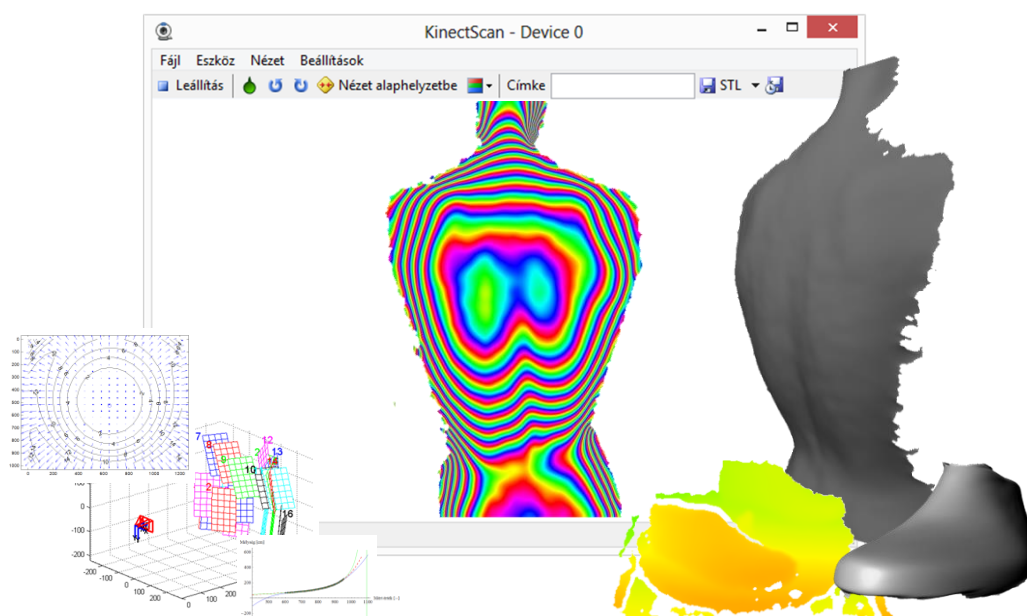


Mélység szenzorok orvostechnikai alkalmazásai



Készítette:

Major Péter

Budapesti Műszaki és Gazdaságtudományi Egyetem, Gépészmérnöki Kar,
Mechatronikai mérnöki mesterképzési szak, I. évfolyam

Témavezető:

Dr. Tamás Péter, egyetemi docens

Budapesti Műszaki és Gazdaságtudományi Egyetem, Gépészmérnöki Kar,
Mechatronika, Optika és Gépészeti Informatika Tanszék

A dolgozat bemutatását a TÁMOP - 4.2.2.B-10/1--2010-0009 "Új tehetséggondozó programok és kutatások a Műegyetem tudományos műhelyeiben" című program támogatta.

1 Tartalomjegyzék

1	Tartalomjegyzék	2
2	Bevezetés.....	4
2.1	A mélylégleképezési eljárások áttekintése	5
2.2	Alkalmazott szoftver és hardvereszközök	6
3	A Kinect kalibrációja	8
3.1	Hardverfelépítés	8
3.2	A Kinect leképezési modellje	11
3.3	A kalibráció menete.....	13
3.3.1	Kalibrációs felvételek elkészítése	13
3.3.2	Kalibráció MATLAB segítségével.....	16
3.4	Mérési eredmények elemzése	21
4	A képfeldolgozás lépései	24
4.1	A képfeldolgozás vezérlése.....	24
4.1.1	A driver kiválasztása	24
4.1.2	A Kinect kezelése	25
4.2	A képek feldolgozása – a hardveres képfeldolgozás	27
4.3	A képfeldolgozás menete	29
4.3.1	A mérési eredmények elérése	29
4.3.2	Átlagolás	30
4.3.3	Torzítás korrekció	32
4.3.4	Gauss szűrés	34
4.3.5	Háromdimenziós modell létrehozása.....	36
4.3.6	A modell árnyalása	39
4.4	Alkalmazás greincferdülés diagnosztizálásához	42
5	Emberi láb háromdimenziós szkennelése	43
5.1	Modellezési módszerek	43
5.2	Modellezés polár-koordináta-rendszerben	45
5.3	Modellképzés keresztmetszetenként.....	51
5.3.1	Részmodellek megalkotása	51
5.3.2	Modellépítés.....	52
5.3.3	Modellfinomítás	52
5.3.4	Háromszögelés alapalgorithmusa	53

5.3.5	Háromszögelési algoritmus finomítása	54
6	Sztereoszkópikus megjelenítés.....	58
7	Továbbfejlesztési lehetőségek.....	61
8	Konklúzió	62
9	Irodalomjegyzék	63
10	Köszönetnyilvánítás	64

2 Bevezetés

Dolgozatomban a mélylégszenzorok orvosi célra való hasznosítási lehetőségeit szeretném bemutatni. A mélylégszenzorok, amelyek a hétköznapi kamerákhoz hasonló leképezésű, de az egyes pixelek által leképezett térbeli pontok képsíktól mért távolságát megadó képalkotást végeznek, az egészségügy számos területén alkalmazhatóak. Segítségükkel az emberi testfelületről háromdimenziós modell alkotható, amely lehetőséget ad különböző deformációk (pl. gerincferdülés) automatikus vagy fél-automatikus diagnosztizálására, a testhez jól illeszkedő eszközök, speciális ruhadarabok, támasztékok létrehozására és számos egyéb megoldásra. Előnyük, hogy a képalkotás gyorsan, a betegek zavarása nélkül végrehajtható, nem igényel előkészítést és tetszőlegesen sokszor megismételhető.

Bár mélylégi képalkotásra alkalmas eszközök már régóta léteznek, az elmúlt évek kivételével ezek alatt általában drága, sok esetben még nem kiforrott megoldásokat értettünk. Laboratóriumi prototípusok már a 70-es, 80-as években fejlesztés alatt álltak, az első alkalmazási területek egyike a gépipar volt, ahol az alkatrészek egy részének minőségellenőrzése ilyen módon igen gyorsan, és elég precízen vált elvégezhetővé. Fontos terület volt a robotok navigációja is, hiszen a mélylégszenzorokkal képesek voltak az előttük álló akadályok megfelelő érzékelésére. A kétezres évektől kezdve több új elgondolás is jelentős fejlődésnek indult, amelyek kompakt, kis számítási igényű, valós-idejű, relatíve pontos és olcsó megoldásokat szültek. A mélylégkamerák széles körű elérhetőségével új eszközök széles választékának létrehozására nyílt lehetőség. A számos új alkalmazási terület közül kiemelhető például a gesztusvezérlés, valós-idejű modellgenerálás, mobil háromdimenziós szkennerek, 2,5D fényképezés/filmfelvétel vagy a környezet modellezése. A mélylégszenzorok segítségével a környezet háromdimenziós modellje úgy hozható létre, hogy az érzékelő működése nem zavarja a jelenlévő embereket, állatokat, technológiától függően gyors mozgások is rögzíthetőek, és magas mélylégi felbontás is elérhető. A módszerek jelenleg még folyamatos fejlődés alatt állnak, amelyek elsősorban a mérési zaj csökkentését és a felbontás növelését célozzák, illetve a szenzorok méretének és teljesítményfelvételének csökkentése is fontos, hogy azok mobil eszközökbe is beépíthetőek legyenek.

A dolgozatban két különböző orvosi feladat megoldására alkalmas eszközt szeretnék bemutatni: az első a gerincproblémák diagnosztizálásában kíván segítséget nyújtani, míg a másik a páciens lábáról háromdimenziós modellt készítve speciális, személyre szabott cipők, kaptafák tervezését és gyártását kívánja segíteni. Bemutatom az ezekhez készült mérőrendszer felépítését, a mérési eredményeket feldolgozó program működését, annak implementációja során felmerült problémákat és megoldásokat. A dolgozat emellett részletesen tárgyalja az alkalmazott mélylégszenzorok kalibrációját, leképezésük modelljének megalkotását, a felhasználható szoftvercsomagokat és az ezekkel kapcsolatos nehézségeket. Így két konkrét példa ismertetése mellett általánosságban is áttekintem a mélylégszenzorok alkalmazását. A dolgozat első részében ismertetem a felhasznált eszközöket, illetve a választási lehetőségeket, ezt követi a mérőrendszerek kalibrációjának és leképezésének részletes bemutatása. A dolgozat törzsében a már említett gyakorlati megoldásokat mutatom be, a lezáró rész az eredmények elemzésével és a továbbfejlesztési lehetőségek ismertetésével foglalkozik.

2.1 A mélylégleképezési eljárások áttekintése

A mélylégszenzorok kalibrációja segítségével felírhatóak azok képkalkotását leíró egyenletek, és ezek alapján a leképezés inverze is elvégezhető, és a mélylégképből a leképezett tárgyak háromdimenziós modellje előállítható. A mélylégszenzorok a környezetből csak a kamera felé néző felületrészeket látják, tehát a modell inkább 2,5 dimenziósnak tekinthető. Ennek ellenére kidolgozhatóak olyan megoldások, amelyekkel több felvétel kombinálásával már igazi háromdimenziós modell is nyerhető. Az általam alkalmazott szenzor a teljes látóteret egy lépésben képezi le, nem szkennelést végez, így valós idejű felvételek készíthetők. Mivel az egyes felvételek egyszerű képekként kezelhetők, így azok tárolása, esetleges későbbi feldolgozása könnyen megoldható. Hátrány ugyanakkor, hogy az itt vizsgált megoldás zajos, és alacsony felbontású kimenetet ad. A mélylégérzékenység pedig a távolság növekedésével erősen lecsökken.

Mélylégképek előállítására több módszer is rendelkezésünkre áll, amelyek közül a legfontosabbakat szeretném a következőkben ismertetni [14]:

- a) Lézer háromszögelők: ezen gépek egy eltéríthető sugarú lézerből és egy kamerából állnak. A lézer egy pontban vagy egy vonal mentén megvilágítja a tárgyat és az arról visszaverődő fényt a kamera érzékeli. A pont háromdimenziós helye ezután egyszerű háromszögeléssel meghatározható a rendszer paramétereinek ismeretében. E megoldás előnyei a nagy pontosság és a felületi textúrára, reflektivitásra, megvilágításra vonatkozó érzéketlenség. Mivel a felület pontonként, vagy vonalak mentén kerül leképezésre, így valós idejű mérésre csak kevésbé alkalmas, bár gyors működésű kamera és hasonlóan gyorsan pásztázó lézer segítségével ez is megoldható.
- b) Strukturált fény: ezen szenzorok a megméréendő felületekre egy előre definiált mintát vetítenek, és a pontok térbeli pozícióit hasonlóan az előző esethez háromszögeléssel határozzák meg. Mivel ezen módszerek egyszerre az egész felületet vizsgálják, így valós idejű működés is megoldható. A kivetített minták lehetnek például különböző rácsok, pontmátrixok stb. Amennyiben a minta egyes elemeit sikerül a kamera által felvett képen meghatározni, úgy azok térbeli pozíciója meghatározható.
- c) Sztereo látás: ez a megoldás bizonyos szempontból az előző passzív változatának tekinthető, ugyanis itt nem vetítünk mintát a modellre, hanem annak természetes textúráját használjuk fel és azt két kamerával is megfigyeljük. A kamerák képein a jellemző részeket felismerve és azokat a két kamera képén összepárosítva a távolságuk meghatározható. A módszer hátránya, hogy csak akkor alkalmazható jól, ha a vizsgált felület megfelelően sok jól azonosítható részlettel rendelkezik. A módszer kiegészíthető egy a felületekre mintát vetítő projektorral, ami biztosítja a felületek megfelelő textúráját.
- d) Fotógramméria: ez a módszer hasonló az előzőhöz, de itt egy kamerával különböző helyzetekből készült képek alapján lehet háromdimenziós modellt alkotni. Ehhez jól kell ismerni a felhasznált kamera optikai paramétereit és az egyes képek között a kamera relatív helyzetváltozását is. Ezek után a sztereo leképezéshez hasonló feladatot kapunk. A módszer valós időben nem alkalmazható, hiszen hozzá a kamera és környezet relatív mozgása szükséges.
- e) A fény utazási idejének mérése: ilyenkor a megméréendő tárgyat egy lézerimpulzussal megvilágítjuk, majd pixelenként megmérjük, hogy a kamerába milyen gyorsan ér vissza a fény. Ez a módszer elsősorban nagyobb hatótávolság esetén használható jól, mert kis

távolságok esetén nagyon nagy felbontású időmérő áramkörök szükségesek. A módszer hátránya hogy a tükröződő felületek kezelése nagyon rossz, illetve hogy a lézer teljesítményét életvédelmi szempontok miatt korlátozni kell.

- f) Interferometria: ez a módszer térben vagy időben változó periodikus mintákat vetít a felületre, majd a visszavert fényt egy megfelelően választott referencia ráccsal demoduláljuk. Az így kapott képen az interferenciajelenség miatt láthatóvá válik a felület pontjainak távolságkülönbsége. Ezen rendszerek előnye a nagy elérhető pontosság.
- g) Moaré kontúrok: ez a módszer a képet egy rácson átvezetett fénynyalábbal világítja meg, majd a visszaérkező fényt egy a projektortól oldalirányban eltolt kamerával figyeli meg, ami előtt egy az előbbivel identikus rács található. A keletkező interferencia minta ad lehetőséget a mélységi vizsgálatokra.
- h) Fókuszálás alapján: a részletek elmosódottsága utal a fókusztávolságtól való távolságra.
- i) Árnyalás alapján: az objektum körül fényforrást mozgatva felvételeket készítünk, és a felületről visszaverődő fény változásait figyeljük. Amennyiben a felületre jellemző fény-visszaverődési szabályokat (Lambert stb.) ismerjük, úgy meghatározható a felület geometriája.

A felsorolásból én a strukturált fény elvét használó Kinectet alkalmaztam a feladat megoldására, de a feldolgozó algoritmus nagy része nem függ a mélységkép előállítás módjától, így az itt bemutatásra kerülő módszerek nagy része más mélységérzékelési eljárással is működőképes. A strukturált fény alkalmazásának előnyei:

- a szükséges hardver egyszerű és olcsón megvalósítható,
- valós idejű feldolgozás végezhető,
- kompakt méretű szenzorok hozhatóak létre,
- megvilágítás hiányában is működik,
- a legtöbb hardver mélységképet továbbít, így nincs szükség gépigényes feldolgozási műveletre ennek előállításához.

Hátrányként jelentkezik a zajos kimenet és az alacsony felbontás (mélységi és a képsíkkal párhuzamos irányban egyaránt).

2.2 Alkalmazott szoftver és hardvereszközök

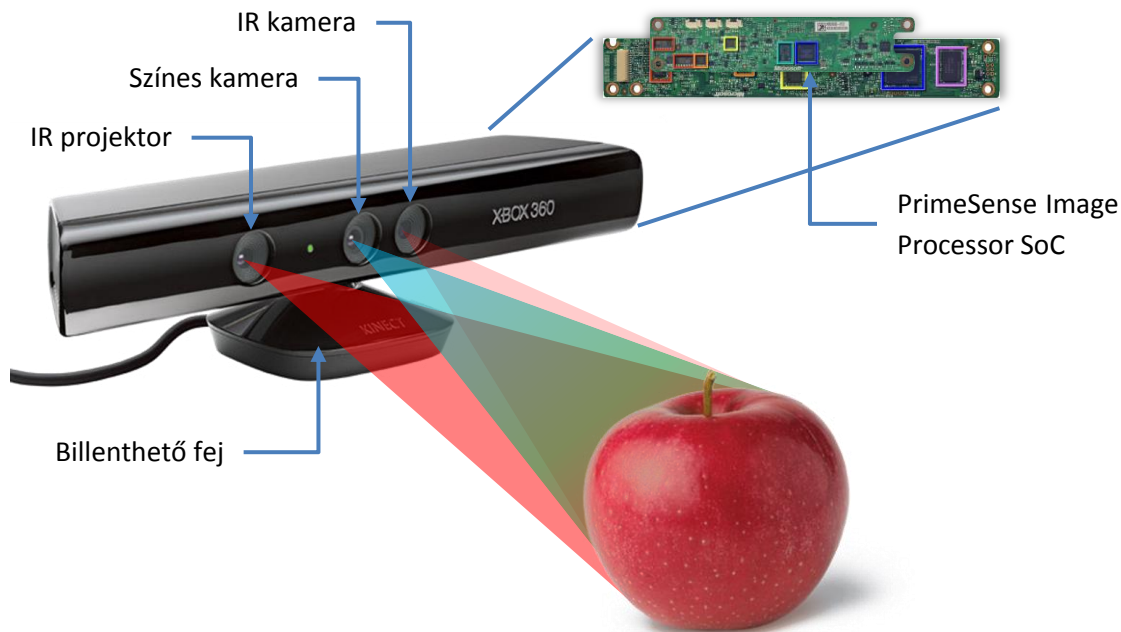
A képfeldolgozást végző program nagy része C# nyelven a Microsoft .Net keretrendszerben [22] íródott, kis része C++ és C++/CLI-ben. A képfeldolgozás szinte teljesen a GPU-n kerül végrehajtásra, amelyet HLSL nyelven [18, 21] programoztam a DirectX API segítségével. Mivel a képfeldolgozást a GPU végzi, így a CPU terhelés viszonylag alacsony. A grafikus kártya elérése .Net környezetből az XNA keretrendszer segítségével történt, amely a DirectX API-t képezi le .Net osztályokban. A program elkészítéshez az OpenKinect projekt nyílt forráskódú freenect driverét használtam fel a Kinecttel való kommunikációra. A fejlesztés nagy része Microsoft Visual Studio környezetben történt. A kalibrációhoz felhasználtam a MATLAB szoftver Camera Calibration Toolbox modulját is.

A felhasznált mélyégszenzor a Microsoft Kinect, amelynek egy példányát a Mechatronika, Optika és Gépészeti Informatika Tanszék bocsátotta rendelkezésemre. A forgószkenner hardverét Basch László fejleszteti, és ő biztosította az ezzel készült felvételeket is. A program futtatásának legfőbb hardverkövetelménye a grafikus kártyával szemben lép fel, amelynél szükséges a DirectX

Shader Model 3 vagy annál újabb verzió teljes támogatása. A CPU és GPU teljesítményigény szempontjából a ma használt átlagos gépekkel kielégíthető, ugyanis a feldolgozás jól optimalizált: a CPU kevés műveletet végez, a GPU végzett műveletek száma ehhez képest ugyan nagy, de a mai kártyák teljesítménye ezt sokszorosán meghaladja. A memóriaigény kb. 70-200MB használati módtól függően, amelynek nagy részét a képfeldolgozás során használt nagy mennyiségű képi és modellinformáció adja.

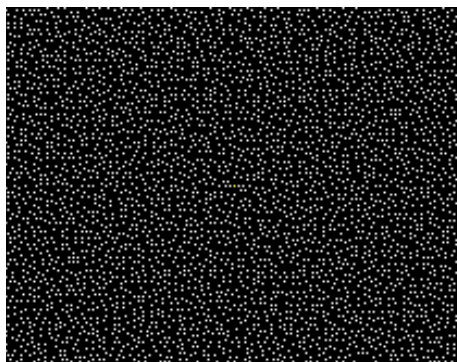
3 A Kinect kalibrációja

3.1 Hardverfelépítés



1. ábra A Kinect felépítése

A Kinect mélységmérési és modellezési szempontból négy lényeges modulból épül fel. Található benne egy infravörös projektor, amely lézertüneten keresztül vezetve létrehoz egy pontokból álló mintát [1]. A pontok bár szabályos rácson kerültek kialakításra, azok elrendeződése lokálisan nem szabályos. A kivetített pontsokaság jól láthatóan 3x3 szegmensből áll, amelyeken belül a pontok elrendeződése már megegyezik. A PrimeSense cég egy másik szabadalma alapján valószínű, hogy csak egy mintát állítanak elő, majd ezt optikai úton másolják [2]. Mivel a minta mindkét irányban csak háromszor ismétlődik, így ez felhasználható az egyes pontok mintabeli pozíciójának meghatározásához. Ezt a technikát a cég Light Coding-nak nevezte el. A minta bináris, így az egyes pontok jól kiemelhetők a képből, ezt segíti az is, hogy a világos területek sokkal kisebbek a mintában, mint a sötétek. A projektor az infravörös tartományban működik, ami lehetővé teszi, hogy annak működését, pontosabban az általa kivetített minta láthatóságát ne befolyásolják a környezeti

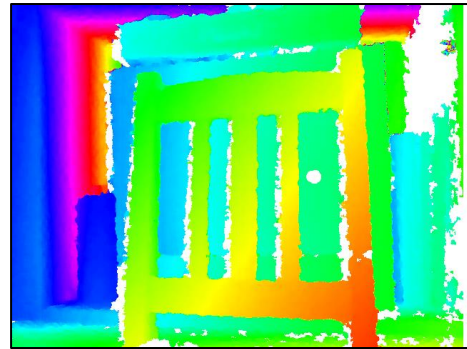


2. ábra A Kinect által kivetített minta egy szegmense [3]

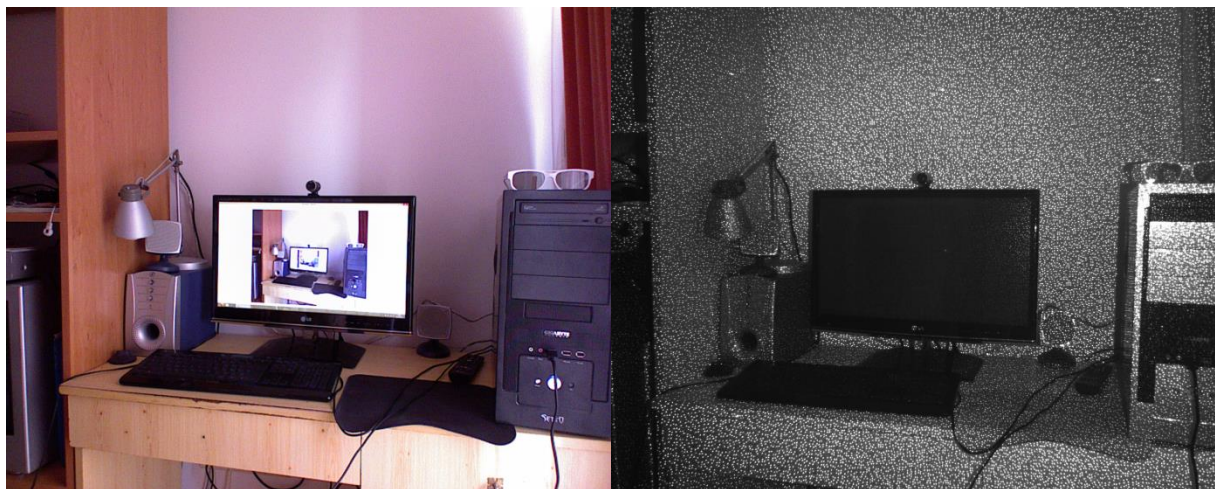
fényviszonyok. Ennek ellenére a módszer a szabadban nehezen alkalmazható, ugyanis a napfény infravörös tartományban elég erős ahhoz, hogy hozzá képest a projektor által vetített kép nem vehető észre. A minta elsősorban a matt felületekről visszaverve észlelhető jól, ha a felület tükröződik, akkor az elvakítja a mintát megfigyelő kamerát, ha az közel merőlegesen néz a felületre. A tükröződés mellett még gondot okoznak azok a felületek, amelyek nem vernek vissza elegendő fényt, vagy közel párhuzamosak az optikai tengellyel. Az egyes szegmensek közepén egy különösen nagy fényerejű, és nagyobb átmérőjű pont áll (4. ábra). Ezek a rendszer kiindulási

pontjai, amelyek érzékelése nagyobb erejűektől fogva sokkal megbízhatóbb, mint az átlagos pontoké.

A készülék másik fontos eleme az infravörös kamera, amely szürkeárnyaltos képet ad. Ez figyelmeztet a projektor által kivetített képet. Mivel a projektor és kamera kitérő optikai tengelyekkel rendelkeznek, ezért az infravörös kamera képén árnyékok jelenhetnek meg. Ennek az az oka, hogy a kamera olyan felületekre is rálát, amelyekre a projektor nem. Ez a jelenség különösen erőteljesen jelentkezik, ha egy objektum közel van a Kinecthez, és a háttér jóval távolabb, ilyenkor a nagy árnyéktér miatt a háttér nagyrészt nem látható (3. ábra). A fénynyalábok széttartása a távolság növekedésével láthatóvá válik, azonban a kamera vetítése miatt ez a kimeneten csak kevésbé látható. A mélységi látótávolságot ugyanakkor a fényelnyelődés korlátozza és az kb. 6-8 méter. Az infravörös kamera fókuszáló mechanikával is rendelkezik, hogy képes legyen különböző távolságtartományokban működni. A mélységérzékelés nem csak felső, hanem alsó határral is rendelkezik, ennek oka, hogy közeli tárgyak esetén az infravörös projektor fényerejénél fogva elvakítja az infravörös kamerát. Ez a jelenség a fényerőt csökkentő szűrő felhelyezésével javítható (bár ilyenkor a felső határ is csökkenni fog). Az elvégzett kísérletek szerint erősen nem reflektív és sötét színű felületek esetén minimum 40 cm távolság tartására van szükség, azonban a kimenet 60 cm-ig még tartalmazhat nem érzékelt foltokat.

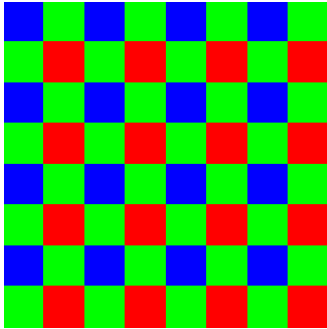


3. ábra Az infravörös projektor és kamera eltérő helyzete miatt árnyékok keletkeznek balra a szék háttámlájának léceitől



4. ábra A Kinect színes és infravörös kamerájának kimenetei 1280x1024-es felbontáson, az infravörös képen jól látható a 9x9-es rács néhány eleme is

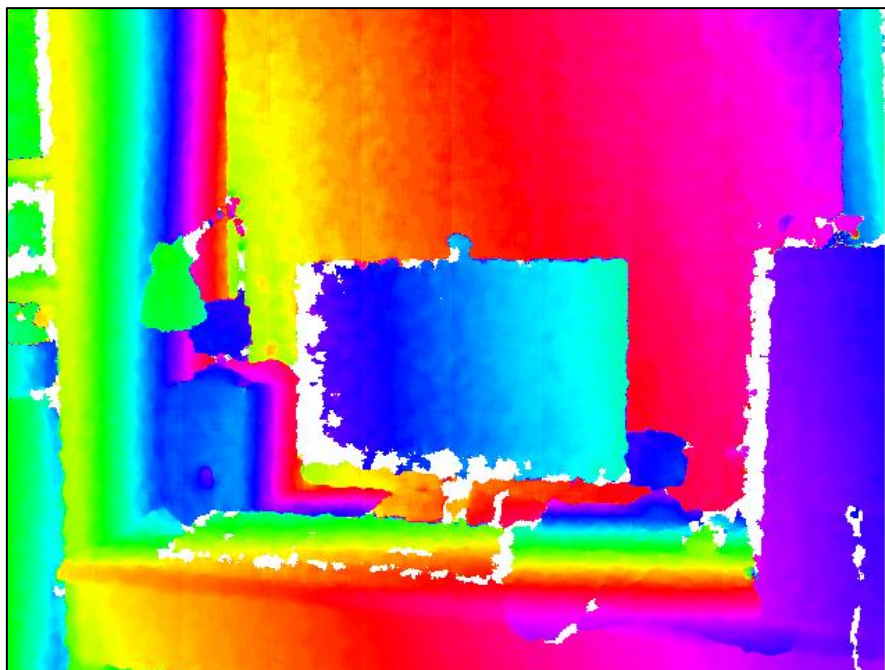
A Kinect harmadik fontos eleme a színes kamera, ez egy felső kategóriás webkamera képességeivel rendelkezik. Bár a színes kamera nem vesz részt a mélység érzékelésében, annak szerepe azért válik fontossá, mert használatával a geometriai modellek mellé megfelelő textúrákat is tudunk készíteni. Bár mind az infravörös, mind a színes kamerák nagyfelbontásúak (1280x1024 vagy nagyobb), ekkora felbontásban az információk mozgatása a készülék és a feldolgozó számítógép között nem lehetséges - az USB 2 által képezett korlátok miatt - 7-10 FPS-nél nagyobb sebességgel. Ráadásul a rendszer jól láthatóan ilyenkor növeli az exponálási időt, így a képen elmosódások jelennek meg. További probléma, hogy a képek továbbítása alatt a kép tartalma megváltozhat, így pl.



a kamera forgatásakor a környezet „elhajlik”, mivel a felülről lefelé haladva az egyes sorok más és más időben kerültek felvételre. Megfelelő sebességgel (30FPS) csak 640x480-as felbontáson továbbítható a videó (mélység és színes együtt). Azonban ilyenkor is szükség van tömörítésre, így például a színes videó nem tartalmazza minden pixelre a külön a színtkomponenseket, hanem az ún. Bayern minta (5. ábra) szerint kódolja a színeket, ami miatt a kép élessége kisebb lesz.

5. ábra A Kinect által színes képek esetén használt kódolás

Amint az említésre került, alapesetben a Kinect mélység- és színinformációkat továbbít. A mélység információt a Kinectbe beépített feldolgozóegység állítja elő. Ez úgy történik, hogy a feldolgozóegység összeveti az infravörös kamerából nyert képen látható mintát egy referenciával, és megfigyeli a pontok elmozdulását, ami a nem sima felületekre eső vetítésből adódik [1]. A pontok elmozdulásából kiszámítható azok távolsága. Mivel a pontelháló felbontása alacsony, és a pontok elhelyezkedése nem szabályos ezért a Kinect interpolációt végez, amely simább felületek esetén kevésbé visz be hibát, az élek mentén viszont erős, véletlen zajjal is terhelt csipkéződés (6. ábra) jelentkezik. Mivel a Kinect elég sok pontról nem kap folyamatosan információt, ezért ezeket a lyukakat a rendszer megpróbálja kitölteni. A rendszer a mért értékeket nemlineáris skálájú 11-bites formában továbbítja. A mélységképen jól megfigyelhetőek függőleges, egyenlő szélességű sávok, amelyek eredete a rendszer nem publikus implementációja miatt nem ismert. Ezek a sima felületeket enyhén recéssé teszik (hatásuk elsősorban nagy távolság esetén látható a rendszer jelleggörbéje miatt, lásd később), így ki kell őket simítani. Kedvezőtlen, hogy ezek mozognak, és szélességük az eltérő képkockákon erősen különböző lehet. Ennek oka szintén nem ismert, de valószínűleg kapcsolatban állhat a rendszer autofókusz algoritmusával, ha a látott kép tartalma nem változik, akkor a függőleges sávok mozgása megszűnik egy kezdeti beállási idő után.



6. ábra A mélységkimenet félméteres periódusú szírványszín árnyalás mellett és 32 képkocka átlagolásával

A Kinect belső jelfeldolgozó egysége tehát egy 640x480-as felbontású, 11-bites bitmélységű mélységképet ad. A Microsoft a szenzorokat a gyártás során kalibrálja, és a kalibrálási eredményeket az egység nem-felejtő memóriájába írja. Ezeket az információkat a driverek kiolvassák, így lehetőség van az érzékelt pontok háromdimenziós pozíciójának meghatározására. A probléma ezzel a megoldással, hogy mivel a Kinect célja gesztusvezérlés megvalósítása, így az elvégzett kalibráció háromdimenziós mérések végzéséhez nem elég pontos. Ennek kiküszöbölésére saját kalibrációs módszert dolgoztam ki, amely precíz és relatíve gyorsan elvégezhető.

3.2 A Kinect leképezési modellje

A kalibráció első lépése a kalibrálandó készülék modelljének megalkotása. Ennek elkészítéséhez meg kell ismerni a Kinect képalkotásának leképezési szabályait, amelyeket a tüllyuk kamera modell és Brown-féle torzítási modell segítségével modelleztem. Először a mélységi képalkotáshoz szükséges infravörös kamera leképezési modelljét mutatom be. Tekintsünk egy háromdimenziós objektumot, illetve ezen egy felületi pontot (P)! P pont helyzete legyen ismert a kamera koordinátarendszerében ($\mathbf{P}_w(x_w, y_w, z_w)$), amely egy derékszögű koordinátarendszer. A koordinátarendszer origója a képzeletbeli kamera apertúrájánál helyezkedik el. A koordinátarendszer X és Y tengelyei a képsíknak is megfelelő vízszintes és függőleges elhelyezkedéssel rendelkeznek (irányuk az egyszerűbb kezelés miatt a kép digitális tárolásának megfelelően alakul, így a képen balról jobbra, illetve fentről lefelé irányba mutatnak), a Z tengely a kamera optikai tengelyén helyezkedik el. Ebben a koordinátarendszerben minden pont három koordinátával rendelkezik. Legyenek ezek x , y és z ! Ebből a leképezés első lépéseként Z -normált koordináták nyerhetők ($\mathbf{P}_n(x_n, y_n)$), amelyek így a $z = 1$ síkban helyezkednek el. Ez a lépés azt is szemlélteti, hogy a távolabbi objektumok kisebbnek látszanak. Ezután kerül sor a képtorzítások modellezésére.

A képtorzítások a Brown modellnek megfelelően [4, 19] két összetevőből épülnek fel. A radiális torzítás esetében a képpontok optikai tengelytől mért távolsága megváltozik egy nemlineáris összefüggés szerint:

$$r_n^2 = x_n^2 + y_n^2 \quad (1)$$

$$\mathbf{P}_d = \begin{bmatrix} x_d \\ y_d \end{bmatrix} = (1 + k_1 r_n^2 + k_2 r_n^4 + k_3 r_n^6) \begin{bmatrix} x_n \\ y_n \end{bmatrix} + \mathbf{d}_t \quad (2)$$

Ahol:

- $\mathbf{P}_d(x_d, y_d)$: a torzított pont koordinátái
- $\mathbf{P}_n(x_n, y_n)$: a Z -normált pont koordinátái
- k_i : a radiális torzítási együtthatók
- $\mathbf{d}_t$: a tangenciális torzítás

A képletben szereplő együtthatókból általában csak az első kettőt, nagy látószög esetén pedig az első hármat használják fel. Fontos figyelembe venni, hogy a legtöbb kalibrációs módszer a harmadik vagy ennél magasabb tagokat már csak azok értékével összemérhető bizonytalansággal tudja meghatározni, így ezek használata a hibát tovább növelheti. A másik torzítási összetevő az ún. tangenciális torzítás, amely a mai modern kamerák esetén általában kicsi, de nem mindig elhanyagolható.

A tangenciális torzítás következő képlettel írható le:

$$\mathbf{d}_t = \begin{bmatrix} 2t_1x_ny_n + t_2(r_n^2 + 2x_n^2) \\ t_1(r_n^2 + 2y_n^2) + 2t_2x_ny_n \end{bmatrix} \quad (3)$$

Ahol t_i -k a tangenciális torzítási együtthatók. A két torzítás összege írja le a Brown-féle torzítási modellt. A torzítások felírása után történik meg a kamera perspektivikus vetítési transzformációjának második része (az első a Z komponenssel való normálás volt). Ez a transzformáció bár síkbeli koordinátarendszerek között megy végbe, leírásához a homogén koordinátás felírás miatt 3x3-as mátrix szükséges. Ez a mátrix tartalmazza a rendszer függőleges és vízszintes fókusz távolságát pixelben, továbbá az optikai főtengely képsíkkal vett metszéspontjának a kép közepétől vett távolságát. Ez utóbbi úgy is értelmezhető, hogy a képet alkotó CCD lapka elcsúszott az optikai rendszerhez képest. Ezen újabb transzformáció leírása:

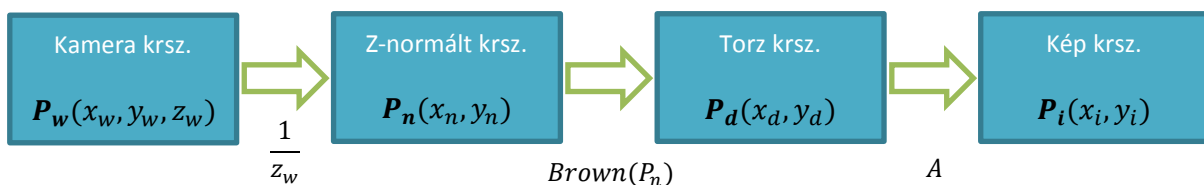
$$\mathbf{A} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \quad (5)$$

Ahol:

- $\mathbf{A}$: a kameramátrix
- f_x, f_y : a fókusz távolság x és y irányban
- c_x, c_y : a képsík és az optikai tengely metszéspontjának koordinátái
- s : a tengelyek merőlegestől való eltérését írja le, itt 0
- $\mathbf{P}_i(x_i, y_i)$: a leképezett pont helye a képen

Ezzel a lépéssel megkaptuk egy valós a kamera koordinátarendszerében felírt térbeli pont képsíkba leképezett pozícióját. A leképezés összefüggése a képtorzítás miatt nem egyértelmű vissza irányba: a torzítási képlet egy torzítatlan ponthoz ugyan csak egy torzítottat rendel, de ez ellenkező irányban már nem igaz. Ilyenkor egy képponthoz a képlet alapján több pont is tartozhatna, azonban ezek közül csak egy igazi megoldás, a többi a modellből ered. A leképezési képlet visszafelé nemlineáris egyenletrendszerre válik, aminek megoldása összetettebb, és számításigényesebb, mivel csak numerikus megoldás lehetséges. Amint azt azonban később látjuk a grafikus kártya működése miatt, a vissza irányt majd nem kell megtenni a torzítás megszüntetéséhez.



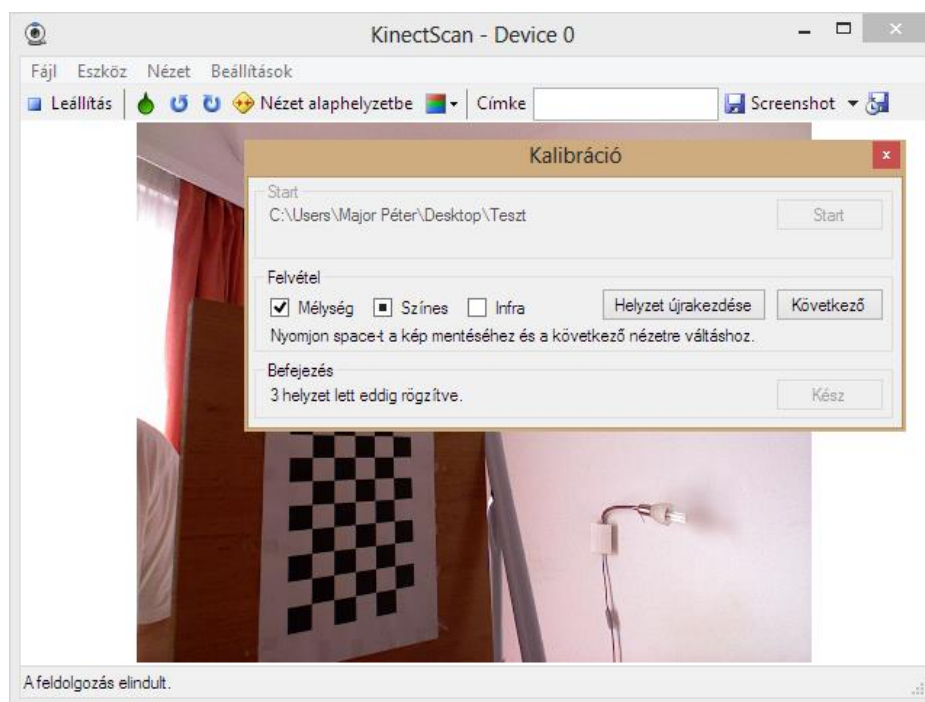
7. ábra A kamera leképezési modellje

Ez a modell mind az infravörös, mind a színes kamera leírására alkalmas. Mivel a kamerák optikai tulajdonságai erősen különböznek, így a transzformációkat, torzításokat leíró jellemzők is erősen eltérőek lehetnek. Mivel a két kamera különböző nézőpontból látja a világot, ezért az általuk leképezett képeken a pontok összekapcsolása további számításokat igényel.

3.3 A kalibráció menete

3.3.1 Kalibrációs felvételek elkészítése

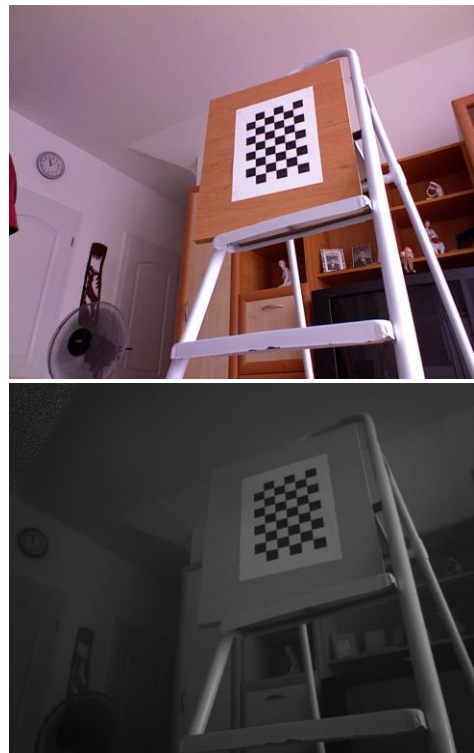
Ebben a részben a kalibráció menetét fogom bemutatni. A kalibráció elvégzése előtt több módszert is elemeztem, amelyet mások használtak [15, 16, 17]. A legtöbb módszer a mélységképet használja fel a kalibrációhoz, amely azonban nem oldható meg jól, annak erős zaja, torzítása, és az autofókusz okozta elcsúszások miatt, ezért én egy másik módszert választottam, amely az infravörös kamera képéből indul ki. A két kamerát először külön-külön kell kalibrálni, amelyet Zhang módszerével [5] végeztem el. A kalibrációhoz szükséges egy sakktábla minta, amely általában négyzetekből épül fel, de téglalapok is alkalmazhatóak. A mezők száma tetszőleges lehet, de ha egyik irányban páros a másikban páratlan a mezők száma, akkor a tábla nem lesz forgásszimmetrikus, így annak helyzete a kép alapján pontosan megállapítható. Azok a szoftverek, amelyek maguk végzik el a sakktábla helyzetének megállapítását, általában elvárják a páros x páratlan felépítést. (Amikor a sakktábla helyét nekünk kell meghatározni a képeken, a sarokpontok bejelölésének sorrendje megadja a tábla irányát is.) A sakktáblamintára azért van szükség, mert a mezők sarkai jól azonosíthatóak a kontrasztos mintán, ami lehetővé teszi azok méretének ismeretében a leképezést leíró konstansok meghatározását. A megoldás során két koordináta-rendszer közötti kapcsolat feltárása a feladat. Az egyik koordináta-rendszer a sakktáblához van rögzítve (pl. egyik sarkához), a másik pedig a kép koordináta-rendszere, amelynek origója a bal felső sarokban található, tengelyei pedig a pixelek tárolásának megfelelően balról jobbra, illetve fentről lefele kerülnek elhelyezésre. Mivel a pontok helyzete a sakktábla és a képp koordináta-rendszerben is ismert, valamint a köztük felírható transzformációs egyenletek paraméteres formában rendelkezésre állnak, így lehetőség van a paraméterek kiszámításra. Mivel a pontpárok pozíciója zajjal terhelt, ezért csak akkor lehet megbízható, pontos eredményre jutni, ha azokból elegendő számú áll rendelkezésre. Mivel a transzformáció nem-lineáris és a feladat túlhatározott ezért az csak összetett számításokkal kezelhető.



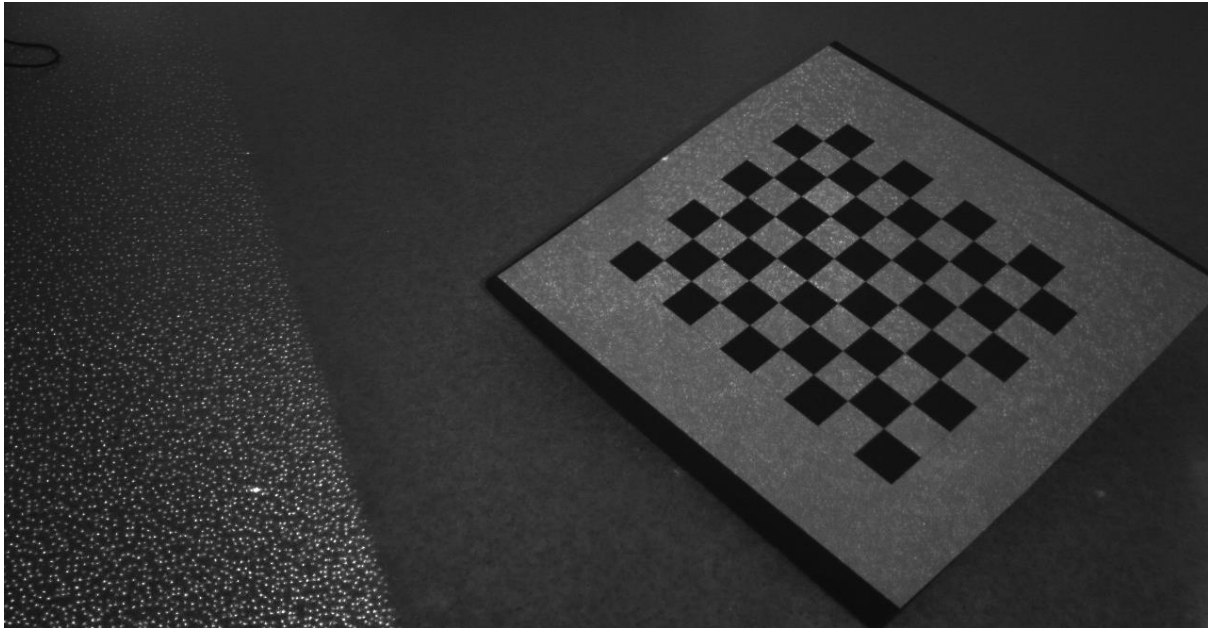
8. ábra A KinectScan kezelőfelülete kalibrációs módban

A kamerák kalibrációját a MATLAB segítségével végeztem el. A programcsomaghoz készült ingyenesen elérhető Camera Calibration Toolbox [6] implementál több kalibrációs módszert is, és megfelelően testre szabható. Mivel a modul nyílt forráskódú, így megfelelően módosítható, bővíthető. A kalibrációhoz egy saját MATLAB szkriptet készítettem, amely minimális felhasználói beavatkozás mellett elvégzi a Kinect kalibrációját. A kalibrációhoz a sakktábláról különböző helyzetekben felvételeket kell készíteni. Egy-egy helyzetben – az általam kidolgozott módszerrel – színes, infravörös és mélységfelvételekre van szükség. Ezek elkészítése a Kinect elérése miatt nem végezhető el időben egyszerre, ugyanis közben meg kell változtatni a Kinect működési módját, amely azonban időigényes művelet, akár több másodpercbe is telhet. Az átváltás után pedig nem lehetséges azonnal képet készíteni, mert még az expozíciós időt is be kell állítani a készüléknek. Ez utóbbi elsősorban a színes képek készítésénél fontos, ahol több másodperc is eltelhet a kezdeti beálláskor. A képek készítéséhez az 1280x1024-es felbontású színes és infravörös, valamint a 640x480-as mélység módokat használtam. Ezek mérete eltér a méréskor alkalmazottól, de ez nem probléma, mert az arányos nagyítás miatt könnyen átszámíthatóak a koordináták. A korlátozott sáv szélesség miatt ezen módok csak egymás után aktiválhatók, egyszerre nem (sőt a rendszer feldolgozó egysége a színes és infravörös csatorna párhuzamos olvasását sem teszi lehetővé, csak a színes és mélység kombináció használható). A képek elkészítését a programba épített külön segédeszköz segíti, amely a program beállításait a kalibráció idejére a megfelelőre változtatja (8. ábra). A felvételek készítése esetén a nyers adatok kerülnek lementésre, torzítás korrekció, visszavetítés nem történik, hiszen ezen műveletek paramétereit kellene megállapítani. Az infravörös képek elkészítése során problémát okoz a beépített infravörös projektor, amely automatikusan bekapcsol az infravörös kamera elindításakor. Erre két megoldást találtam:

- A gépet napfény jelenlétében kalibráljuk, például egy ablak mellett, úgy hogy a fény jól megvilágítsa a táblát. Ha elég erős a napfény, akkor a minta, amit a projektor kivetít nem lesz látható, így egyszerűen csak le kell menteni a képeket. Ha a napfény nem elég erős, akkor el kell takarni az infravörös projektort. (9. ábra)
- A infravörös projektort homogén fényforrássá alakítjuk, ez elérhető például egy átlátszó műanyag fólia összehajtogatásával és az érzékelő elő helyezésével. A rétegelt fólia csak minimális mértékben csökkenti a fényerőt, de viszonylag homogén fényforrást ad. A projektor fényereje ebben az esetben elég erős ahhoz, hogy magában elég fényt juttasson a sakktáblára. Ezt a megoldást csak akkor javasolnám, ha nincs infravörös fényforrásunk, mert a megvilágítás inhomogén lesz, és enyhe zaj is látható lesz. (10. ábra)
- Külső, mesterséges infravörös fényforrást is lehet alkalmazni. Ez az optimálmegoldás, ugyanakkor külön felszerelést igényel.



9. ábra A kalibrációs minta: a) színes, b) infravörös kamerával és természetes infravörös fényforrással



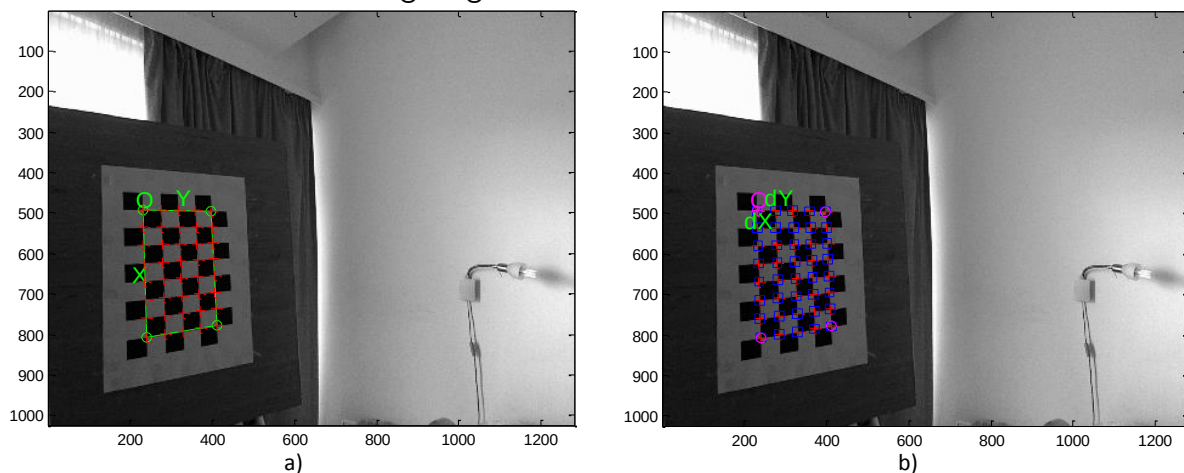
10. ábra Az infravörös projektor fényének homogenizálása rétegzett átlátszó fólián való fényátbocsátás segítségével, ezen a felvételen a projektor csak részben van kitakarva, a bal oldalon a normális intenzitás látható, míg tőle jobbra a kitakarással kapott eredmény. A kitakarás hatása ugyanakkor nem tökéletes, hiszen halvány szemcsék a jobb oldalon is láthatóak

Én az első két megoldást használtam. Mivel az automatika hajlamos nagy expozíciós időt alkalmazni az infravörös képek nagy felbontású olvasása esetén, ezért fokozottan ügyelni kell a minta és kamera helyzetének biztosítására (ez csak megfelelő állvány segítségével érhető el, így kézben tartott minta nem alkalmazható). Az infravörös képekhez hasonlóan történik a színes képek elkészítése is. Ügyelni kell arra, hogy a két kamera látótere különbözik, így a minta lelóghat az egyik képen, míg a másikon tökéletesen belefér a látható zónába. Az infravörös kamera látótere kisebb, ezért a kalibrálás során a rendszer a mélységképpel kezdi egy helyzet felvételét, mert ha abban elfér a minta, akkor általában a többi működési módban is (és így az is látható, hogy a mélységérzékelés az adott távolságban jól működik-e).

A mélységfelvételek elkészítése több körülményt igényel. Ennek az az oka, hogy szemben az infravörös és színes képekkel, a mélységkimenet igen zajos. A véletlen zaj csökkentése kétféle módon történik időben és térben (pontosabban síkban). A képnek mindig van számos olyan pontja, amelyek mélységi láthatósága bizonytalan, így nem szerepel róluk információ minden képkockán. Ennek kiküszöbölésére egy átlagoló algoritmust írtam, amely minden pixelre összeveti több képkocka adatait, és ha van rendelkezésre álló érték, akkor azok összegzésre, majd átlagolásra kerülnek. Ezzel a módszerrel elég, ha egy képkockán szerepelt a sorozatból az adott pont. Mivel a Kinect nem lineáris skálán jelöli a mélységet, ezért a mélységértékeket egy referencia Kinect kalibrálása során kapott paraméterekkel valós távolsággá alakítom, majd mentés előtt ugyanazzal a képlettel vissza. A referencia kalibráció elvégzését majd később ismertetem. Az átlagolással ugyan jelentősen csökken a vibráló, hol látható, hol nem látható területek aránya, azonban ez nem oldja meg a pixeleken megjelenő zaj problémáját teljesen. Ezért a képek további feldolgozása előtt egy Gauss szűrést is végzek. Mivel a sakktábla felülete sima, ezért Gauss szűrés nem torzítja annak alakját, viszont megszünteti a mélységképen a Kinectre jellemző függőleges hasábokat és a csak kevés képkockáról átlagolt adatok kiugrásait.

A képek rögzítése Windows Bitmap formátumban történik, kivéve a mélységképeket, ott ugyanis az átlagolásból keletkező pontosabb értékek miatt 32 bites lebegőpontos formában kerülnek tárolásra az információk. A kalibrációs modul végigvezeti a felhasználót a kalibráció egyes lépésein, és segít a képek elkészítésében. Az egyes helyzetek kiválasztása esetén érdemes ügyelni arra, hogy olyan képeket készítsünk, amelyeken a minta erősen oldalirányból látható (természetesen a sakktábla minden mezőjének még azonosíthatónak kell lennie), mert ezekből pontosabb eredmények nyerhetők. Az is fontos, hogy a kép teljes területéről kapjunk információt, tehát a mintát a kép minden részére el kell juttatni. Ügyelni kell arra, hogy a minta megfelelően sík legyen, így ha egy papírlapra kinyomtattuk azt rá kell ragasztani egy merevebb lemezre. Amikor legalább 20 helyzet rögzítésre került, akkor elkezdődhet a kalibráció a MATLAB-ban.

3.3.2 Kalibráció MATLAB segítségével

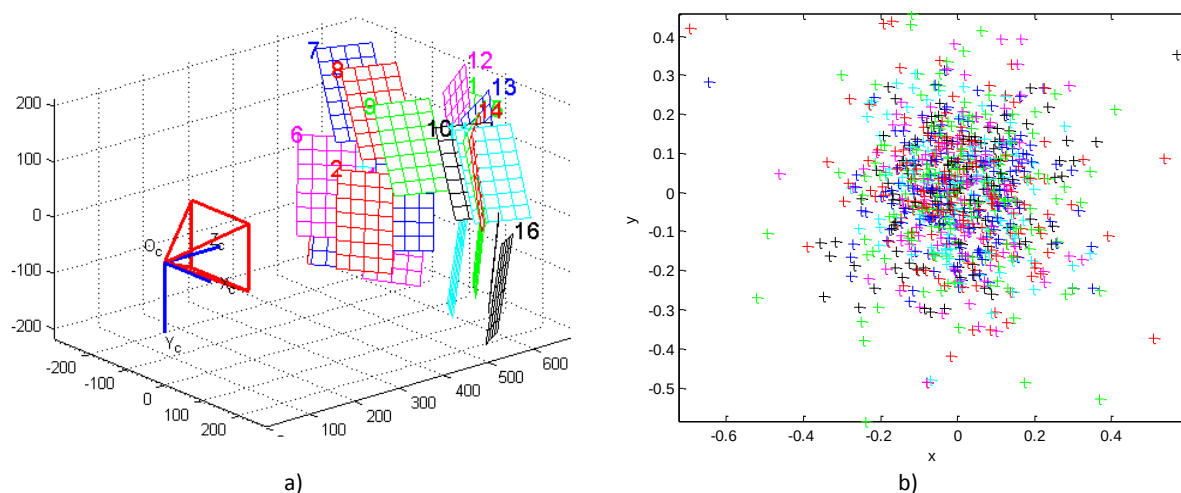


11. ábra A sakktábla csomópontjainak megkeresése: a) a sarokpontok kézi kijelölése alapján jósolt sarokpozíciók, b) a sarokpontok pontos helye a keresés után

Először meg kell adni az elkészített képek helyét, amelybe a kalibrációs modul azokat elhelyezte. Itt külön mappába kerülnek a színes, infravörös és mélységi felvételek. Ezeket az általam írt MATLAB szkript beolvassa, és listát készít róluk. A képek összepárosítását a megfelelő fájlnevek használata biztosítja. Ezután meg kell adni a rács méreteit. Én két kalibráló sakktáblát is használtam: kezdetben egy 6x9-es méretű mintát 25x25 mm-es mezőkkel, később egy 8x9-es méretűt 50x50 mm nagyságúakkal. A következő lépésben meg kell adni, milyen nagyságú területen keresse a csomópontokat a rendszer a várt pozíciókhoz képest. Ezt érdemes akkorára állítani, mint a négyzetek mérete a képeken. Ez azért is fontos, mert a minta négy sarkát a felhasználónak kell a képeken megjelölnie, ami problémát jelenthet, ha igen pontosan kell elvégezni. A mezők számának megadása nem szükséges, mert a rendszer azokat magától megszámlolja. A négy sarok és a mezőszám ismeretében a rendszer megkeresi sarkokat a képen, amelynél ismét használatra kerül a megadott keresési ablak mérete. Igen nagy torzítással rendelkező kamerák esetén a sarkok megkereséséhez meg kell adni egy becsült értéket a radiális torzítási együtthatókra, de erre a Kinectnél nem volt szükség (a radiális torzítás jelenléte szabad szemmel látható, de még elfogadható). A sarkok kijelölési sorrendje biztosítja a MATLAB számára a minta helyzetének egyértelmű definícióját, ezért ügyelni kell arra, hogy ezt mindig ugyanabban a sorrendben végezzük el. A rendszer jelzi a képen az általa megtalált sarkok helyét. Amennyiben néhány képen egyes sarkok kijelölése nem megfelelő, akkor annak korrigálására van lehetőség automatikus módon is (lásd később). A sarkok keresését úgy

implementáltam, hogy biztosítsa azok helyes kezelését akkor is, ha a képen a fekete vagy fehér négyzetek összeönnének (fényviszonyoktól függően).

A sarkok megkeresése után a rendszer kiszámolja a kamera torzítási paramétereit, a fókusztávolságokat, és optikai főtengely helyét, illetve ezen értékek bizonytalanságáról is ad információt. Ezek ismeretében a rendszerrel visszavetíthetők a pontok a képekre, és meghatározható az átlagos pixelhiba, azaz mennyivel kerültek odébb a tökéletes mintapontok visszavetítés után az eredetileg megtalált sarokpontokhoz képest. A visszavetítés egyben felhasználható a pontok kijelölésének pontosítására, mert új keresési kiindulópontokat szolgáltat. Például ha egyik képen megcsúszott a kezünk, és nem sikerült a sarkok pontos megjelölése, akkor a rendszer az első kalibrációs számítások után képes a pontok kijelölését javítani. A nagyobb torzítások esetén is fontos szerepet játszik az újravetítés, hiszen a torzítással visszavetített keresési kiindulópontok közelebb állnak a valódi pozíciókhoz. A visszavetítési hibagrafikonon is ábrázolható (12. ábra/b), amelyen látható a kalibráció minősége: ha sok olyan pont van, amelyek nagy visszavetítési hibával rendelkeznek, akkor a kalibráció nem elég pontos. Hogy ha csak néhány kiugró pont van, az általában azt jelenti, hogy ott vagy a pontok keresése hozott rossz eredményt (pl. zaj miatt vagy azért, mert nem jól jelöltük meg a sarokpontokat), vagy a kalibráló minta rossz. Amint az az ábrán látható, ebben az esetben a visszavetítés kis hibát mutatott. A kalibrációt újra lefuttatva, az újravetítésre és az előzetes eredményekre alapozva pontosabb, optimalizált eredmények nyerhetők. A színes és infravörös kamerák kalibrációja ugyanúgy történik. Az egyes kamerák kalibrációja után az eredményeket külön fájlokban tárolom, amelyek a sztereo kalibráció során kerülnek felhasználásra, illetve a Kinect kalibráló szkript lefutása után külön vizsgálhatóak.



12. ábra a) A sakktábla térbeli pozíciójának meghatározása a kamera koordinátarendszerében (mm), b) A sarokpontok kalibrálási paraméterekben visszavetített pozíciója és a képen felvett valódi pozíció közti eltérés (pixel)

Eddig ismerjük a színes és az infravörös kamerák különálló paramétereit, de nem tudjuk azokat egymással összekapcsolni. Ezért toolbox sztereo kalibrációs algoritmusát is fel kellett használnom. A sztereo kalibráció során kiszámításra kerül egy transzformációs mátrix, amely segítségével az infravörös kamera koordinátarendszeréből átjuthatunk a színes kameráéba. Ehhez a gyakorlatban természetesen a mélységi információk is kellenek, hiszen nélkülük a képen látható pontok térbeli helyzete csak egy félegyenes erejéig definiált.

A sztereo kalibráció során nyert eredmények:

Az infravörös kamera intrinzik paraméterei	
Fókusz távolság	$[1161.95960, 1169.64938] \pm [6.70019, 4.94887]$
Főtengelypont	$[639.86540, 521.46052] \pm [5.49222, 6.91659]$
Torzítás	$[-0.04421, 0.06934, 0.00181, -0.00114] \pm [0.00450, 0.00781, 0.00180, 0.00153]$
A színes kamera intrinzik paraméterei	
Fókusz távolság	$[1051.45009, 1053.78104] \pm [5.82529, 4.30443]$
Főtengelypont	$[641.15443, 521.79049] \pm [3.16070, 3.81499]$
Torzítás	$[0.14354, -0.24478, 0.00384, -0.00032] \pm [0.00595, 0.01021, 0.00122, 0.00080]$
Extrinzik paraméterek	
Forgatási vektor	$[0.00271, 0.00793, 0.00667]$
Eltolási vektor	$[-25.95501, -0.03472, 5.28277]$

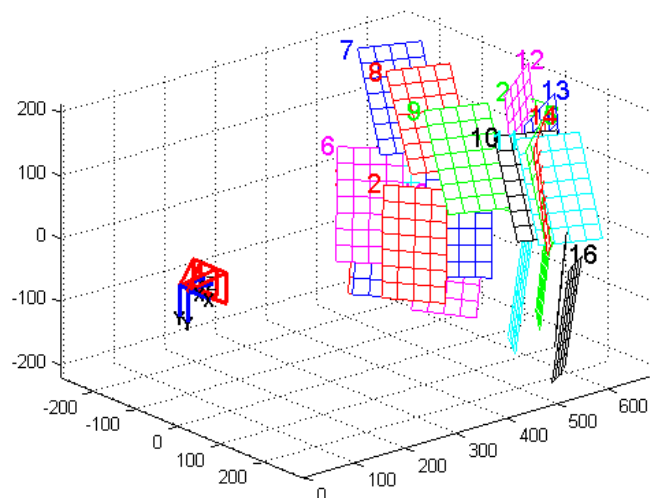
1. táblázat A kamera kalibráció eredményei

Az eredmények formátuma:

- Fókusz távolság $[f_x, f_y]$: a fókusz távolságok nagysága vízszintes és függőleges irányban pixelben.
- Főtengelypont $[c_x, c_y]$: a főtengelypont helye vízszintes és függőleges irányban pixelben.
- Torzítás $[k_1, k_2, t_1, t_2]$: az első két radiális (k) és tangenciális (t) torzítási együttható Brown modellje szerint.
- Forgatási vektor $[r_x, r_y, r_z]$: a mélységkamera koordinátarendszeréből a színes kamerába vivő transzformáció forgató komponensét leíró vektor, amely a Rodrigues féle formulával alakítható forgatási mátrixszá.
- Eltolási vektor $[t_x, t_y, t_z]$: a mélységkamera koordinátarendszeréből a színeskamerába vivő transzformáció eltolási komponensének vektora.

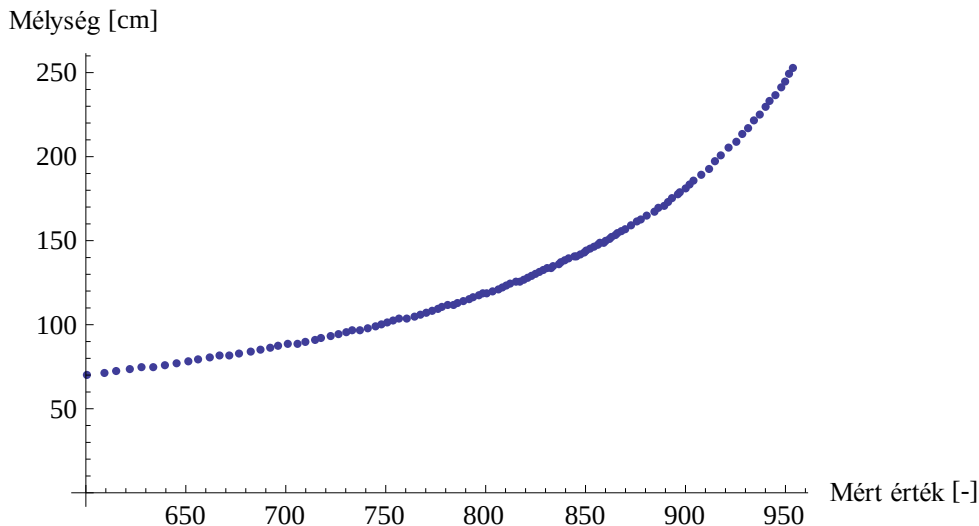
A bizonytalanságok 99,7%-os valószínűségre értendők (3σ). A sztereo kalibráció eredményeinek ismeretében a mélységkép alapján alkotott modellre textúra is illeszthető, ehhez azonban először még meg kell határozni a mélység karakterisztikáját is.

A mélységi kalibráció előtt meg kellett vizsgálnom, hogy a Kinect milyen mélységi karakterisztikával rendelkezik. Erre a célra egy matt felületű, a képsíkkal párhuzamos tengelyű henger palástját használtam. A henger megfelelő elhelyezésével biztosítottam, hogy a henger alkotói a képen függőleges irányban álljanak. A henger Kinect által mért távolságát úgy határoztam meg, hogy a henger mélységképen látható alkotói közül a legközelebbi mentén átlagoltam a mélységértékeket. Ez utóbbi megfelelő zajelnyomást biztosított. A



13. ábra A sztereo kalibráció eredménye (a távolságok mm-ben)

hengerfelület alkalmazása azért is előnyös, mert pl. egy sík tábla képsíkkal való tökéletes párhuzamosítása nehezen megoldható. A vonal menti átlagolás pedig biztosítja, hogy a véletlen és esetleges helyfüggő állandó hibát kiküszöböljem. (Természetesen időbeli átlagolás is alkalmazásra került.) Eközben mértem a henger valós távolságát is. Így az alábbi pontokat vettem fel:



14. ábra A Kinect mélységmérése során nyert értékek

Jól látható, hogy a kapcsolat erősen nemlineáris, így kérdéskén merült fel, hogy azt milyen paraméterekkel kellene leírni. Az alábbi megoldásokat fontoltam meg (az egyes megoldási módok után egy illesztéssel kapott képletet is megadtam, a függvényillesztést a Wolfram Mathematica szoftverrel végeztem el):

- Polinom segítségével egyszerűen leírható a kapcsolat, és az együtthatók is könnyen kiszámolhatók a legkisebb négyzetek módszerével. Hátránya az, hogy csak olyan távolságra ad megbízható eredményt, ahonnan voltak is adatok. Extrapolációra viszont alkalmatlan, ami azért kedvezőtlen, mert nagy távolságkülönbségek biztosítása a kalibrációnál nehézkes lehet (nincs elég hely, különböző nagyságú minták különböző távolságokra stb.).

$$p(x) = -1879.44 + 8.19864x - 1.16564 \cdot 10^{-2}x^2 + 5.66001 \cdot 10^{-6}x^3 \quad (6)$$

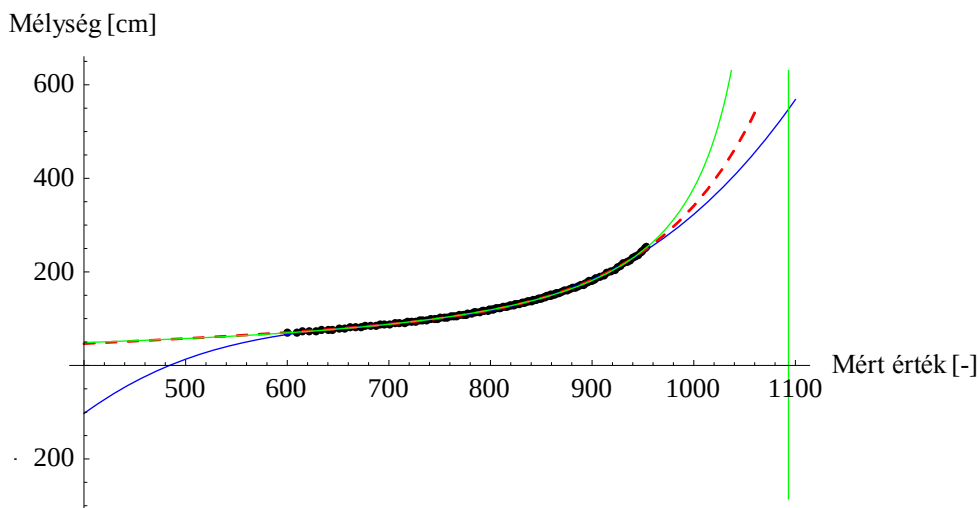
- Modellkezés egy megfelelő függvényterben: a modell első ránézésre hatványfüggvény jelleget ölt, az illesztési kísérletek azonban azt mutatják, hogy nagyobb távolságkülönbségek esetén nem lehet megfelelően elvégezni az illesztést. A második esetben exponenciális görbével próbálkoztam, amely a felfele ívelő szakaszon jó eredményt ad, azonban a kezdeti lapos szakaszon nem. Ekkor a választás egy konstans, egy elsőrendű és egy exponenciális tag összegére esett, ezzel már igen jól közelíthető a pontthalmaz. Ennek ellenére egy ilyen karakterisztika elég valószínűtlennek tűnt a működést figyelembe véve.

$$e(x) = 5.40092 + 0.0106353 e^{\frac{x}{100}} + 0.101291 x \quad (7)$$

- A végső és alkalmazott modell egy eltolt hiperbolikus tag és egy konstans összege lett. Azért esett erre a választás, mert tökéletesen ráilleszthető a pontokra, és mérésekkel ellenőrizve azt kaptam, hogy extrapoláció esetén is pontos marad. Nemlineáris függvényillesztés nem végezhető a legkisebb négyzetek módszerével, és a kezdő értékeket is meg kellett adni.

$$h(x) = -2.33458 - \frac{35498.8}{x - 1093.09} \quad (8)$$

A kalibráció során kérdés lett, hogy a 11-bites mélység felírásnál, ami 0..2047 tartományban vehet fel értékeket, hova fog kerülni a hiperbolikus tag szingularitása. A próbák során láthatóvá vált, hogy az értékek 950 fölé nehezen mennek, mert olyan messze nem lát el a Kinect. Ahová pedig nem lát el, oda a 2047-es értéket rendeli. A kezdeti próbák során a szingularitás 1000..1100-as tartományba került, amiből arra lehet következtetni, hogy 1023-nál kellene lennie, és a tizenegyedik bit egy flag, ami jelzi, hogy lát-e a rendszer abban a pontban valamit. Az illesztések során kapott eredmények grafikonon ábrázolva:

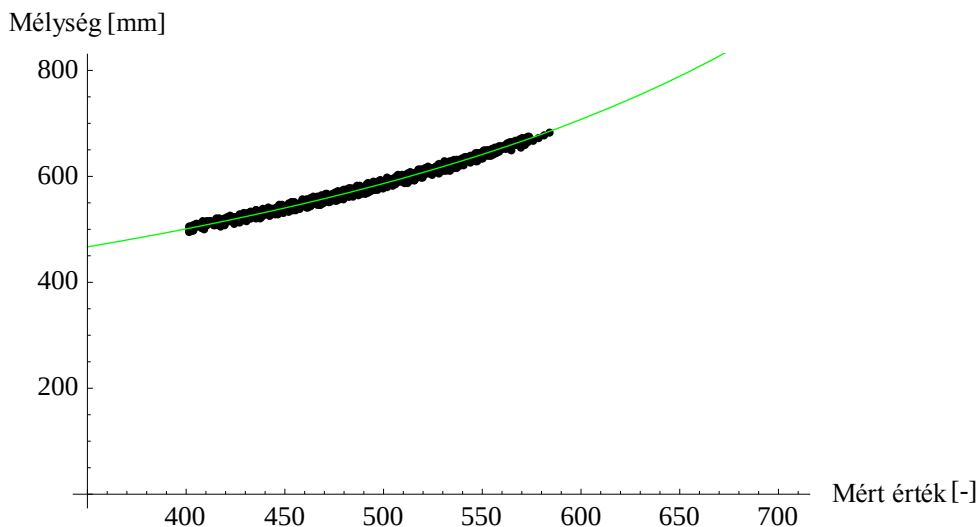


15. ábra A vizsgált mélységi függvények: kék) polinomfüggvény, piros) exponenciális és lineáris tag, zöld) hiperbolikus függvény

Amint az a grafikonon látható, mindhárom függvény jó közelítést ad a vizsgált tartományon, azonban praktikussága (egyszerű, gyorsan számítható) miatt a hiperbolikus függvényt választottam. A Kinect felhasználók az internetes források alapján több különböző megoldást alkalmaznak, ezek közül legelterjedtebb a hiperbolikus megoldás. A kézi kalibráció hibája, hogy a távolságot nem ugyanattól a viszonyítási ponttól méri, mint amit a sztereo kalibráció is használ, ennek kiküszöbölésére a mélységfüggvény előállításával bővítettem ki a Kinect kalibráló MATLAB programomat.

Mivel a mélységi kamerák kezelésére a Camera Calibration Toolbox nem képes, ezért az itt elvégzendő számítások kódolását magam végeztem el. Ehhez felhasználtam, hogy a rendszer minden kalibrálandó kép esetén megadja a kamera és mintához rögzített koordináta-rendszer közötti átszámításhoz szükséges forgatási és eltolási vektorok hosszát. Ez a többi kalibrációs paraméter ismeretében lehetővé teszi, hogy kiszámoljuk, hogy az egyes sarokpontok hova esnek a mélységi képen. Először az algoritmus felveszi az egyes sarokpontok helyzetét a minta koordináta-rendszerében, és ezeket áttaszformálja a kamera koordináta-rendszerébe. Ezután a torzításokat és projekciót is elvégezzük, amivel az infravörös képen értendő képkoordináták előállnak. Ezután már csak ki kellene olvasni a mélységi kép megfelelő pozícióiban felvett értékeket. Ez azonban így nem lenne pontos, ugyanis a rendszer tanulmányozása során kiderült, hogy a mélységi és infravörös képek nem köthetőek össze egyszerű átméretező transzformációval. Ennek oka, hogy a Kinect levágja az infravörös kép széleit, mielőtt előállítja a mélységi kimenetet. A vizsgálatok arra is fényt derítettek, hogy a levágás nem egyenlő szélességű az egyes oldalakon és egy enyhe nem aránytartó nagyítás is történik. Mivel a pontos implementációt a Microsoft nem publikálja, így ennek pontos oka nem ismert. A jelenlegi kalibrációs módszerben ezen eltolási és nagyítási paraméterek

manuálisan kerülnek megállapításra, az egyező helyzetből felvett mélységi és infravörös képek egymásra igazításával. (A nagyobb pontosság esetében ehhez felhasználtunk egy térbeli sakktáblát is, ahol a megfelelő mezők kiemelkednek. Ennek a térbeli sakktáblának részletgazdagsága és több nézetből felvett képei segítik az infravörös és mélységi képek közötti kapcsolat pontos megállapítását.) Ezután a rendszer, ezt a módosítást figyelembe véve, lekérdezi a sarokpontokhoz kapcsolódó mélységértékeket. Mivel a kiszámolt pozíciók a képen nem lesznek egészek, ezért a lekérdezés bilineáris interpoláció segítségével történik. A rendszer az így kapott értékpárokat összegyűjti két vektorban. A valódi és a Kinect által mért értékek grafikonon ábrázolva az alábbi formát adják:

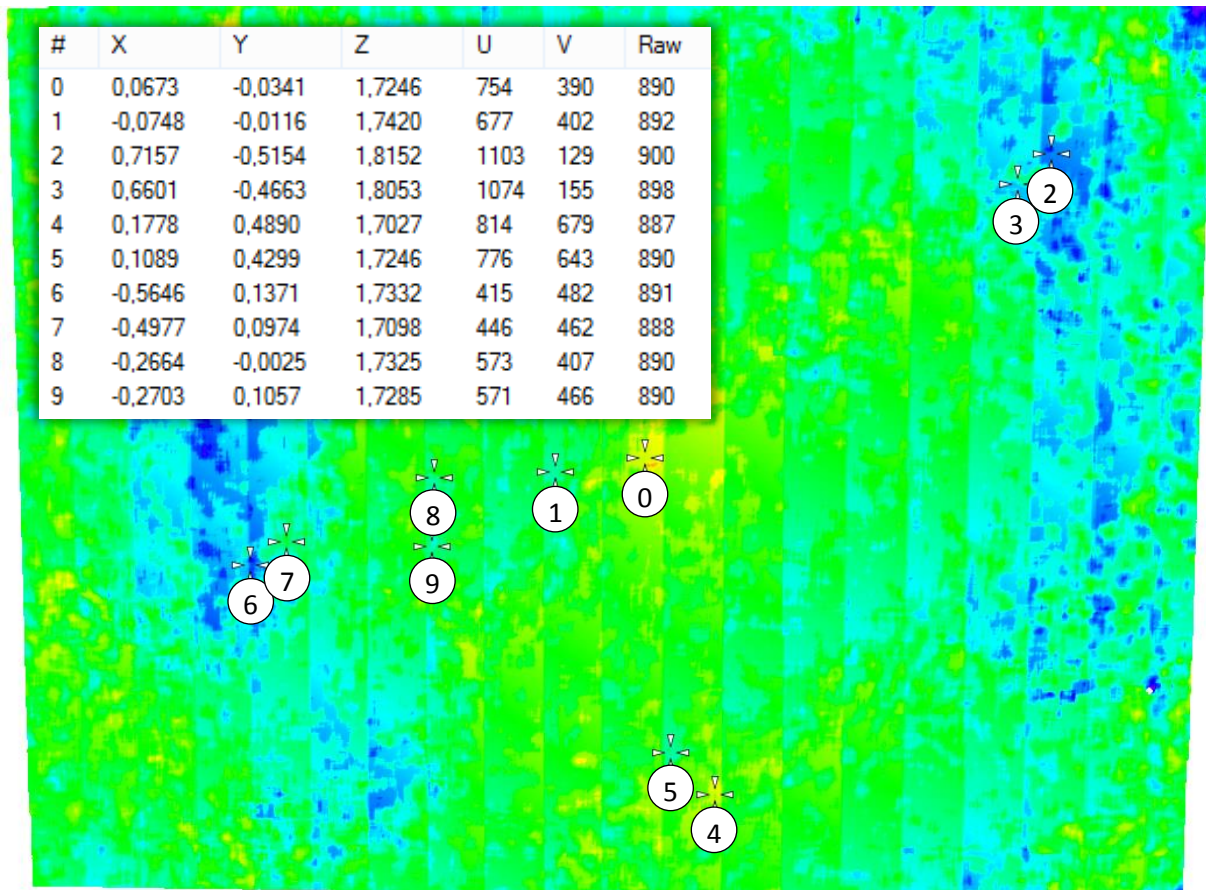


16. ábra Mélységfüggvény MATLAB használata mellett

A MATLAB-os kalibráció során azt tapasztaltam, hogy a minta pozíciójának háromdimenziós meghatározása akkor a legpontosabb, ha az nagy szöget zár be a képsíkkal. Ha a minta majdnem párhuzamosan helyezkedik el, akkor az adott kép alapján kapott pontokat ábrázolva a görbe egymás feletti vízszintes görbedarabokká esik szét néhány extrém esetben. Amint a grafikonon is látható, a pontok egy szélesebb sávban húzódnak. Ennek oka vizsgálataim alapján a Kinect autófókusz rendszere, nagyobb távolságok esetén a görbén emiatt kisebb törések is megjelennek. A MATLAB program ezután az eredményeket egy XML fájlban tárolja, amelyet a feldolgozó program is tud kezelni.

3.4 Mérési eredmények elemzése

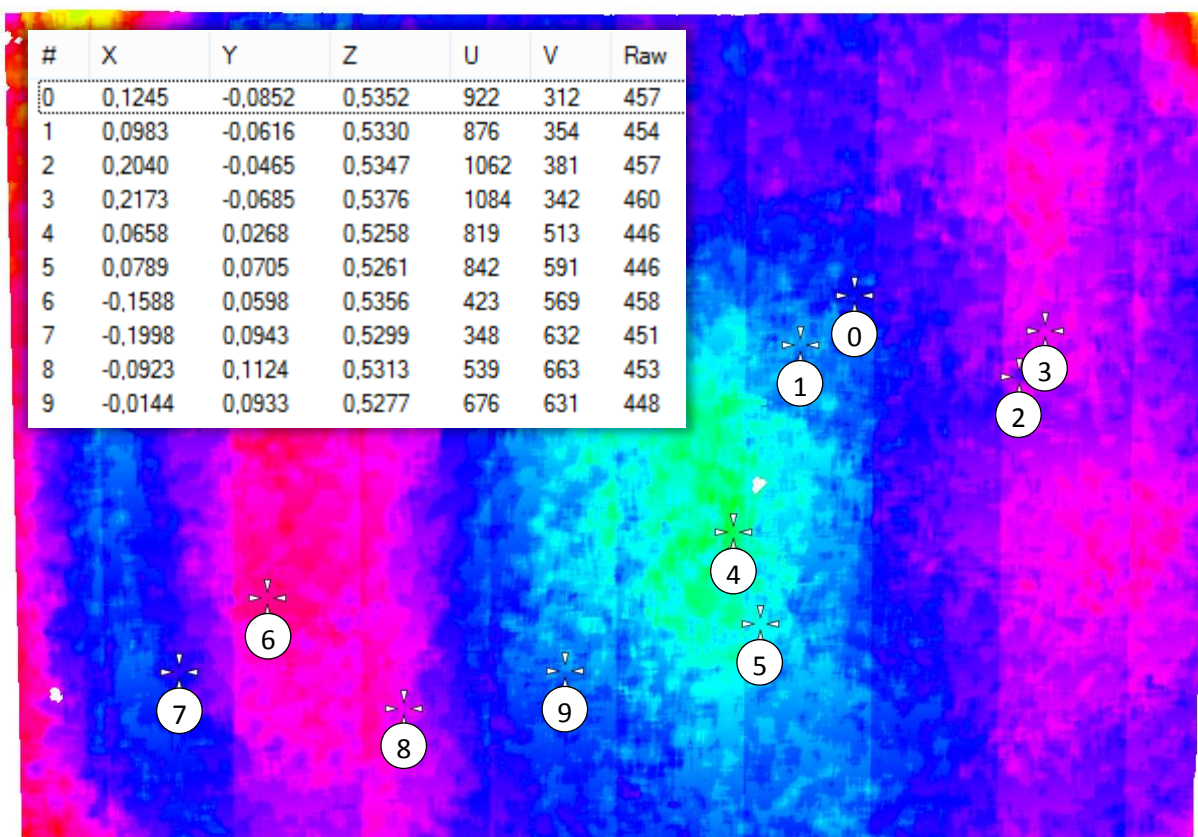
A mérési eredmények elemzésének tárgya elsősorban a mélységmérés ellenőrzése, hiszen a létrehozott modellek precizitása ebben az irányban lesz a legkisebb. Ennek talán legegyszerűbb módja, ha a Kinecttel egy sima felületről készítünk felvételeket különböző távolságokból, majd azokat elemezzük. A Kinect karakteristikájából adódóan arra számíthatunk, hogy a pontosság a távolság növekedésével rohamosan romlani fog a hiperbolikus görbe felszálló ága mentén. Ezt a jelenséget többen is megfigyelték a kalibráció során [15, 17]. Én a méréshez készítettem egy „szondák” funkciót, amely segítségével a valós időben generált modell pontjainak helyzete a modell felületéhez „tapadó” szondákkal lekérdezhető.



17. ábra A Kinect pontossága nagy távolság esetén egy sima felületű fal segítségével tesztelve, a táblázat oszlopai: X, Y, Z: a kamera-koordináta-rendszerben mért pozíció (m), U, V: a képen kiválasztott hely (pixel), Raw: a Kinect által adott érték.

Amint az látható a távolság növekedésével a mérési pontosság jelentősen romlik: míg 40-50cm távon 0,5mm-es pontosság is elérhető, addig 2 méter távolságában már 3-4cm nagyságú felületi hibák is jelentkeznek. A képeken az is megfigyelhető, hogy a Kinect mélységi karakterisztikája nem teljesen megegyező a teljes kép területén, hanem az optikai tengely távolságának függvényében enyhén változik. Ennek hatása közeli mérések esetén, mint amelyek az alkalmazásokban bemutatásra kerülnek, kisebb (a megfelelő szűrési eljárások alkalmazása mellett), a szoftver továbbfejlesztési lehetőségei közé ugyanakkor felvehető ezek lekezelése is. Ennél sokkal zavaróbb a már említett függőleges sávok jelenléte. Ezek szintén inkább nagyobb távolságoknál okoznak gondot. Mivel a sávok helyzete, szélessége időnként megváltozik, így ezek eltávolítása csak speciális algoritmussal lenne elvégezhető.

Kis távolságok esetén a hiba mértéke is sokkal kisebbnek adódik. Itt az egyenlőtlenségek néhány milliméter nagyságúak lesznek. A tapasztalatok alapján elmondható, hogy precízebb mérések elsősorban 40-60cm távolságban végezhetőek, ezen túl a hiba fokozatosan nő, a maximálisan elérhető 6-7 méteres távolságban a felületi hibák értéke elérheti a 10cm-t is. Ebben a tartományban inkább csak tájékozódásra használható a rendszer. A Kinect PC-khez készült változatán elérhető egy ún. makro mód, amelyben még közelebből készíthetünk pontosabb felvételeket. Ha a Kinectet közelebb helyezzük a mérés tárgyához, akkor a látótérben egyre kisebb rész fog látszódni, ami korlátozza e módszer alkalmazási lehetőségeit. A felületi egyenlőtlenségek nagyságát én időbeli átlagolás, és Gauss szűrés segítségével csökkentettem.



18. ábra A Kinect pontossága kis távolságok esetén egy sima falfelülettel tesztelve

4 A képfeldolgozás lépései

Ez a fejezet a Kinect kezelését és az általa mért adatok alapvető feldolgozását kívánja bemutatni. Az itt bemutatott művelet sor a megfigyelt környezet felületi modelljének megalkotásához vezet. Az itt leírt művelet sor végrehajtására valós időben kerül sor, amelyet a hardveres gyorsítás tesz lehetővé. Ennek segítségével a rendszer valójában a Kinect által nyújtott képkocka sebességek többszörösét is képes lenne kezelni. Mivel a grafikus kártyára a hatékony háromdimenziós megjelenítés miatt szükség van, így nem jelent plusz erőforrásigényt a hardveres gyorsítás kivitelezése.

4.1 A képfeldolgozás vezérlése

4.1.1 A driver kiválasztása

Annak ellenére, hogy a Kinectet eredetileg játékvezérlőnek tervezték, a fejlesztők meglátták benne az egyéb lehetőségeket is. Mivel kezdetben a Microsoft nem adott ki drivereket a rendszer mellé, ezért a felhasználók maguk láttak neki azok elkészítésének. A támogatás kezdeti hiánya miatt fordulhatott elő, hogy a készülékekhez jelenleg legalább három eltérő driver, és ahhoz tartozó programozási interfész érhető el, amelyek erősségei jelentősen különböznek. A fejlesztés során mind a három drivert teszteltem és a következő tapasztalatokat nyertem:

- OpenNI driver [7]: ezt a drivert egy konzorcium, illetve magát a Kinect szenzorát gyártó PrimeSense készítette. Ez a driver egy middleware része, amely egyéb gesztusfelismerésre szolgáló eszközökkel együtt képes a Kinect kezelésére. Hátránya, hogy a gépigénye jelentős a Kinect használata közben, illetve az, hogy a keretrendszerében a hardverek inicializációja nehézkes és nem elég rugalmas. Előnye, hogy egy Kinectet ezzel a rendszerrel több program is elérhet egyszerre és relatíve jól hozzáférhetőek a beállítások is.
- Microsoft driver [8]: mivel a Microsoft is észlelte a Kinecttel kapcsolatban megjelenő nem játékfejlesztési lehetőségeket (és az ezekben rejlő piacot), így ők is kiadtak egy hivatalos drivert. Ezen driver előnye többek között az egyszerű terjesztés (automatikus telepítés a Windows Update driveradatbázisából), a megbízhatóbb támogatás és a stabilabb működés. Hátránya többek között a magas gépigény a Kinect használata alatt (valószínűleg a gesztusfelismerés és képkorrekciók állandó futása miatt), illetve az, hogy a Kinect számos funkciójához nem lehetséges vele a hozzáférés, ezek közül számomra kiemelt jelentőségű, hogy nem érhető el az infravörös kamera képe, amely kell a kalibrációs módszeremhez.
- OpenKinect driver [9]: ezt a drivert szabad fejlesztői közösség készítette a Kinect és az XBOX közötti kommunikáció lehallgatása és elemzése alapján. Ez a driver multiplatform és számos operációs rendszerre elérhető. Előnye a minimális erőforrás használat, és az, hogy számos olyan hardverelemhez is hozzá lehet vele férni (infravörös kamera, gyorsulásérzékelő, állapotjelző LED, további kimeneti módok stb.), ami a többi driverrel nem érhető el. Ez volt az első driver, amivel a Kinect 640x480-as felbontású mélység kimenetét is el lehetett érni, ami a fejlesztés elején még nem volt biztosított a többi megoldásnál.

A tesztek után a választás az OpenKinect driverére esett, mivel ez felelt meg legjobban az igényeknek. A driver használatához annak forráskódját le kellett tölteni, majd a megfelelő további modulok (pl. libusb, pthreads) beszerzése után azt le kellett fordítani. A telepítés egyszerűen elvégezhető az eszközközkezelő segítségével a driver helyének kiválasztása után. A Kinect

csatlakoztatása esetén a Windows automatikusan letölti és telepíti a Microsoft driverét, így erre az időre meg kell szakítani az internetkapcsolatot (vagy a Windows Update automatikus driver letöltési funkcióját kell letiltani). Ha a driver telepítése befejeződött, akkor azt már nem bántja a Windows.

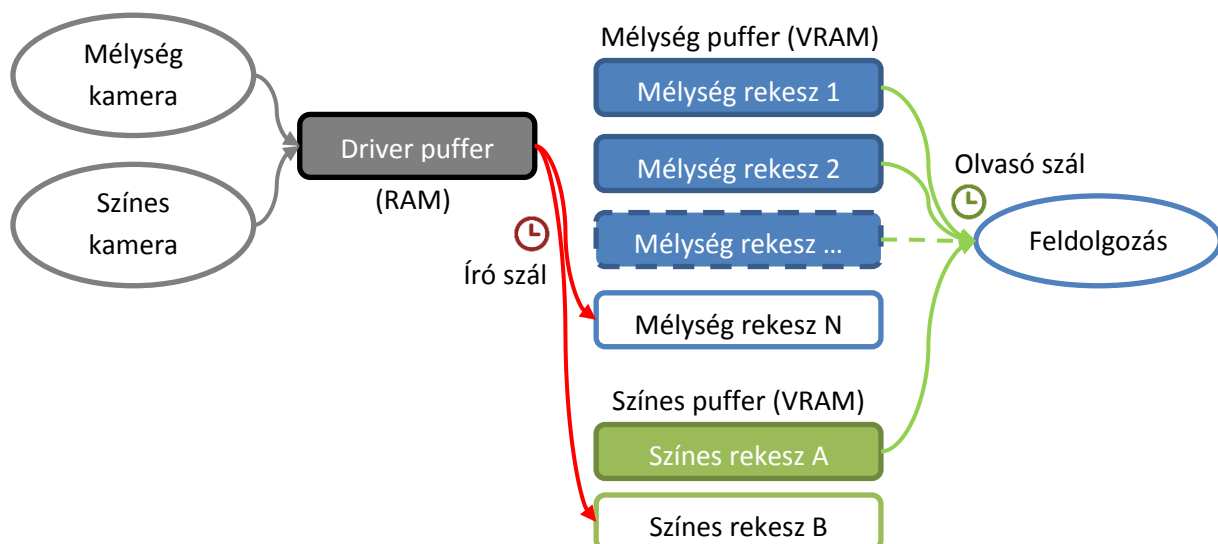
Ebben a bekezdésben néhány a hardverrel kapcsolatos megjegyzést foglalkozok össze, amelyek segítséget kívánnak nyújtani a dolgozat mellé benyújtott program használatához. Egy gépen lehetőség van több Kinecttel párhuzamosan dolgozni. Korlátozó viszont, hogy minden Kinectet külön USB gyökhöz kell csatlakoztatni a számítógépen. Ha ez utóbbi nem teljesül, akkor csak egyetlen Kinect lesz használható az adott gyökhöz lévők közül, és a kapcsolat sávszélessége is szűk lehet (eldobott képkockák). Fontos megjegyezni, hogy a Kinect normál üzemállapotban a teljes effektív USB 2.0 sávszélességet felhasználja, így arra az USB vezérlőre, ami a Kinectet kezeli, már nem lehet jelentősebb sávszélességet igénylő eszközt csatlakoztatni anélkül, hogy képkocka kihagyás történne. Bizonyos számítógépek (például sok noteszgép) USB vezérlői nem képesek a Kinectnek szükséges sávszélességet biztosítani, és akadni fog a kép (ennek oka az is lehet, hogy az adott USB vezérlőhöz kívülről nem látható belső perifériák is csatlakoznak, pl. webkamera, hálózati adapter). A Kinect kommunikációs protokolljának egyik gyengesége, hogy minden képkockát megpróbál elküldeni, akkor is, ha a sávszélesség nem elég nagy. Ez ahhoz vezet, hogy ha kicsi a sávszélesség, akkor a képkockák vége szinte mindig lemarad, és egy képet sem kapunk meg teljesen. A mélységképek előnyt élveznek a színesekkel szemben, így ha kicsi a sávszélesség a kettőnek, akkor javarészt csak mélységképeket fogunk megkapni (azokat teljes sebességgel), a színes képekből pedig nagyon keveset (pl. egy általam tesztelt notebooknál véletlenszerűen 4-6 másodpercenként egyet) vagy egyet sem. Így elmondható, hogy szűk sávszélesség esetén a Kinect nem használható (alacsony képkocka-sebesség sem érhető el), ugyanis az új képkockák megszakítják a régiek küldését. A Kinect három külön hardverelemként jelenik meg a számítógép számára, amelyeket külön kell telepíteni: Kinect NUI motor (ez a vezérlőegység, és amíg ez nincs telepítve, addig nem látszik a többi), Kinect NUI kamera, Kinect NUI Audio. Egy Windows installáción csak egy fajta Kinect driver lehet telepítve, a helyes működéshez az új driver telepítése előtt a régit le kell törölni.

4.1.2 A Kinect kezelése

Bár a Kinect többfajta driverrel is rendelkezik, és ezek funkcionalitása nem egyezik meg, a szenzor vezérlésének sémája nagyjából ugyanaz. Az első lépés a csatlakoztatott Kinectek számának lekérdezése, így megtudható hány Kinect is használható. A Kinectek azonosítása a sorszámuk alapján történik. A Kinectet használatbavétel előtt inicializálni kell, amely csak akkor lesz sikeres, ha az nincs használatban. E művelet a Kinectet alaphelyzetbe állítja. A használat után a Kinecttel a kapcsolatot megfelelő függvényhívással kell megszakítani, különben az nem definiált állapotban marad (ilyenkor a kamerák és feldolgozás hardveres oldalon nem áll le, ami felesleges energiafogyasztáshoz vezet). A Kinect feje egy motor segítségével előre-hátrabillenthető, de a motor nem folyamatos igénybevételre lett tervezve, így az állítások számát korlátozni kell. Az inicializáció során a Kinect megpróbál vízszintes helyzetbe állni (ehhez segítséget nyújt a beépített gyorsulásmérő), ezidő alatt fix alátámasztást kell biztosítani. Ha kapcsolatfelvétel sikeres volt, akkor elérhetővé válnak az egyes hardveregységek. A kamerák, illetve a központi adatfeldolgozó egység vezérlése úgy történik, hogy ki kell választani a kért működési módot két csatornán. Ezek a mélységi és a videó csatorna, amelyek tulajdonképpen nem felelnek meg egy az egyben a hardverelemeknek, hanem a következő funkciókkal rendelkeznek:

- A videó csatornán keresztül érhető el a színes és az infravörös kamera képe is. Ezek kétféle felbontásban 1280x1024@7Hz és 640x480@30Hz használhatóak. A nagyobb felbontás mellett szereplő képkocka sebesség akkor érhető el, ha csak ez a csatorna aktív.
- A mélységi csatorna csak mélységinformációt ad, amely 640x480@30Hz és 320x240@30hz módokban kapható meg.

A két csatorna egyszerre is használható, de a sávszélesség-korlátok miatt párhuzamosan csak 640x480-as felbontáson lehet képeket kapni (ilyenkor 30 FPS mellett). Ha az infravörös kamera működik, akkor az infravörös projektor is automatikusan elindul, az külön nem kapcsolható. A módok kijelölése után van lehetőség az adatfolyamok indítására, ez azonban több másodpercig is eltarthat, így nincs lehetőség gyors váltásra (pl. kalibrációhoz). Az adatfolyamokat a driver automatikusan fogadja és pufferben gyűjti. A megérkezett képkockák fogadásához egy ciklusban kell kérni a beérkezett képkockákat (pollozás). Hogy az emiatti processzorhasználat minimális legyen, én ezt egy külön szálon végzem, amely minden lekérdezés után 5 ms-ot vár, ez lehetővé teszi, hogy minden képkockát kis késleltetéssel megkapjunk, de a processzorhasználat minimális legyen. A rendszer működési módjából adódik, hogy a driver egy hívásra néha több képkockát ad át a belső puffer erejéig. A driver a képkockákat egy külön szálon futó visszahíváson keresztül adja át (mindig ugyanazon a szálon). A színes és mélységi képkockák nem szinkronban érkeznek, és teljesen eltérő képkocka sebesség is könnyen előfordulhat (pl.: képkocka eldobások a színes kameránál). Ezért az általam írt feldolgozó rendszer a képkockákat kettős pufferekben gyűjti, külön a színes és a mélységi képeket. Az általam kialakított kettős puffer a mélységi csatorna esetében kettőnél több képkocka tárolására is képes, ami felhasználható statikus vagy kvázistatikus mérések esetén átlagolásra. Az átlagolás további előnye, hogy a Kinect mélységi képén megjelenő ideiglenes lyukakat (ahonnan nincs mélységi információ) az átlagolás erősen visszaszorítja. Az átlagolás a színes kamerán nem indokolt, mivel azzal nem mérünk, és a kimenetén megjelenő zaj is csekély mértékű. A pufferek tartalmát a kamerákkal aszinkron futó szálon olvasom ki:



19. ábra A rögzített képkockák kezelése

A megoldás lehetőséget ad több feldolgozó szál párhuzamos működtetésére, de az így elérhető plusz teljesítmény kicsi, mivel a feldolgozás nagy része a grafikus gyorsító segítségével megy végbe, így az már amúgy is erősen adatpárhuzamos (SIMD). Mivel a pufferek beírása/olvasása

párhuzamosan történik, ezért folyamatosan figyelni kell az épp használatban lévő puffereket. A szoftver mind az olvasás, mind az írás alatt álló rekeszeket nyilván tartja, és eszerint választja ki az írások és olvasások célpontjait. Ez elsősorban akkor fontos kérdés, ha kettőnél több képkockát kell tárolni. Ilyenkor az írás körbeforgó módon történik, míg olvasásnál az átlagoláshoz az összes elemet kiolvasom, kivéve, ha az adott elem éppen írás alatt áll. Ez utóbbi azt okozza, hogy időnként a kijelölt képkocka-mennyiségnél eggyel kevesebb kerül átlagolásra. A pufferek kiolvasása után kezdődik meg a képek valódi feldolgozása.

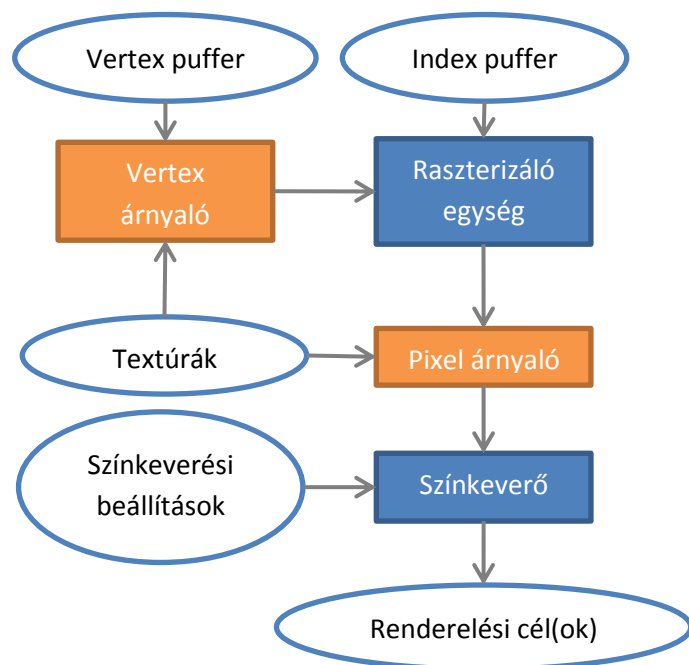
4.2 A képek feldolgozása – a hardveres képfeldolgozás

A képek feldolgozása a grafikus gyorsítón történik, amely programomban a Kinecttől kapott nyers színes, illetve mélységi adatokból különféle árnyalási technikákkal megjeleníthető, textúrázott háromdimenziós modelleket hoz létre valós időben. A grafikus kártyán való feldolgozás ismertetéséhez röviden áttekinteném annak architektúráját programozási szempontból. Ma már kidolgozásra kerültek olyan programozási eszközök, amelyek lehetővé teszik a grafikus kártya számítási kapacitásának kihasználását nem grafikai problémák megoldására is anélkül, hogy a problémákat grafikussá kellene átalakítani. Ezekre ebben az esetben azonban nincs szükség, mivel a megoldandó feladatok mind grafikusak, vagy ahhoz hasonló jellegűek. A programozás során a DirectX környezetet a .Net rendszer számára elérhetővé tévő XNA keretrendszert használtam, amely programozás szempontjából minden funkciót elérhetővé tesz, amit a DirectX kínál, de letisztult, jól strukturált és objektum-orientált formában.

A különböző grafikai feladatok megoldásának bemutatása előtt ismertetném egy standard rajzoló feladat megoldását DirectX rendszerben. A rajzoláshoz szükség van egy geometriai modellre, amely alapesetben csúcsok tömbjéből és a hozzájuk kapcsolódó indextömbből áll. Minden csúcst egy struktúra ír le, amely minimálisan azok térbeli helyét tartalmazza, de a legtöbb esetben ezek mellett textúra-koordinátákat, normálvektorokat, színinformációkat is magába foglal. Az előre elkészített struktúrák használata mellett lehetőség van saját létrehozására is, így tetszőleges számmal leírható információ továbbítható. Az indextömb azt adja meg, hogy az egyes kirajzolandó primitívekhez hányadik csúcsok tartoznak. A primitívek itt háromszögek vagy szakaszok lehetnek. Az indextömbök alkalmazása nagyon előnyös, ugyanis a legtöbb modell esetében egy csúcs több primitív részét is képezi, és mivel a csúcsokat rajzolás előtt transzformációknak kell alávetni, így elég azokat csak egyszer elvégezni. Mielőtt a tömböket alkalmazni lehetne, azokat fel kell tölteni a grafikus gyorsító memóriájába, ezt index- és vertexpufferek segítségével lehet megtenni. Ezen puffereket előre létre kell hozni, és fix méretüket is meg kell adni. Előny ugyanakkor, hogy ezek a pufferek bármikor frissíthetők új tartalommal, amennyiben az belefér a pufferbe. A modell információk mellett textúra információkra is szükség van, amelyek számára szintén előre le kell foglalni egy megfelelő méretű memóiahelyet a grafikus kártyán. A memóiafoglalás során a textúra mérete mellett annak típusát is meg kell adni. A képek tárolására többféle bitmélységű és csatornaszámú típus áll rendelkezésre, ugyanakkor a modern kártyákon már lebegőpontos leírás is használható, amely számos új alkalmazási lehetőséget hoz magával, hiszen az egész típusoknál csak 16 bites csatornaszélesség érhető el (maximum). A textúrákat használat előtt mintavételezőkhöz kell rendelni, amelyek azokból a megadott beállítások mellett olvasni képesek. Ilyen beállítások például a használandó interpoláció (pont, bilineáris, bikubikus), vagy a textúráról lelógó részek kezelése (ismétlés, korlátozás, tükrözés). Hasonlóan az index- és vertexpufferekhez, a textúrák tartalma is frissíthető. A frissítések során lehetőség van egy adott rész különálló frissítésére is. A pufferek előnye,

hogy az egyes képkockákhoz szükséges adatok közvetlenül, a grafikus memóriából érhetőek el, azokat nem kell ismételtlen átküldeni rajzolásonként. A számítások további input paraméterekkel is rendelkeznek, amelyeket a grafikus kártyán futó programok használhatnak. Ezek fontos jellemzője, hogy nem változhatnak meg rajzolás közben, csak azok között frissíthető tartalmuk. Ilyen „változók” definiálják a transzformációs mátrixokat, a számítások során alkalmazott vektorokat, logikai beállításokat stb. Ezek mellett még számos regiszter értéke is fontos szerepet játszik, amelyek a kártya működési módját befolyásolják. Ilyenek például a színkeverési beállítások, a kamerának hátat fordító háromszögek elrejtése (triangle culling), a használandó textúrák és pufferek megadása. Néhány hasonló paraméter beállítására csak a kártya inicializálása során van lehetőség, később az nem tehető meg. Ilyen például a képernyőpuffer mérete, színmélysége, a mélységpuffer típusa, a mintavételezési minőség stb. További fontos elemek még a renderelési célpontok, amelyek olyan képernyőn közvetlenül megjelenítésre nem kerülő memóriaterületek, amelyekre rajzolás lehetséges és amelyek később mint textúra alkalmazhatóak.

Ebben a bekezdésben a rajzolás folyamatát ismertetem. A rajzolás során először a vertexek feldolgozására kerül sor, amely ún. vertexárnyalók segítségével történik. Minden vertexárnyaló egy olyan függvény, amelynek bemeneti paramétere is visszatérési értéke is egy vertexstruktúra (a be- és kimenet típusa eltérhet). A vertexárnyalók végzik el a csúcspontok képernyő-koordinátarendszerbe való leképezését. A vertexárnyalók a modern kártyákon hozzáférnek a textúrákhoz is, amelyek kulcsszerepet játszanak a program számos részében. Ezután a feldolgozott vertexek és az index-puffer tartalmából előállításra kerülnek a kirajzolandó primitívek. Ezek az adatok ezután a raszterizáló egységbe kerülnek, amely megállapítja, hogy az adott primitív mely pixeleket fedi le, és a vertexek által tárolt információkat minden egyes pixelre kiinterpolálja. Ezen már pixelenként eltérő vertex-információkat ezután a pixelárnyalók kapják meg. A pixelárnyalók feladata, hogy meghatározzák a kirajzolandó pixel színét, ehhez az interpolált vertexinformációk mellett a textúrákat is igénybe vehetik. Ezek után az egyes pixelek a színkeverőbe kerülnek, amely meghatározza, hogy mi legyen a kimenetben a pixel színe az új kirajzolandó szín és annak helyét elfoglaló eddigi figyelembevételével. A legtöbb esetben, ha mélységpuffert is használunk, akkor csak azon pixelek kerülnek kirajzolásra, amelyek közelebb találhatók a kamerához.



20. ábra A rajzolás menete DirectX környezetben

Mivel a Direct3D kétdimenziós rajzolást nem támogat közvetlenül, ezért ha csak a pixelárnyalókra van szükség, akkor két háromszögből összerakott téglalapot rajzolok ki. A pixel- és vertexárnyalókat úgynevezett technikákba kell rendezni. Minden technika több renderelési menetből állhat, és minden menet egy vertex- és pixelárnyalót tartalmaz. A többmenetű renderelést én a

projekt során nem alkalmaztam, mert nem volt rá szükség. Mivel egy vertex- vagy pixelárnyaló több technikában is használható, így könnyen megvalósíthatóak különböző megjelenítési módok ugyanarra a modellre.

Végezetül néhány korlátozást ismertetnék, amelyeket figyelembe kellett venni a program elkészítése során. A HLSL támogatja a függvények használatát, de azok minden esetben inlineolásra kerülnek, így nem okoznak teljesítménycsökkenést. A grafikus kártyák ma általánosan nem támogatják az ún. dynamic braching-et, mert túl nagy overhaddel jár a kártya ütemezőjében. Ez azt jelenti, hogy a kártya az elágazások mindkét ágát minden esetben végrehajtja, de csak az egyik eredményét hasznosítja. Erre azért van szükség, mert egy modern kártya 1500-3000 árnyaló maggal rendelkezik, amelyek ugyanazon programot erős párhuzamosítással futtatják, és enélkül a számítások ütemezése, egyenlő ciklusszáma nem biztosított. A ciklusokon emiatt ún. unroll-t kell végezni, ami azt jelenti, hogy a gépi kódban nem igazi ciklus fog szerepelni ugrásokkal, hanem ugyanazon utasítások megismétlése. Ehhez persze ismerni kell előre, hogy hányszor fog lefutni egy ciklus, így tetszőlegesen sokszor lefutó ciklus sem hozható létre. Az árnyaló programok hossza is korlátozott, és a különböző műveletek (aritmetikai, textúraolvasási) maximális száma egységekben van megadva. Minden utasítás adott számú egységet ér, és ezek összege nem lehet nagyobb, mint az adott árnyaló verziója által kikötött maximum. Annak ellenére, hogy a DirectX azért készült, hogy a különböző gyártók hardverei egységesen legyenek kezelhetők, gyakorlati tapasztalataim szerint az árnyalók működőképességét mindig ellenőrizni kell a különböző hardvereken (az általam használt nVidia és az AMD kártyák egyes esetekben igen nagy eltéréseket mutattak egyes árnyalók értelmezésében, de általában található olyan megvalósítás, ami mindkét cég kártyáin jól működik). Továbbá mivel a hardverek a DirectX eszközkészletének csak egy részhalmazát implementálják, a többi funkció egy része CPU emulálásra kerül, ami nagy lassulást okozhat. A programban használt lehetőségek egy részénél a szoftveres emuláció nem áll rendelkezésre, ezért az egyes régebbi számítógépeken nem futtatható.

4.3 A képfeldolgozás menete

A következőkben bemutatom, hogyan állítható elő textúrázott, háromdimenziós modell a mélységi és színes kamerák képének felhasználásával. Fontos megjegyezni, hogy az egész folyamat a grafikus gyorsítón megy végbe így az jelentéktelen CPU használattal rendelkezik.

4.3.1 A mérési eredmények elérése

A Kinectről érkező adatokat aszinkron visszahívásokon adja át a programozási interfész, amelyeken még egy kevés feldolgozásra is sor kerül, hogy az adatok a grafikus kártyával kompatibilis formába kerüljenek. Így például a mélységi adatokat 11-bites pixeleit 16-bitessé kell tenni, illetve a színes képeket Bayer kódolásról egyszerű RGB képpé kell alakítani. Ezen műveleteket már a driver elvégzi, amennyiben a kimeneti formátumot megfelelően kikötjük. A driver által létrehozott visszahívó szálakon történik a beérkezett adatok feltöltése a grafikus kártyára, hogy ezzel a feldolgozás többi részét ne késleltessem. A feltöltésnél kiemelt fontosságú a céltextúra típusa, ugyanis ez jelentősen befolyásolja, hogy milyen formában lesznek elérhetőek az adatok. A textúrák esetén az adatok elérése mindig lebegőpontos formában történik, és a textúra típusától függetlenül az összes mintavételező egy négyelemű vektort ad vissza. Amennyiben a textúra kevesebb színcsatornát tartalmaz, mint négy, úgy a használatlan csatornák értéke nulla lesz. Amennyiben a textúra maga is lebegőpontos leírással rendelkezett, úgy átalakításra nincs szükség. Ha a tárolás

egész típusú, akkor viszont az adatok a 0..1 tartományba kerülnek átírásra. A színes kép feltöltéséhez azt 32 bites színmélységűre kell bővíteni, ugyanis a hardver csak az 1, 2 és 4 csatornával rendelkező képeket támogatja, a forrásként rendelkezésre álló 24 bites 3 csatornás felírást viszont nem. Az átalakítás során a negyedik, átlátszóság értéként értelmezett csatornába nulla érték kerül, így a kép átlátszatlan lesz. A mélységpuffer esetében 16 bites színmélységre van szükség, amely a választható formátumok közül a BGRA4444 formátummal volt legjobban leírható, amely 4 db. 4 bites színcsatornával rendelkezik. Mivel a tárolt 16 bites érték 4 db. 4 bitesre tagolódik fel, így az eredeti érték visszaállítását a használat előtt el kell végezni.

Mivel a színes és mélységi képkockák aszinkron érkeznek, ezért úgy döntöttem, hogy a feldolgozás és a beérkező képkockák kezelése aszinkron módon történjen. (Természetesen, ha adott esetben a feldolgozást a mélység vagy színes képkockák beérkezéséhez szeretnénk kötni, az is egyszerűen megoldható, ilyenkor elég a feldolgozó függvényt a visszahívás végén meghívni.) A feladat elvégzésére saját, kettős pufferelési rendszert hoztam létre, hogy az adatok párhuzamos kiolvasása és beírása megoldható legyen. Amennyiben bekapcsoljuk a pufferelést, akkor átlagolás is történik a későbbi feldolgozás során. Ilyenkor a rendszer az átlagolandó képkockaszámnak megfelelő számú rekesszel rendelkező puffert használ, ahol a beírás körbeforgó módon történik. A visszahívások ezen kívül más feladatot alapesetben nem végeznek. A program lehetőséget ad a Kinectről érkező nyers adatok elmentésére, ami ugyanitt megy végbe. Lehetőség van sorozatfelvételek készítésére is, ilyenkor azonban a nyers adatok egy külön pufferbe kerülnek, ahonnan a feldolgozó száakkal párhuzamosan futó IO szálak írják ki őket a lemezre. Mivel a lemezre való írás lassú művelet, ezért a sorozatfelvételek jelentősen megnövelnék a visszahívások szálain a várakozással eltöltött időt, ezért van szükség a párhuzamos lemezre írásra. A mentő szálak egy sorból olvassák ki az elmentendő adatokat, amennyiben a sor kiürül, akkor várakozó állapotba küldöm őket, amelyből a főprogram felébreszti őket, ha új képkockák várnak írásra.

A nyers kimentett adatok később – egy igazi Kinect jelenléte nélkül is - betölthetőek a programba. Ezen funkció csak a fejlesztés későbbi szakaszában került implementálásra, emiatt fontos volt, hogy a program többi részén csak minimális változtatásokra legyen szükség. Ezért létrehoztam egy absztrakt Scanner osztályt, amely tetszőleges mélységszenzor – nem csak Kinect – esetén, mint közös interfész használható. Az eredeti Kinectet leképező osztály pedig ebből lett leszármaztatva, és emellett létrehoztam egy fájl szkennert is, amely nyers mélység és színes adatok segítségével képes a Kinect helyettesítésére. Mivel a program nagy része a közös interfészen keresztül éri el a szenzort, így ezek számára nincs különbség egy igazi és egy virtuális eszköz között, amit a leszármaztatott osztályok polimorfizmusa biztosít. Természetesen néhány programrész továbbra is a leszármaztatott osztályokat használja a specifikus funkciókhoz (fájlbetöltés, fej billentés stb.). Ezen architektúra további előnye, hogy később a program jelentősebb változtatása nélkül lehetőség van más elven működő szkennert alkalmazására, amennyiben a színes és mélységképek előállíthatóak. A fájl szkennert használatkor lehetőség van az adott könyvtár elemei közötti léptetésre, ami hatékonyabbá teszi annak használatát.

4.3.2 Átlagolás

A feldolgozás ütemezése egy időzítő segítségével történik. Az időzítő minden egyes ütésére egyszer lefut a feldolgozási folyamat. Mivel a feldolgozás a grafikus kártyán történik, és így igen gyorsan végbe megy, nincs szükség többszálú feldolgozásra (a grafikus kártya eleve jelentős adat párhuzamosítást végez). Az első lépés a mélységi képkockákat tároló pufferből a következő képkocka

kiolvasása. Ez kétféleképpen is történhet, attól függően, hogy a felhasználó bekapcsolta-e a mélységképkockák átlagolását. Amennyiben nincs átlagolás, úgy egyszerűen a kettős puffer adott pillanatban előlő képkockája kerül felhasználásra. Ellenkező esetben az összes képkocka kiolvasásra kerül a képkocka pufferből, kivéve persze az éppen írás alatt álló elemet. Az átlagoláshoz először össze kell adni a képkockákat. A képkockák összeadására a legkézenfekvőbb módszer az lenne, ha a színkeverőt úgy állítanánk be, hogy az összeadja a forrás és a célszint. Ugyan a színkeverő függvénynek tényleg kiválaszthatunk összeadást, azonban ezt nem lehet lebegőpontos kimenet esetén használni. A lebegőpontos renderelési célra azért van szükség, mert a többi formátumban nem lehet megoldani tetszőleges számú kép összeadását anélkül, hogy jelentősebb adatvesztés történne a korlátozott értékkészlet miatt. Mivel nem csak az összegeket, hanem az összeadott képkockák számát is nyilván kell tartani (pixelenként az érzékelési lyukak miatt), így ehhez kétdimenziós vektorok szükségesek. Az összegzéshez két renderelési cél kell, jelölje ezeket A és B , továbbá szükség van a hozzáadandó képre, jelölje ezt $S_0, S_1..S_N$. Az összeadási ciklus a következőképpen megy végbe: $B = S_0, A = S_1 + B, B = S_2 + A, A = S_3 + B$ és így tovább.

Mivel a Kinect erősen nem-lineáris karakterisztikával rendelkezik, így az átlagolás előtt az általa adott értékeket valós mélységértékekké kell alakítani, amely az alábbi képlet alapján történik:

$$d(x) = a - \frac{b}{x - c} \quad (9)$$

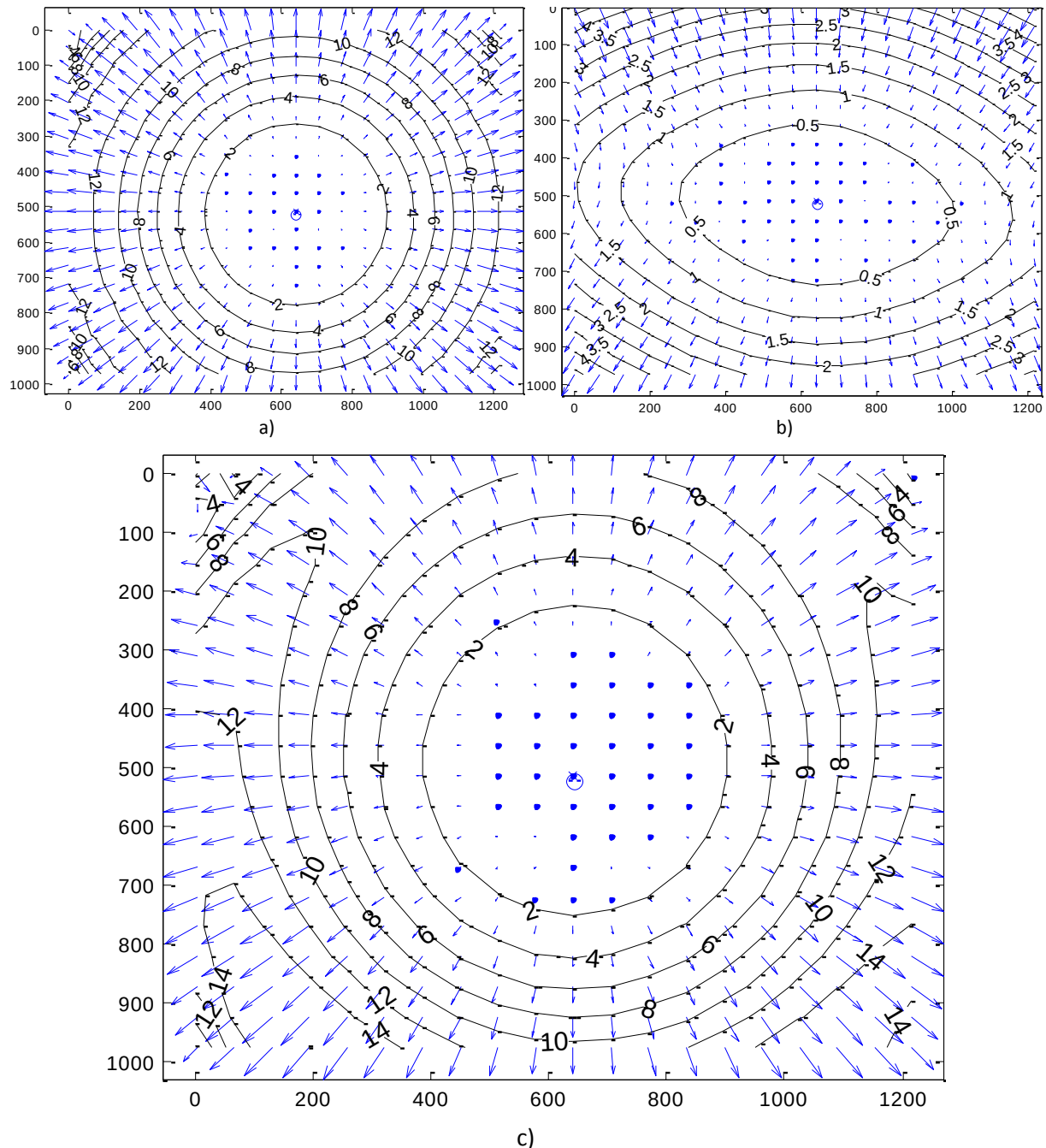
Ahol:

- a, b és c az adott hardverre jellemző, kalibráció során meghatározott paraméterek
- d a mélység valódi egységben (m)
- x a mélység a Kinect egységében

Az összeadás során a rendszer eldobja a 2047 és 0 értékekhez tartozó nem látható területeket, illetve a felhasználó is kiválaszthat egy maximális látótávolságot.

A képkockákat összegző árnyalók lefutása után kerül sor az eredmény átlagolására. Maga az eredmény az két renderelési célban is elhelyezkedhet az összeadások számától függően. Az eredményt átlagoló árnyaló esetén lehetőség van az átlagolásnál figyelembe veendő képkockák minimális számának kikötésére. Ez akkor célszerű, ha garantálni szeretnénk, hogy minden pixel legalább n db. érték átlagolásából származó mélységgel rendelkezzen. Az átlagolást elvégző árnyaló a teljesítmény növelése érdekében egybe van forrasztva a torzítás korrekcióval. A torzítások korrekciója során a rendszer felhasználja a már előre meghatározott torzítási térképet. A torzítási térkép megadja, hogy egy adott a torzítatlan képen elhelyezkedő ponthoz melyik pont kapcsolódik a torzított képről. A torzítási térképek kiszámítására a program első indításakor, illetve a hardver megfelelő kalibrációs paramétereinek megváltozása esetén kerül sor. A térkép generálása után azokat a program könyvtárában elmentem, így nincs szükség azok újbóli kiszámítására a program következő indításakor, amely jelentősen gyorsítja a program elindulását is. A torzítások javítása úgy történik, hogy az adott pixel kirajzolása során a térképből kiolvasom, melyik pixelnek kellene oda kerülnie, majd ellenőrzöm, hogy az még a képen van-e vagy esetleg lelóg arról. Amennyiben a képen van, akkor beírom azt, különben 0 értéket írok be. Ezen műveletnél kihasználom a kártya mintavételező egységei által nyújtott interpolációs képességet, így szép, sima kimenet érhető el.

4.3.3 Torzítás korrekció

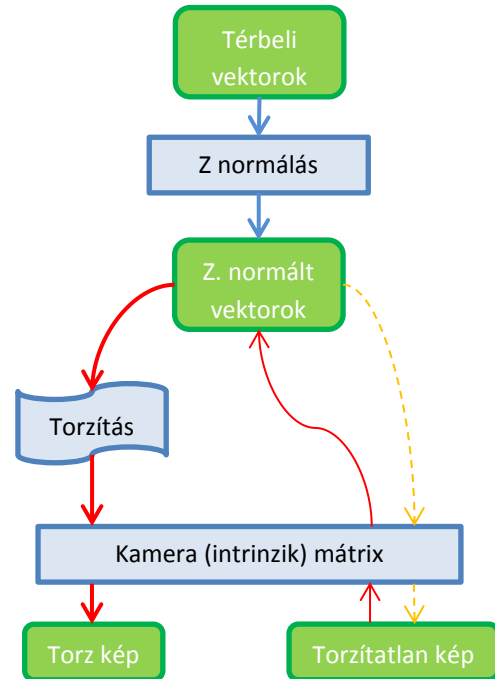


21. ábra A Kinect színes kamerájának torzítása: a) a torzítás radiális komponense, b) a torzítás tangenciális komponense, c) a teljes torzítási modell

A torzítások korrekciója során az a cél, hogy a korrekció után úgy lehessen a kameramoddellel dolgozni, mintha annak optikai leképezése tökéletesen torzításmentes lenne, anélkül, hogy a többi paraméter is megváltozna. Legyen adott egy objektum a valós térben, ez a képre három lépésben képezhető le: a) z-normálás, b) torzítás, c) szorzás az intrinzik kamera mátrixszal. A cél az, hogy a képet úgy változtassuk meg, hogy a torzítási lépés elhagyhatóvá váljon. Ilyenkor mivel a megfigyelt valóság és a többi transzformáció ugyanaz marad, így a képen kell változtatni. Bár a leképezés modellje a valós térből a kép felé haladva kerül felírásra, és csak ebben az irányban egyértelmű a modell, a gyakorlatban a visszafelé irány a fontos, hiszen a képek alapján kell a háromdimenziós modellt előállítani.

Ha van egy pontunk a képen, akkor visszafelé a következő lépéseket kell elvégezni, ha annak térbeli helyzete kellene: a) szorzás az intrinzik mátrix inverzével, b) vissza-torzítás (nem-lineáris egyenletrendszer megoldás, több eredmény, választás), c) szorzás a távolsággal. A torzítás korrekció során a képet úgy kell megváltoztatni, hogy a valós térből a Z normálás és az intrinzik mátrixszal vett szorzás közvetlenül a képhez jutasson. Ehhez a kezdeti elképzeléssel ellentétben nem kell lineáris egyenletrendszer megoldást végezni, ugyanis a GPU képkalkotása logikailag visszafelé működik: azt tudjuk, hogy hova akarunk rajzolni, és meg kell határozni, hogy mi kerül oda (szemben a modellünkkel ahol adott, hogy mit szeretnénk rajzolni, és el kell dönteni, hogy hova). A Z normált térből indulva a torzításon keresztül haladva kapjuk a Kinect által szolgáltatott képet, míg anélkül a torzításkorrigált képet kapjuk. Ezért ha ki akarjuk rajzolni a torzítatlan kép egy pontját, akkor a következőt tehetjük: a) szorzunk az intrinzik mátrix inverzével (megkapjuk, hogy a Z normált tér mely pontja fog ide leképeződni), b) alkalmazzuk a torzítási képleteket az egyértelmű irányban, c) szorzunk az intrinzik mátrixszal.

Ezzel a műveletsorral megkapjuk, hogy a torzítatlan képpontnak melyik pont felel meg a torzított képen, ami lehetőségét ad a torzítás kiküszöbölésére. Minden torzítatlan pontba beírjuk a neki megfelelő torzított pontban kapott mélységértéket, így a kép torzítását kiküszöböltük:



22. ábra A torzítás eltávolítása: a vastag nyílak az eredeti leképezést jelölik, a piros szín a torzításkorrekció során megvalósított transzformációt jelöli, míg a sárga szaggatott vonalak az így kapott új rendszer modelljét mutatják be.



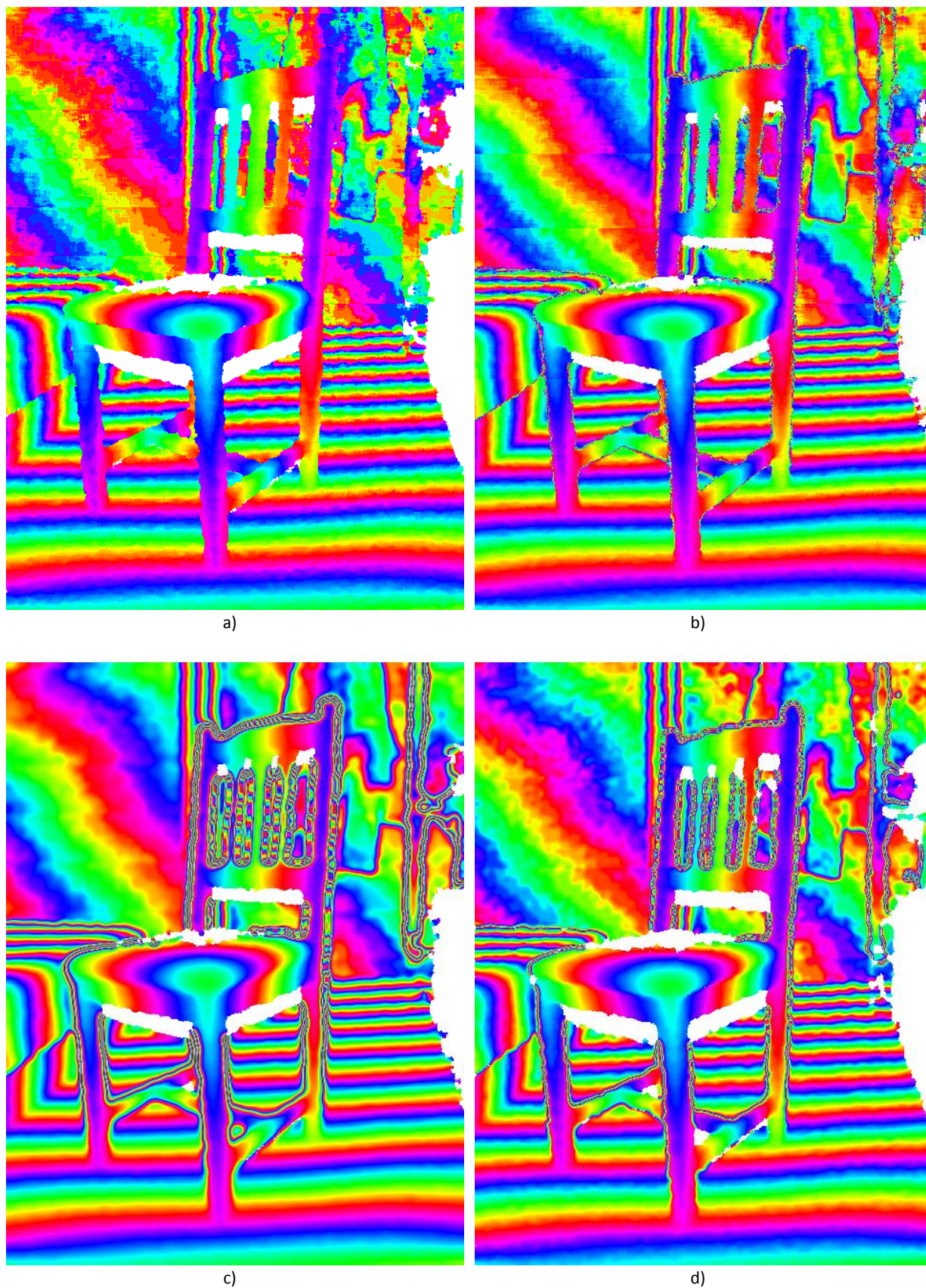
23. ábra A torzítás korrigálása: a) eredeti (torz) kép, b) torzításkorrigált kép - a kép felső része a programom által végzett korrekciót, még az alsó része a MATLAB CCT eredményeit mutatja, amelyek a várakozásnak megfelelően fedésben vannak

E művelet során nincs szükség mélységi információra, tehát a művelet független a látottaktól. További előny a GPU megfelelő beállítások mellett – a mintavételezési frekvencia növelésével (multisampling) – képes „nagy simaságú” képek előállítására. Ez a korrekció mind az infravörös, mind a mélységképen elvégezhető a kalibráció során meghatározott paraméterekkel. Mint már említésre

került a továbbiakban a transzformációk során elhagyható a torzítás kezelése, így a továbbiakban a kamerát úgy kezeljük, mintha egyből a torzításkorrigált eredményt adná, és így ideális lenne.

4.3.4 Gauss szűrés

A feldolgozás következő lépése a Gauss szűrő alkalmazása, amely opcionális. A szűrésnek elsősorban akkor van haszna, ha az átlagolás funkció nem alkalmazható. A Gauss szűrő kétdimenziós Gauss eloszlásnak megfelelő súlyozással átlagolja a szomszédos pixeleket, ezzel jelentősen megnöveli a kimenet simaságát. A Gauss szűrésre azért van szükség, mert a Kinect átlagolás nélkül nagyon zajos kimenettel rendelkezik, és működési módja miatt nem csak véletlen, hanem a bemenettől függő hibával is rendelkezik. Ez utóbbi a gép belső processzora által megvalósított mélységszámítási algoritmusnak köszönhető. A Kinect által érzékelt felületek egyenlőtlenségei jól kisimíthatóak Gauss szűrés segítségével. A szűrő implementálásának kulcskérdése a textúra olvasási és számítási műveletek számának alacsony értéken tartása. Ezért ahelyett, hogy egy kétdimenziós szűrést végeznék az ehhez szükséges mátrix konvolúcióval, inkább két egydimenziós lépésben végzem el a műveletet (egyszer vízszintes, egyszer függőleges irányban). Matematikailag levezethető, hogy például egy 9x9-es konvolúciós mátrixszal elvégzendő konvolúció helyett, két 9 elemű vektorral elvégzendő konvolúciót kapunk. Pixelenként így csak 18 textúra lekérdezést kell végezni az eredeti 81 helyett, ami jelentősen hatékonyabb működést jelent. További teljesítménynövekedés érhető el úgy, ha nem egy nagyobb elemszámú vektorral végzünk konvolúciót, hanem több kisebbel egymás után. Ebben az esetben az egyes szűrések hatása összeszorozódik, így az elért kernelméret drasztikusan megnövekszik. A szűrés végrehajtása az átlagoláshoz hasonló módon két renderelési cél segítségével került megvalósításra (A , B), ezek közül egyik mindig az átlagolás eredményét tartalmazza. A szűrés során A simításának eredménye B -re, majd B -é A -ba kerül. Minden iteráció során egy vízszintes és egy függőleges simítást végzek. A rendszerben megadható a Gauss görbe szélessége, illetve megadható a mintavételezési pontok egymástól vett távolsága. Ez utóbbit csak a kisebb teljesítményű számítógépek esetén érdemes magasabbra állítani az alapértelmezett 1 értékről, mert ugyan a simítás mértéke növekedni fog, egynél nagyobb érték esetén különféle képhibák jelentkeznek. A szűrés a súlyozott átlag számítása során figyelembe veszi a hiányzó pixeleket, így a súlyok összegével való leosztáskor csak azokat a súlyokat veszi figyelembe, amelyekhez tartozott mélységinformáció is. A szűrés hátránya, hogy elmosza az éleket is. Ez a jelenlegi felhasználási területeken nem jelent problémát, de egyes esetekben gondot okozhat. Ilyenkor érdemes lehet kiszámítani a pixelekhez tartozó élintenzitás értékeket (pl. Canny féle éldetektálással [10]), és ezt felhasználni arra, hogy csak azokat a részeket simítsuk ki, ahol nincsenek erős élek. Ezt a módszert az tény alapozza meg, hogy a zaj amplitúdója jelentősen kisebb a lényeges éleknél. A beállításoknak megfelelő konvolúciós vektorok kiszámítására a CPU-n kerül sor, majd innen kerülnek be a GPU memóriájába.

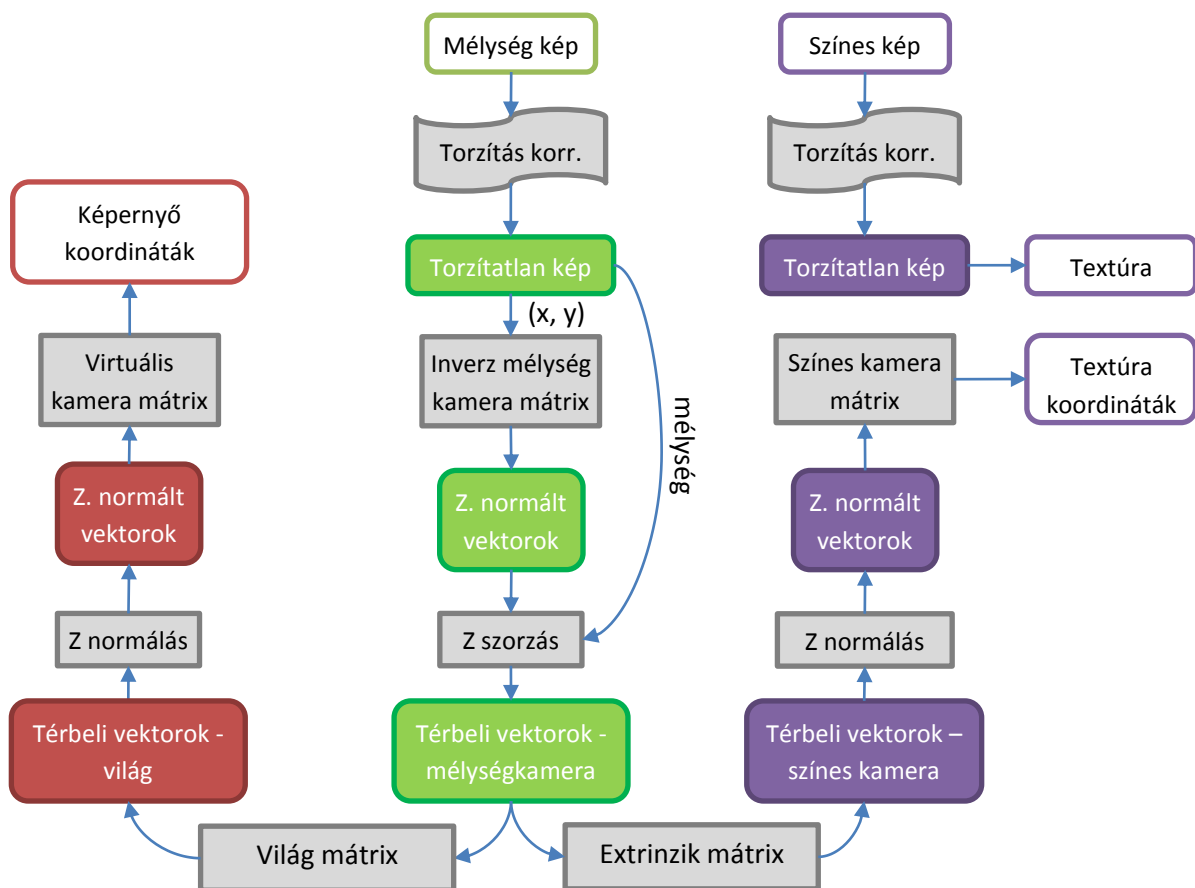


24. ábra Az temporális átlagolás és a spátialis Gauss szűrés hatása: a) eredeti mélységkép, b) kimenet átlagolással, c) kimenet átlagolással és Gauss-szűréssel, d) kimenet Gauss-szűréssel (a színezés periódusa 20 cm)

4.3.5 Háromdimenziós modell létrehozása

A feldolgozás legfontosabb része a háromdimenziós modell létrehozása. A feladat megoldásához szükséges, hogy egy textúrából – adott szabályok szerint – egy háromdimenziós modellt állítsunk elő. Ha az lenne a feladat, hogy a mélységképet egy háromdimenziós grafikonon, mint egy felület jelenítsük meg, az egyszerűen megoldható lenne úgy, hogy először megadnánk a grafikus kártya vertex és index puffereibe egy téglalapot, amely a csúcsok szabályos rácsára feszített háromszögpárokból épül fel, majd ezen csúcsok pozícióit a rajzolás vertex árnyalási szakaszában odébb tennénk a mélység képből nyert magasság információknak megfelelően. Ilyenkor tulajdonképpen úgy kezeljük a mélységképet, mintha az egy ortogonális vetítés eredménye lenne, ami természetesen nem felel meg a valóságnak. Ha a vertexek pozícióit a kamera leképezésének megfelelően állítjuk elő, akkor valódi, arányos háromdimenziós modellt kapunk. A módszer hátránya, hogy számos régebbi grafikus kártya, különösen az integrált grafikus megoldások nem támogatják a textúrák elérését a vertex árnyalókból, amely nélkül ez a módszer nem valósítható meg.

A modell létrehozásához az alábbi műveletek elvégzése szükséges:



25. ábra A modell létrehozásához szükséges műveletek elvi vázlata

A bemeneti kép létrehozásakor a kamera koordináta-rendszerében felírt pontokat Z-normálni, torzítani kell, és végül beszorozni a kameramátrixszal. Most az ellenkező irányban kell a műveleteket elvégezni. Mivel képtorzítások már nincsenek, így azokat már nem kell figyelembe venni. Ez azt jelenti, hogy a Z normálás áthelyezhető a kameramátrixszal való szorzás utánra is, mivel ez nem változtatja meg az eredményt. Ez azért hasznos, mert így visszafelé végezve a műveletet, csak egy vektort kell felírni, majd azt a megfelelő mátrixszal megszorozni, és előállnak a térbeli koordináták.

Azt, hogy pontosan hol vagyunk, azt az aktuális feldolgozandó vertex textúra pozíciójából tudjuk meg, ami nálam mindkét irányban a 0..1 tartományban változik a rács tengelyei mentén. Ezért a következő vektort kell felírunk:

$$Q_D = \begin{bmatrix} w \cdot u \cdot d \\ h \cdot v \cdot d \\ d \\ 1 \end{bmatrix} \quad (10)$$

Ahol a mélységet a mélységkép adja meg (d) méterben, míg az u, v értékek a képen elfoglalt helyet írják le (0..1), a w, h pedig a kép szélessége illetve magassága pixelben. Ezután kell elvégezni a kameramátrixsz inverzével való szorzást. Mivel a kalibrálás során használt nagyfelbontású infravörös képek pixelei nem feleltethetők meg közvetlenül a mélységképek pixeleinek (más a felbontás és a képarány is, és eltolás is van a középpontok között), így a kalibrálásnál kapott mátrixot a következőképpen módosítani kell:

$$A_d = \begin{bmatrix} s_{x_{IR,d}} & 0 & 0 & t_{x_{IR,d}} \\ 0 & s_{y_{IR,d}} & 0 & t_{y_{IR,d}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} A_{IR} \quad (11)$$

Ahol:

- $s_{x_{IR,d}}, s_{y_{IR,d}}$: a mélység és infravörös kép közötti nagyítást írja le
- $t_{x_{IR,d}}, t_{y_{IR,d}}$: a mélység és infravörös kép közötti eltolást írja le
- A_{IR} : az infravörös kamera intrinzik mátrixa
- A_d : a mélységi „kamera” intrinzik mátrixa

Az inverz szorzás után a háromdimenziós térben elfoglalt pozíciók valós egységekben állnak rendelkezésre:

$$P_D = A_d^{-1} \cdot Q_D \quad (12)$$

Ahol P_D a mélységkamera koordinátarendszerében értelmezett vektor. Ahhoz, hogy ebből a képernyőn megjeleníthető háromdimenziós modellt kapjunk, a vektorokat a képsíkba kell vetíteni. A visszavetítés előtt lehetőség van a pontok transzformálására, például adott pont körüli elforgatására, nagyítására, eltolására. Miután elvégeztük ezeket a transzformációkat ki kell számolni a pontok helyét a képernyő-koordinátarendszerben. Ehhez tetszőleges vetítési paraméterek felhasználhatóak, én az egyszerűség kedvéért a Kinect mélység kamerájának kamera mátrixát használtam. Ezzel gyakorlatilag azt érem el, hogy az objektum forgatásával olyan képet kapunk, mintha a Kinect azt egy másik irányból vette volna fel. Erre tekintettel ezt a műveletet újravetítésnek neveztem el. Mivel csupa mátrixszorzással megvalósított műveletet kell elvégezni, így az újravetítés egyetlen mátrixszorzással megoldható. Az így kapott eredményt még át kell írni a DirectX képernyő-koordinátarendszerébe, amely a képernyő bal-felső sarkát a (-1,-1), jobb-alsót pedig a (1; 1) koordinátákkal jelöli. Ezen kívül még le kell tárolni az új mélységértéket a Z osztás előtt, mert erre szükség van a Z puffer használatához. Mivel a képernyőpuffer képaránya az ablak átméretezésével

változik, ezért a kapott pontok vektorait x és y irányban nagyítani, majd tolni kell. Az itt használt tényező meghatározására a képernyőpuffer méretének meghatározásakor kerül sor.

Amennyiben textúrázni is kellene a modellt, úgy ki kell számítani, hogy a mélységkamera által látott pontok hova esnek a színes kamera képén. Ehhez a kalibráció sztereo részében nyert transzformációs mátrixot, illetve a színes kamera intrinzik mátrixát használok fel, így a szükséges transzformáció a következőképpen adódik:

$$Q_C = A_C' \cdot T_{D,C} \cdot A_D^{-1} \cdot P_D \quad (13)$$

Ahol:

- Q_C : a színes kamera vetítési terében értelmezett koordináta (ezt Z normálva adódik a képpont koordináta)
- A_C' : a színes kamera korrigált intrinzik mátrixa
- $T_{D,C}$: a színes és mélység kamera koordinátarendszere közötti transzformáció
- A_D : a mélység kamera intrinzik mátrixa

Ez a transzformáció Z normált vektorok között értelmezett. Mivel a kalibrálásnál alkalmazott nagy és az üzem közben használt kisebb felbontású kép képaránya eltér, így a színes kamera intrinzik mátrixát is módosítani kellett a megfelelő eredményhez. Az újraprojektáló vertex-árnyaló bemente a vetex pozícióját (3D) és a textúra koordinátáit (2D) tartalmazta, míg a kimenete a képernyőkoordinátarendszerben értelmezett pozíciót (4D¹), a mélységet (1D) és a textúra-koordinátákat (2D) tartalmazza.

Problémát jelent, hogy mivel a rendszer csak „2,5” dimenziós, így a kapott felület folytonos lesz, nem lesznek benne lyukak. Ami azt jelenti, hogy az optikai tengelyre közel merőleges irányban hamis felületek jelennek meg, amik valójában nincsenek jelen. Ezek eltávolításához meg kell becsülni a felület normálvektorának irányát. Probléma azonban, hogy a beállításoktól függően a bemenet zajos lehet, illetve az, hogy a jó minőségű deriválást végző metódusok számításigénye jelentős. Ennek kiküszöbölésére egy egyszerűbb módszert választottam, amely egyszerűen kiszámítja a vizsgált, mélységképen található képpont adott távolságra lévő felső, alsó, illetve jobb és bal oldali szomszédjainak háromdimenziós pozícióját, majd ebből a következő összeget képezi:

$$\left(\frac{p_u - p_d}{d}\right)^2 + \left(\frac{p_r - p_l}{d}\right)^2 \quad (14)$$

Ahol:

- p_u, p_d, p_r, p_l : felső, alsó, illetve jobb és bal oldali szomszéd pozíciója
- d : a vizsgált pont távolsága a kamerától

Ha ez az összeg nagyobb, mint egy előre megadott érték, akkor az árnyaló a vertex pozíciókoordinátáit nem szám (NaN) értékűre állítja. Mivel az árnyalóhoz használt HLSL nyelv nem teszi lehetővé a nem szám konstans használatát, ezért azt a külső, a CPU-n futó programkódból futás

¹Ahol az első két koordináta a képernyőn elfoglalt helyet írja le, a harmadik komponens a mélységpufferbe írandó távolság, a negyedik tag egy osztó tényező, amivel a kártya használat előtt lenormálja az előző három értéket.

közben írom be egy globális változóba a kártya inicializálása során. A NaN érték hatására a raszterizáló eldobja az érintett háromszögeket, így a hamis felületdarabok eltűnnek.

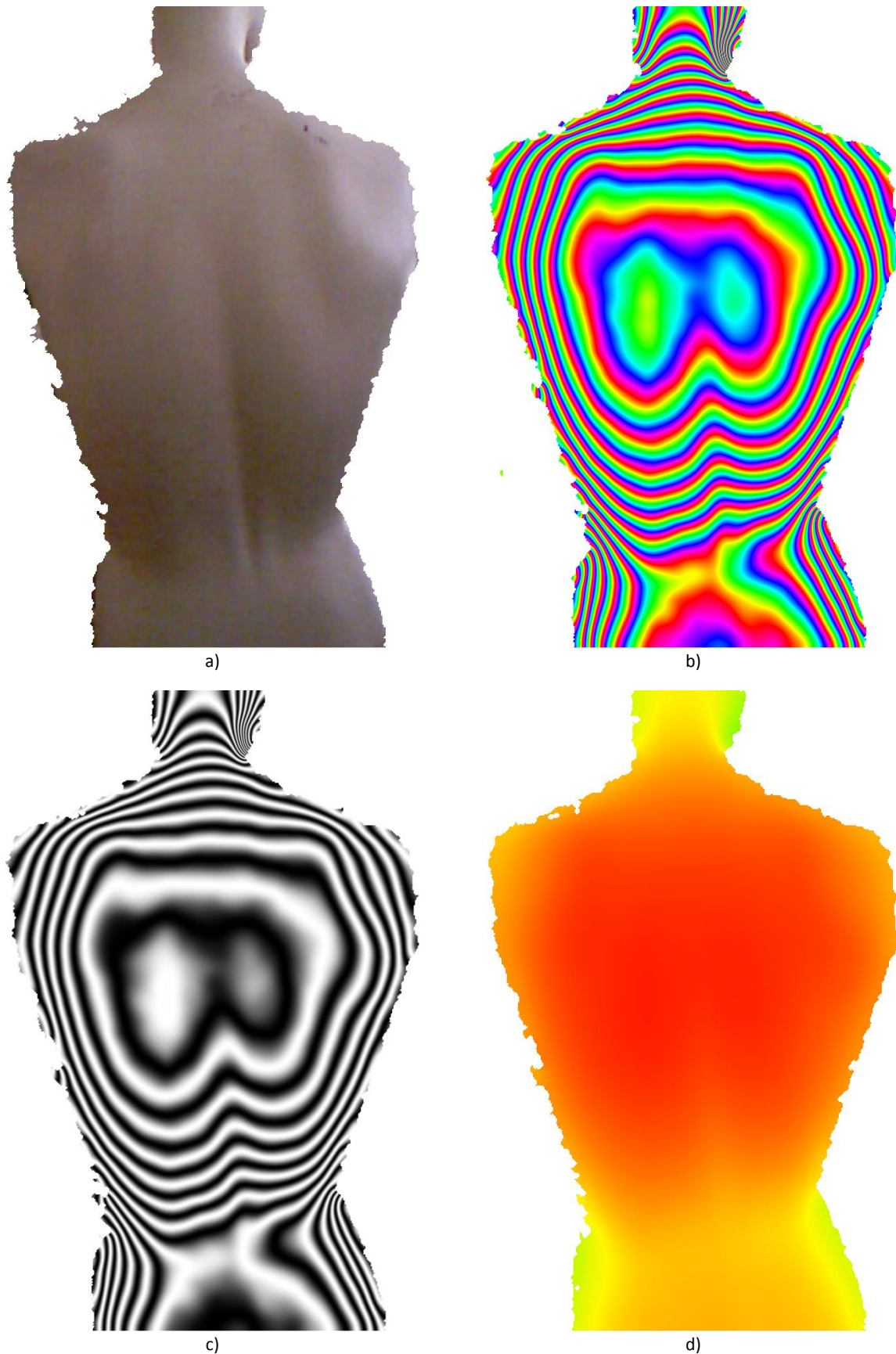
4.3.6 A modell árnyalása

Miután a transzformált vertexek eljutnak a raszterizáló egységbe a pixel árnyalók munkája veszi kezdetét. Többféle pixel árnyalót is írtam, amelyek az alábbi megjelenítési formákat teszik lehetővé:

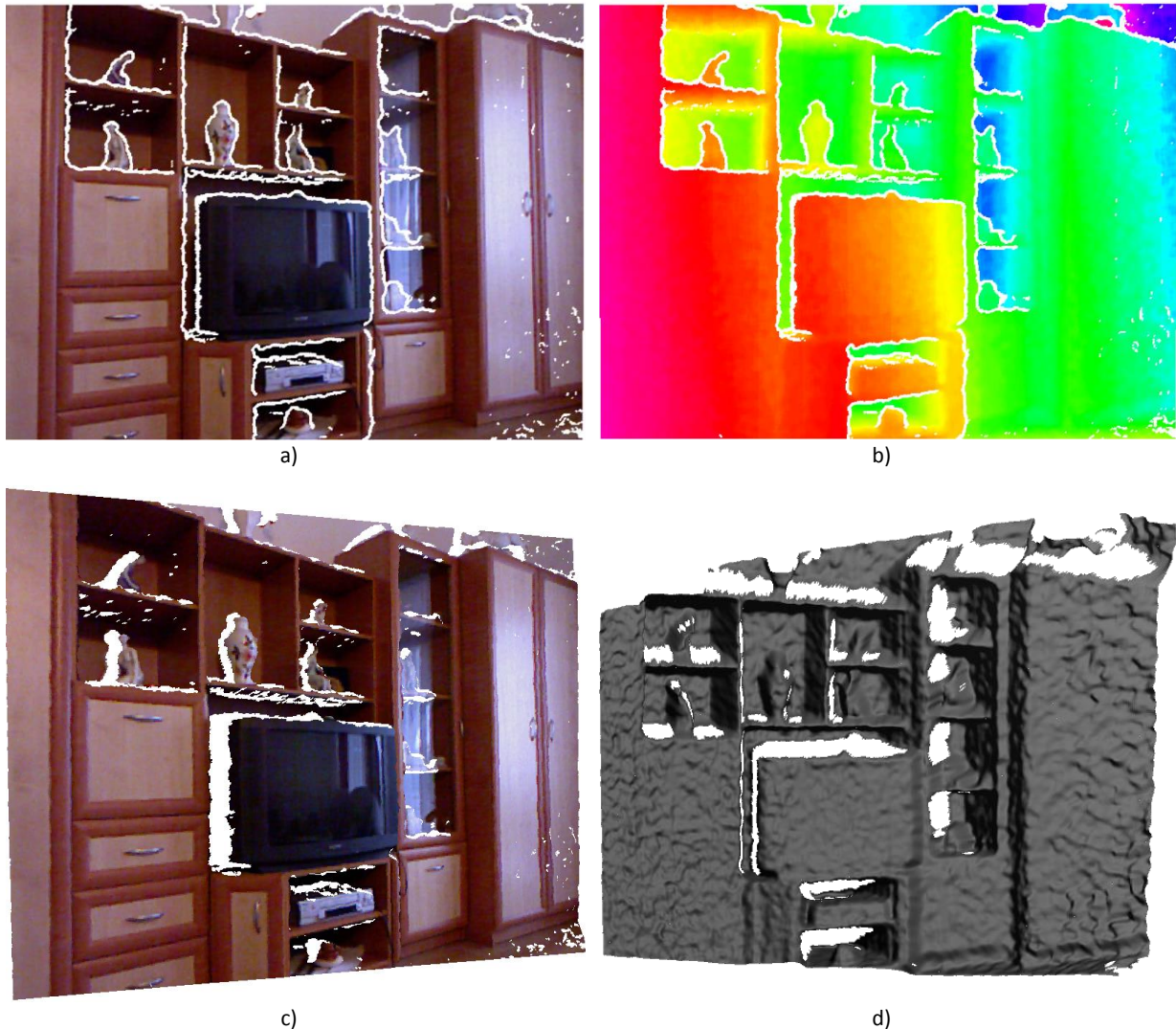
- **Textúrázott:** a modellt a színes kamera által felvett kép alapján textúrázom. A vertex árnyaló által kiszámított - a csúcsokra vonatkozó – textúra-koordináták interpolálva jutnak el a pixel árnyalóhoz (mint az összes lebegőpontosan tárolt érték). Ahhoz, hogy a textúrát megjelenítsem, először ki kell olvasni a kapott torzítatlan textúra-koordinátákhoz tartozó torzított koordinátákat, hogy a torzítást megszüntessem. Az így kapott pozícióban mintavételezem a színes képet.
- **Zebra:** ebben az esetben az árnyalás alapja az újrapetítés során kapott új mélység érték, amelynek adott periódusú szinuszos függvénye a színezés alapja. A végeredmény fekete és fehér sávjai tekinthetők egy hamis Moaré képnek is.
- **Szivárvány:** ebben az esetben kiszámítom a mélység maradékát egy adott, a felhasználó által megadott periódusra, majd el is osztom azzal. Így egy 0 és 1 közötti értéket kapok, ami a kamerától távolodással ciklikusan ismétlődik. Ezután ezt az értéket, mint színárnyalatot tekintve egy HSL-RGB színtér transzformációt végzek, aminek eredménye egy periodikusan ismétlődő szivárványszínű színezés.
- **Árnyalt szivárvány:** ez az eset csak abban különbözik az előzőtől, hogy a HSL-RGB transzformáció során a világosság értéke a felület becsült normálvektorának z komponense lesz. A normálvektor becslése során a nem kívánt háromszögek eltávolítása során alkalmazott megoldáshoz hasonlóan jártam el, de itt a vízszintes és függőleges szomszédok távolságainak keresztszorzatát határoztam meg, majd ezt normalizáltam. Az így kapott vektor z komponensét mint világosságot tekintve a domborzati térképekhez hasonlító árnyalás kapható, amin jól kivehető a felület részletes textúrázata is.
- **Paletta:** ebben az esetben nem HSL-RGB transzformációt végzek, hanem a színárnyalatot leíró értékkel egy 1 sorból álló textúrából választok színt, amit a felhasználó előre megadhat egy bitkép formájában.

Az árnyalás után a képernyőpuffer készen áll a megjelenítésre. A rajzolás nem közvetlenül a képernyőre, hanem a back buffer-be történik, ezért a rajzolás befejeztével egy utasítással azt át kell helyezni a front buffer-be. Ha a grafikus adapter inicializálása során bekapcsoltuk, akkor a képkockák kirajzolása szinkronizálva van a monitor képfrissítési frekvenciájához. Ilyenkor a kép kirajzolását kiváltó parancs megvárja a következő frissítést, addig megakasztja a program futását.

A háromdimenziós modell mellett lehetőség van a színes, infravörös és mélységi üzemmódok nyers kimeneteinek megtekintésére és elmentésére is. Mivel a grafikus kártyán tárolt textúrák tartalma átmásolható a központi memóriába, így lehetőség van az eredmények exportálására, elmentésére. Bár az XNA rendszer maga is biztosít képmentésre szolgáló metódusokat, egyes esetekben ezek nem várt eredményeket produkáltak, így ezeket nem tudtam használni. A .Net keretrendszer a GDI-n keresztül lehetőséget ad különböző formátumú képek írására. Ehhez a képeket pontterek segítségével át kell másolni a GDI kép objektumába, aminél arra kell figyelni, hogy a színcsatornák sorrendje eltér, illetve, hogy minden sor hossza négy többszöröse tároláskor (végükön lehetnek nem használt pixelek).



26. ábra Egy bábu hátáról készült felvételek különböző árnyalásokkal: a) textúrázott, b) szivárvány (színezési periódus 1cm), c) zebra (színezési periódus 1cm), d) skála



27. ábra Felvételek egy szekrényorról: a) textúrázott modell, b) szírvárnyszín kitöltés mélység szerint, c) modell kiforgatva az eredeti nézetből, d) modell renderelve az Autodesk Maya szoftverrel (amint látható nagy távolságok esetén a Kinect pontossága alacsonyabb)

A program nemcsak képek mentésére ad lehetőséget, hanem biztosítja azt is, hogy a felvételeket modellként exportáljuk. A modell exportálása két módon történhet meg:

- háromdimenziós vektorok tömbjeként, ahol minden vektor a neki megfelelő pixel térbeli helyét írja le
- STL formátumban, amelyben háromszögek csúcsainak koordinátái vannak megadva, az egyes háromszögek normálvektorai mellett

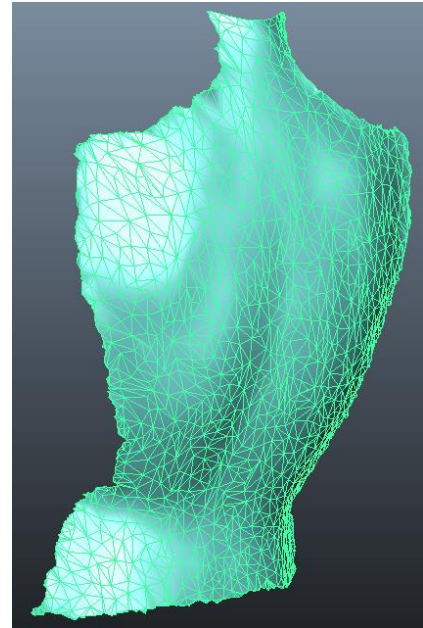
A modellek mentése mellett sor kerül a textúrák mentésére is.

A feldolgozási folyamat megtervezését jelentősen befolyásolta, hogy a grafikus kártya, mint erőforrás, egyes esetekben a feldolgozás során elérhetlenné válik, vagy egyes beállítások módosításához azt újra kell inicializálni. Mivel a grafikus adaptert reprezentáló objektum hozzá van kötve a képernyőpufferhez, illetve az ábrázolást biztosító ablakhoz, ezért annak felbontása nem módosítható menet közben, ehhez új objektumpéldány létrehozása szükséges. A Windows virtualizálja a grafikus gyorsítót, emiatt a több program párhuzamosan használhatja azt. A

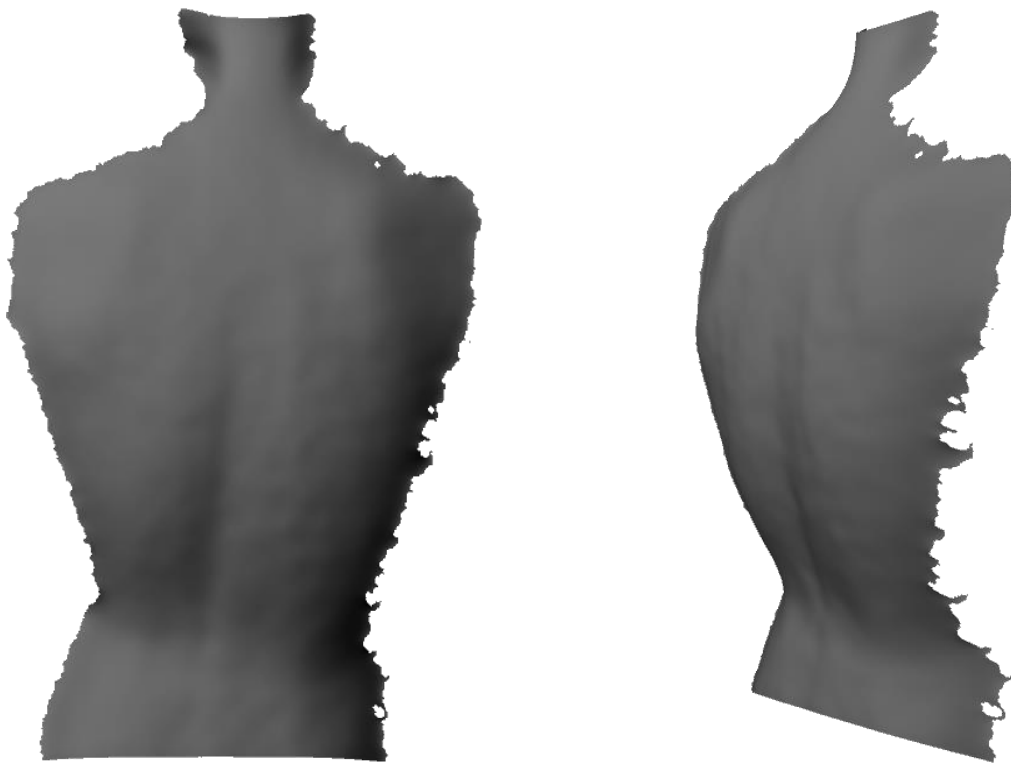
virtualizáció azt is biztosítja, hogy az egyes felhasználók egymás memóriaterületéhez ne férjenek hozzá. Ez akkor is igaz, ha a kártyára egy programon belül többször is kapcsolódunk. Mikor a grafikus kártyáról a neki megfelelő objektum eldobásával lekapcsolódunk, akkor minden használt memóriaterület tartalma elvész. Így ablak átméretezéskor az új objektumpéldány létrehozása során az összes adatot, textúrát stb. újra fel kell tölteni a kártyára. Így ezalatt a feldolgozás szünetel, és csak a feltöltés végén folytatódik. Az adatok újratöltését akkor is el kellett végezni, ha az eszköz „elvész” (DeviceLost kivétel), ezt okozhatja a grafikus kártya driverének hibája vagy például a Windows Security Prompt (képernyő biztonsági elsötétítés) megjelenése.

4.4 Alkalmazás gerincferdülés diagnosztizálásához

A program gerincferdülés diagnosztizációs célra is felhasználható, ehhez az alapot a hátról készült háromdimenziós modellek és ál-moiré képek (26. ábra) jelentik.



28. ábra A hát modellje a háromszögek számának 98%-os csökkentése után



29. ábra A kimentett modell renderelése Autodesk Maya segítségével

Az ál-moiré képekhez már kidolgozott diagnosztizációs módszerek állnak rendelkezésre. Míg a háromdimenziós modellek automatikus diagnosztizációs program készítésére adnak lehetőséget. Ennek megvalósítási lehetőségét mutatja be Ayhan Dániel „*Optikai gerincferdülés-diagnosztizáló szoftver fejlesztése*” című TDK dolgozatában [11].

5 Emberi láb háromdimenziós szkennelése

A dolgozat második felében az eddig bemutatott program olyan kiterjesztését szeretném bemutatni, amely lehetővé teszi háromdimenziós modellek készítését az emberi lábról. Ez a modul egy olyan projekthez készült, melynek célja egy kisméretű, precíz, hordozható és gazdaságosan előállítható lábskenner létrehozása. Az elkészült modellek használhatóak például orvosi célokra vagy egyedi lábra szabott cipők, kaptafák készítéséhez.

A méréshez szükséges hardver és szoftver fejlesztése párhuzamosan történt, én a fejlesztés szoftveres részével foglalkoztam elsősorban. Mivel a program és a hardver fejlesztése távoli helyeken történt, és csak egy prototípus volt, így a program teszteléséhez írtam egy modult, amely előre elkészített felvételsorozatok segítségével szimulálja a program több része felé a szkennert hardveres részét. Ezt majd a későbbiekben könnyen - a program többi részének módosítása nélkül - ki lehet cserélni a tényleges hardvert kezelő változatra. A hardver gyakori változtatása miatt a forgópados szkennert manuális módszerrel (amely becslésekre, illetve a kimenet elemzésére épült) kalibráltam, amely ugyan nem pontos, de a végleges és számomra fizikailag is rendelkezésre álló hardver elkészültéig megfelelőnek bizonyult az algoritmusok tesztelésére. Az automatikus kalibrációs módszert még nem valósítottam meg, mivel ez a hardver fizikai rendelkezésre állását igényli a sok mérés, kísérlet, tesztelés végrehajtásához, amit a kalibráció pontosságának ellenőrzése igényel. A dolgozatban így elsősorban a mérés, modellképzés algoritmusát fogom ismertetni, az eredmények vizsgálatára csak nagyságrendi, alakhűségi szempontból kerül sor.

5.1 Modellezési módszerek



30. ábra Kaptafák a forgópados

Mivel a Kinect nem képes a tárgyak mögé látni, ezért a lábak szkenneléséhez a szkennert mozgatni kell a lábakhoz képest. Ez megoldható a Kinect vagy az ember mozgatásával is. Az elérendő nagy precizitás nem teszi lehetővé a mélylégszenzor manuális mozgatását, mert így az egyes felvételek közti transzformációk csak kevésbé pontosan állapíthatók meg. A motoros mozgatás esetén a helytakarékosabb megoldást az ember mozgatása jelenti, így egy forgópados szkennert kapunk. A modellezendő objektumot egy fix tengely körül forgatjuk, miközben az elfordulás pontos szögével társított mélylégfelvételeket készítünk. A program célja tehát az, hogy ezeket a mélylégfelvételeket megvizsgálva létrehozza a látottak háromdimenziós modelljét. Ehhez három módszert vizsgáltam meg:

- A háromdimenziós testek felületének geometriai leírására több módszer is létezik: ezek közül a legelterjedtebbek a polygon és spline modellek. A lábak formájának leírására mindkét

modell alkalmas, de az előbbi szélesebb szoftvertámogatással bír, és egyszerűbben megvalósítható, hiszen ehhez „csak” háromszögeket kell fektetni az egyes felvételek összegzésével kapott pontfelhőre. A „csak” szó idézőjelbe vételét indokolja, hogy a pontfelhő feldolgozása számos nehézségbe ütközik. Ideális esetben a pontok egy sima felületen helyezkednének el, a gyakorlatban azonban a mérés hibája miatt a pontok vastag rétegeket alkotnak és eloszlásuk sem egyenletes. Amennyiben a szenzor paraméterei nem ismertek elegendő pontossággal, úgy előfordulhat, hogy a modell nem rakható össze, mert az egyes pontfelhők a forgatás miatt különböző irányokban tartalmazzák ugyanazon torzítást, ami ellentmondó felvételeket eredményez. A pontfelhők kezelésére több módszer is létezik. Ezek közül jól ismert a legkisebb távolság alapú növesztés [12] módszere. Ennél arra építünk, hogy kiválasztunk egy pontot a pontfelhőből, majd megkeressük, mely pontok helyezkednek el hozzá legközelebb. A távolság vizsgálatokor felhasználjuk a felületi normálvektorokat is, amelyeket az egyes felvételek létrehozásakor számítunk ki, és a távolságmérést az ezekre merőleges síkban végezzük. Ez az utóbbi módszer lehetővé teszi a zaj miatt sima felületet nem közelítő pontcsoportok kezelését. A módszer lényege, hogy a legközelebbi pontok közé háromszögeket illesztünk, majd ezt ismételve növesztjük a felületet a modell mentén. Ezen módszer előnye, hogy összetett formák is egyszerűen leképezhetők segítségével. Problémák akkor jelentkeznek, amikor a pontok vastag rétegeket alkotnak a mérési hibák miatt, sok esetben a felületi normálvektorok meghatározása is nehéz a felületi zaj jelenléte miatt. További problémát jelent, hogy az ilyen algoritmusok általában nem eredményeznek lyukmentes modelleket és egy élt kettőnél több háromszög is tartalmazhat. Mivel a lábak modelljein később méréseket kell végezni, illetve a lábnak megfelelő kaptafát kifaragó CNC esztergagépet is jó lenne közvetlenül ezzel irányítani, így ezt a módszert elvetettem.

- Az előző módszer fontos jellemzője, hogy az általa létrehozott modell nem strukturált, a csúcspontok szabálytalanul helyezkednek el a térben. Ezzel szemben egy jobban áttekinthető leírási módszert nyerhetünk akkor, ha a lábakat a tengelyükre merőleges keresztmetszetek segítségével írjuk le. Ehhez metszetenként mindössze két zárt töröttvonalra van szükségünk. A lábak alakja jól modellezhető úgy, hogy képezünk egy polygon tengelypalástot, majd ennek csúcspontjait sugárirányban elmozgatjuk. Egy ilyen modell csak egy láb leírására alkalmas, illetve biztosítani kell azt is, hogy legyen a láb tengelyére merőleges síkokon egy-egy olyan pont, amelyről a láb körvonalának egésze belátható. Erre azért van szükség, mert különben alámetszési problémák jelentkeznének. Ilyen alámetszés létrejön például a lábujjknál, de ez nem okoz külön problémát, mert ilyen részletek megfelelő érzékelése a Kinect korlátai miatt amúgy sem lehetséges, és erre az információra nincs is szükség kaptafa vagy cipő gyártás esetén. A többi alámetszés elkerülhető egészséges láb esetén a középpont megfelelő elhelyezésével. Az implementáció relatíve egyszerű, mert csak a pontok henger polárkoordináta-rendszerbe való transzformációját kell elvégezni, és a tengelytől mért távolságukat felírni. A módszer nagy előnye, hogy nagy pontosság érhető el, lyukmentes modell kapható, és könnyen, hatékonyan implementálható, illetve, hogy pontatlan kalibrációs paraméterek esetén is használható és a torzítások mértékét is csökkenti. Fő hátránya az, hogy középpont helytelen megválasztása esetén, illetve bizonyos konkáv alakzatoknál nem tudja helyesen leképezni a geometriát a kialakuló alámetszések miatt.
- A harmadik ötlet abban különbözik a másodiktól, hogy az egyes metszetekben tetszőleges töröttvonalal dolgozik, de emiatt a megoldási módszer gyökeresen különbözni fog az

eredetitől. Ebben az esetben a rendszer a modellt szeletenként hozza létre, és a forgatás során folyamatosan összeveti a már meglévő modellt az adott nézetben látható részmodellel és fokozatosan bővíti azt. A nehézséget itt a már leképezett és új részek megfelelő azonosítása, illetve a kezdetben kedvezőtlen szögből látszó részek cseréje jelenti. Ez utóbbi művelet érzékeny a kalibrációra és pontatlanság esetén durva felületet eredményez. Összetett feladat az egymásmelletti keresztmetszetek közötti háromszögelés megvalósítása is. A módszer előnye, hogy összetettebb formák is leképezhetőek vele, mint a második esetben. Hátrány az összetett implementáció, és a kalibrációs paraméterekre való érzékenység.

A három módszer közül végül a második kettőt valósítottam meg. A dolgozat következő részében bemutatom ezek implementációját és összehasonlítom a velük elért eredményeket is.

5.2 Modellezés polár-koordinátarendszerben

A polár-koordinátarendszer alapú modellezés első lépése hasonlóan a dolgozat első felében bemutatott folyamathoz a képkockák átlagolása. Abban az esetben még statikus környezetet feltételeztünk, ebben az esetben viszont már egy tálcán körbeforgatott objektumról kell felvételeket készíteni. A tálcát léptetőmotor mozgatja, és így az elfordulás szöge ismert. Mivel a rendszer mélységképeket rögzít, amelyekből háromdimenziós modell is készíthető, így lehetőség van arra, hogy az egymás utáni nézeteket egy közös helyzetbe forgassuk. Ehhez először korrigálni kell az optikai torzításokat, hogy azokat a feldolgozás további részében már ne kelljen figyelembe venni. Ezután fel kell építeni a kamera koordinátarendszerből a forgótálcához rögzített koordinátarendszerbe vivő transzformációt leíró mátrixot:

$$T_f = T_z^{-1} R T_z T_{x,y} \quad (15)$$

Ahol:

- $T_{x,y}$: a forgópad középpontját a kamera koordinátarendszerének középpontjába vivő eltoló mátrix x és y irányú eltolást tartalmazó komponense
- T_z : a forgópad középpontját a kamera koordinátarendszerének középpontjába vivő eltoló mátrix z irányú eltolást tartalmazó komponense
- R : ez a mátrix több forgatómátrix szorzataként adódik: először a már középpontba tolt forgópadot vízszintes síkba forgatom, majd a függőleges forgástengely körül a fúzióhoz szükséges helyzetbe hozom

A transzformáció után a forgópad a vízszintes síkba kerül, és középpontjának a kamera koordinátarendszerében értelmezett z irányú helyzete változatlan marad. Ezt a transzformációt ezután az alábbi módon kiegészítem:

$$T_r = A_d T_f A_d^{-1} \quad (16)$$

Ahol A_d a mélységi kamera mátrixa. T_r transzformációja a z-szorozott, egyessel négydimenziósra bővített képkordinátákra vonatkozik, azaz a következő formájú vektorokat transzformálja:

$$P = \begin{bmatrix} x \cdot d \\ y \cdot d \\ d \\ 1 \end{bmatrix} \quad (17)$$

Ahol (x, y) a képsíkban értelmezett koordináták (bal felső sarokból balról jobbra, illetve fentről lefelé mutató koordinátatengelyeknél), d a mélység (a képben „befelé” mutat). A következőkben az összes koordinátarendszerek közötti transzformáció ilyen vektorokra lesz értelmezve.

Így olyan újrapetítési mátrix hozható létre, amellyel az egyes eltérő helyzetben készült felvételek egyesíthetőek. A közös helyzet a pufferben lévő középső képkocka helyzete lesz, hiszen az adott képkockakészletből erről a pozícióról áll rendelkezésre a legtöbb információ:



31. ábra Kaptafák közös helyzetbe transzformálva a fúzióhoz

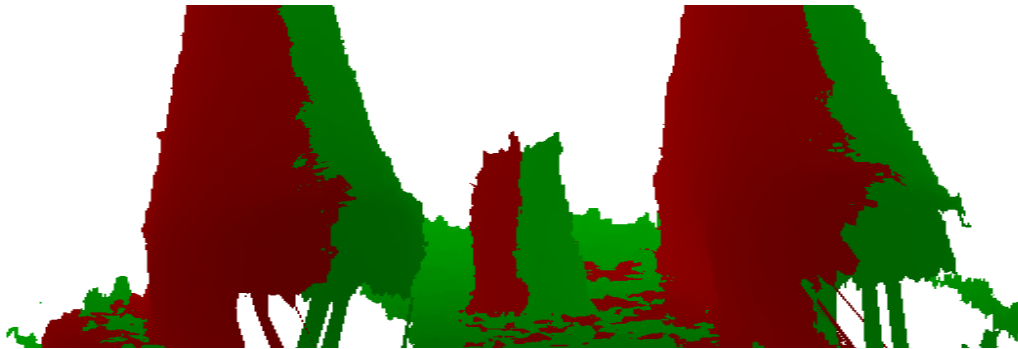
Mivel csak a forgótálcán elhelyezkedő objektumok forognak, ezért az újrapetítés során le kell vágni azokat a háromszögeket, amelyek forgótengelyhez rögzített megválasztható sugarú hengeren kívül vannak. Az újrapetítés után a képkockák tartalmát összeadom, majd elvégzem az átlagolást is. Az összeadás során ügyelni kell arra, hogy ha egy adott nézetben készült 2,5D-s modellt elforgatunk, akkor egyes olyan felületek is láthatóvá válhatnak, amelyek nem látszanak az adott nézetből, mert egy másik – az előbbi nézetből nem látható – felület eltakarja őket. Ezért az összeadás során figyelem, hogy az aktuális képkocka által adott mélységérték hogyan viszonyul az átlaghoz. Ha az átlagtól nem tér el erősen, akkor egyszerű összeadást végzek, azonban ha jelentős az eltérés, akkor az átlag és az aktuális érték közül az kerül a kimenetbe, amelyik közelebb van a kamerához. A művelet után kapott kép és az eredeti közötti minőségjavulás nagyon látványos:



32. ábra Az összeforgatás utáni átlagolással kapott eredmény

Kérdésként merülhet fel, hogy hogyan határozzuk meg a Kinect és a forgópad kölcsönös helyzetét. Amennyiben nincs lehetőség pontos kalibrációra, úgy a paraméterek megadhatóak manuálisan is, ehhez az nyújt segítséget, hogy nem megfelelő beállítás esetén az átlagolás után torzult képet kapunk, amelyen az objektum mérete nagyobb, mint lennie kellene, ugyanis az egyes helyre vetítések nem hoznak létre teljes fedést az átlagolt képkockák között, hanem az objektum megnyúlik, eltorzul. Az egyes paraméterek megfelelő értéke úgy ismerhető fel, hogy abban a helyzetben az átlagolás után kapott objektum képe a lehető legkisebb területű. A másik kalibrációs

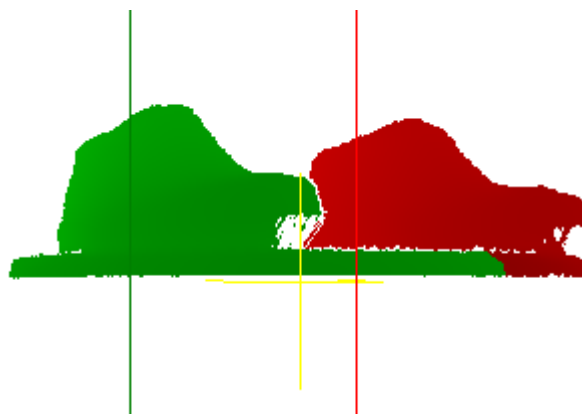
lehetőség esetében a program az egymás utáni képkockákból kapott háromdimenziós felületmodelleket egy közös koordinátarendszerbe vetíti, ahol a felületek a paraméterek állításával fedésbe hozhatóak:



33. ábra Két különböző felvétel kiegészíti egymást (a két kaptafa közé egy harmadik tárgy került elhelyezésre), ez a kalibrálási nézet nyers képkockákon alapszik nem az átlagoltakon (ahhoz ugye már eleve kell a kalibráció), ilyenkor a zaj sincs kiszűrve

A feladat elvégezhető a kalibráló sakktábla segítségével is, úgy, hogy azt a forgótálcára helyezzük, majd megforgatjuk, és a sarokpontok koordinátáit összevetve az egymás utáni képeken kiszámolható az azok közötti forgatás mátrixa és ebből a tálca koordinátarendszerének kamerához képest felvett helyzete is. Amennyiben a forgatás tengelyéhez központosítjuk a sakktáblát, úgy egy lépésben is meghatározható a transzformáció.

Ha az így kapott kép minősége még nem lenne kielégítő, lehetőség van arra is, hogy azon Gauss szűrést is végezzünk. Ezen művelet azonban nem végezhető el úgy, mint az előzőekben, ugyanis számos képkockán a két láb részben takarja egymást. Ez azért okoz problémát, mert ebben az esetben a Gauss szűrő a két láb közti váltást is elmossa, ami a lábak „összenövéséhez”, de legalábbis torzulásához vezethet. Ez azonban elkerülhető, olyan módon, ha a két láb közötti felezősíki helyzetét megadjuk, és csak azokat pontokat vesszük figyelembe a Gauss szűrés súlyozásos átlagszámításánál, amelyek ugyanazon oldalon helyezkednek el, mint az a pont, amelyen ép a szűrést végrehajtjuk. Ily módon mind a két láb felülete kisimítható egyetlen menetben:



34. ábra A két kaptafa a lábszárak helyének kijelölésével, a két lábat elválasztó vágósík a piros és zöld árnyalással került megjelölésre

A következő lépés végrehajtásához ismerni kell a lábszár helyzetét a forgópadon. Ennek beállítása jelenleg úgy történik, hogy a szkenneléskor a páciens mindig a kamerával szemben álló

helyzetből indítjuk, így a két láb közötti felezősíki helyzete már adott, a lábak távolsága illetve a lábszár pozíciója oldalirányból nézve a forgási tengelyhez képest manuálisan állítható be. Illetve az is megoldható, hogy annak helyét a forgólapon előre kijelöljük és a páciens oda állítjuk. A fejlesztés későbbi szakaszában megvalósítható automata tengelyválasztás is, hiszen a Kinecttel a lábokról készített mélységképen egy alkalmas magasságban elhelyezkedő sort végignézve és a kép két oldalán egy-egy távolsági minimumot keresve, a két láb távolsága illetve a mélységirányú helyzetük is könnyen leolvasható, ebből pedig a tengelyek helye is kiadódik. Manuális beállítás esetén a program a tengelyek kijelölését a képre vetített vonalak segítségével könnyíti meg, amelyek a kamera leképezésének megfelelően helyezkednek el.

A modell előállításához az eddigiek során előállított mélységkép pontjait a térbe kell visszavetíteni és meg kell határozni a pozíciójukat a forgótálca koordináta rendszerében. Ez a következőképpen végezhető el:

$$T_{sf} = R_y T_z A_d^{-1} \quad (18)$$

Ahol:

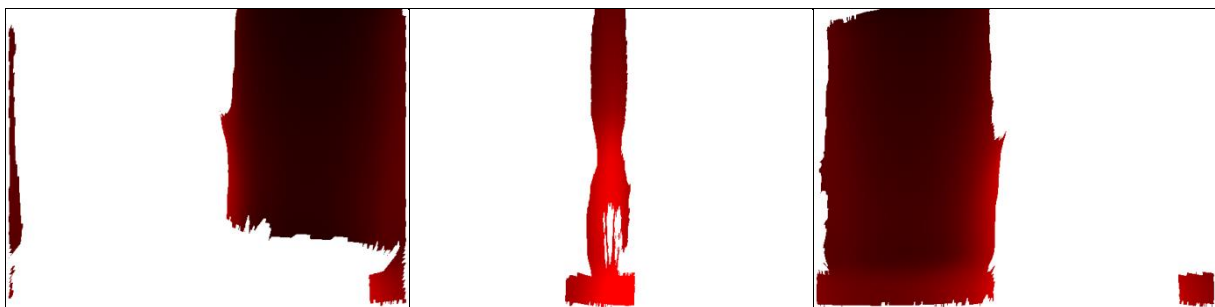
- A_d : a mélységi kamera mátrixa
- T_z : a forgópadot a középpontba toló mátrix (csak z irányú komponense van, hiszen a forgópad a $z = 0$ síkban van)
- R_y : a világ koordinátarendszerbe forgat a forgópad helyzete alapján (a forgópad forgásának tengelye körül)

Ha megkaptuk a pozíciókat a tálcához rögzített derékszögű koordinátarendszerben, akkor azok átírhatóak egy ugyancsak a tálcához rögzített henger-koordinátarendszerbe, amelynek origója a láb már beállított tengelyénél van:

Ennél a műveletnél, a vágósíkkal a teret kettéválasztva, a feldolgozást lábanként végezem el, így a két láb külön modellbe kerül. Az így kapott pozíciókat ezután a képsíkba vetítem a következőképpen:

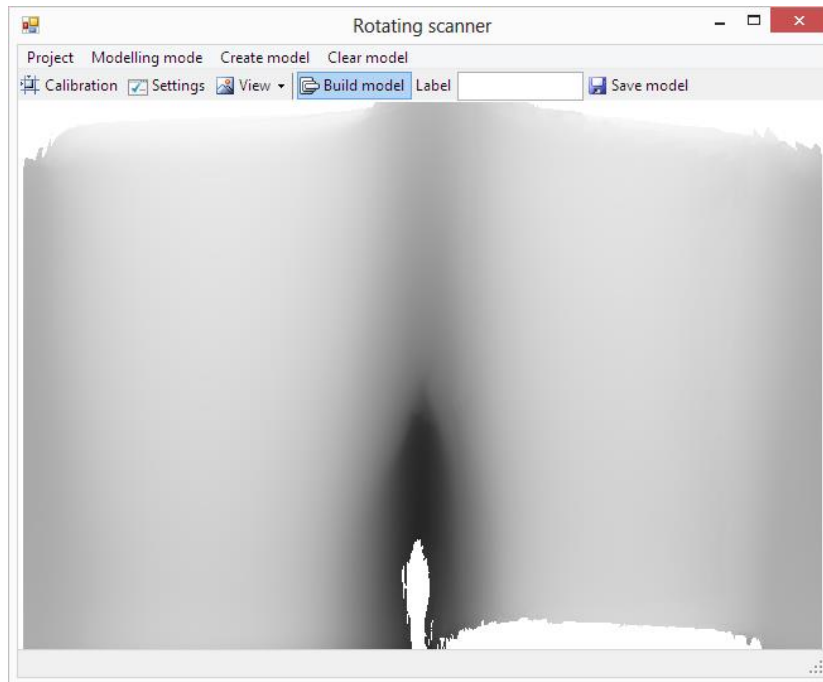
- a kép függőleges tengelye meg fog felelni a henger-koordinátarendszer fő tengelyének
- a kép vízszintes tengelye írja le a henger-koordinátarendszerben az elfordulás szögét
- a tengelytől mért távolság kerül beírásra az egyes pixelekbe

A leképezés során problémát okoz a szögkoordináta átbillenése, mert ilyenkor a háromszögek a kép egyik széléről a másikra nyúlhatnak át, ezért ezt külön le kell kezelni.



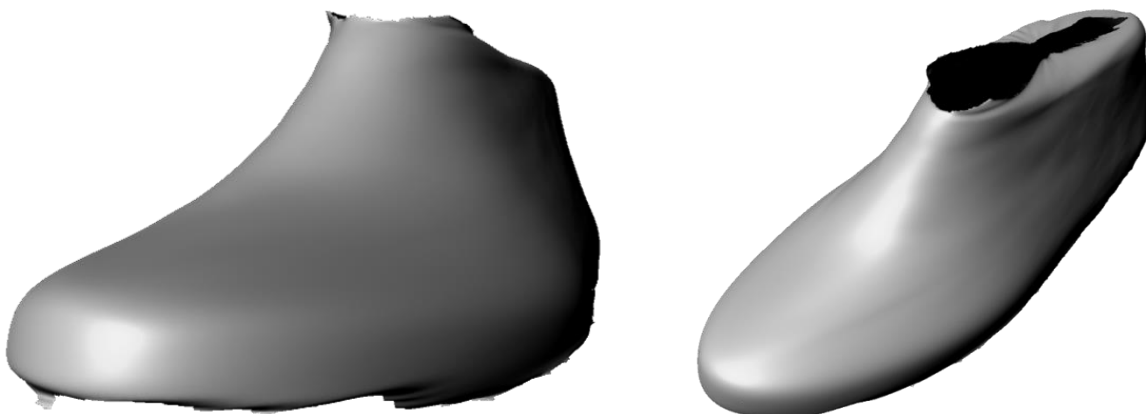
35. ábra A hengerkoordináta-rendszerbe vetített kaptafa különböző irányokból készített felvételekből előállítva

Ezt a műveletet a megfelelő elforgatás figyelembevételével elvégzem az összes képkockán forgatás közben, az eredményeket összeadom, majd átlagolom. Így egy a térképészetben használt henger-koordináta-rendszer alapú vetítésekkel egyező tulajdonságú, nyújtott képet kapok. Ebből a képből úgy állítható elő háromszögekből felépített felület, hogy minden egyes pixel esetén kiszámítom az oda leképezett pont forgópadhoz rögzített derékszögű-koordináta-rendszerbeli pozícióját, majd az így kapott pontokra a következőképpen fektetek háromszögeket:



36. ábra A kaptafa hengerkoordináta-rendszerben kapott képe

A feldolgozás után az előállított modell STL formátumban exportálható. Amennyiben színes képeket így rögzítünk, úgy a modell csúcsaihoz tartozó textúra-koordináták minden helyzetben meghatározhatóak a dolgozat első felében bemutatott módon, ez a funkció azonban még nem került implementálásra. Az adatfeldolgozás után a következő eredményt kaptam:



37. ábra A modellezés során nyert eredmény

Amint, az a képeken látható, a kimenet sima felületű és alakhelyes leképezést mutat. A felületen néhol kis törésvonalak jeletkeznek (például a sarok felőli részen), amelyek arra utalnak, hogy a különböző felvételek közös koordináta-rendszerbe való transzformációja nem teljesen pontos. Ennek

egyik nyilvánvaló oka az, hogy a kalibráció a paraméterek próbálásán keresztül történt, amellyel nehéz tökéletesen megfelelő eredményt elérni. A jelenség másik forrása az, hogy a felvételek készítése során az elfordulást még – a hardver rendelkezésre állásának hiányában – nem mérték, így az csak a feldolgozás során egyenközű osztással került megadásra, amely a valóságos helyzettől eltér. További problémát jelenthet a Kinectbe épített autofókusz funkció, ennek hatása elsősorban akkor észlelhető a modellezés során, ha a felvételek készítése során változik meg a fókusztávolság, ilyenkor ugyanis minden elcsúszik valamennyivel. Mivel ilyen nagy elcsúszásra nincs jel, illetve a felvételen lévő statikus objektumok távolsága is állandó, ez a hibalehetőség itt gyakorlatilag nem játszik szerepet. Ezek a kis törésvonalak igazi kalibráció és elfordulásmérés megvalósításával tűntethetők majd el. Ennek elkészültéig a felvételek számának növelése, a kalibrációs paraméterek finomhangolása és simító algoritmusok alkalmazása ad lehetőséget a pontosabb eredmények elérésére.

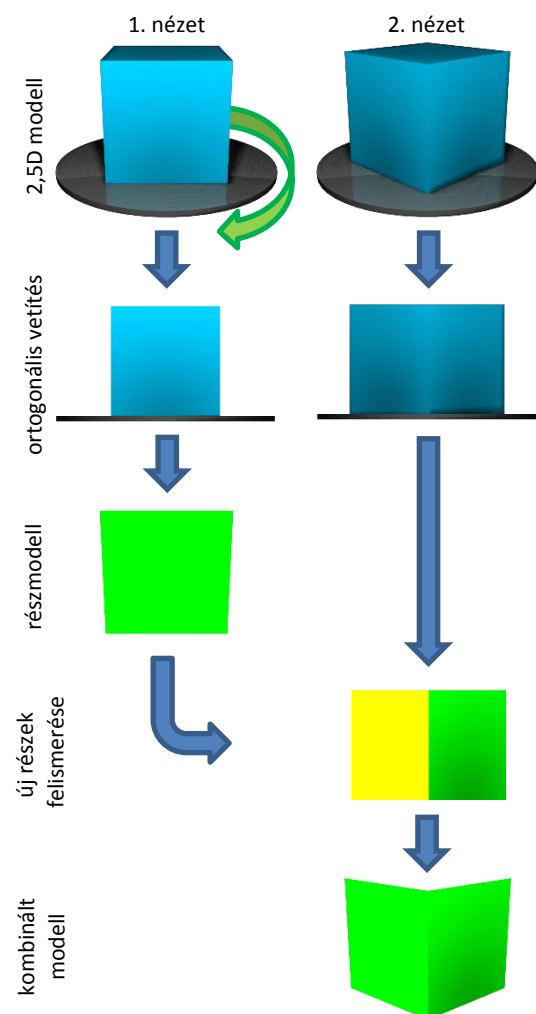
5.3 Modellképzés keresztmetszetenként

Ebben a részben egy olyan módszert mutatok be, amely a modellt egymással párhuzamos szeletekből építi fel. A két láb modelljének megalkotása itt is egymástól elkülönítetten megy végbe. A feldolgozás folyamata az átlagolási lépésig megegyezik, de ezután jelentősen eltér. Amint az majd látható, az egyes nézetek közös koordináta-rendszerben való felírása nem jelentett nagy problémát. Az igazi kihívás inkább az volt, hogy hogyan lehetne az egyes részmodelleket egyesíteni. Ennek az egyesítésnek ugyanis előfeltétele, hogy az egyes részmodellek azonos modellfelületeket leképező részeit össze lehessen párosítani. Ami a zaj, a különböző torzítások, illetve kalibrációs hibák miatt nem oldható meg könnyen.

5.3.1 Részmodellek megalkotása

A bemenet szűrését ilyenkor is újrapetítés követi, azonban ez nem egy hengerkoordináta-rendszerbe, hanem egy ortogonális vetítéssel történik. Az ortogonális vetítés vetítési síkja megegyezik az eredeti képsíkkal, és lehetővé teszi, hogy az egy sorban álló pixelek magassága egyenlő legyen, amely a modell létrehozása során használható majd fel. Az újrapetítési renderelési célba a felületet felépítő pontok háromdimenziós pozíciói kerülnek. A két lábat külön-külön kirajzolva a kapott eredményt visszamásolom a GPU-ról a központi memóriába, mivel a modell elkészítésének algoritmusai a CPU-n kerültek megvalósításra annak természete miatt.

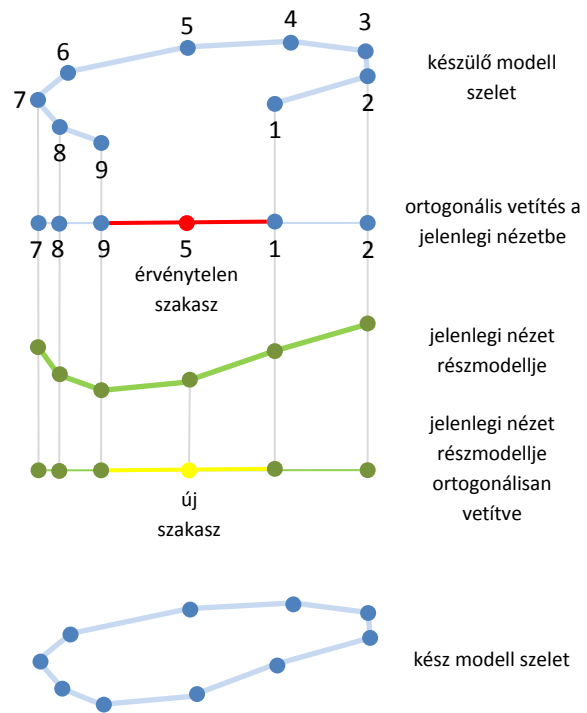
A modell leírása láncolt listák tömbjével történik. Ahol a tömb minden eleme egyet ír a párhuzamos keresztmetszetek közül. A modellezés kezdetén az összes láncolt lista üres. Az első képkocka feldolgozása úgy történik, hogy a sorokat egymás után beolvasom, és az összes olyan pontot, amelyre történt rajzolás felveszem a sor láncolt listájára. A pontok pozíciója mellett rögzítem a kamera optikai tengelyének vektorát a forgótálcához rögzített koordináta-rendszerben, amelynek felhasználására a háromszögelés során kerül sor. Ezzel a művelettel megkapom a modell első képkockán látható részét.



38. ábra A részmodellek összerakásának folyamata

5.3.2 Modellépítés

A feladat ezután a modell fokozatos bővítése, ehhez pedig ismerni kell, hogy a modell mely részei ismertek és melyek nem. Hogy ezt megtudjam az eddig elkészült modellt kirajzolom abból a szögből, ahogy a feldolgozandó képkockán is látszanak az ortogonális vetítés után. A rajzolást szeletenként végzem el töröttvonalak segítségével. Ez a módszer azért előnyös, mert így nem kell háromszögeket fektetni a csomópontokra, így a számolási idő csökkenthető. Az ortogonális vetítés miatt minden sorba egy töröttvonal esik, így egy tömör érzést keltő kép keletkezik, annak ellenére, hogy valójában csak szakaszok sokasága kerül a kirajzolásra. A rajzolás során a képre a szakaszok sorszáma kerül, így nem csak azt lehet megmondani, hogy a feldolgozandó képkocka mely pixelai hordoznak már ismert információt, hanem azt is, hogy azokhoz melyik elem tartozik az adott sor láncából. Ezután



39. ábra Egy szelet felépítése

elkezdem a sorok végigolvasását, amit a félkész modell azonos nézetben készült képén és a feldolgozandó újrajvetített képkockán párhuzamosan végzek. Amennyiben az eddig elkészült modell képén olyan pixelre jutok, amelyre már történt rajzolás, úgy onnan kiolvasom, hogy az hányadik szakaszból készült. Ezt a sorszámot felhasználva a láncolt listán a megfelelő szakaszra állok. Ha ilyenkor a feldolgozandó képkockán sem üres a megfelelő pixel, akkor sem kell tenni semmit, hiszen az itt található rész már le lett modellezve. Ha a feldolgozandó képkockán úgy találok nem üres pixelt, hogy a megfelelője a félkész modell rajzán üres, akkor az adott rész még nem szerepel a modellen, ezért hozzá kell adni ahhoz. Ilyenkor az új pont beszúrása történhet a lánc elejére, belsejébe és végére is, aszerint, hogy épp hányadik elemen állunk a listában. A listába való beszúrások miatt a beszúrás után szereplő elemek sorszáma megváltozik, ezért az aktuális elem eredeti sorszáma egy külön változó tartja nyilván, amely beszúrás esetén nem kerül növelésre, csak az ismert modellrészekre való áthaladáskor. Mivel a modell még befejezetlen, ezért előfordulhat, hogy a felületének belső oldalát látjuk a vonalas rajzon. Ez úgy ismerhető fel, hogy ilyenkor a szakaszok sorszáma nő és nem csökken (39. ábra).

5.3.3 Modellfinomítás

Ha teljes körbeforgást ugyanezen műveletsornak vetjük alá, akkor megkapjuk a háromdimenziós modellt. A modell felületének nagy része azonban durva lesz. Ez annak a következménye, hogy minden felületrészt akkor modellezünk le, amikor azt először láttuk. Ez azonban előnytelen, mert a felületek először oldalról láthatóak, amikor a normálvektoruk nagy szöget zár be az optikai főtengellyel. Ilyenkor az adott felületrész csak néhány pixel szélességben látható, ráadásul a Kinect mélységérzékelése az oldalról látható felületek esetén nagyon bizonytalan és általában nagy hibával is rendelkezik. A megoldás fűrészfogszerű ugrásokkal tagolt felületeket eredményez. Mivel ez így nem fogadható el, ezért módszert dolgoztam ki a jelenség megszüntetésére. A hiba azért keletkezik, mert

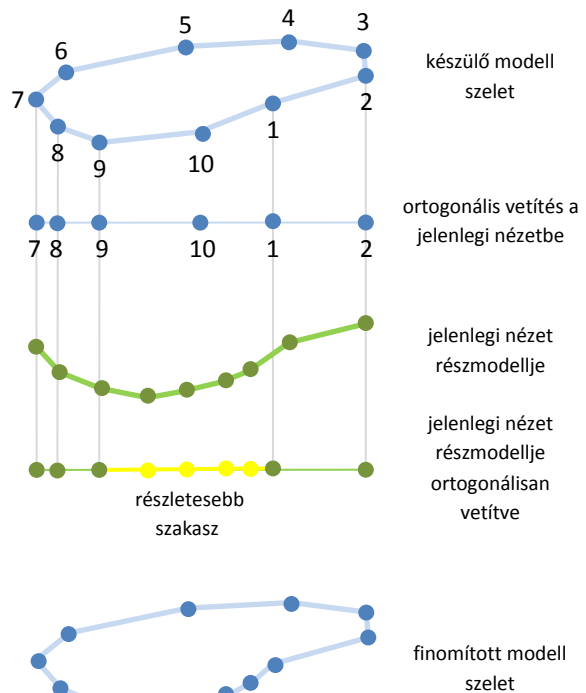
az adott felületrészt nem abból a szögből nézve rögzítjük, ahonnan az a legjobban látszik. Kézenfekvő megoldás tehát annak felismerése, ha egy felületrészt jobban látunk, mint eddig. Ez megtehető úgy, hogy figyeljük, milyen széles egy szakasz a készülő modell vonalas rajzán. Amennyiben a szakaszok az adott nézetben több mint egy pixel hosszúak, úgy az adott rész az aktuális helyzetben jobban látható, mint amikor az eredetileg rögzítésre került. Ilyenkor a rendszer kicseréli a már rögzített részt a jobban látható verzióval. Ezzel elértük, hogy a felület minden pontja a lehető legjobb helyzetből legyen felvéve. A forgópad teljes körbefordulása után már nem történik változtatás, hiszen minden felület leképezésre került.

5.3.4 Háromszögelés alapalgorithmusa

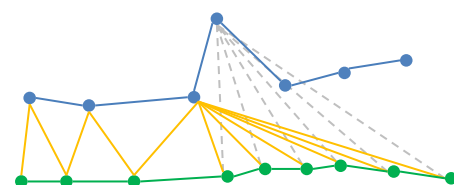
Ezzel elkészült a lábak párhuzamos keresztmetszeteinek sorozata. Azonban ezzel a modell még nincs kész, ugyanis ahhoz még az egyes szeletekre háromszögeket is kell fektetni. Amelynek jó minőségű megoldása nem könnyű feladat.

A háromszögelés során egyszerre mindig két réteg összekötésével foglalkozom, amennyiben minden csúcspont felhasználásra kerül, úgy az egymás feletti rétegek lyukmentesen fognak kapcsolódni egymáshoz. A két szomszédos réteget jelölje A és B . A háromszögelés megkezdéséhez kiválasztom az első pontot az A rétegből, majd megkeresem a legközelebbit a B -ből. Ezzel megkapjuk az első háromszög két csúcsát. A továbbiakban az aktuális élhez egy-egy új csúcs hozzáadásával - amely az A és B rétegről is történhet - egy-egy új háromszöget nyerek. A háromszög kezdőoldalát megadó élet ezentúl vándorló élnek fogom nevezni. Így a feladat egy olyan algoritmus kidolgozása, amely eldönti, hogy az A és B réteg közül melyiken kell továbblépni. A döntés alapja az kell, hogy legyen, hogy minél „szebb” eredményt kapjunk. Ehhez el kellett döntenem, hogy mi számít szép eredménynek:

- a) Arra kell törekedni, hogy a vándorló él minél rövidebb legyen, azaz a réteget úgy kell kiválasztani, hogy a továbblépési csúcsot a másik réteg aktuális pontjával összekötve rövidebb szakaszt kapjunk, mintha a másik rétegen lépnénk tovább. Bár a módszer elsőre jónak látszik, könnyen megadható pontok olyan rövid sora, amellyel a rendszer folyamatosan csak az egyik oldalon fog továbblépni, és onnan nem is tér vissza. Amint, látható ez a módszer nem alkalmas a feladat megfelelő megoldására.

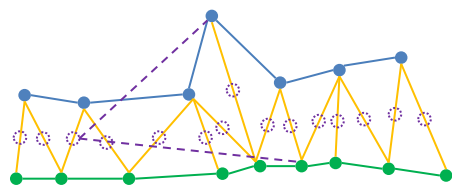


40. ábra Egy szelet finomítása



41. ábra Háromszögelés a legrövidebb vándorél választásával (perspektivikus nézet), annak ellenére, hogy a sárga élék rövidebbek, mint alternatívák (a hibás részen szaggatott vonallal jelölve), hibás eredményt kapunk egyes esetekben.

- b) Az előző esetben bemutatott hiba egyes esetekben elkerülhető, ha nem a vándorlóért minimalizálom, hanem az új háromszög területét. A 41. ábra szerinti alakzat azonban ezt a módszert is „kiakasztja”. (Felmerülhetne a terület minimalizálása, de ez rossz elképzelés, mivel a nagyon elnyúlt háromszögek területe is kicsi.)
- c) Azon a rétegen lépek tovább, amelyen a következő vertex az n lépéssel ezelőtti vándorló él közép-pontjához közelebb van. Ezt az indokolja, hogy így az az öngerjesztő folyamat, amely ahhoz vezet, hogy csak egyik oldalon lépünk nehezen indul el. Ez a módszer nagyon bízható eredményeket ad (42. ábra).



42. ábra Háromszögelés lemaradó referenciaponttal (perspektivikus nézet), a módszer a két rétegen következő vertexek távolságait hasonlítja össze, és a közelebbi vertexhez húzza a következő élet.

A programban elsősorban a c) módszert használtam, az algoritmus későbbi kiegészítései lehetővé teszik az a) és b) módszerek használatát is.

5.3.5 Háromszögelési algoritmus finomítása

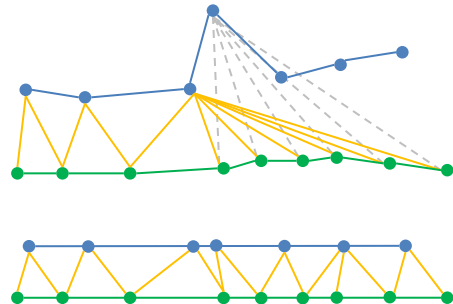
Ahhoz, hogy az előbb kiválasztott algoritmus minden esetben jó eredményeket adjon, számos kiegészítésre volt szükség. Ezeket ismertetem ebben a fejezetben.

Azt is megtehetjük, hogy folyamatosan figyeljük, hogy nem alakul-e ki öngerjesztő folyamat. Ha az új élék hosszát folyamatosan figyeljük, akkor felismerhető a kedvezőtlen helyzet, hiszen ilyenkor az élék túl hosszúvá nyúlnak. Ebben az esetben az élék visszatörölhetők addig a pontig, ami utoljára megfelelt az optimális élhossznak. Majd új kezdőpontot keresünk, ami úgy történhet meg, hogy egyik réteg elemein végighaladva minden ponthoz megkeressük a legközelebbit a másik rétegről, ilyenkor persze azt is folyamatosan figyelni kell, hogy ne fussunk bele már leháromszögelt szakaszba (ha ez megtörténik, akkor körbeértünk, a kimaradt részek nem háromszögelhetőek). A módszer képes kijavítani az előzőleg jelentkező öngerjesztő jelenséget, de hatására lyukak maradhatnak a felületen (például akkor, ha a bemeneti szeletekből szakaszok hiányoznak, azaz az egyes csomópontok túl messze helyezkednek el egymástól).

A megoldás során kérdés lehet, hogy a háromszögek visszatörölése során meddig kell visszamenni és az is, hogy mikor mondhatjuk, hogy a háromszögek túl hosszúak lettek. Kezdetben egy hosszlimitet alkalmaztam, amely alatt az élék jók voltak, míg fölötte hibásak. Ez a módszer azonban nem bizonyult jónak, mivel egyes helyeken, kimaradtak háromszögek, amelyek jók lettek volna. Ilyen hely például a lábfej, ahol a felület meredeksége függőlegesen nagy és még vízszintes irányban is erősen görbül a felület. Ezért két mérethatárt vezettem be: az alsó határ azt kívánja jelezni, hogy meddig tekinthető kiválónak az adott szakasz hossza, efelett ugyan a szakasz hossza már kicsit nagy, de ha ez csak néhány elem erejéig áll fent, akkor még elfogadható. Ha a hossz az alsó határ felé nő, akkor a rendszer elkezd számolni a hosszú szakaszok számát, ha ez elér egy határt, akkor a rendszer elkönnyveli a hibát. Van továbbá egy felső határ is, amely felé növekedő élhossz már nem tolerálható. A rendszer emellett folyamatosan tárolja az utolsó, alsó határméretnél rövidebb él csúcspontjainak sorszámát, és ezt használja alapként a kereséshez. A keresés mindig egy csomóponttal később indul mindkét oldalon, mint az utolsó optimális hosszúságú él, hogy ugyanoda ne lehessen visszajutni. A keresés eredménytelenül ér véget, ha az algoritmus visszaér az első a két réteg közé beszúrt élig, vagy ha még egy él sem került beszúrásra, akkor a láncok végét elérve. A keresés során az elsődleges

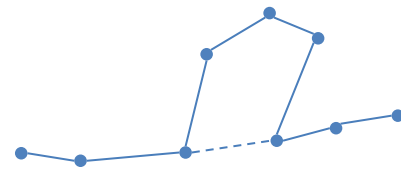
lépéseket a felső élen teszem, és itt állva nézem az éllehetőségeket a túloldalon haladva. Amennyiben a keresés sikeres, úgy a program a haladást leíró változókat a megfelelő értékre állítja, mivel az ugrási lehetőségek összetett megállapítása nagyban növelik a végtelen ciklusok keletkezésének esélyét, ezért ezt a részt kiterjedt tesztelésnek vetettem alá.

A háromszögelés minősége nagymértékben javítható úgy, hogyha nem csak a csúcsok pozícióját rögzítem, hanem a kamera optikai tengelyének helyzetét is a fúziós koordinátarendszerben. És az b) vagy c) módszereket úgy alkalmazom, hogy a távolságszámításnál az optikai tengelyre merőleges komponenst veszem csak figyelembe. Ez azért célszerű, mert egy képen az egymás melletti pontok háromszögelése az ortogonális vetítés miatt triviális feladat. Amennyiben a vándorló él két végpontja nem ugyanabban a helyzetben lett rögzítve, akkor a hozzájuk tartozó optikai tengelyvektorok átlaga használható. Ilyen módon a háromszögelés minősége nagymértékben javul azokon a helyeken, ahol eddig érzékelési hiba miatt a két réteg megfelelő szakaszai nagymértékben eltértek. Ezeket a helyeket az eddigi módszerekkel nem lehetett jól kezelni, mert a csak a két réteget vizsgálva nem egyértelmű, hogy milyen irányban áll a felület (43. ábra). Amint az látható a nézőpont információkat is felhasználva a megoldás már egyértelműen megtalálható a kezdetben hibát okozó algoritmussal is.



43. ábra Ha az elrendeződést megfelelően megválasztott síkba vetítjük, akkor a kezdetben hibát okozó a) algoritmus is használhatóvá válik ebben az esetben.

A kimenet minősége úgy is feljavítható, ha a kiugró csúcspontokat eltávolítjuk a töröttvonalból. Ennek egyik megoldása az lehetne, hogy a szakaszok láncára valamilyen görbét fektetünk, majd kiszámítjuk az egyes csúcspontok ettől mért távolságát, és azokat a vertexeket, amelyek túl távolra esnek eltávolítjuk, vagy közelebb helyezzük. A tüskék eltávolítása ennél hatékonyabban is elvégezhető oly módon, hogy az n . és az előtte elhelyezkedő $n - 1$, $n - 2$, ... $n - k$. csúcspontok távolságát kiszámítjuk páronként (k nagyságát 5-10 tartományban érdemes megválasztani, egy réteg kb. 300-500 elemből áll). Majd az így kapott k távolság közül kiválasztjuk a legrövidebbet. Amennyiben ez a távolság nem az $n - 1$. csomópontozhoz tartozik, úgy a legközelebbi és vizsgált vertex közötti elemeket törölni kell. Ez a módszer azzal az előnnyel jár, hogy az első pontokban tapasztalt egy oldalon ragadást kiváltó elrendezések típusainak nagy részét megszünteti, illetve eltünteti a fűrészfogszerű felületeket is.



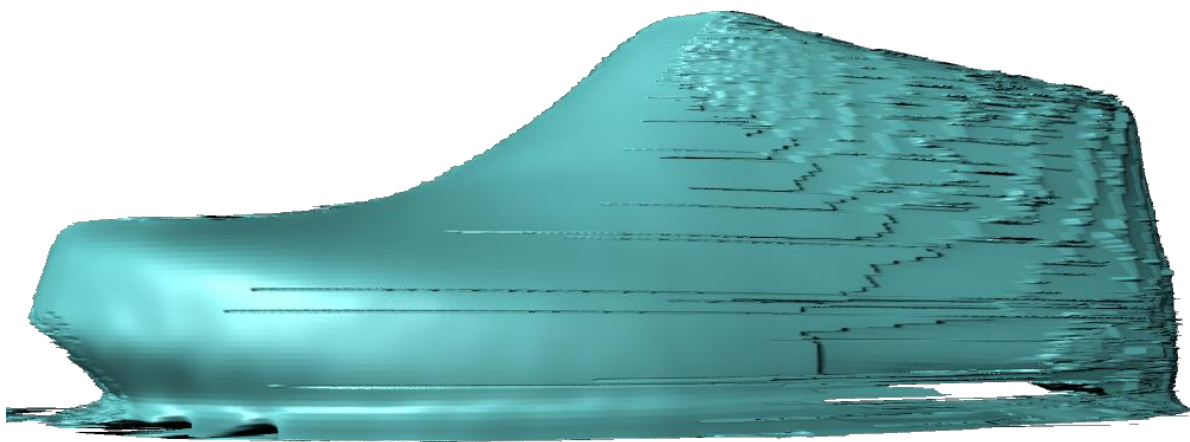
44. ábra Kiugró pontok/pontsorozatok eltávolítása a láncból.

A kidolgozott modellezési módszer a teljes körbefordulás után befejezi működését, és a modellt leíró pontsorok állandó állapotba kerülnek. Az eltérő nézőpontok és a kalibrációs hibák miatt azonban rendszeresen előfordul, hogy az egyes szinteket leíró töröttvonal végei nem pontosan találkoznak, hanem az egyik vég alábukik a másiknak. Ennek kiküszöbölésére kódot dolgoztam ki a probléma elhárítására. A módszer a töröttvonal végét részesíti előnyben és annak elejéből törli a felesleges szakaszt. A program a töröttvonal elejétől elindul és kiszámítja annak utolsó csomópontjától mért távolságot. Amennyiben a távolság egy bizonyos határérték alá csökken, úgy elkezd keresni a legközelebbi eső csomópontot, amit egészen addig végez, amíg a csúcsok közti távolság újra nagyobb nem lesz a már említett határértéknél. Ezután, amennyiben a közel eső

csomópontok száma elér egy minimális értéket (pl. 5), akkor a legközelebb eső pont előtti elemeket törli a láncból, máskülönben folytatja a keresést. Az esetleges hibás döntések elkerülése érdekében a program csak akkor végzi el a csomópontok törlését, ha azok mind a töröttvonal szakaszsorszám szerinti első kétharmadában vannak. Ilyen módon sosem érhet körbe a rendszer.

Még jó minőségű háromszögelés esetén sem biztosított, hogy miután körbe érünk, pont a megfelelő helyre illesztjük be az utolsó háromszögeket. Ezért, ha az egyik rétegben körbeért a program, akkor utána már csak a másik oldalon végez lépést, még akkor is, ha ez ellentmond a modellezés többi szakaszán használt háromszögeket létrehozó algoritmusnak. Amennyiben az adott réteg befejezése sikertelen keresés miatt történt, úgy a rendszer ellenőrzi a háromszögelt szakasz végeinek távolságát, és ha az nem túl nagy, akkor lezáró háromszögeket szűr be.

A háromszögelő a modellt index és vertex tömb segítségével írja le, amely egyrészt egyszerűsíti az adatok tárolását, hiszen minden vertex csak egy helyen szerepel, így pl. simítás végrehajtása során csak egy helyen kell módosítani az adatokat, másrészt a memóriahasználatot is csökkenti. Az index tömb a háromszögeket adó csúcspontok sorszámaikat tárolja. Mivel a program az adatokat a későbbi felhasználáshoz a többi modul számára STL formátumban kell, hogy nyújtsa, amely hármassával a vertexek pozícióit és a háromszög normálvektorát tárolja, így a mentés során a vertex- és indexpuffer tartalmát ilyen formába kell hozni. Hogy ne kelljen a teljes modellt többször tárolni a memóriában, a program a modellt fokozatosan egy kisebb írási pufferbe írja (a modell átalakítása és a pufferbe írás egy lépésben megy végbe, tehát nincs szükség további memóriára), majd ha ez betelt, akkor a lemezre viszi át az adatokat és újra elkezdi feltölteni a puffert. A kimentett modellek többfajta programmal is megjeleníthetők, a következő képernyőképeket Autodesk Maya segítségével készítettem:



45. ábra A kaptafáról készült modell a felületi simítás nélkül, amint az látható a felületen 0,5 - 1mm nagyságú kiszögellések találhatók, amelyek a tesztfelvételek becslésen alapuló kalibrációjának, és a felvételek manuális elkészítésének köszönhetőek

Amint látható, itt erőteljesebben jelentkezik a félkész, precíz módon még nem kalibrálhatott hardver, illetve az elfordulásmérés hiányának hatása. Ennek az az oka, hogy az előző esetben a henger-koordinátarendszerben történő átlagolás a felületet simította, minél több felvétel készült, az annál simább lett. Itt ez a hatás nem jelentkezik, a különálló nézetekből kiemelt szakaszok láthatóvá teszik az „összehegesztési” vonalakat. Jól látható, hogy a háromszögelő és a modellező algoritmus helyesen működik, megfelelően rakja össze a modellt. A felületi hibák ebben az esetben is a

hardveres oldal elkészültével lesznek orvosolhatóak. Működési elvéből adódóan, itt valószínűleg nagyon precíz kalibráció esetén is jelentkeznek majd elsősorban esztétikai hibát jelentő mikro-törésvonalak, ezek simítással eltávolíthatóak.

6 Sztereoszkópikus megjelenítés

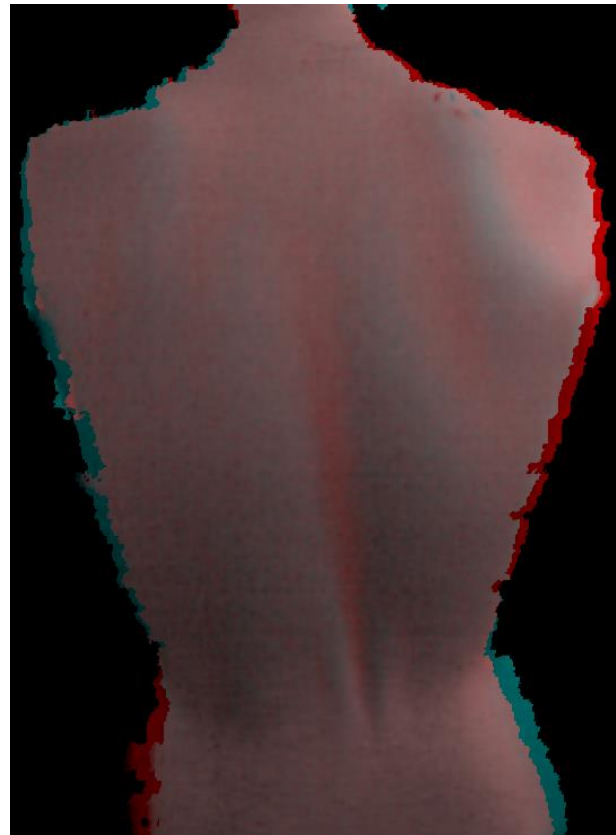
Napjainkban egyre többet találkozhatunk sztereoszkópikus megjelenítést alkalmazó eszközökkel. Ma már nem csak a moziban, hanem otthon a televíziókészülékek, monitorok, sőt mobiltelefonok révén magunk is elérhetjük a megoldásokat. A sztereoszkópia a történelem során már számos időszakban divatba jött, de általában a technológiai megoldások kezdetlegessége miatt később mindig háttérbe szorult. Ma újra egy ilyen időszakot élünk, amelyben minden eddiginél jobb megoldások állnak rendelkezésre. Ezért döntöttem úgy, hogy programomban is lehetőséget biztosítok sztereoszkópikus megjelenítésre.

A sztereoszkópikus kimenet előállítását általában egyszerű, ha már a hagyományos háromdimenziós programrészeket elkészítettük. Két műveletet kell hozzá elvégezni:

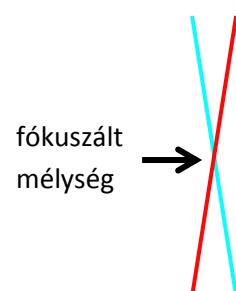
- A kamerát vízszintes tengelye mentén el kell tolni balra illetve jobbra: ezzel állítjuk elő a két különböző nézőpontból készült, a bal és jobb szemünknek szóló információt.
- A két így előállított képet „össze” kell tolni. Máskülönben agyunk nehezen tudja értelmezni a látottakat, a kép „szétesik”.

A kamera vetítését úgy módosítottam például a jobb szem esetén, hogy a vetítés előtt a világot eltolom balra, majd a vetítés után kapott vektorokat eltolom jobbra. Ebből látható, hogy egy ilyen megoldásnak két paramétere van: a kamerák távolsága és a fókuszált mélység. Ez utóbbi az a mélység, ami a megjelenítő felszínével megegyező távolságban látszik. A név onnan származik, hogy a bal és jobb szemnek szóló képeket egymásra helyezve ez a mélység szint ugyanott fekszik a két képen.

A paraméterek beállítására érdemes lehetőséget biztosítani a programban, mert az eltérő emberek más és más beállítást találhatnak optimálisnak. Amennyiben a két nézet rendelkezésre áll, már „csak” azt kell megoldani, hogy az egyiket a jobb, a másikat a bal szemünk elé tárjuk. Erre számos különböző megoldást fejlesztettek ki. Ezek közül én most csak hármat ismertetnék, amelyek közül az első kettőt használják ma elterjedten, míg a harmadik azért kerül ismertetésre, mert dolgozatban ezzel készült sztereoszkópikus képeket helyeztem el:



46. ábra Egy már bemutatott kép anaglif megjelenítésben.



9. ábra A fókuszált mélység értelmezése

- **Active-shutter technológia:** ez a technológia azon alapul, hogy a megjelenítő magas frekvencián és nagy fényerő mellett időosztásosan hol a bal, hol a jobb szemnek szóló képet jelenít meg. A megjelenítővel szinkronban az emberi szemeket felváltva eltakaró szemüveget visel a felhasználó. Így az idő közel felében csak a jobb, míg a másik felében csak a bal szemével lát (az átváltási idő ezt valamivel csökkenti). Ha a kitakargatás elég nagy frekvenciával történik, akkor az ember azt már nem észleli, így nem zavaró. A technika problémája, hogy nagy képváltoztatási sebességet és pontos szinkronizációt igényel. Hátrány továbbá, hogy a szemüvegek elektronikát tartalmaznak (innen származik az aktív 3D elnevezés), ami energiát igényel, és nagyobb súlyt, méreteket eredményez. A legnagyobb probléma azonban a legtöbb ember számára a vibrálás érzékelése, ami gyorsan elfárasztja a szemet és fejfájást is okozhat.
- **Polarizációs szűrés:** ennek a megoldásnak a lényege, hogy az egyes szemeknek szóló fényt eltérően polarizálják. A néző egy polarizációs szűrőt tartalmazó szemüveget visel, ami csak az adott szemnek szóló fényt engedi át. Az eltérő polarizáció ma általában cirkuláris polarizációt takar, mert ennél nem akkora probléma, ha a szemüveget viselő a fejét megdönti. Így kisebb az áthallás, ugyanakkor a fej megdöntése a felvételhez képest a szemek látóterének és a felvételnek a vízszintes tengelyét egymáshoz képest elforgatja, ezért az agy egyre kevésbé tudja értelmezni a látottakat. A módszer megvalósítható több elrendezésben, például két eltérően polarizált fényt kibocsátó projektor és ezüstözött vászon segítségével, vagy a monitorok, televíziók esetében a páros és páratlan sorok eltérő polarizációjával (Film-type Patterned Retarder, FPR alkalmazásával). Egy ilyen monitorral magam is rendelkezem, így ezzel teszteltem a megjelenítés működését.
- **Anaglif sztereográfia:** ennél a bal és jobb szemnek szóló információk szín szerint vannak szűrve megfelelő szemüveg alkalmazásával. A módszer legegyszerűbb verziói azok, amelyek csak két színre szűrnek, így színhű megjelenítést nem biztosítanak. Ilyenek például a jól ismert vörös-cián szemüvegek, amelyekből egyet a dolgozathoz mellékeltem. A fejlettebb megoldások vörös, zöld és kék színek egy-egy eltérő, szűk tartományát szűrik ki a két szem számára, így színes megjelenítés is létrehozható. Ezen módszer nagy előnye, hogy nyomtatásban, alacsony költségű papírszemüvegek segítségével alkalmazható.

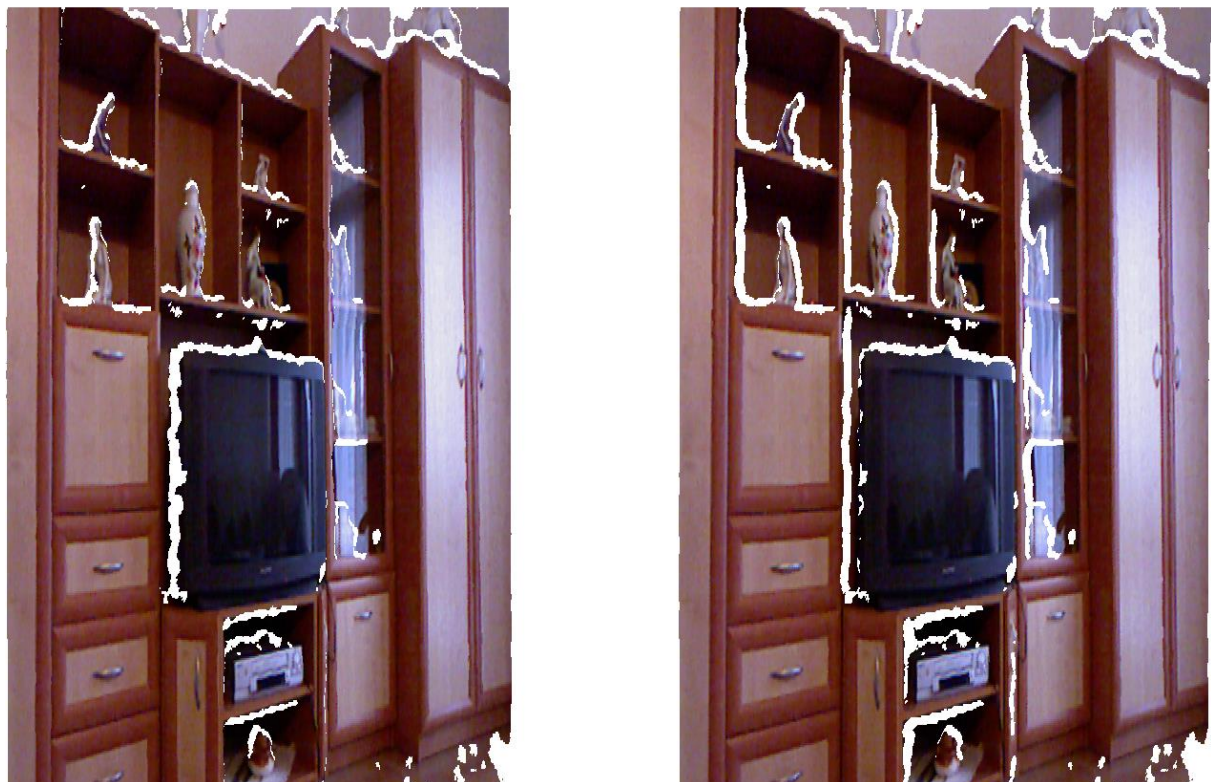
Én e három módszer közül a második kettőt próbáltam ki. A passzív megjelenítéshez elegendő volt osztott képernyős kimenetet biztosítani. Ennél a bal illetve jobb szemnek szóló információk a egymás mellett vagy alatt kerülnek megjelenítésre (48. ábra). Az általam használt monitort megfelelő módra állítva az a képeket összefűzi, és a két képet a páratlan illetve páros sorokra küldi. Ezzel az egy-egy szem által látott vízszintes felbontás megfelelődik, emiatt érdemes a fent-lent tagolású kimenetet alkalmazni. A megoldással így teljesképernyős módban valós időben tudok a Kinecttel megfigyelt térrészről sztereoszkópikus megjelenítést biztosítani.

Az anaglif megjelenítéshez felhasználtam a színes nézeteket. Először kiszámoltam a pixelek világosságát, majd ezt a vörös, illetve cián színekkel beszoroztam. Ezután a két képet összegeztem. A mellékelt szemüveggel jól látható a dolgozatban elhelyezett anaglif képek térhatása.

A program mindkét megjelenítési módot a grafikus kártya segítségével állítja elő valós időben. Mivel a sztereoszkópikus képek előállításához a modellt kétszer kell kirajzoltatni, így ez a terhelést valamelyest növeli. Ugyanakkor ez globális szinten nem jelentős, mert a grafikus kártya terhelésének nagy része nem az eredmények megjelenítéséből ered.



47. ábra Egy szekrény sor anaglif képe



48. ábra Egy szekrény sor sztereoszkópikus összenyomott vízszintes bal-jobb módban renderelve 3D megjelenítők számára

7 Továbbfejlesztési lehetőségek

A program fejlesztése a dolgozat írásának időpontjában is igen aktívan zajlik, így számos jelenleg is folyó továbbfejlesztési irányt mutathatok be. Ilyen például:

- A Microsoft nem sokkal ez előtt adta ki a Kinect fejlesztőkészlet egy új verzióját [13], amely többek között képes az eddig csak az OpenKinect driver által elérhető infravörös videófolyam elérésére, és számos kamera beállítás (pl. expozíciós idő) beállítására. Ez lehetőséget adhat az autofókusz által okozott problémák megoldására, illetve a Kinect infravörös kamerájának önelvakítására is. További előny, hogy a hardvertelepítés is egyszerűbbé válna.
- A forgópados szkennerek fejlesztése még folyamatban van, az itt bemutatott modellezési eljárások már elég szilárdak ugyan, de sok fejlesztés még hátra van. Ezek közé tartozik például a program végleges hardverrel (ez a dolgozat leadásának idején készült el) való összekötése, így az automata működés biztosítása is.
- A Kinect kalibrációjának teljes automatizálása és a MATLAB szkript alkalmazása helyett annak a programba való beépítése.
- A forgópados szkennerek elkészülte után, annak pontos kalibrációját el kell végezni, ezután kerülhet elemzésre az eredmények minősége.
- A program kezelőfelületének egy része – a forgópados szkennerek felvételeinek feldolgozásával foglalkozó - még csak tesztelési célokat szolgál, ezeket a gyakorlati alkalmazás előtt felhasználóbarátabbá kell alakítani.
- A program hardverfüggetlenségének növelése: jelenleg a program az újabb (DirectX Shader Model 3-at és vertex shaderok textúra olvasását teljesen támogató) grafikus kártyákkal működik [20]. Régebbi, illetve integrált grafikus feldolgozót tartalmazó (kivéve az utolsó generációkat) számítógépek esetén a program nem indítható el. Ennek megoldásához át kell alakítani a grafikus kártyán futó részeket, hogy azok szűkebb utasításkészleten operáljanak, illetve rövidebbek legyenek.
- A program sztereoszkópikus részei nem kompatibilisek számos hardverrel, így indokolt lehet valamelyik elterjedtebb, mindenféle kimenetet előállítani tudó programcsomag alkalmazása. Jelenleg a TriDef Ignition szoftverrel való integráció fejlesztése folyik.
- Kinect modelljének további fejlesztése, például a kvázistatikus helyfüggő képhibák hatásának csökkentése, vagy a mélység függvény helyfüggésének vizsgálata.

A fejlesztés folyamán természetesen folyamatosan történik az esetlegesen felbukkanó programhibák javítása is. Az általános és gerincdiagnosztikát célzó részek széleskörű és hosszú tesztelésnek lettek alávetve már igen megbízhatóan, hibamentesen működnek (éles helyzetben is már alkalmazásban vannak). Az új forgópados részek még fejlesztési-tesztelési fázisban vannak, a kalibráció elvégzése, a hardverrel való összekapcsolás és a kezelőfelület végleges kialakítása után következik majd a programstabilitás vizsgálata, amely után a program e része is kiadásra kerül. Ez utóbbira valószínűleg 2013 első negyedévében kerül majd sor.

A program automata on-line frissítési modullal van ellátva, így az egyes fejlesztéseket, javításokat a felhasználók folyamatosan megkapják. A frissítő modul a program minden indítása esetén ellenőrzi, hogy elérhető-e újabb verzió, és ha igen, akkor felkínálja ennek telepítését a felhasználó számára. Amennyiben ő a frissítést engedélyezi, akkor az automatikusan végbemegy. A programhoz telepítő is készült, amely biztosítja az ahhoz szükséges komponensek megfelelő installációját, a fájltypusok társítását, illetve a program megfelelő eltávolítását.

8 Konklúzió

A dolgozatban bemutatam két különböző programot, amelyek a mélylégszenzorok orvostechnikai alkalmazására nyújtanak lehetőséget. Amint az látható volt, ezen szenzorok segítségével jó minőségű modellek állíthatóak elő hatékony módon, anélkül, hogy a leképezendő környezetet elő kellene készíteni. A szenzorok kis helyigénye lehetőséget ad arra, hogy azok mobileszközökben is alkalmazhatóak legyenek. Az elkészült modellek jól felhasználhatóak mind deformációk, sérülések vagy egyéb rendellenességek elemzésére, azok alakjának rögzítésére, mind testreszabott orvosi segédeszközök vagy ruházati cikkek előállításához. A jelenlegi szenzorok felbontása még nagy növekedés előtt áll, ami később újabb területeken teszi használhatóvá azokat. A felbontás mellett még a mai szenzorok kimenetének zajszintje hagy némi kívánnivalót maga után.

Az algoritmus szempontjából elsősorban az egyes háromdimenziós felvételek összeillesztése és egy minden feltételt összegző modell építése jár a legtöbb nehézséggel. Ezen műveletek igénylik a legtöbb processzoridőt is a feldolgozás során. Ugyanakkor az itt bemutatott eredmények is mutatják, hogy megfelelő algoritmusok kidolgozásával már a mai szenzorok mellett is relatíve jó minőségű eredmények előállítására van lehetőség. Informatikai szempontból elsősorban a hardveres gyorsítás használatának szükségessége emelhető ki, hiszen enélkül az itt elvégzendő műveletek valósidejű végrehajtása csak igen nagy teljesítményű processzorok mellett lenne biztosítható.

A dolgozat a mélylégszenzorok általános használata mellett két különböző orvostechnikai feladat megoldását is bemutatta. Mindkét esetben olyan megoldások születtek, amelyek a mélylégszenzorok nélkül, ilyen egyszerűen használható, gyors, kis helyigényű és hatékony módon csak magas költségek mellett volnának megvalósíthatók. A gerincferdülést vizsgáló program már valódi használatban van a Heim Pál Kórházban és az Ortopédiai Klinikán. Míg a forgópados rendszer elkészülte 2013 első negyedévére várható. A fejlesztéseket, javításokat a program használói automatikusan megkapják az on-line frissítő rendszernek köszönhetően. A program strukturált felépítése biztosítja, hogy az további irányokban is könnyen bővíthető legyen. Ezt szolgálja a kiépített plug-in rendszer is, amely biztosítja az egyes modulok (pl. forgópados szkennerek) szabályozott interfészen keresztüli beépülését, így a kód jobban átlátható, és a több fejlesztő is könnyen dolgozhat egyszerre.

Mint minden orvostechológiai fejlesztés esetében a mérnöki munka, a tudományos eredmények fontosak ugyan, de a legfontosabb az, hogy az elkészült eszközök, kifejlesztett eljárások minél inkább segítségére legyenek beteg embertársainknak. Így én is remélem, hogy a projekt eddigi és jövőbeli fejlesztései sok ember életét tehetik teljesebbé.

9 Irodalomjegyzék

1. Barak Freedman, Alexander Shpunt, Meir Machline és Yoel Arieli: „Depth mapping using projected patterns”, United States Patent Application Publication, Pub. No.: US 2010/0118123, 2010
2. Alexander Shpunt: „Optical designs for zero order reduction”, United States Patent Application Publication, Pub. No.: US 2009/0185274, 2009
3. „Kinect Pattern Uncovered”, <http://azttm.wordpress.com/2011/04/03/kinect-pattern-uncovered/>, lekérés időpontja: 2012. 09. 15.
4. D. C. Brown: „Close-Range Camera Calibration”, Photogrammetric Engineering, vol. 37, no. 8, 1971, 855-866. oldal
5. Zhengyou Zhang: „A flexible new technique for camera calibration”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, 2000, 1330-1334 oldal.
6. Jean-Yves Bouguet: „Camera Calibration Toolbox for Matlab”, http://www.vision.caltech.edu/bouguetj/calib_doc/index.html, lekérés időpontja: 2012. 09. 17.
7. „OpenNI open source projects repository”, <http://openni.org/Downloads.aspx>, lekérés időpontja: 2012. 09. 28.
8. „Kinect for Windows SDK”, <http://www.microsoft.com/en-us/kinectforwindows/development/developer-downloads.aspx>, lekérés időpontja: 2012. 09. 28.
9. „The OpenKinect project”, http://openkinect.org/wiki/Main_Page, lekérés időpontja: 2012. 09. 28.
10. John Canny: „A Computational Approach to Edge Detection”, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8, Issue: 6, 1986, 679-698. oldal
11. Ayhan Dániel: „Optikai geinccferdülés-diagnosztizáló szoftver fejlesztése”, TDK dolgozat, 2012
12. „Fast triangulation of unordered point clouds”, http://www.pointclouds.org/documentation/tutorials/greedy_projection.php#greedy-triangulation, lekérés időpontja: 2012. 09. 04.
13. Kinect for Windows features: Highlights of October 2012 Release, <http://www.microsoft.com/en-us/kinectforwindows/discover/features.aspx>, lekérés időpontja: 2012. 10. 23.
14. Giovanna Sansoni, Marco Trebeschi and Franco Docchio: „State-of-The-Art and Applications of 3D Imaging Sensors in Industry, Cultural Heritage, Medicine, and Criminal Investigation”, Sensors 2009, no. 9, 568-601. oldal
15. K. Khoshelham: „Accuracy Analysis of Kinect Depth Data”, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XXXVIII-5/W12, 2011, ISPRS Calgary 2011 Workshop, 2011. augusztus 29-31., Calgary, Canada, 133-138. oldal
16. Jan Smisek, Michal Jancosek and Tomas Pajdla: „3D with Kinect”, 2011 IEEE International Conference on Computer Vision Workshops, Proceedings, 1154-1160. oldal
17. Daniel Herrera C., Juho Kannala, and Janne Heikkilä: „Accurate and Practical Calibration of a Depth and Color Camera Pair”, CAIP 2011, Part II, LNCS 6855, 437-445. oldal
18. Riemer G.: „XNA 3.0 Game Programming Recipes – A Problem-Solution Approach”, Apress, New York, 2009.: 231. oldal
19. Gergely Vassy and Tamás Perlaki: „Applying and removing lens distortion in post production”, The Second Hungarian Conference on Computer Graphics and Geometry, 2003, Budapest, 2nd best speech

20. Microsoft Corporation: „Pixel Shader Differences”, Microsoft Developer Network, [http://msdn.microsoft.com/en-us/library/bb219846\(v=VS.85\).aspx](http://msdn.microsoft.com/en-us/library/bb219846(v=VS.85).aspx), lekérés időpontja: 2011. október 15.
21. Microsoft Corporation: „Programming Guide for HLSL”, Windows Development Center, [http://msdn.microsoft.com/en-us/library/windows/desktop/bb509635\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/bb509635(v=vs.85).aspx), lekérés időpontja: 2011. július 22.
22. Microsoft Corporation. „C# Language specification, version 4.0”, Microsoft Press, Redmond, 2010.

10 Köszönetnyilvánítás

A dolgozat írója szeretné köszönetét kifejezni a projekt támogatóinak:

- A Mechatronika, Optikai és Gépészeti Informatikai Tanszéknek, amely a projekt során használt hardvereszközöket rendelkezésre bocsátotta.
- Dr. Tamás Péter tanár úrnak, aki számos hasznos tanáccsal, ötlettel, szervezéssel támogatta a program létrehozását.
- Basch László úrnak, aki a forgószenzor hardveroldalt fejlesztte és a program elkészítéséhez szükséges felvételeket rendelkezésemre bocsátotta.
- Dr. Wenzel Klára tanárnőnek és Molnár József tanár úrnak, akik az anaglif szemüvegeket biztosították.