetés (Raytracing)

A sugárkövetés algoritmus segítségével 2D-s képeket állíthatunk elő 3D-s grafikus objektumokról úgy, hogy a nézőpontból vagy fényforrásokból induló fény útját követjük, és figyeljük, hogy a fény a 3D-s térben milyen grafikus objektummal ütközik. Az ütközési pontokból további sugarakat indítunk útjára, majd ezen sugarak ütközési pontjaiból újabb és újabb sugarakat és így tovább egy bizonyos lépésszámig. Az első ütközési pontból indított sugarak árnyék, reflexió és refrakciós sugarak melyeknek segítségével képesek vagyunk fotorealistikus árnyékok, fénytörés és visszaverődés modellezésére. Ha a nézőpontból indítjuk a fénysugarakat akkor backward raytracing-ről, ha a sugarak a fényforrásból indulnak akkor forward raytracing-ről beszélünk. A forward raytracing számításigényesebb, mivel sok olyan sugarat is követni kell, amik soha nem érkeznek meg a nézőpontba.

A sugárkövetés algoritmusnak azt a változatát melynek során a sugarak útját csak az első ütközési pontig követjük raycastingnek nevezzük. A raycasting algoritmus nem alkalmas fotorealistikus árnyékok, reflexió és refrakció modellezésére, viszont kevésbé számításigényes.

A raytracing és raycasting előnye, hogy képes optikai kölcsönhatások modellezésére hátrányuk viszont, hogy számításigényesek, emiatt egy-egy kép kiszámítása akár órákig is eltarthat.

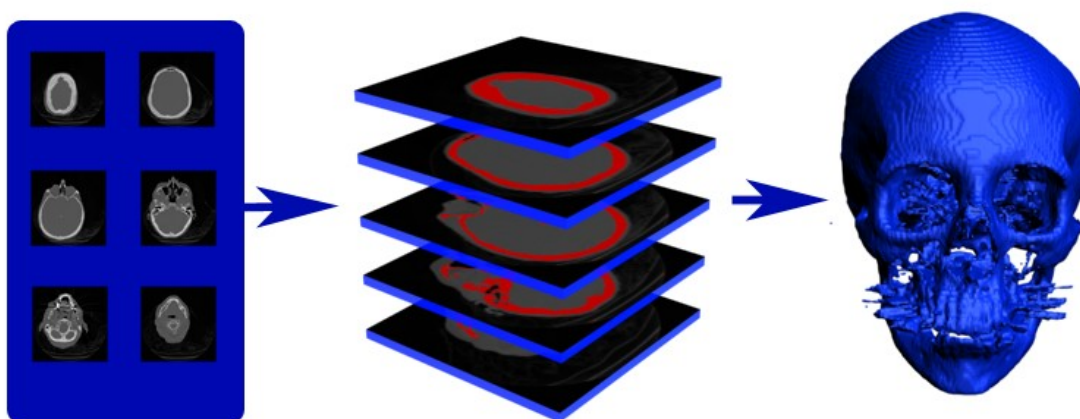
2.3. A szakterület jövője

A jövőt nem lehet megjósolni, de azt lehet látni, hogy merre halad ez a terület. Jelenleg néhány gigabyte adat vizualizálására képes a technológia, de a hardver eszközök fejlődése révén a közeljövőben valószínűleg több terabyte adat vizualizálására leszünk képesek. A hardver eszközök fejlődése maga után fogja vonni a térfogat vizualizáció fejlődését, emiatt várhatóan szélesebb körben fogják alkalmazni, ezáltal egyre nagyobb felbontású háromdimenziós modelleket tudunk majd vizualizálni.

Várhatóan el fognak terjedni az intelligens képalkotó berendezések, melyek a vizsgálati helyzethez való alkalmazkodásuk miatt kevés emberi beavatkozást igényelnek [4][20], ezután már nem kell sok idő ahhoz, hogy a folyamat emberi beavatkozás nélkül történjen.

3. A MedVis térfogat-vizualizációs alkalmazásban használt módszerek

Kutató munkám során céлом egy olyan alkalmazás elkészítése volt, amely képes a CT és MRI képsorozatok kontraszthibáinak javítására, zajcsökkentésére, kétdimenziós rekonstrukciójára, valamint lehetővé teszi a képi adatokból a háromdimenziós modell előállítását mind poligon, mind térfogat vizualizáció segítségével. Munkám során több különböző háromdimenziós megjelenítési eljárás megvalósítása mellett döntöttem, mert a szakirodalmi álláspontok nem egyértelműek a téren, hogy melyik a legjobb megoldás. Ebben a fejezetben arról olvashatnak, hogy milyen eszközöket, megoldásokat, ötleteket használtam fel az alkalmazásom elkészítéséhez, miközben végig követjük a képek útját a beolvasástól egészen a 3D-s modell elkészüléséig.



Ábra 3.1.1: Képek feldolgozási folyamata

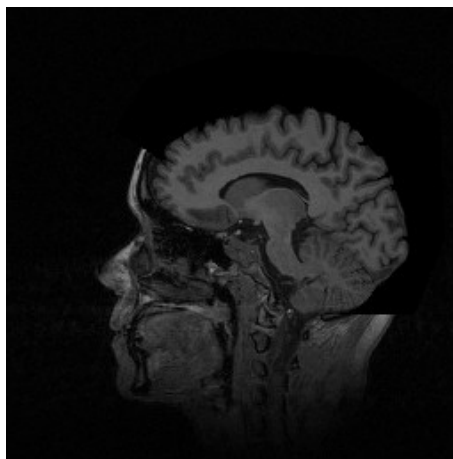
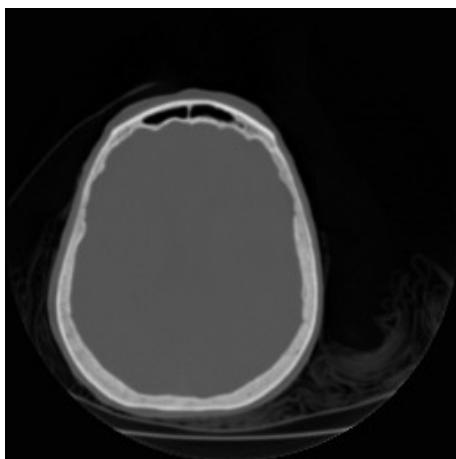
A 3.1.1-es ábrán a képek feldolgozásának folyamata látható röviden összefoglalva. Először végre kell hajtanunk az előfeldolgozási műveleteket. Ide tartozik a kontraszt feljavítás, a zajcsökkentés, és a szegmentálás. Ezután előállíthatunk egy háromdimenziós tömböt (ez egy köztes tároló, ennek alapján állapítjuk meg, hogy a pontok hol helyezkednek el a háromdimenziós térben), majd ezután térfogat vizualizáció vagy poligon alapú megjelenítés segítségével előállíthatjuk a háromdimenziós modellt.

Ha egyből a kiválasztott megjelenítéshez szükséges formában adnám át az adatokat, a tömböt „átugorva”, akkor gyorsabb lenne az alkalmazás, de ezt, mivel több megjelenítési módszer kerül megvalósításra az alkalmazásban, körülményes lenne megoldani. A tömbből mindegyik megjelenítési módszerhez könnyen előállíthatók a szükséges adatok. A tömb elhagyása akkor lenne célravezető, ha csak egyféle megjelenítési módszer került volna implementálásra az alkalmazásban.

3.1. Felhasznált adathalmazok

Néhány külső forrástól engedélyt kértem arra, hogy használhassam a CT és MRI képsorozataikat ehhez a munkához, ezáltal többféle minőségű és felbontású különböző eszközökkel készült képsorozat került a birtokomba. A forrásokról további információk a függelékben találhatók.

A legkisebb méretű rendelkezésre álló képanyag 200*160 pixel, a legnagyobb pedig 512*512 pixel-es felbontású. Mindegyik képsorozat szürkeárnyaltos. Előnyösebb lenne nagyobb felbontású képekkel dolgozni, mivel akkor a háromdimenziós megjelenítésnél szebb eredményt érhetnénk el. Képsorozatanként különböző távolságra vették a mintákat, emiatt az X:Y:Z arányaik is különbözőek, ezt a későbbi feldolgozásnál több helyen is figyelembe kell majd vennünk. Több képet, és az adathalmazok részletes leírását mely információkat tartalmaz a felbontásukról, arányaikról, származási helyükről és történetükről függelékben találunk.



Ábra 3.2.1: Bal oldalt CT, jobb oldalt MRI által előállított kép.

3.2. Kétdimenziós feldolgozás

Ebben a fejezetben olyan módszereket mutatok be, amelyek segítségével javíthatunk a digitalizált kép minőségén, előkészíthetjük a háromdimenziós megjelenítést. Célom az volt, hogy kiválasszam az irodalomban ismert legfontosabb algoritmusokat, és megvalósítsam azokat. A CT és MRI képeken megjelenhet zaj, anomáliák alakulhatnak ki a fémimplantátumok, fogtömések miatt, valamint a páciens lélegzése és mozgása is rontja a képek minőségét, és ha nem javítjuk ezeket a hibákat, az a későbbiekben rontani fogja a háromdimenziós grafikus objektum minőségét [3].

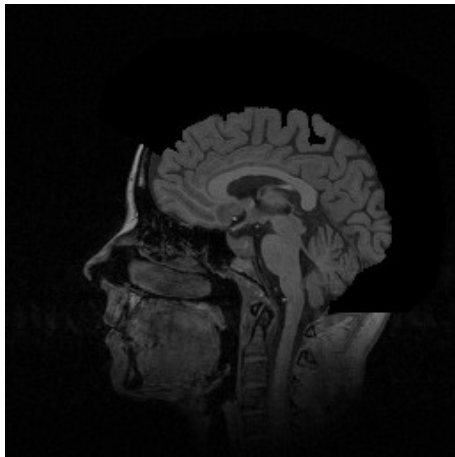
3.2.1. Kontraszt javítása

Hisztogram fogalma

A digitális kép különböző intenzitásértékű pixel-ekből áll. A hisztogram segítségével összeszámolhatjuk és függvényszerűen ábrázolhatjuk az egyes szürkeárnyalat szinteken lévő pixel-ek számát. A hisztogram egy olyan H függvény, amely a $[0, ..., k, ..., 255]$ intervallumon értelmezett pixelintenzitások eloszlását írja le:

$$H(k) = n, k$$

Ahol k a szürkeárnyalat értéke, n a k értékű pixel-ek száma a vizsgált képen. A hisztogramból megállapíthatjuk a kép telítettségét. Két fajta hisztogram számítási mód létezik: lineáris és kumulatív [6].



Ábra 3.2.1: Az 54. kép az MRI brain képsorozatból és lineáris hisztogramja

Hisztogram kiegyenlítés

Az ábrán látható, hogy a képen található legnagyobb intenzitásértékű pixel nem 255 intenzitásértékkel bír. Azt az eljárást, amikor a legsötétebb pixel intenzitásértékét 0-ra, a legvilágosabbat 255-re írjuk át, a közte található értékeket pedig egyenletesen eloszlatjuk a két érték között, hisztogram kiegyenlítésnek nevezzük. Ennek segítségével javíthatunk a kontraszthibás képeken [7]. Ahhoz, hogy ezt megtegyük, először elő kell állítanunk a teljes képsorozatról készült hisztogramból a kumulatív hisztogramot.

3.2.1. Kontraszt javítása

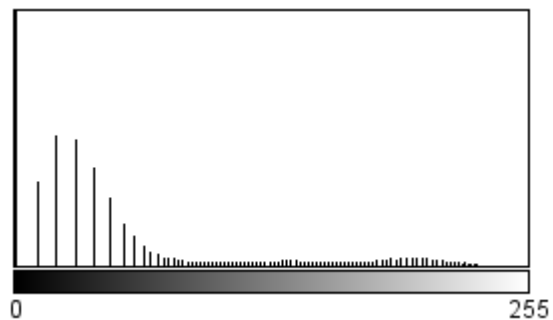
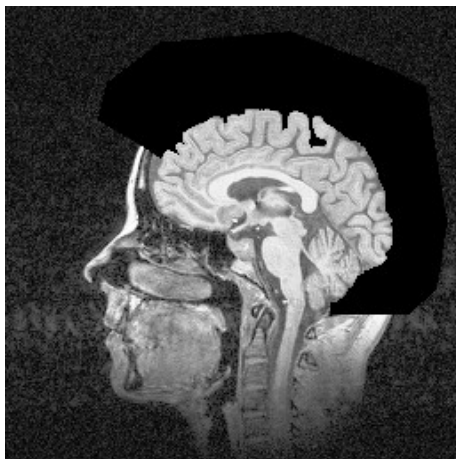
A kumulált hisztogramot a következő függvénnyel adhatjuk meg:

$$C(i) = \sum_{j=0}^i H(j) \quad \text{Ahol } 0 \leq i \leq 255, i \in N$$

A kumulatív hisztogramból a következő formula segítségével megkaphatjuk, hogy a j intenzitásértékű pixel-ekhez milyen új intenzitásértéket rendelünk.

$$h(j) = \text{Round}\left(\frac{C(j) - C_{\min}}{M * N - C_{\min}} * (L - 1)\right)$$

Ahol $M * N$ a kép pixel-einek száma, L pedig azt jelöli, hogy hány intenzitásérték van. (legtöbb esetben 255)



Ábra 3.2.2: Az MRI brain képsorozat 54. képének kontrasztjavított változata és lineáris hisztogramja

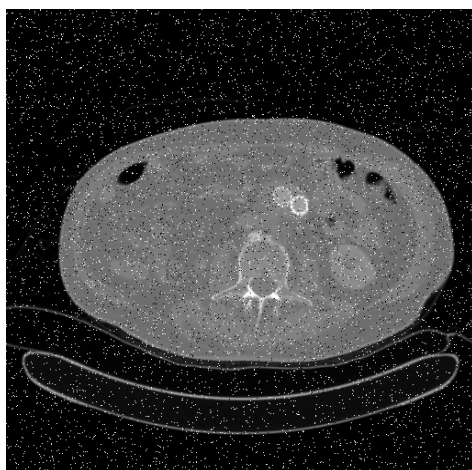
A 3.2.2-es ábrán azt vehetjük észre, hogy a képen olyan részletek is megjelentek, amik a kontrasztjavítás előtt nem voltak láthatók. Az agy körül látható fekete folt amiatt van ott, mivel a képről digitálisan eltávolították a koponyát [2][13][18].

3.2.2. Zajcsökkentés

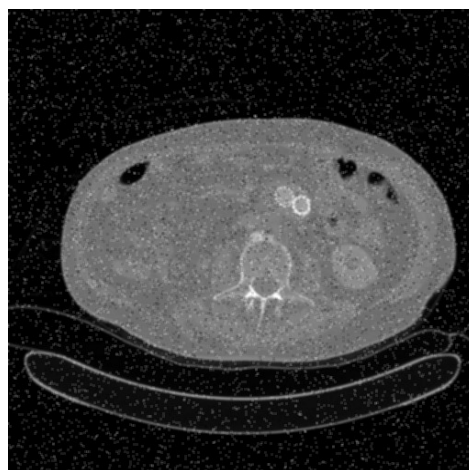
A képeken a digitalizálás miatt zaj jelenhet meg, ami rontja a képek minőségét. Ebben a fejezetben a zaj csökkentésére, azon belül is az intenzitás zaj csökkentésére szolgáló megoldások kerülnek ismertetésre.

Mean filter

A Mean filter egyike a konvolúciós maszkoláson alapuló zajszűrő eljárásoknak [12]. A maszkot $((2K+1)*(2L+1))$ -es mátrixként szokás megadni, ahol a mátrix elemei az $1/((2k+1)*(2l+1))$ értéket veszik fel. A Java nyelvben van beépített osztály a konvolúciós maszkolás megvalósítására.



a)



b)

$$\frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

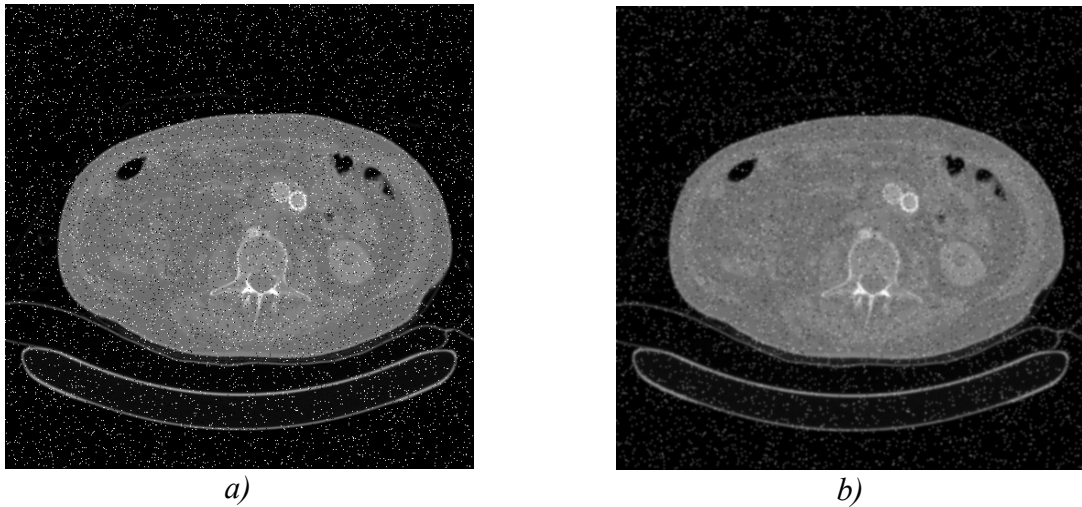
c)

Ábra 3.2.3: a: intenzitás zajos kép, b: Mean filterrel megszürt kép, c: 3x3 as maszk

Gaussian Filter

A Gaussian filter is konvolúciós maszkoláson alapuló zajszűrő eljárás. Abban különbözik a Mean filtertől, hogy a mátrix elemei súlyozást kapnak. A mátrix átlóinak metszéspontjában található elem kapja a legnagyobb súlyozást. Ahogy távolodunk ettől a ponttól, a súlyozás mértéke csökken.

A Gaussian filter előnye a Mean filterrel szemben, hogy a kép kevésbé lesz elmosódott, mivel súlyoztuk a mátrix elemeit [8]. Hátránya viszont, hogy a kisebb elmosás miatt kevésbé szűri a zajt. A 3.2.4-es ábra b) jelű képén láthatjuk a Gaussian filterrel megszürt képet.



$$\frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

c)

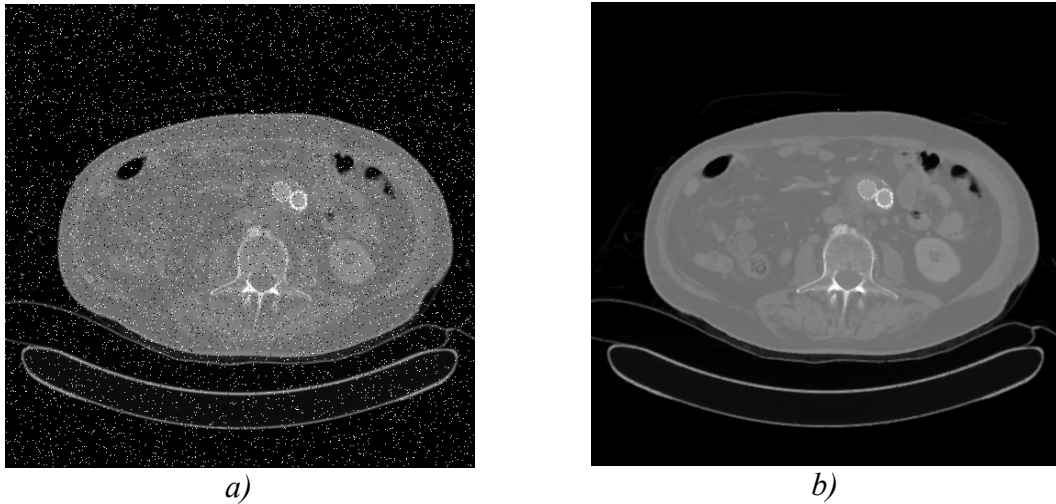
Ábra 3.2.4: a: intenzitás zajos kép, b: Gaussian filterrel megszürt kép, c: 3x3 as maszk

Median filter

A Median filter is maszkoláson alapul, de nem tartozik a konvolúciós szűrők közé. A környező pixel-eket intenzitásértékük szerint sorba rendezzük, és kiválasztjuk a sorból a középső elemet (mediánt). Az alkalmazásomban 3x3 as maszkot használtam a szűréshez. A rendezéskor a kiugró intenzitásértékű (zaj) pixel-ek a sor szélére kerülnek, ezáltal nagyon jól alkalmas az intenzitás zaj csökkentésére.

Ennek a filternek egy másik változata is létezik, ez pseudo-median filter néven ismert, melynek több fajtája létezik. Vezetjük külön külön a sorok mediánját, majd ezen értékek mediánját számoljuk, egy másik megoldás, hogy hasonlóképpen járunk el az oszlopokkal [19].

A Median filter előnye az előzőekben ismertetett szűrőkkel szemben, hogy a képen nem jelennek meg új intenzitásértékek, továbbá a konvolúciós maszkoláson alapuló szűrések csak nagyobb területre oszlatják szét a zajt, míg a Median filter szinte teljesen eltünteti. Figyeljük meg a 3.2.5-ös ábrát [10].



Ábra 3.2.5: a: intenzitás zajos kép, b: Median filterrel megszürt kép

Az ábrákon látszik, hogy a Mean és a Gaussian filterrel hasonló hatást értünk el, míg a Median filter szinte teljesen eltüntette a zajt a képről, és az éleket sem mosta el olyan mértékben mint a Mean vagy a Gaussian filter.

3.2.3. Szegmentálás

Miután megtörtént az előfeldolgozás, a képeket szegmentálhatjuk. A szegmentálás az a folyamat, melynek során a digitális kép pixel-eit olyan struktúrákba (pixel halamzokba) képezzük le, amelyek egy speciális feladathoz szükségesek [3]. A szegmentálást alkalmazzák az orvosi képfeldolgozásban, gépi látásban, műholdképek feldolgozásánál, ujjlenyomat felismerésnél és még sok más területen.

Szegmentálásra amiatt van szükség, mivel a képeken található olyan adatok, amiket nem szeretnénk megjeleníteni (ilyen például a fekete háttér), továbbá az sem lenne célravezető, ha a lágy szöveteket, és a csontot ugyanolyan színnel jelenítenénk meg, emiatt ezeket is külön struktúrába rendezzük. Többféle szegmentálási eljárás létezik. A következőkben a thresholding eljárást fogom ismertetni.

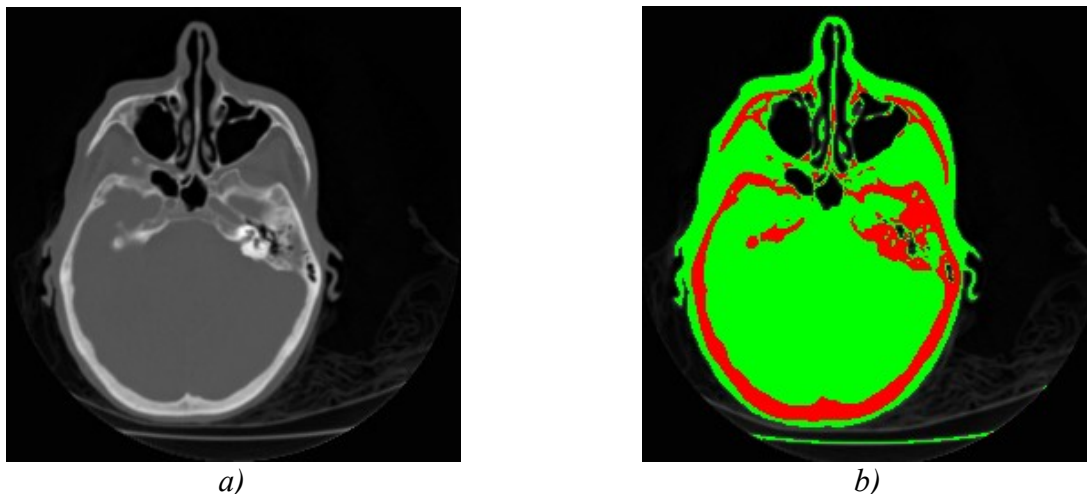
Thresholding (küszöbértékekkel történő szegmentálás)

Az 3.2.6-os ábra a) képén látható, hogy a csont, a lágy szövetek, és a háttér különböző intenzitásértékű pixel-ekből állnak. A háttér alacsony (fekete színhez közeli), a lágy szövetek közepes, míg a csont magas intenzitásértékű pixel-eket tartalmaz.

3.2.3. Szegmentálás

A thresholding eljárás lényege, hogy megadunk két küszöbértéket, és azokat pixel-eket, amelyek intenzitásértékei ezen két küszöbérték között találhatók, egy struktúrába soroljuk. A küszöbértékek megadása történhet manuálisan, és algoritmus által. Az alkalmazásomban a manuális megadási módot választottam. Ez nagyobb szabadságot ad a felhasználónak, mivel maga választhatja meg a küszöbértékeket. Ha pl. a 3.2.6 a) képen a csontot és a lágy szöveteket szeretném a képről elkülöníteni, akkor két magasabb intenzitásértéket adok meg küszöbértéknek, ezáltal a háttér nem kerül kiválasztásra. A 3.2.6-os ábra b) képen a csont (piros szín) és a lágy szövetek (zöld szín) kerültek külön struktúrába. A szegmentálást elvégzem képenként a teljes képsorozaton a megadott küszöbértékekkel, és az adatokat egy háromdimenziós byte tömbben tárolom. Azok a byte-ok ahol nincs „semmi” 0 értéket kapnak, ahol piros színű a kép 1-set, ahol zöld 2-est, ahol kék 3-ast.

Ezt az eljárást amiatt választottam, mivel könnyű implementálni, és a felhasználónak csak a két intenzitásértéket kell megadnia, és már látja is a végeredményt, emiatt a felhasználó dolgát is megkönnyíti.

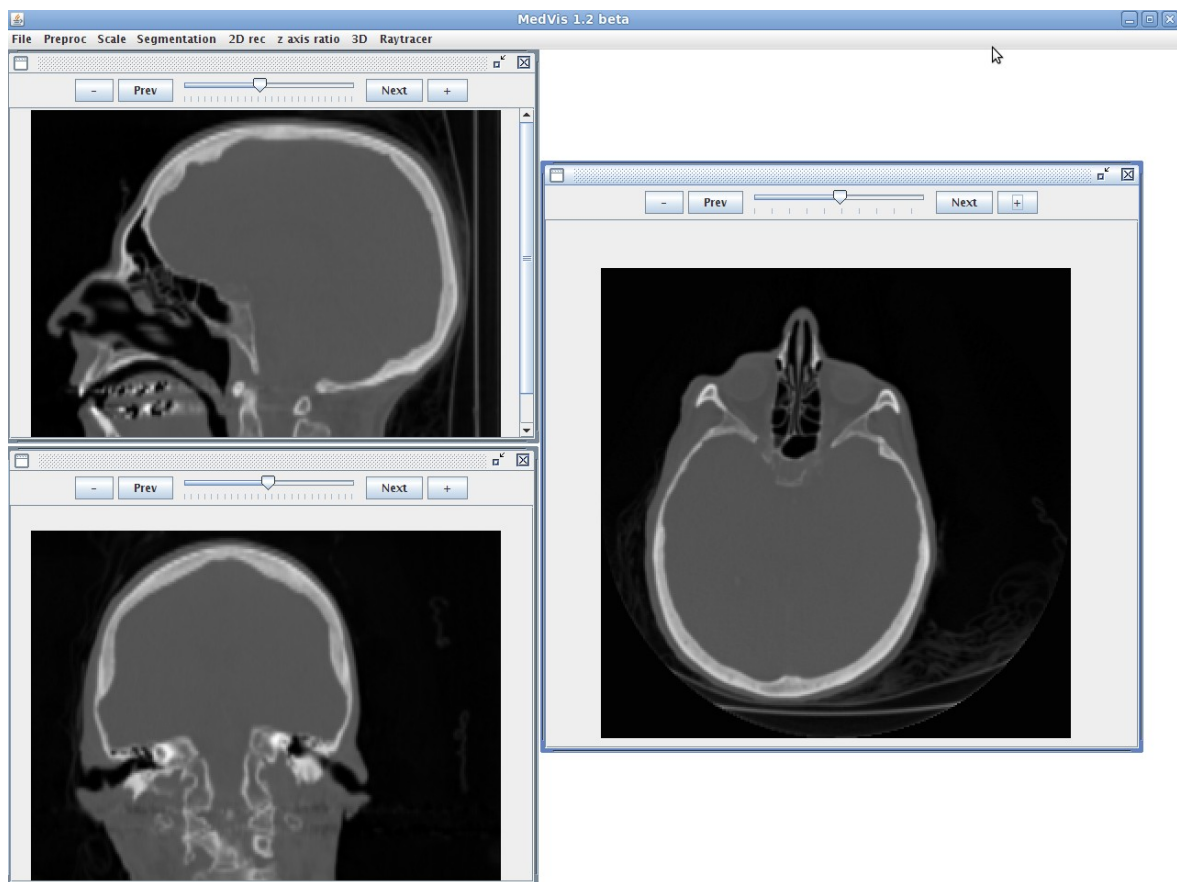


Ábra 3.2.6: a) szegmentálás előtti, b)szegmentálás utáni kép

3.2.4 . Multi-planáris rekonstrukció

A CT és MRI képsorozatok egy adott tengely mentén vannak előállítva. Adódhatnak olyan problémák melyek hatékony megoldásához szükség van a másik két tengely menti képsorozatra (3.2.7-es ábra). A több tengely menti képsorozat előállítását multi-planáris rekonstrukciónak (MPR) [3] nevezik. A pixel-ek újracsoportosításával ez a probléma gyorsan és egyszerűen megoldható.

3.2.4 . Multi-planáris rekonstrukció



Ábra 3.2.7: Egy képernyőmentés az elkészült szoftverből amelyen mind a három tengely menti képsorozat szerepel

3.3. Háromdimenziós képalkotás

Több olyan feladat létezik, amelyek megoldásához a kétdimenziós képsorozat vagy annak egy részét három dimenzióban kell megjeleníteni. Háromdimenziós modellekre van szükség a sebészi tervezésnél, sebészeti szimulátor alkalmazásokban, interaktív anatómia atlaszokban. Ezen modellek nagy segítséget jelentenek az olyan orvosok számára, akik a képsorozatból nehezen tudják maguk elé képzelni a háromdimenziós ábrát. Ebben a fejezetben azon módszerek kerülnek ismertetésre melyeket az alkalmazás fejlesztésénél felhasználtam. Először a térfogat vizualizáció, azután a poligon alapú megjelenítés kerül bemutatásra.

OpenGL (Open Graphics Library)

Az alkalmazásomban a háromdimenziós képalkotáshoz az OpenGL-t használtam. Az OpenGL a The OpenGL Graphics System: A Specification című szabványokban meghatározott grafikus API. Az OpenGL programok ennek a szabványnak a megvalósításait használják kétdimenziós és háromdimenziós geometriai adatok és képek megjelenítésére. A hardveresen nehezen megvalósítható magasszintű szolgáltatások nem részei az OpenGL-nek, továbbá nem tartalmaz támogatást felhasználói felületi szolgáltatások kezeléséhez [14]. Java-ban a JOGL adja az elérést az OpenGL szolgáltatásaihoz, továbbá hardver specifikus kiegészítéseket is tartalmaz.

3.3.1. Térfogat vizualizáció (Volume rendering)

A térfogat vizualizáció fogalma a 2.2.2-es fejezetben már ismertetésre került. Most azok a módszerek kerülnek bemutatásra, amik helyet kaptak az alkalmazásomban. Ezek a volume slicing, valamint a voxel-ek kétféle reprezentációjának megvalósítása.

3.3.1.1. Térfogat metszés (Volume slicing)

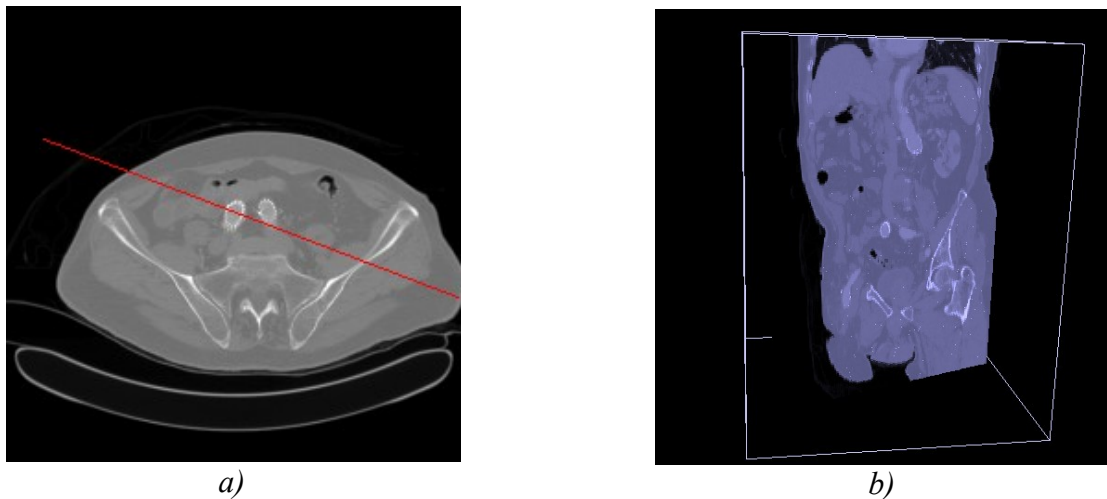
A 3.2.4-es fejezetben bemutatásra került, hogyan állíthatunk elő egy adott tengely menti képsorozatból a másik két tengely menti képsorozatot. A belső szervek nem kötöttek a koordinátatengelyekhez, ezáltal egy nem tengely menti metszet több információt tartalmazhat számunkra, mint egy tengely menti. Ennek előállítását volume slicing-nek nevezik, de a szakirodalom sík alapú térfogat vizualizáció (plane-based volume rendering) néven is említi.

A szakirodalom ennek megvalósításához olyan eljárást használ amely a voxel-eket egy felhasználó által definiált síkkal metszi, ezáltal számolja ki a metszetet. Ez eléggé számításigényes, különösen ha több tízmillió vagy netán százmillió voxel-nél kell ellenőriznünk, hogy van-e metszéspontja a síkkal [3].

Ennek a problémának a megoldására egy egyszerű és gyors megoldást találtam ki, ahol nem szükséges metszéspontot számítani a síkkal. Nem a voxel-ekkel dolgoztam, hanem visszavezettem a problémát kétdimenzióba. Mint az a 3.2.7-es ábrán látható fejlesztettem egy több kép kezelésére alkalmas Java komponenst. Ezen a komponensen (bármelyik

3.3.1. Térfogat vizualizáció (Volume rendering)

tengely menti képsorozaton) a felhasználó egy egyenes rajzolásával definiálhatja a síkot. Egy vektorban tárolom azt, hogy a képen mely pixel-eket fedte le az egyenes (3.3.1-es ábra a) kép), ezután a képsorozat minden képéről kiválasztom ezeket a pixel-eket. A kiválasztott pixel-ek kétdimenziós koordinátaíhoz hozzárendelem a harmadik koordinátát, és előállítom a voxel-eket. Ez a megoldás sokkal gyorsabb és egyszerűbb mint sík és voxel metszéspontját számolni.



Ábra 3.3.1: a) Egy kép a stent dataset-ből, b) Térfogat metszet (volume slice)

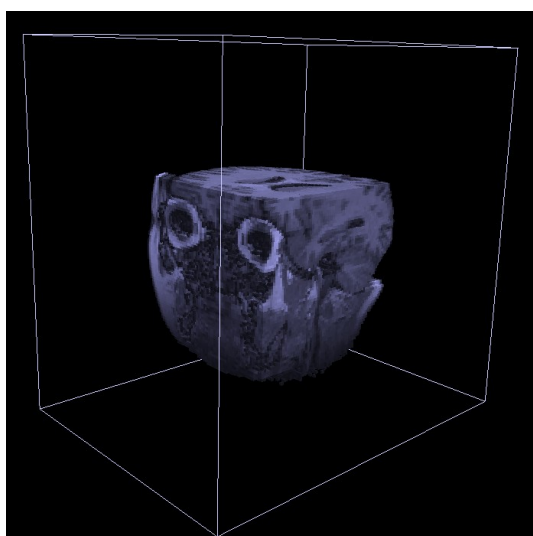
3.3.1.2. Voxel-ek reprezentációja téglatestekkel

Miután a 3.2.3 fejezetben leírt módon szegmentálási eljárást végrehajtottam a képsorozaton, és az adatok bekerültek a byte tömbbe, megkezdődhet a voxel-ek előállítása. A voxel-ek téglatesttel történő reprezentálására külön osztályt hoztam létre, amely gyorsan elő tudja állítani és tárolni a voxel megjelenítéséhez szükséges adatokat.

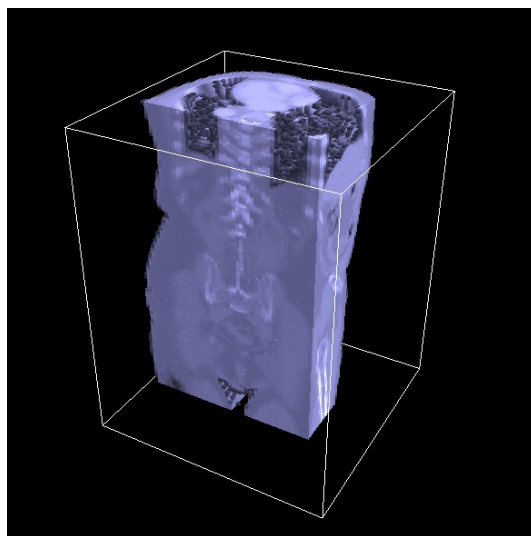
A voxel osztály

Ennek az osztálynak az elkészítésével az volt a célom, hogy a byte tömb adataiból kevés számításal és gyorsan voxel-eket tudjak előállítani. Az osztálynak két konstruktora van, mivel ennél a megoldásnál kétfajta megjelenítést alkalmaztam. Az egyik konstruktor nem ad át szint a voxel-nek, ekkor kívülről kapja meg a szint a megjelenítésnél (3.3.2-es ábra c)), a másik esetben pedig a voxel a képsorozatról annak a pixel-nek a színét kapja meg (3.3.2-es ábra a) és b) képek), amit reprezentál. Az objektum mindkét konstruktorának hívása esetén megkapja a voxel geometriai középpontját, és a 3.1-es fejezetben említett harmadik tengely menti arányt, ezáltal ki tudjuk számítani a voxel csúcspontjait. Az osztály tartalmaz kirajzoló metódust, ezáltal a voxel-ek kirajzolása szolgáló Java kód átláthatóbbá válik.

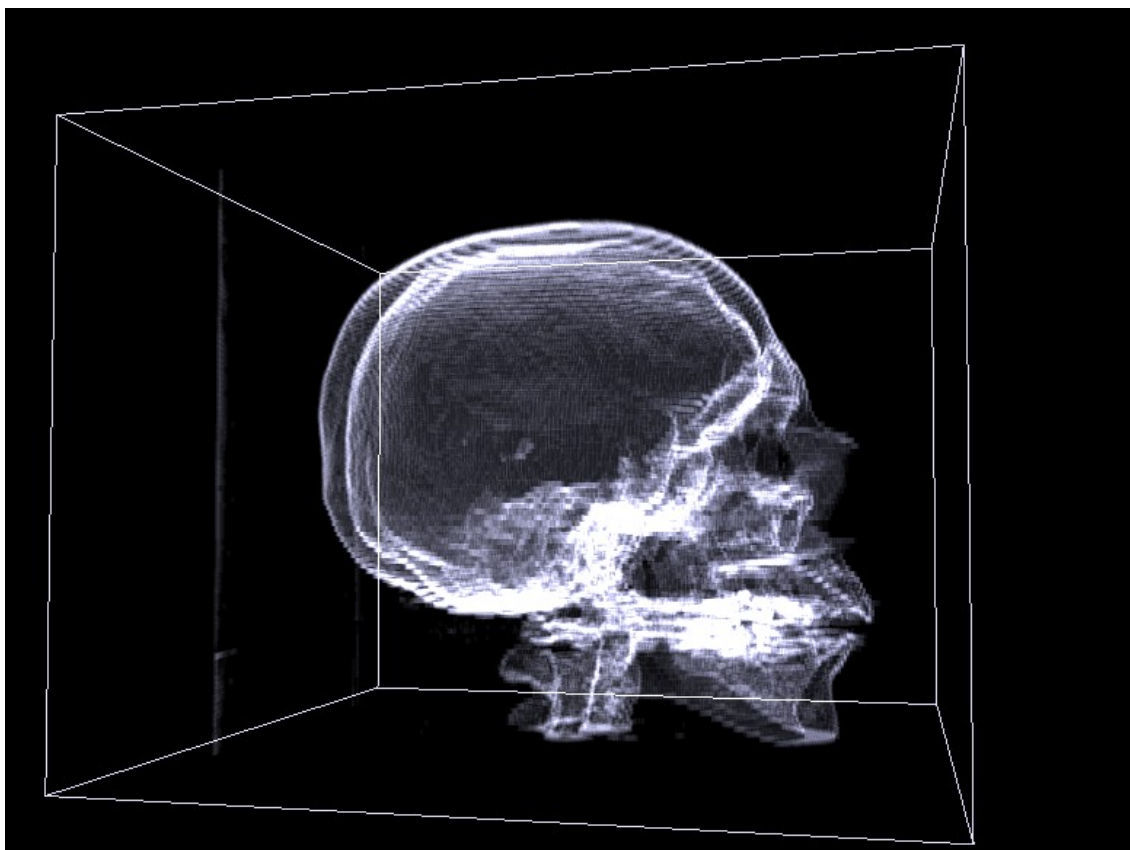
3.3.1. Térfogat vizualizáció (Volume rendering)



a)



b)



c)

Ábra 3.3.2: Háromdimenziós rekonstrukciók a) MRI brain dataset-ről, b) Stent dataset-ről, c) CT Head dataset-ről.

3.3.1. Térfogat vizualizáció (Volume rendering)

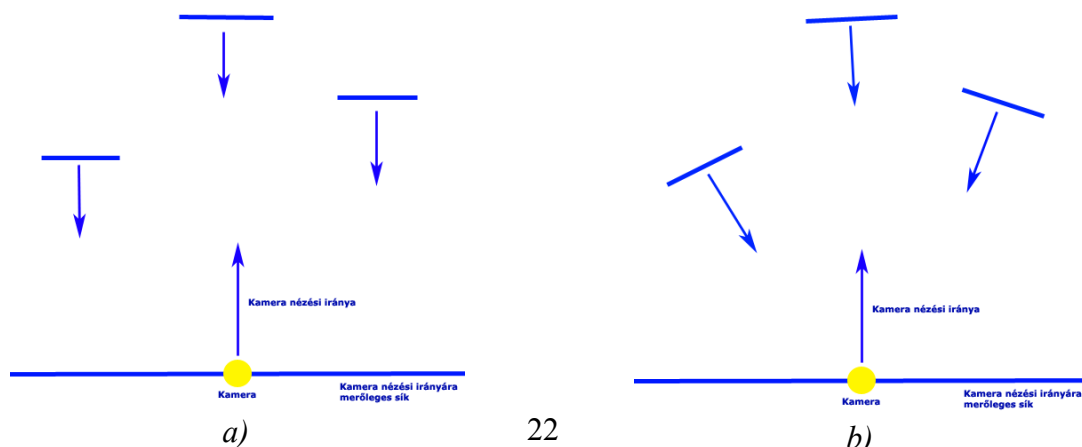
A 3.3.2-es ábra a) és b) képén látható, hogy a modellek tengelyek menti síkokkal vannak vágva. Ezt az alkalmazásban valós időben tehetjük meg, mivel a térfogat vizualizáció használatával ezt a modellt újraszámolása nélkül teszi lehetővé. A c) képen a CT Head dataset-ből szegmentált koponya látható. Természetesen a modell tengelyek menti vágása az egyszínű rétegek esetén is elvégezhető. A vágás minden elkülönített rétegre külön alkalmazható, ezáltal minden rétegből kiválaszthatjuk azt a részt amit látni szeretnénk, a képsorozat újraszegmentálása nélkül. További a voxel-ek téglateszt reprezentációjával készült képeket a függelékben találunk.

A modell egyszínű rétegekkel történő megjelenítése esetén a felhasználónak lehetősége van a modelltől azon voxel-ek eltávolítására, melyen nem a határfelületen található, ekkor kevesebb voxel-t jelenítünk meg, emiatt gyorsul a megjelenítés. A felhasználónak lehetősége van továbbá a rétegekhez átlátszóság értékeket társítani. (3.3.2 ábra c) kép) Ez akkor hasznos, amikor a rétegek takarják egymást, erre vonatkozó képeket a függelékben találunk.

3.3.1.3. Voxel-ek reprezentálása point-sprite-okkal

A voxel-ek téglatesztokkal történő reprezentálásánál meg kellett határoznunk, és tárolnunk kellett a voxel-ek csúcspontjait, ezután téglalapokból állítottuk össze őket. A voxel-ek csúcspontjainak kiszámítása időigényes, tárolásuk pedig sok memóriát foglal, emiatt alkalmaztam a voxel-ek point-sprite-okkal történő megjelenítését. Ahhoz, hogy a point-sprite fogalmát tisztázzuk, először meg kell ismerkedni a billboarding fogalmával.

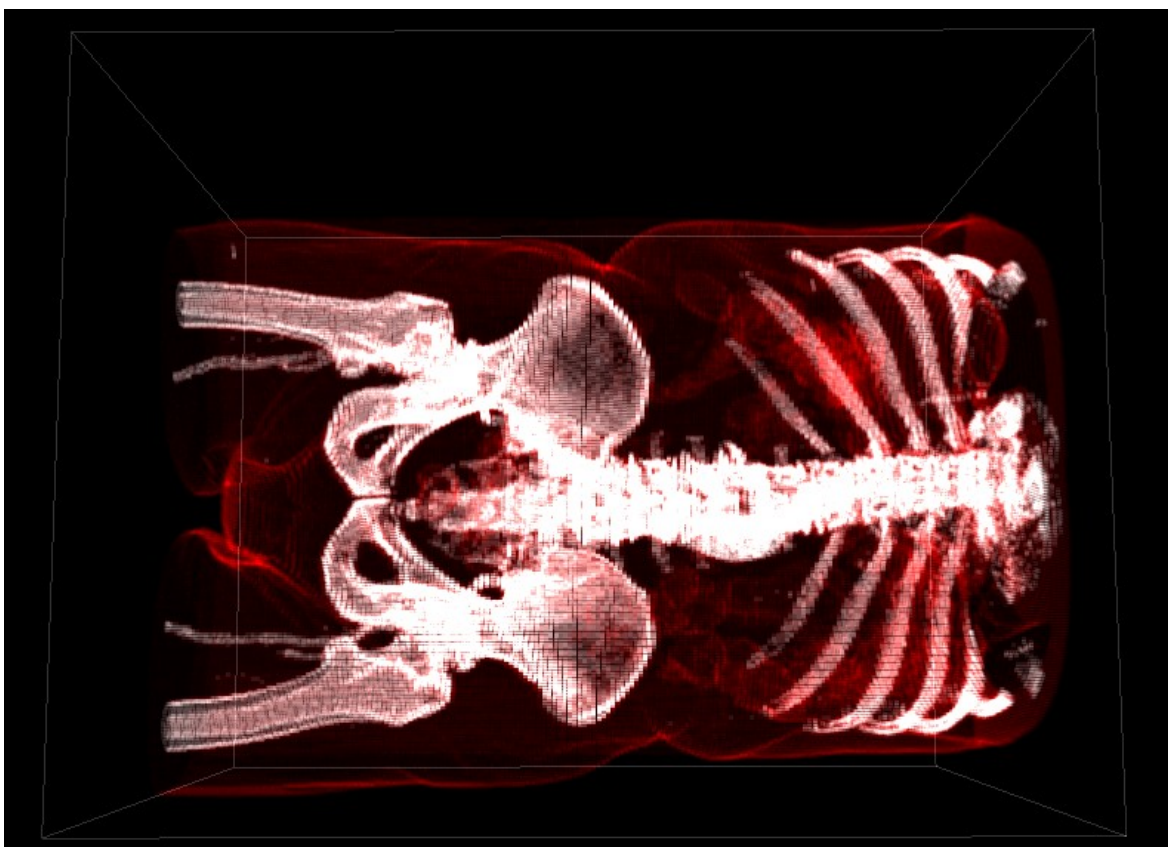
A billboarding eljárás során poligonok helyett kétdimenziós textúrákat (billboard-okat) jelenítünk meg, amelyek mindig egy adott pont (legtöbb esetben a kamera) felé néznek. (Ábra 3.3.3 b) kép) Egy kis trükközéssel az eljárás gyorsítható azáltal, hogy a billboard-okat a kamera nézési irányára merőleges sík felé fordítjuk és nem a kamera felé (Ábra 3.3.3 a) kép). Ez az eljárás sokkal gyorsabb mint a poligonok kirajzolása, mivel egy poligon megadásához legalább három térbeli pontot kell megadom, ezen eljárás esetén pedig csak egyet. Ez a technika nagyon népszerű olyan alkalmazásokban, ahol egyszerre sok adatot kell megjeleníteni ilyenek például a fizikai szimulációk. Az igazi háromdimenziós motorok kora előtt játékokban is sokszor alkalmazták (pl.: Doom). A mai játékokban is találkozni lehet vele. Legtöbbször füst, tűz, eső és hó modellezésére alkalmazzák.



3.3.1. Térfogat vizualizáció (Volume rendering)

Ábra 3.3.3: a) a billboard-ok a kamera nézési irányára merőleges sík felé néznek, b) a billboard-ok a kamera irányába néznek.

A point-sprite hardveresen (GPU) gyorsított billboard, ennek következtében gyors. (GPU: Graphics Processing Unit (grafikus processzor)) Hátránya viszont, hogy ha a point-sprite-októl távolodunk a kamerával vagy közelítünk hozzájuk, nem változik meg a méretük. Ennek következtében ha a modellhez közelítünk, egy idő után nem fogjuk összefüggő alakzatként látni, ha pedig távolodunk tőle elvesznek a részletek, emiatt a kamerától mért távolságuk függvényében változtatnunk kell a méretüket. A point-sprite-ok segítségével készült képek kicsit rosszabb minőségűek, mintha a voxel-eket téglatestekkel reprezentálnánk, de sebességbe messze meghaladja a téglatestes megjelenítést.



Ábra 3.3.4: A Stent dataset-ről point-sprite-ok segítségével készült háromdimenziós rekonstrukció

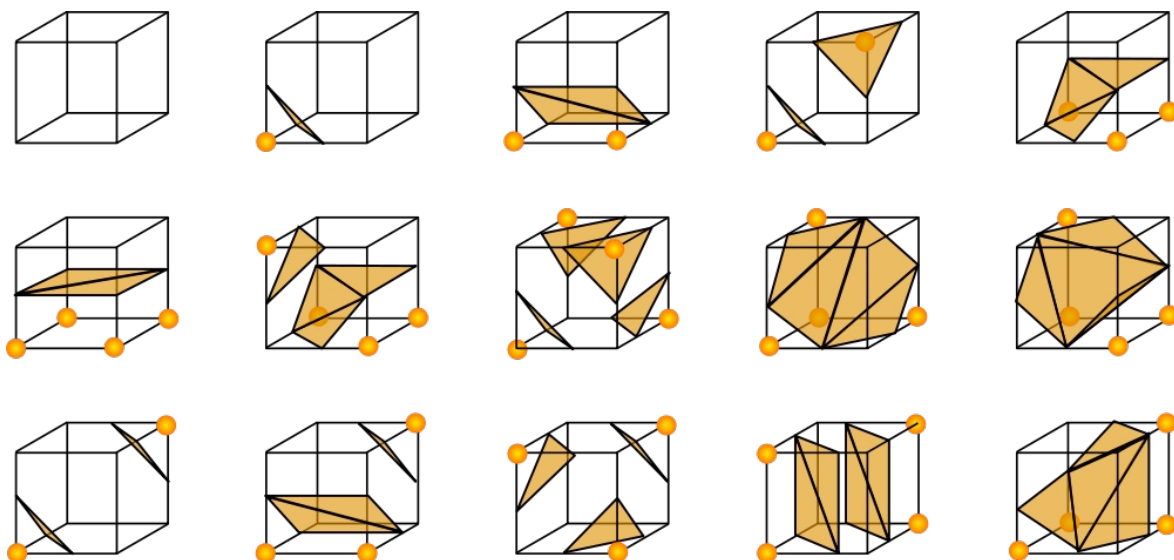
3.3.2 Poligon alapú megjelenítés

Voxel-ekkel megadott grafikai objektumokból elő tudjuk állítani azok poligonhálós változatát. Ez úgy lehetséges, hogy azonosítjuk a háromdimenziós térfogat szintfelületeit. Ezen probléma megoldásához az egyik legnépszerűbb algoritmus a masírozó kockák [17]. Az alkalmazásomban ennek az általam optimalizált változatát használtam.

3.3.2.1 Masírozó kockák

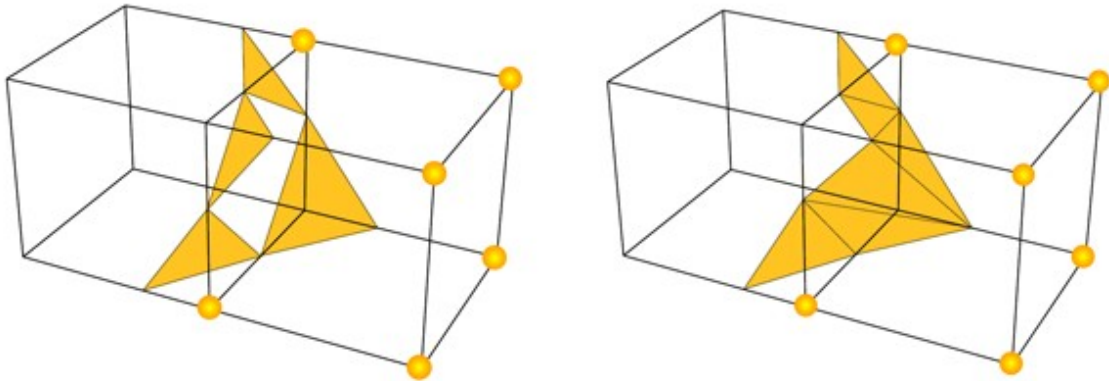
A masírozó kockák algoritmust 1987-ben publikálta Lorensen és Cline. Az algoritmus segítségével voxel-ekből poligonhálós modellt tudunk előállítani. Az algoritmus először eldönti minden voxel-re, hogy a voxel a grafikai objektum belsejében található vagy pedig kívül azon, ezáltal megkereshetjük a modell határfelületét [5]. Azt, hogy melyiket tekintse belső és melyiket külső voxel-nek, egy skalár érték segítségével adhatjuk meg. Jelen esetben a voxel-ekhez csak a színük tartozik, de vannak olyan problémák, ahol más skalár értéket is társítanak egy-egy voxel-hez, például geológiai adatok vizualizációjánál az adott helyen lévő kőzet sűrűsége. Ha például a nagyobb sűrűségű kőzeteket akarom „bekeríteni” poligonhálósval akkor nagyobb skalárértéket adok meg.

Az algoritmus kockák segítségével dönti el, hogy hova hozzon létre háromszöget. A teret kockákra osztja fel, és minden kocka csúcsaihoz hozzárendeli a kocka csúcsainál található voxel-eket. A voxel-ek vagy külső vagy belső voxel-ek (az előbb említett skalár értékek alapján eldönthető), ennek megfelelően összesen 256-féle kocka lehetséges. Ezek a következő 14 alapeset forgatásával és tükrözésével jönnek létre.



Ábra 3.3.5: Masírozó kockák alapesetei

Azonosítanunk kell a megfelelő kockát, azután a háromszög csúcspontjai a kocka élein pontosan hol helyezkednek el, lineáris interpolációval számolhatjuk. Az algoritmus előnye, hogy könnyen megérthető, hátránya viszont, hogy sok háromszöget generál, és vannak olyan esetek, amikor rossz végeredményt generál (Ábra 3.3.6) [11]. Payne és Toga kidolgozott egy a Masírozó kockákhoz hasonló algoritmust. Ők nem kockákkal, hanem tetraéderekkel dolgoztak [1]. A masírozó kockák segítségével előállított képet a függelékben találunk.



Ábra 3.3.6: Bal oldalt a hibás, jobb oldalt a jó háromszög elrendezés

Masírozó kockák optimalizálása

A 3.3.2.1-es fejezetben említettem, hogy a háromszögek csúcsait az algoritmus a voxel-ek skalár értékeinek megfelelően lineáris interpolációval számolja. Ez annyit tesz, hogy a nagyobb skalár értékkel rendelkező voxel-től a háromszöget távolabb helyezi el, ezáltal „kidomborítva” ott a poligonhálót, érzékeltetve, hogy ott egy nagyobb skalár értékkel rendelkező voxel van.

A jelenlegi problémánál csak a szint adhatnám meg az interpolációhoz, mint skalár értéket, mivel ezen kívül mást nem tárolunk, annak pedig nem sok értelme van, hogy nagyobb színintenzitás esetén az adott pontban jobban „kidomborítsa” a poligonhálót. Emiatt a lineáris interpolációt teljesen kiszedtem az algoritmusból, és az így számolt csúcspontok helyett a kocka élein előre definiált pontokkal adtam meg a háromszögeket. A pontokat a kocka éleinek felénél definiáltam. Emiatt az algoritmus gyorsabb lett, viszont több memóriát használ. Az algoritmus memória felhasználása csökkenthető. A kockáknak az őket körülvevő kockákkal vannak közös csúcspontjai, ezáltal ugyanazt a pontot többször is tároljuk. Ezt a problémát úgy oldottam meg, hogy először létrehoztam a csúcspontokat, ezután rendeltem őket a kockákhoz.

Normálvektor számítás

Egy felület adott pontban vett normálvektorán egy olyan vektort értünk, mely adott pontban merőleges a felületre. A normálvektorok meghatározása amiatt fontos, mivel az OpenGL ez alapján számítja ki a háromszög csúcspontjainak színét. Függvénnyel megadott felületek esetén ez a probléma megoldható differenciálszámítás segítségével. Jelen esetben a normálvektort csak becsülni tudjuk. Ehhez ismernem kell az adott pont szomszédságában levő pontokat, és ezen pontok helyvektorainak összeadásából kaphatjuk meg az adott ponthoz tartozó normálvektort.

3.4. A bemutatott módszerek összehasonlítása

Az előzőekben megismerhettük, hogyan állíthatunk elő pontfelhőből képeket térfogat vizualizáció és poligon alapú megjelenítés segítségével. A voxel-ek megjelenítésére több megoldás is ismertetésre került. Amiatt került bemutatásra több megoldás ugyanazon probléma megoldására, mivel a szakirodalom nem említi, hogy melyik a jobb. Emiatt kerülnek ezek a megoldások összehasonlításra.

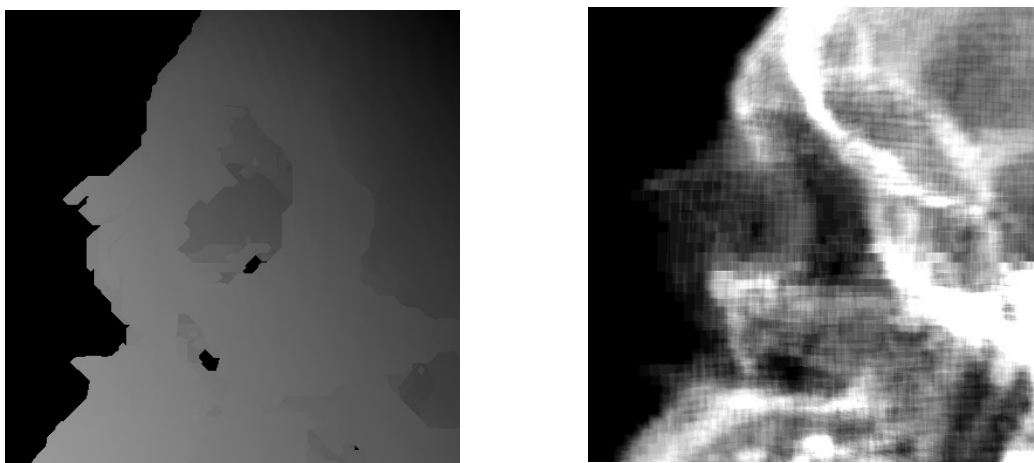
Módszer	Sebesség	Képmínőség	CPU használat	Memória használat	Megvalósítási nehézség
Voxel-ek téglatestekkel	közepes	jó	nagy	nagy	közepes
Voxel-ek point- sprite-okkal	nagyon gyors	közepes	közepes	közepes	nehéz
Poligon alapú	közepes	jó	nagy	nagyon nagy	nagyon nehéz

Táblázat 3.1: Háromdimenziós képkötő módszerek összehasonlítása

A táblázatban közzétett adatok az én alkalmazásomra vonatkoznak. Más fejlesztői eszközök és implementációs módszerek használatával a táblázatban eltérések mutatkoznának. Mint az a táblázatban is látszik, a point-sprite-os megoldás produkálta a leggyorsabb sebességet. Ez a GPU támogatottságából adódik. A 3.3.2.1-es fejezetben bemutatásra került, hogy a voxel-ek megjelenítésénél lehetőségünk van csak a határfelületen lévő voxel-eket megjeleníteni. A táblázat adatai erre az optimalizált változatra vonatkoznak. Ennek a sebessége közel azonos a poligon alapú megjelenítés sebességével. A képmínőség a téglatestekkel reprezentált voxel-es megoldás és a poligon alapú megoldás esetén közel azonos. A point-sprite-os megjelenítés esetén látni egy kis minőségromlást. A CPU és a Memória használat a poligon alapú megjelenítésnél meglehetősen magas lett, mivel a modell előállításához sok adatot kellett tárolni és sok számítást kellett elvégezni. (Gondoljunk a normálvektorok kiszámítására.) A normálvektorok kiszámítása nélkül kisebb lenne a memória és CPU használat, de a képmínőség leromlana. A megvalósítása a voxel-ek téglatestekkel történő reprezentációjának volt a legkönnyebb, mivel kevés probléma adódott. A point-sprite-os megvalósításnál a kamera pontoktól vett távolságának függvényében kell változtatnom a pontok méretét, emiatt a megvalósítása nehezebb. A legnehezebb a poligon alapú megjelenítés, mivel a modell pontfelhőből történő előállítása sok ötletet igényel.

3.4. A bemutatott módszerek összehasonlítása

Az egyes módszereknek megvannak a maguk előnyei és hátrányai. A térfogat vizualizáció előnye a poligon alapú megjelenítéssel szemben, hogy az összes rendelkezésre álló adatot meg tudjuk jeleníteni vele, és transzformációnál a modellt nem kell újraszámolni mint a poligon alapú megjelenítésnél, továbbá nem tesz kárt a grafikai objektum éleiben. A 3.3.7-es ábrán látható, hogy a poligon alapú megjelenítéssel készült képen az élek sokkal durvábbak, a térfogat vizualizációval készült képen pedig eredeti állapotukban maradtak meg. A térfogat vizualizáció hátránya viszont, hogy sok memóriát igényel. A poligon alapú megjelenítés előnye, hogy a grafikus hardver ezt támogatja, emiatt a megjelenítésére jobban kiforrott algoritmusok állnak rendelkezésünkre, továbbá a memóriaigénye is kisebb. Hátránya viszont, hogy transzformációnál(pl.: vágás) a modellt újra kell számolni.

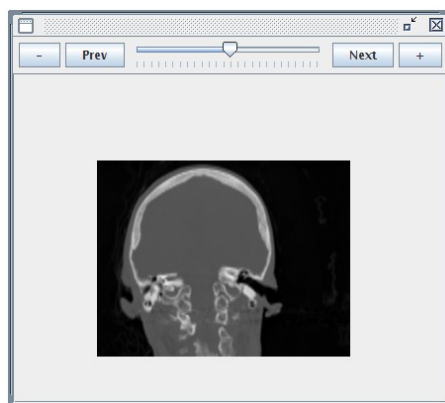


Ábra 3.3.7: Bal oldalt poligon alapú megjelenítéssel előállított, jobb oldalt voxel-ekből álló modell egy részlete.

Ha az én alkalmazásomat nézzük, a point-sprite-os térfogat vizualizáció bizonyult a legjobbnak, mivel GPU támogatás miatt nagyon gyors, és képminőségben nem sokkal marad el a másik két megoldástól. Az, hogy melyik megoldást érdemes használni az adott alkalmazás céljától függ. Ahol csak a felületeket szeretnénk látni (pl.: interaktív anatómia atlasz), ott a poligon alapú megjelenítés előnyösebb, ahol pedig transzformálni szeretnénk valós időben (pl.: sebész szimulátor), ott pedig a térfogat vizualizáció.

4. Az alkalmazás bemutatása

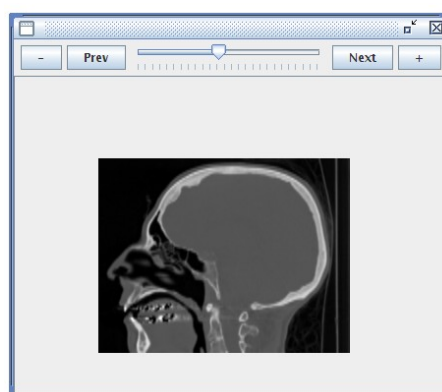
Az alkalmazás személyi számítógépen történő használatra lett tervezve. A Java technológia platformfüggetlensége miatt bármely operációs rendszeren használható. Ajánlott gépigénye 2 gigabyte memória és 2 gigahertzes processzor. Az alkalmazás elindítása után a File menü Open almenüjére kattintva lehetőségünk van képsorozat beolvasására. Ha a képek beolvasása sikeres volt, akkor megjelenik a képsorozat egy általam erre a célra fejlesztett Java komponensben (4.1.1-es ábra). Ezek után ha szükségesnek látjuk, végrehajthatjuk a képeken a 3.2-es fejezetben bemutatott kontrasztjavítást és zajszűrést. Ezeket a Preprocessing menüben találhatjuk meg, továbbá a képek felbontásának megváltoztatására is lehetőségünk van. Miután megtörtént a képek beolvasása és az esetleges előfeldolgozási lépések, lehetőségünk van a 3.2.4-es fejezetben ismertetett multi-planáris rekonstrukció megvalósítására. A menüből kiválasztva megjelennek az újabb java komponensek, bennük a rekonstruált képsorozatokkal.



a)



b)



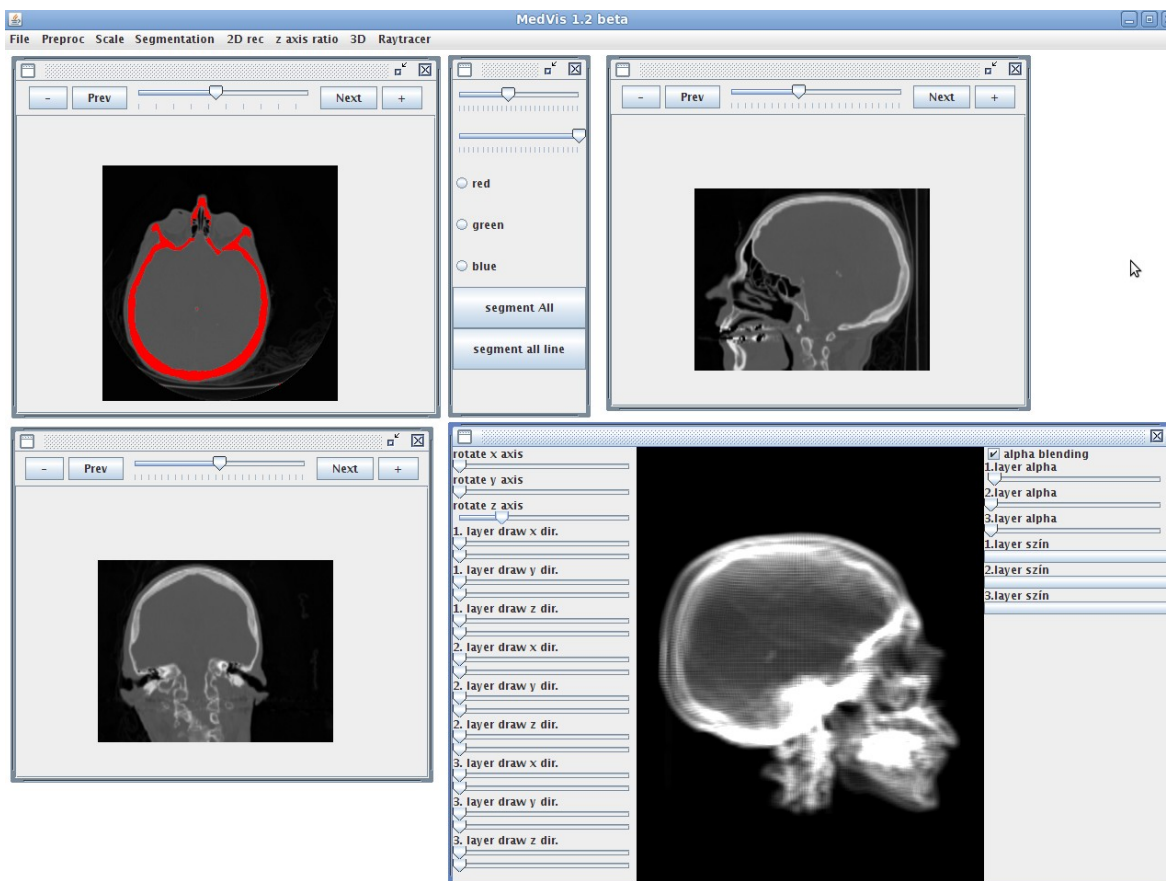
c)

Ábra 4.1.1: a),b),c): Multi-planáris rekonstrukcióval készült X,Z és Y tengely menti képsorozatok a CT Head képsorozatról. Az ábrán látható a képsorozatot megjelenítő Java komponens.

4. Az alkalmazás bemutatása

Ezután szegmentálhatjuk a képeket. A komponens melynek segítségével végrehajthatjuk a szegmentálást, a Segmentation menü Thresholding almenüjével hozható elő. Ha térfogatmetszetet (volume slice) szeretnénk létrehozni, a szegmentálást végrehajtó komponens előhívása után kiválasztjuk azt a képsorozatot (A multi-planáris rekonstrukció miatt több képsorozatunk is lehet) amelyikről a metszetet szeretnénk előállítani, majd mint egy rajzprogramban, megrajzolhatjuk a vonalat, ezután a szegmentálást végző komponens „Segment all line” gombjának (4.1.2-es ábra) segítségével előállíthatjuk a metszetet.

Háromdimenziós grafikai objektum előállításához a szegmentálást végrehajtó komponensen ki kell választanunk két küszöbértéket, majd a komponensen a Thresholding gombra (4.1.2-es ábra) kattintva az alkalmazás végrehajtja a szegmentálást 3.2.3 as fejezetben bemutatott thresholding eljárással. Ezután elő kell állítanunk a háromdimenziós byte tömböt, ami a 3. fejezetben többször is említésre került. Amint ezt megtettük, a menüből kiválaszthatjuk, hogy milyen megjelenítési módszert szeretnénk használni, és az alkalmazás előállítja a grafikai objektumot.



Ábra 4.1.2: Képernyőmentés a MedVis alkalmazásból. Láthatók az X,Y és Z tengelyek menti képsorozatok, a szegmentálást megvalósító komponens, és a point-sprite-ok segítségével megjelenített háromdimenziós grafikai objektum.

5. Továbbfejlesztési lehetőségek

Az alkalmazás továbbfejlesztésénél több főbb irány jelenik meg. Bármelyik irányban is indulunk el, a fő célunk, hogy minél több adatot a lehető leggyorsabban vizualizáljunk. Az első tennivaló, hogy az eddig implementált módszereket, amennyire csak lehet optimalizáljuk memória és CPU felhasználás szempontjából, és az alkalmazás felhasználói felületét barátságosabbá tegyük. Ezután több irányban is elindulhatunk a fejlesztéssel.

Az egyik út a megjelenítés fejlesztése. Az alkalmazáshoz fejleszthetünk valós idejű sugárkövetést alkalmazó grafikai motort ezáltal fotorealistikus árnyékok modellezésére nyílik lehetőségünk. A sugárkövető grafikus motort gyorsabbá tehetjük párhuzamosításokkal mind CPU és GPU szinten. CPU szinten úgy, hogy a képernyőt több részre osztjuk, és minden egyes részhez külön CPU magot rendelünk. GPU szinten pedig használhatunk hardver specifikus API-kat például ilyen az nVidia CUDA nevű API-ja. Az alkalmazáson a voxel-ek GPU-val történő megjelenítése is gyorsít. A sugárkövető motort gyorsabbá tehetjük térfelosztási módszerek alkalmazásával. Ilyenek például a BSP-tree, Octree.

A másik út a deformálható és animálható voxel modellek megvalósítása. Ennek segítségével animálhatjuk az előállított modelleket, és a jelenlegi alkalmazást összekapcsolva egy multi-touch terminállal (olyan érintőképernyős beviteli eszköz mely egyidejűleg több „érintést” kezel) sebész szimulátor fejlesztésére nyílik módunk.

A harmadik út, hogy a képek analízisére alkalmas osztályokat fejlesztünk. Ezen osztályoknak alkalmasnak kell lennie a képsorozatokon található bizonyos betegségek felismerésére, továbbá mérési feladatok ellátására (pl.: daganat térfogatának, felszínének meghatározása), és statisztikák készítésére.

6. Konklúzió

A munkám során a CT és MRI képek kétdimenziós és háromdimenziós vizualizációjánál felmerülő problémákkal foglalkoztam. Ilyen problémák a képek előfeldolgozása (zajszűrés, kontraszt feljavítás, szegmentálás), kétdimenziós rekonstrukció, háromdimenziós megjelenítés. A munkám célja egy olyan alkalmazás kifejlesztése volt, mely képes CT és MRI képsorozatok előfeldolgozására (kontraszt feljavítás, zajszűrés, szegmentálás), a képsorozatok kétdimenziós rekonstrukciójára, a képsorozatok háromdimenziós rekonstrukciójára térfogat és poligon alapú megjelenítés segítségével. A célt sikerült elérnem.

A dolgozatban ismertetésre került a szakterület rövid története és a manapság használt eljárások. Ezután az olvasó betekintést nyerhetett a CT és MRI képsorozatok kétdimenziós illetve háromdimenziós rekonstrukciójába. A volume slicing megvalósítására egy olyan eljárás került ismertetésre melynek sebessége többszöröse a szakirodalomban említett síkmetszetes megoldásnak, és egy olyan megoldás is ismertetésre került, mely a masírozó kockák algoritmus sebességét növeli. Ezután az alkalmazásomban használt háromdimenziós vizualizációs módszerek kerültek összehasonlításra.

A témakör aktualitását adja, hogy a betegségek gyors felismerése emberek életét mentheti meg, és ebben az orvosi képfeldolgozás nagy segítséget ad az orvosok számára. Amíg a világon lesznek beteg emberek, addig szükség van az orvosi képfeldolgozásra. A hardver eszközök fejlődésével mindig újabb lehetőségek nyílnak meg a terület kutatói előtt, biztosítva ezzel azt, hogy a szakterület dinamikusan fejlődjön.

Irodalomjegyzék

- [1] B.A. Payne, A.W. Toga: Surface mapping brain function on 3D models, Computer Graphics & Applications, September 1990.
- [2] Bernd Jahne: Practical handbook on image processing for scientific and technical applications ISBN: 0-8493-1900-5, Crcpress 2005.
- [3] Bernhard Preim, Dirk Bartz: Visualization in Medicine: Theory, Algorithms, and Applications, ISBN: 978-0-12-370596-9, Elsevier 2007.
- [4] Dr. Palkó András: Képkalkotó eljárások befogadáspolitikai kérdései, ime VI. évfolyam egészség-gazdaságtani különszám 2007.október
- [5] Dr. Szirmay-Kalos László: Számítógépes grafika, ISBN:963-618-208-6, ComputerBooks , 1999.
- [6] Eizenberger Máté: Digitális képfeldolgozás alapjai, Ental Kutatási, Fejlesztési és Tanácsadó Kft.
- [7] Elek István: Digitális szűrési eljárások a geoinformatikában, egyetemi jegyzet, 2005.
- [8] Emanuele Neri, Davide Caramella, Carlo Bartolozzi: Image Processing in Radiology, ISBN:978-3-540-25915-2 , Springer, 2008.
- [9] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, Gabriel Taubin: The Ball-Pivoting Algorithm for Surface Reconstruction IEEE Transactions on Visualization and Computer Graphics, Vol. 5, No. 4, 1999.
- [10] Fazekas Gábor, Hajdu András: Képfeldolgozási módszerek egyetemi jegyzet, 2004.
- [11] Gregory M. Nielson, Bernd Hamann: The asymptotic decider: Resolving the ambiguity in Marching Cubes, IEEE Visualization, 1999.
- [12] Isaac N. Bankman, PhD: Handbook of Medical Imaging: Processing and Analysis, ISBN: 0-12-077790-8, Academic Press, 2000.
- [13] Jiri Jan: Medical image processing, reconstruction and restoration ISBN:978-0-8247-5849-3, Taylor&Francis 2006.
- [14] Paul Martz: OpenGL röviden, ISBN: 978-963-9637-25-2, Kiskapu Kft, 2007.
- [15] Pecz Vilmos: Ókori Lexikon I–II. kötet, Franklin Társulat, 1904.
- [16] Valentin E. Brimkov, Reneta P. Barneva: Analytical Honeycomb Geometry for Raster and Volume Graphics, The Computer Journal 48(2), 2005.
- [17] William E. Lorensen, Harvey E. Cline: Marching Cubes: A high resolution 3D surface construction algorithm, ACM SIGGRAPH Computer GraphicsVolume 21 , Issue 4, 1987.
- [18] William K. Pratt Digital Image Processing ISBN:978-0-471-76777-0, John Wiley, 2007.
- [19] Zajszerűs variációk

<http://vision.vein.hu/~kla/kepfeld/kepfeld10.html>, 2003.

[20] Siemens robot revolutionizes interventional imaging

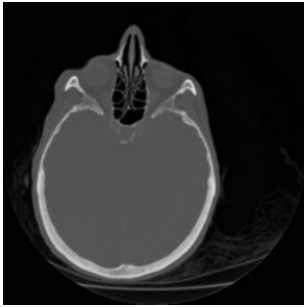
<http://www.diagnosticimaging.com/conferencereports/rsna2007/article/113619/1188397>, 2007.

[21] A boncolás története

<http://www.sulinet.hu/tart/fcikk/Kiam/0/24876/1>, 2004.

A. Függelék

CT Head



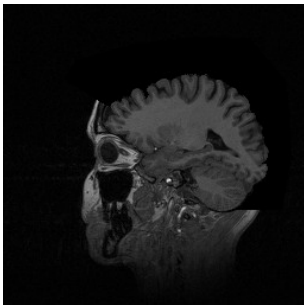
a)

Egy halott nőről készült teljes koponya CT, amely része a „University of North Carolina Volume Rendering Test Data Set” archívumnak. A száj környékén anomáliák láthatók, ami a fogászati töméseknek köszönhető. A képsorozatról nehéz őket eltávolítani, mivel a pixel-ek a csontokéhoz közeli intenzitásértékekkel bírnak.

A képsorozat 99 képből áll melyek 256*256 pixel felbontásúak.

Arányok: X:Y:Z 1:1:2

MRI Brain



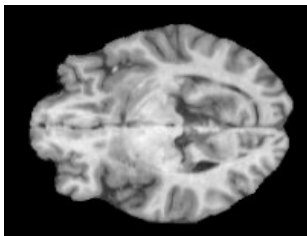
b)

Emberi fejről készült teljes MRI scan, ez is része a „University of North Carolina Volume Rendering Test Data Set” archívumnak. A képsorozatról Dr. Julian Rosenman a North Carolina Memorial Hospital munkatársa eltávolította a koponyát.

A képsorozat 99 képből áll melyek 256*256 pixel felbontásúak.

Arányok: X:Y:Z 1:1:2

MRI Brain 2



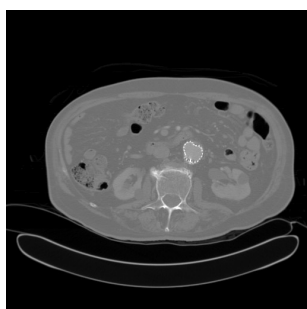
c)

Agyi MRI felvétel.

A képsorozat 160 képből áll, melyek 200*160 pixel felbontásúak

Arányok: X:Y:Z 1:1:1

Stent



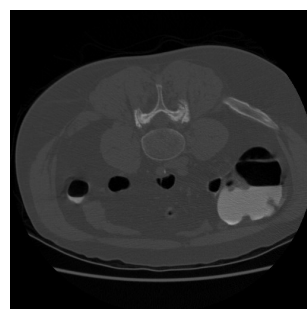
d)

CT képsorozat a hasról és a medencéről, továbbá a hasi aortában stent látható.

A képsorozat 174 képből áll, melyek 512*512 pixel felbontásúak

Arányok: X:Y:Z 1:1:3.81

Prone



e)

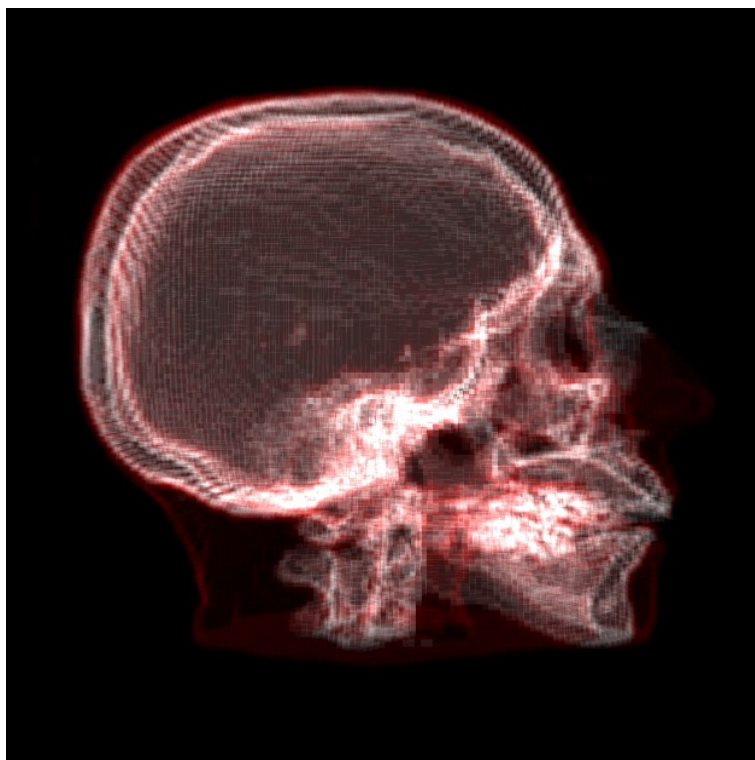
Hasi CT scan.

A képsorozat 463 képből áll, melyek 512*512 pixel felbontásúak

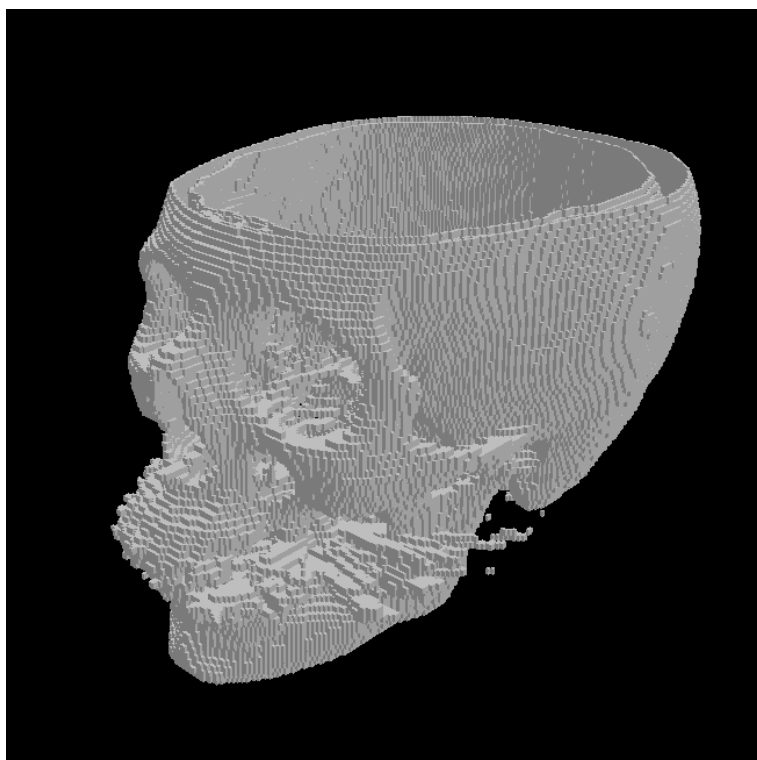
Arányok: X:Y:Z 1:1:1.6

Ábra 5.1: a): Egy kép a CT Head képsorozatból, b): Egy kép az MRI Brain képsorozatból, c): Egy kép az MRI Brain2 képsorozatból, d): Egy kép a Stent képsorozatból, e): egy kép a Prone képsorozatból

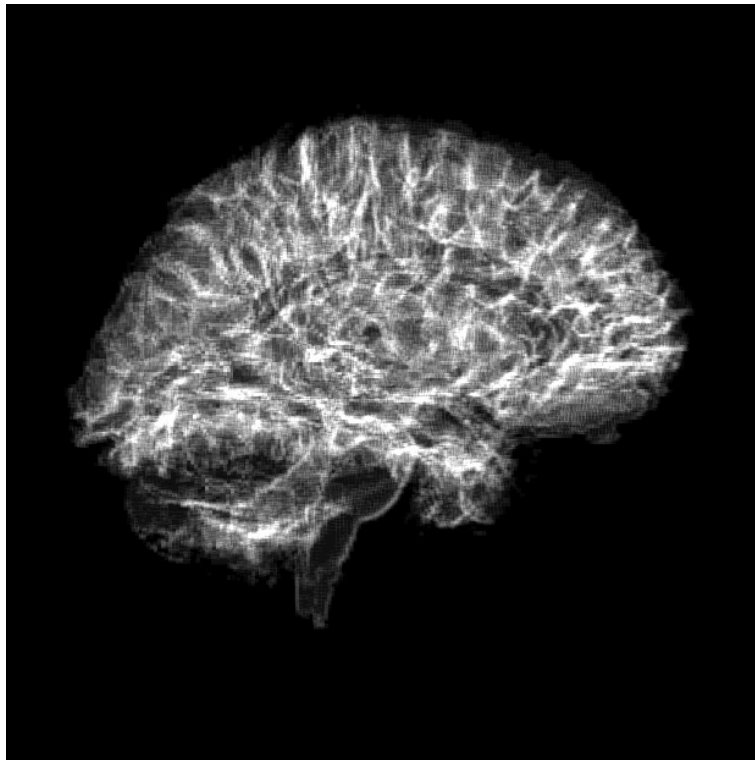
A CT Head és MRI Brain adathalmazok elérhetők a <http://graphics.stanford.edu/data/voldata/> címen, a stent és a prone adathalmazok pedig innen tölthetők le: <http://www.volvis.org/>



a)



b)

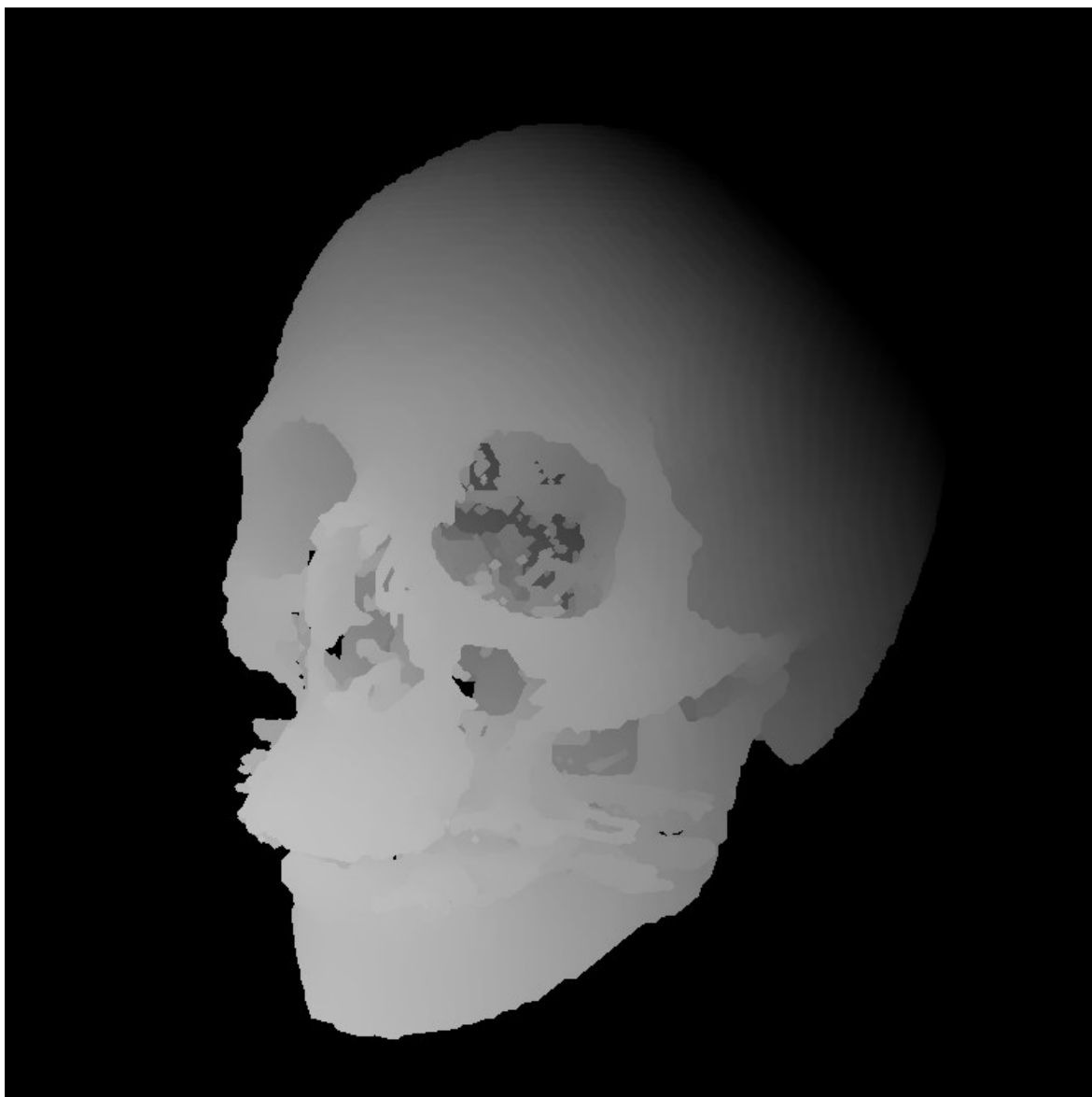


c)

Ábra 5.2: Voxel-ek téglatestekkel történő reprezentációjával előállított képek. a) és b) a CT Head képsorozatról c) az MRI Brain 2 képsorozatról. Az a) képen 2 réteget láthatunk. A piros a lágy szöveteket, míg a fehér a csontot jelöli. A b) képen átlátszóság nélkül megjelenített voxel-eket látunk. A koponya felső része nincs megjelenítve.



Ábra 5.3: Stent képsorozatról point-sprite-ok segítségével előállított kép



Ábra 5.4: Masírozó kockák algoritmus segítségével előállított kép a CT Head képsorozatról diffúz árnyalással.