

TDK-dolgozat

Készítette:
Kovács Aurél
Gálfi Norbert

Debreceni Egyetem
Informatika Kar



Háromdimenziós szkennelés, textúrázás, mesh optimalizálás

Konzulens:

Dr. Cserhádi Csaba
egyetemi docens
Szilárdtest Fizikai Tanszék

Készítette:

Kovács Aurél
Gálfi Norbert
III. éves
Mérnökinformatikus hallgatók

Tartalomjegyzék

1. Bevezetés.....	1
1.1 Fejlesztési célkitűzésünk	1
1.2 A fejlesztőkről.....	2
2. Valós tárgyak 3D leképezési lehetőségei.....	3
2.1 A 3D grafika alapjai	3
2.2 A 3D grafika alapfogalmai	3
2.3 OBJ Fájlformátum.....	5
2.4 Szkennelés 3D-ben.....	7
2.4.1 A 3D szkennelés technológiai megvalósításai.....	7
2.4.2 Kontakt típusú szkennerek.....	7
2.4.3 Nonkontakt típusú szkennerek.....	8
2.4.3.1 Aktív nonkontakt típusú szkennerek	8
2.4.3.2 Passzív nonkontakt típusú szkennerek	10
3. Saját fejlesztésű 3D szkennер szoftver és hardver	12
3.1 A szkennер szoftver főbb moduljainak feladatai	12
3.2 A szkennер hardvere.....	12
3.2.1 Forgótálca	12
3.2.2 Vonallézer.....	14
3.2.3 Kamera	14
3.3 A hardver és a szoftver használata.....	15
3.4 A tárgy anyaga	15
3.5 A tárgy felülete	16
3.6 Tárgy pozicionálása	16
3.7 Fényviszonyok	17
3.7.1 Vonallézer beállítása	17
3.7.2 Fények beállítása a textúra készítéshez	18
3.8 Kalibrálás.....	18
3.8.1 Saját kalibrálási és sugár meghatározási módszereink.....	18
3.8.2 Távolság meghatározás háromszögeléssel	19
3.9 Képek készítése.....	20
3.10 ROI (Range of Interest).....	22
3.11 Méretarányosság	22
3.12 Optimális képfeldolgozási beállítások.....	23
3.12.1 Brightness, Contrast, Gamma Correction.....	23
3.12.2 Kiugró élek eltávolítása	24
3.12.3 Hézagok összekötése	26
3.12.4 Mintavételek átlagolása és polygonok számának csökkentése	27
3.12.5 Moduláris, Pipe-Line szerű kép elő- és utófeldolgozás.....	28
4. Képek feldolgozására használt algoritmusok.....	29
4.1 Brightness, Contrast, Gamma Correction.....	29
4.2 Lézervonal középpontjainak meghatározása és letárolása 2D tömbbe	30
4.3 Kiugró élek eltávolítására használt algoritmusok	31
4.4 Hézagok kitöltése	32

4.5 Mesh optimalizálás.....	33
4.6 Vonalsimítás B-Spline görbével	34
4.7 A tárgy forgástengelyétől való távolság meghatározására használt módszerek	35
4.8 Sorfolytonos adattárolás előnyei.....	38
4.9 Pontfelhő generálása	39
4.10 Szomszédos pontok összekötése.....	40
4.11 Textúrázáshoz használt algoritmusok	41
4.12 A szoftver által generált fájlok.....	43
4.13 Pontosság.....	43
5. Összehasonlítás más eszközökkel	45
6. Továbbfejlesztési lehetőségek	46
6.1 Mechanika automatizálása National Instruments eszközökkel	46
6.2 Lézervonal középpontjának automatikus megkeresése.....	46
6.3 Más módszerek a textúra készítéséhez.....	47
6.3.1 Több mintavétel a textúrázáshoz.....	47
6.3.2 Kevesebb mintavétel a textúrázáshoz.....	47
6.4 Realtime képfeldolgozás NI Smart Camera használatával.....	47
6.5 Kompatibilitás 3D Printer eszközökkel.....	48
6.6 Arc szkennelés	48
6.7 Optimalizálás	48
7. Összefoglalás	50
Irodalomjegyzék.....	51
Köszönetnyilvánítás	52
Függelék	53

1. Bevezetés

Egyre több területen jelentkezik igény a legkülönbözőbb tárgyak, vagy azok részleteinek a számítógépes, háromdimenziós térbeli ábrázolására, kiértékelésére. Így például az orvostudományban, a játékiparban, filmiparban, vagy akár tömeggyártásban történő automatizálás során is.

Az ehhez szükséges erőforrás igényeket - a jelenlegi, és a közeljövőben megjelenő - számítógépeink egyre hatékonyabban ki is tudják majd szolgálni. Ezek a tények is befolyásoltak minket, mikor TDK dolgozati témánknak a háromdimenziós képfeldolgozást választottuk. Közrejátszott az is, hogy szerettük volna, ha közvetlenül, kézzelfogható gyakorlati eredménye lenne munkánknak, továbbá egy olyan hardver és szoftver megalkotása, ami az általunk megfizethető technológiával, az elérhető legjobb minőségű háromdimenziós modellt rekonstruálja.

A fejlesztésbe 2011 őszén kezdtünk bele, a tanulmányaink során megismert LabView-val. Témánk kidolgozását a Mérés és folyamatirányítás szakirány Műszaki képfeldolgozás tárgyának önállóan megalkotott beadandó feladataként folytattuk. Munkánkhoz további motivációt adott a lehetőség, hogy az évente megrendezett „Élő Fizika” kiállításon a Kölcsény Központban is bemutathattuk az általunk készített eszköz és szoftver működését az érdeklődőknek.

1.1 Fejlesztési célkitűzésünk

Célul tűztük ki magunk elé, olyan szoftver és hardver megoldások fejlesztését, melyekkel könnyedén lehet tetszőleges formájú tárgyakat digitalizálni, és olyan algoritmusok kifejlesztését, melyekkel alak és mérethű leképezést valósíthatunk meg, és a legjobban támogatott fájlformátumokkal kompatibilis fájlokat készíteni.

A szoftver első működőképes verzióit látva vált számunkra nyilvánvalóvá, hogy a körülmények javításával, sokkal jobb minőségű 3D modellt is létre tudunk hozni. Több mintavétellel, jobb minőségű képek készítésével, és a szoftverben implementált tetszőlegesen paraméterezhető algoritmusok segítségével törekedtünk a lehető legjobban visszaadni a létrehozott 3D objektummal az eredeti szkennelt tárgy formáját. Természetes igényként merült fel, hogy ne csak a tárgy alakját, hanem felületének mintázatát is képes legyen a szoftver feldolgozni, tehát alkalmas legyen a szkennert szoftvert a textúrázásra is.

Dolgozatunkban a fejlesztés során szerzett tapasztalatinkról, és munkánk jelenlegi állapotáról valamint továbbfejlesztési elképzeléseinkről számolunk be. A dolgot ketten készítettük. A közös érdeklődési terület, valamint egymás munkájának kiegészítése, és segítése adta a lehetőséget, hogy elképzeléseink realizálódhassanak.

1.2 A fejlesztőkről

Kovács Aurél: Már kiskorom óta érdekeltek a műszaki dolgok, és az informatika. A digitális elektronikai áramkörök tervezésének alapjaival tíz éves korom körül kezdtem ismerkedni. Éppen ezért középiskolai tanulmányaimat a Gábor Dénes Elektronikai és Műszaki Szakközépiskolában végeztem, ahol két éves ipari elektronikai technikus képzésben is részt vettem. Programozást az egyetemen tanultam először, ami egyből szakmai érdeklődésem középpontjába került. A LabView azért fogott meg, mert abban egyaránt hasznosíthatom a digitális áramkörök tervezésében, és a programozásban szerzett tapasztalataimat is. A fejlesztés során az én munkám volt:

- A fejlesztés munkafázisainak megtervezése, modulokra osztása
- Főbb algoritmusok logikájának megalkotása, lekódolása
- C, C++, LabView programozás

Gálfi Norbert: Szeretek alkotni, tervezni, és rajzolni. Műszaki és informatikai érdeklődésem miatt választottam a Gábor Dénes Elektronikai és Műszaki Szakközépiskolát. Végzés után egy OKJ-s média szakképzésen vettem részt, emellett webfejlesztéssel és a háromdimenziós grafikák készítésével is foglalkoztam. Az egyetemen a Műszaki Képfeldolgozás tantárgyon használt LabView szoftver keltette fel az érdeklődésemet a háromdimenziós szkennelés iránt. A fejlesztésben az alábbi tevékenységeket végeztem:

- LabView programozás
- Képfeldolgozó algoritmusok programozása IMAQ-val
- Algoritmusok tesztelése, portolása

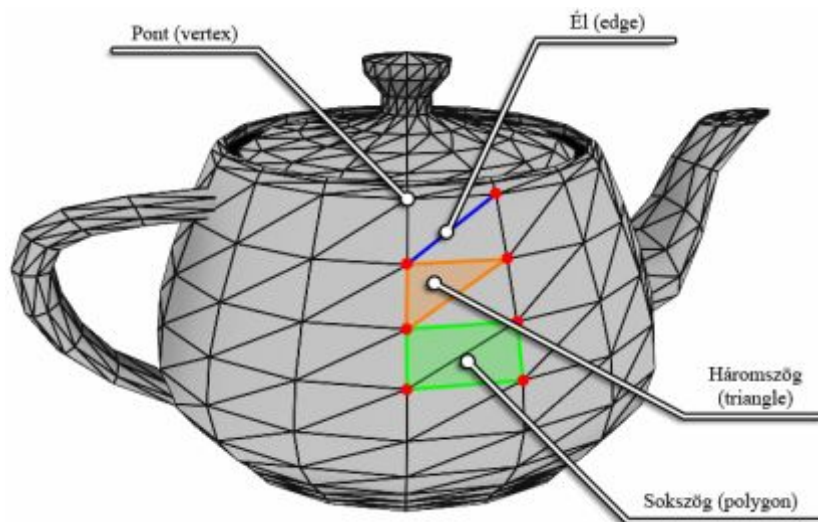
2. Valós tárgyak 3D leképezési lehetőségei

2.1 A 3D grafika alapjai

A számítógépes grafika segítségével egy virtuális világot hozhatunk létre, amely létező vagy nem létező tárgyak modelljeit tartalmazza. A világ leírását modellezésnek nevezzük, ennek során egy virtuális világot írunk le, ami tartalmazza az objektumok geometriáját (alak, kiterjedés, pozíció, orientáció) és megjelenítési attribútumait (szín, optikai tulajdonságok). Az objektumokon kívül még fényforrásokat és kamerákat is elhelyezhetünk ebben a virtuális világban, amik segítségével kétdimenziós képekké képezhetjük le, amit a kamera lát. A számítógép ezt a világot, és annak összes objektumát számok formájában tárolja. A geometria számokkal történő leírásához egy úgynevezett világ-koordinátarendszert használunk, amiben az alakzatok pontjainak koordinátáit adjuk meg, mivel az objektumok óriási pontfelhőkből épülnek fel. Ezeket a pontokat *vertex*-eknek nevezzük a háromdimenziós grafikában, azonosításukra, hogy melyik objektumhoz tartoznak, többféle módszert is alkalmazhatunk. Az egyik ilyen módszer, amikor azon pontokat tekintjük egy-egy alakzat részeként, amelyek x , y , z Descartes-koordinátái egy-egy adott egyenletet kielégítenek. De lehet a pontokat egyénenként is azonosítani, ez annyit tesz, hogy az adott tárgy minden pontját hozzárendeljük az objektumhoz. A tárgyak, a fényforrások és a kamera tulajdonságai az időben nem feltétlenül állandóak, ezekhez a változó tulajdonságokhoz egy-egy időfüggvényt rendelhetünk, ezáltal jön létre az animáció. Ahhoz, hogy a virtuális kamera által látott képet a felhasználó is lássa, egy képszámítási folyamatra, az úgynevezett *renderelésre* van szükség, ez az a folyamat, amikor a háromdimenziós virtuális világot kétdimenziós képpé vagy képek sorozatává, videóvá képezi le a számítógép. [2]

2.2 A 3D grafika alapfogalmai

A számítógépes 3D grafikában az objektumokat úgy kell elképzelni mintha a tárgy felületére egy térbeli rácshálót feszítenénk ki, amik sokszögekből épülnek fel. A rácsháló legkisebb eleme a pont, amit szakszóval *vertex*-nek neveznek. Minden tárgy felülete ilyen pontokból épül fel, két *vertex*-et egy él, vagy más szóval *edge* köt össze, ezek metszéspontja a *vertex*. Három összekötött *vertex* alkot egy háromszöget (*triangle*, *face*), négy vagy annál több összekötött pont pedig egy sokszöget (*polygon*), mint ahogy az általunk 3D Studio Max-ban renderelt 2.1-es ábrán látható. A tárgyak akár több különböző al-objektumokból, elemekből is felépülhetnek, amiket külön *vertex* csoportoknak is tekinthetünk.



2.1 ábra. Háromdimenziós objektumok alkotóelemei

Modellezésnek nevezzük azt a folyamatot, amikor létrehozunk egy objektumot a virtuális térben, és módosítjuk, valamint különböző transzformációkat alkalmazunk a *vertex*-eken és egyéb alkotóelemeken. Magát az objektumot modellnek hívjuk, ez rendelkezhet további tulajdonságokkal is, mint például:

- szín, mintázat (*textúra*)
- anyag (*material*)

Mint ahogyan a valóságban is, a számítógépes grafikában is minden modell rendelkezik valamilyen anyagi tulajdonsággal. Lehet egy modell színes, visszaverheti a fényt, lehet matt, vagy akár átlátszó is. Tehát mindazon tulajdonságokkal rendelkezik, mint amivel egy valódi tárgy is, csak ezeket a tulajdonságokat számokkal reprezentálhatjuk. Az anyag mellett *textúrát* is használhatunk a modell anyagának és mintájának az elkészítéséhez, ez egy kétdimenziós kép a modell térbeli rácshálójára kifesztve. Mielőtt azonban bármilyen *textúrát* használnánk is egy modellen, szükség van a rácsháló úgynevezett kiterítésére. Ezt a folyamatot úgy lehet a legegyszerűbben elképzelni, hogy a modell rácshálóját valamelyik oldalon függőlegesen szétvágjuk egy él mentén, és egy síkra kifesztjük. Ezáltal a háromdimenziós képet két dimenzióba vetítjük. Ezt a folyamatot nevezzük *UV-mapping*-nek.



2.2 ábra. UV map textúrával

A modelleken kívül sok más dolog is elhelyezhető a virtuális világban, mint például:

- fényforrások
- kamerák
- részecske rendszerek (füst, víz, tűz szimuláció)

Fényforrásra és kamerára minden esetben szükség van, ha a háromdimenziós teret le akarjuk képezni valamilyen kétdimenziós képpé. Ezek az elemek szintén rendelkeznek tulajdonságokkal, például egy fényforrásnak van intenzitása, színe, lehet irányított vagy diffúz, és az általa keltett árnyékok tulajdonságai paraméterezhetők. Minden olyan tulajdonság szabályozható egy virtuális kameránál is, amit egy valóságos eszköz tud: fókusztávolság, képarány, felbontás, mélységélesség, szűrők és még ennél is több. Ennek a virtuális térnek a képpé vagy videóvá való leképezését nevezik renderelésnek (*rendering*). A számítógép a folyamat során a kamera által látott képet különböző matematikai algoritmusok alkalmazásával leképezi egy kétdimenziós képpé, figyelembe véve az objektumok, fények, és a kamera beállításait. Ezek az algoritmusok teljesen paraméterezhetők, és a megfelelő beállításokkal akár teljesen élethű, fotórealisztikus lehet a végeredmény.

2.3 OBJ fájl formátum

A legegyszerűbb szöveges ASCII fájlformátum, amely háromdimenziós objektumok számítógépen történő adatainak tárolására szolgál. A mai 3D szoftverek mindegyike támogatja ezt a formátumot. Továbbá tartozik hozzá egy *.mtl* kiterjesztésű fájl is, melyben anyagi tulajdonságok definiálására van lehetőség.[8] [9]

Adattípusok:

- *vertex* (jelölése: *v*): egy pontot jelöl a háromdimenziós térben, melynek három koordinátája van: *x*, *y*, *z*. Ezek tárolása sorfolytonosan történik, melyekre a további típusok sorszáma szerint hivatkoznak.

pl: *v -5.0 5.0 0.0*

- *vertex textúra* (jelölése: *vt*): minden *vertex*-hez tartozik egy *vertex textúra* pont is, melyek szintén sorfolytonosan vannak tárolva, erre csak akkor van szükség ha az objektumhoz szeretnénk textúra fájlt használni. Segítségével a térbeli pontokhoz rendelhetjük hozzá, hogy egy kétdimenziós kép mely része legyen látható a felületen. Két paramétere van *x*, *y*, melyek nulla és egy közötti értékeket vehetnek fel.

pl: *vt 0.342 0.674*

- *face* (jelölése: *f*): három vagy több él által határolt sík felület, amelynek paraméterei a csúcspontok (*vertex*), valamint a *vertex*-ekhez tartozó *vertex textúra* pontok, ezekre sorszáma alapján hivatkozhatunk. A *vertex* és a hozzá tartozó *vertex textúra* pontok sorszámaikat per jellel kell elválasztani.

pl: *f 1/1 2/2 15/15*

Az első, a második és a tizenötödik *vertex* adja a háromszög csúcspontjait.

- *material library* (jelölése: *mtllib*): amennyiben a modellnek a geometriáján kívül a színét, mintázatát és anyagi tulajdonságait is letároljuk, ezeket egy külön fájlban kell megadni. Ezzel a kulcsszóval lehet hozzárendelni az *.obj* fájlhoz egy *.mtl* (*material*) kiterjesztésű fájlt, amely az objektum további megjelenítéssel kapcsolatos paramétereit tartalmazza. Mivel egy *obj* fájl több al-objektum tárolására is képes, ezért akár több anyagi tulajdonságot is lehet definiálni, amik különböző elemeken alkalmazhatóak.

pl: *mtllib scanned_object.mtl*

- *geometry name* (jelölése: *g*): egy *obj* fájl tartalmazhat több al-objektumot is, amiket azonosítani kell valamilyen módon, ezek megadására szolgál ez a kulcsszó. Minden objektumnak egyedi névvel kell rendelkeznie, amely tartalmazhatja az angol abc kis és nagy betűit, valamint számokat.

pl: *g scanned_object*

- *use material* (jelölése: *usemtl*): segítségével az objektumhoz, vagy annak al-objektumaihoz tudjuk hozzárendelni az *mtl* fájlban definiált anyagokat.

pl: *usemtl material01*

2.4 Szkennelés 3D-ben

A 3D szkennер egy olyan eszköz, mellyel különböző tárgyak vagy környezeti elemek alakjáról és lehetséges megjelenéséről (pl: szín, mintázat) gyűjthetünk adatokat. Ezeket az adatokat feldolgozva készíthetünk digitális háromdimenziós objektumokat. A 3D-s adatgyűjtés rendkívül hasznos és széles körben felhasználható. Főként a szórakoztatóiparban, a videojáték iparban, a műszaki tervezésben, az orvostudományban, az ipari termék- és formatervezésben, prototípustervezésben és gyártásban, de ezeken kívül természetesen még számos más területen használnak ilyen eszközöket.

Egy ilyen eszköznél a cél alapvetően egy olyan pontfelhalmaz felépítése, amely egy valós objektum felszínének geometriai mintavételezésével, majd ezen pontok felhasználásával egy digitális objektumot reprezentál. Amennyiben az eszköz képes a geometriai mintavételeken túl szín információt is gyűjteni, akkor ezt hozzárendelhetjük az elkészített digitális objektumhoz, így nem csak a tárgy felülete, hanem annak színe és mintázata is digitalizálásra kerül, ez utóbbit nevezzük textúrának. [1]

2.4.1 A 3D szkennelés technológiai megvalósításai

Számos technológia létezik arra, hogy valós térbeli objektumokat digitalizáljunk, de természetesen mindegyiknek vannak előnyei és hátrányai. Általában két fő csoportra bontják őket: a *kontakt*, és a *nonkontakt* típusú 3D szkennerekre. Utóbbi még további két részre tagolódik, az *aktív* és *passzív* szkennerekre. [1]

2.4.2 Kontakt típusú szkennerek

Ezek a szkennerek ténylegesen, fizikailag tapogatják le a tárgyat egy érintés érzékeny eszköz segítségével, amely egy robot karra van szerelve, de vannak kézzel vezérelhető típusok is. Legnagyobb hátránya, hogy a szkennelési folyamat során a tárgy elmozdulhat, sérülhet, vagy torzulhat is. Emellett a mintavételezés gyakorisága sem túl gyors, csupán pár száz Hz, ezzel szemben a *nonkontakt* eszközök akár 10-500kHz-el is működhetnek. Ezeket a kontakttípusú szkennereket CMM, azaz *Coordinate Measuring Machine*-nek nevezik. Főként a gyártástechnológiában alkalmazzák pontossága miatt. [1]

2.4.3 Nonkontakt típusú szkennerek

Ezek a szkennerek egy mesterséges fényforrás, vagy sugárzás (pl: fény, lézerfény, röntgensugár, elektronnyaláb, mágneses rezonancia) segítségével tapogatták le az objektumot és egy kamera gyűjti az adatokat. Az ilyen szkennerek célja, hogy egy úgynevezett pontfelhőt generáljanak a felületről készített geometriai mintavételek alapján. Ez a pontfelhő nagymennyiségű adattal írható le, mert minden egyes ponthoz három koordináta tartozik (x , y , z) melyek az adott pont térbeli elhelyezkedését írják le. Néhány tízezer vagy akár több millió pont is digitalizálásra kerülhet egy ilyen szkennelés során, de a pontfelhő részletessége a mintavételezéstől, az eszköz pontosságától, és a kamerától függ. Ezen pontok feldolgozásával, és optimalizálásával generálhatunk egy digitális háromdimenziós objektumot. A legtöbb szituációban egyetlen szkennelés nem elég ahhoz, hogy egy teljes modellt generáljunk. A legjobb példa erre a lézer szkennerek. Az objektumot függőlegesen egy vonal lézer világítja meg, ami 1-2mm vastagságú, tehát egy időpillanatban egy vonalnyi adatot képes gyűjteni az ilyen típusú szkennerek. Viszont egy ilyen vonal rengeteg pontból épül fel, emellett a kamera a 30, vagy 25 kép/másodperces mintavételezési sebességével az egyik leggyorsabb technológiának számít. Természetesen vannak olyan eszközök, amelyek több kamerával vannak felszerelve, ezzel csökkentve a szkennelés idejét, és növelve annak pontosságát és részletességét. Ezek a szkennerek több modellt generálnak és ezekből egyesítenek egy véglegeset. [1]

2.4.3.1 Aktív nonkontakt típusú szkennerek

A nonkontakt szkennerek közé tartoznak a *Time of Flight* technológián alapuló, a Háromszögelés elvét, valamint a Strukturált fényt használó megoldások.

Time of Flight technológia (Repülési idő módszer)

A *Time of Flight* technológiával úgy határozható meg a felületen az egyes pontok távolsága, hogy az eszköz méri a lézersugár kivetítése és ezen visszavert fény visszaérkezése között eltelt időt. A fény sebességének és az előbb leírt idő alapján egyszerűen kiszámítható az adott pont távolsága:

$$d = \frac{c \cdot t}{2}$$

ahol c a fény sebessége, t pedig a kivetítés és a visszaérkezés közötti eltelt idő. Az ilyen eszközök annál pontosabbak minél pontosabban képesek mérni az időt (kb. 3,3 pikoszekundum szükséges a fénynek, hogy megtegyen 1mm utat). Másodpercenként akár 10,000-100,000 mintavételre is képes, de ez a sebesség is csak a lézert vagy a tükröket

mozgató mechanikus alkatrészek lassúsága okozza. Nagy előnye, hogy óriási távolságokról (több kilométer akár) is képes viszonylag jó mérési adatokkal szolgálni. Az ilyen szkennereket általában épületek szkenneléséhez és földrajzi mérésekhez használják, például a LIDAR (*Light Detection And Ranging*) eszközök. [1]

A *Time of Flight* technológia hátránya, hogy pontatlan, ez a fény nagy sebessége és az idő mérésének a nehézségből adódik, ezért csak néhány milliméteres pontosságra képes, emellett egy ilyen eszköz igen drága.

Háromszögelés, trianguláció

A háromszögelés elvén működő 3D lézer szkennerek is az aktív eszközök csoportjába tartoznak, ezek az eszközök pont vagy vonallézert használnak a felület letapogatására. A *Time of Flight* módszerrel ellentétben itt szükség van egy külső kamerára, ami érzékeli, hogy hol van éppen a lézervonal. Attól függően, hogy a szkennelendő felületen hol van éppen a lézervonal vagy pont, ez mindig más és más helyen fog elhelyezkedni a kamera képén. Ezt a technikát azért nevezzük háromszögelésnek, mert a lézervonal forrása, a kamera és a lézervonal vagy pont egy háromszöget alkot. Az egyik távolság, amit tudunk az a háromszög egyik oldala, ami a kamera szenzorának közepe és a lézervonal közötti távolság, a másik az általuk bezárt szög. Ezen adatok ismeretében a kamera képén látható lézervonal vagy pont helye megállapítható. A legtöbb esetben inkább vonallézert használnak pontlézer helyet, ami a felgyorsítja a mintavétel gyűjtés sebességét, mivel egy időpillanatban nem csak egy pont helyéről szerezhetünk információt, hanem a felületen végigfutó lézervonal összes pontjáról.

Ha ezt a technikát ötvözzük egy olyan eszközzel, ami a kamerát és a lézervonalforrást együtt, oldalirányban mozgatja a felülettel párhuzamosan, akkor a mintavételek sorozatából rekonstruálható az egész felület. Ezen kívül létezik olyan technika is amikor a szkennelendő tárgyat egy hardver segítségével körbeforgatják a függőleges tengelye mentén és az így gyűjtött adatok alapján rekonstruálható az objektum. A háromszögelés elvén működő lézer szkennerek nagy előnye, hogy megfelelő eszközökkel és jó környezeti körülmények mellett igen pontos méréseket tesznek lehetővé. Az ilyen eszközök akár mikrométeres pontosságra is képesek. Emellett viszonylag olcsó, könnyen beszerezhető alkatrészekből akár még otthon is megépíthető. Viszont csak korlátozott távolságban képes a pontos mérésekre, tehát távoli, nagy objektumokról nem képes adatokat gyűjteni. Ez a lézervonal romlása, és szórása, a nem megfelelő környezeti fények valamint a kamera és a tárgy közötti távolságból adódó képi minőségromlás miatt van. [1] [4]

Strukturált fény

A háromszögelés módszerhez hasonlóan ez a technika is valamilyen fényforrással és egy külső kamera segítségével rekonstruálja az tárgyakat, de itt nem egy pontot vagy egy lézervonalat vetítünk a vizsgálandó felületre, hanem valamilyen egy vagy kétdimenziós alakzatot, mintázatot. Egydimenziós mintaként például egy vonalat, kétdimenziósként pedig egy rácshálót, vagy több párhuzamos vonalat, csíkozást használnak. Ezt vetítik a digitalizálni kívánt felületre, és a kamera által látott képen megjelenő mintázatok torzulásából határozható meg a vizsgált felszín geometriája. A minták vetítéséhez általában projektort használnak. Ennek a technológiának az előnye, hogy a kamera által látott és a strukturált fénnel megvilágított egész felület egyszerre digitalizálható, tehát ennél a módszernél a leggyorsabb a mintavételezési sebesség és itt tapasztalható a legkisebb mértékű torzulás, ami a mozgásból adódik. Ez a technika időben változó geometriájú felületek, alakzatok, vagy akár mozgó tárgyak valós idejű rekonstrukálására is lehetőséget ad. [1]

2.4.3.2 Passzív nonkontakt típusú szkennerek

A passzív nonkontakt szkennerek nem rendelkeznek olyan eszközzel, ami fényt vagy egyéb sugárzást bocsátana ki a szkennelendő felületre, hanem az objektumokról visszaverődő környezeti fények alapján gyűjt információkat a szkenneléshez. Nagy részük olyan gyűjtőlencsével van felszerelve, ami a látható fényt érzékeli, de van olyan is, ami a hosszabb hullámhosszú fényeket érzékeli, mint például az infravörös fény. Legelterjedtebb fajtái a sztereoszkópikus, és a fotometrikus technológiák, valamint a sziluett és a kép alapján modellező rendszerek. [1]

Sztereoszkópikus rendszerek

Ennél a technikánál két kamerára van szükség, melyek egymástól kis távolságra helyezkednek el és azonos pontra tekintenek. Elemzésnél a két kamera által látott kép közötti kis eltérésből lehetséges az egyes pontok közötti távolságok meghatározása. Ugyan ezen az elven működik az emberi látás is.

Fotometrikus rendszerek

Ez a rendszer csak egy kamerát használ, és különböző fényviszonyok mellett gyűjt képeket, melyeket pixelenként vizsgálva képes a felület rekonstrukciójára.

- **Sziluett technika**

A sziluett technikát alkalmazó szkennerek alapvető feltétele egy olyan háttér biztosítása a szkennelendő objektum mögött, ami kiemeli a tárgy körvonalát, sziluettjét. Az ilyen módszerrel készített képekből, a tárgy körvonala alapján építi fel a virtuális objektumot a program.

- **Kép alapján modellező rendszerek**

Ezen technika a felhasználó beavatkozását igényli, ugyanis több különböző kép felhasználásával kell manuálisan referenciapontokat beállítani, ami alapján egy igen pontatlan modell rekonstruálható. Egy már meglévő modell részeit változtatja a képeken felvett referenciapontok alapján, vagy valamilyen nagyon egyszerű, alacsony részletességű modellt készít. A képekből generált *textúra* segítségével javít a modell részletességén és alakhűségén, és nagy előnye, hogy csak egy egyszerű kamera is elég hozzá. Például egy arc szkennelésénél, egy már meglévő, optimális rácshálózattal rendelkező modellen végez változtatásokat, a beállított referenciapontok alapján, mint például szem szélei, száj széle, vagy pupillák.

3. Saját fejlesztésű 3D szkennert szoftver és hardver

Ebben a fejezetben bemutatjuk az általunk készített szkennert szoftver, és hardver komponenseit, valamint részletezzük a fejlesztés közben szerzett tapasztalatokat, hibákat, és azok megoldásait. A munka során egy olyan aktív nonkontakt típusú rendszert állítottunk össze, mely egy forgóasztalra helyezett és körbeforgatott tárgyról készít mérethelyes háromdimenziós digitális objektumot, amit a mérés végeztével tetszőleges textúrával is elláthatunk.

3.1 A szkennert szoftver főbb moduljainak feladatai

- Képszekvenciák felvétele és tárolása
- Képek feldolgozása, lézervonal középpontjainak megkeresése
- Adathalmaz létrehozása a feldolgozott képek alapján
- Adatok optimalizálása
- Térbeli pontok koordinátáinak számítása
- Pontok összekötése
- Textúra összeállítás
- Textúra koordinátáinak számítása
- Az eredmények (feldolgozatlan és végleges) fájlokba mentése

3.2 A szkennert hardvere

A munkánkhoz szükséges hardver elemek egy része kereskedelmi forgalomban is kapható „kommersz” számítástechnikai eszközként könnyen megszerezhető volt, azonban az elképzelt forgótálca elkészítéséhez a szükséges műhelymunkák áthághatatlan akadályt képeztek számunkra. Így eredeti elképzeléseinket átértékelve, szintén meglevő eszközök átalakításával mi magunk készítettük el azt.

3.2.1 Forgótálca

Szkennelési módszerünk feltételezi a tárgy teljes körbeforgatását, ami alatt a vonallézer megvilágítja a tárgyat, és eközben a szoftver rögzíti a képszekvenciát.

Az alapvető feltételek, melyeket ki kell elégítenie:

- A szkennelendő tárgyak befogadása (geometriai méret, tömeg)
- Az adatok valós időben történő rögzítéshez szükséges állandó szögsebességgel történő körbeforgatása.

Forgótálcás megoldásunk a 3.1 ábrán látható. Mechanikai alapját egy erre a célra átalakított lemezjátszó szolgáltatja. A tányér 244 mm átmérőjű és csapágyazása alkalmassá teszi viszonylag nagyméretű tárgyak elhelyezését is. Egy körbeforduláshoz 136 másodperc szükséges. A forgatás állandó szögsebességét egy szinkron motor segítségével oldottuk meg. Ez a megoldás nem biztosít szinkronitást a feldolgozó szoftverrel, így a pozíció-azonosítást manuális módszerrel oldottuk meg.



3.1 ábra. Szkennerünk forgó tárgyasztala két külön nézetből.

Terveink közt szerepel az eszköz továbbfejlesztése úgy, hogy a szoftver vezérelje a motort, a vonallézert, és a megvilágítás fényerejét. Erre a témára az 5.1 fejezetben, a továbbfejlesztési lehetőségek leírásánál térünk ki.

3.2.2 Vonallézer

A tárgy függőleges megvilágításhoz használt fókuszálható vonallézert a 3.2 ábra mutatja. Ezt az internetről olcsón betudtunk szerezni. Az általunk használt Class 3b típusú, vörös vonallézer 5mW-os, és ~650nm-es hullámhosszú fényt bocsát ki. Az általa vetített lézervonal szélessége a tárgy anyagától, a fényviszonyoktól és a fókuszlencse beállításától függően 3-15mm közötti. Annak érdekében, hogy minél kompaktabb legyen a hardver, ezért a lézer az 5 Voltos tápfeszültséget a vezérlő számítógép USB portján kapja.



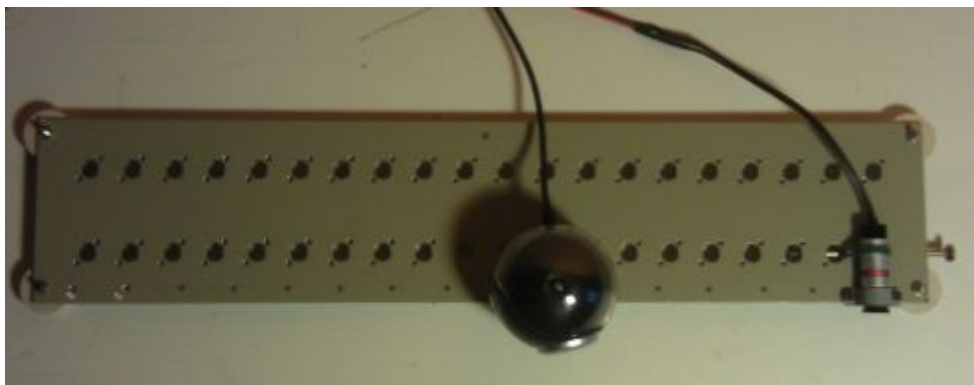
3.2 ábra. A hardver megépítéséhez használt 5mW-os vonallézer.

3.2.3 Kamera

Fejlesztés során többféle kamerával is próbálkoztunk. A webkamerák többsége két tengely mentén mozgatható, így nehéz beállítani, hogy az optikai tengelye pontosan merőleges legyen a tálca forgástengelyére. Jelenleg is egy webkamerát használunk, melyet a 3.3 ábra mutat. Ennek kialakítása olyan, hogy két oldalán csavarral rögzítve volt egy "U" alakú műanyaghoz, amivel a két tengely menti mozgathatóságot oldották meg. Ezt a műanyagot eltávolítottuk, és az oldalán lévő csavarral egy fémlapra rögzítettük. A kamera így 90°-al elfordított képet ad, ezt a szögeltérést szoftveresen korrigáltuk. Ugyanezen a fémlapon van elhelyezve a vonallézer is, amit úgy rögzítettünk, hogy függőleges vonalat vetítsen. Ezen a fémlapon tizennégy darab furatlyukat alakítottunk ki, így a kamera és a lézer által bezárt szög módosítható.



3.3 ábra. A képsorozatok rögzítéséhez használt kamerák.
(Genius Look 1320 V2, és Panasonic NV-GS60)



3.4 ábra. Fémlapra szerelt kamera és vonallézer

Egyik legfőbb problémánk az volt, hogy a LabView gyakran egy ismeretlen hiba miatt lefagy, ha webkameráról akarunk képet rögzíteni. Ez sajnos egy elég gyakori hibajelenség, interneten is nagyon sokan panaszkodtak, hogy ugyanezt tapasztalták, de működőképes megoldást nem találtunk erre a problémára. Végül egy rendszer újratelepítés megoldotta ezt is, tehát tovább tudtunk haladni a fejlesztéssel.

A legutóbbi szkenneléskor egy Panasonic NV-GS60 típusú kézi kamerát használtunk, melyet IEEE 1394 - FireWire kábellel csatlakoztattunk a számítógéphez. A DV kamera felbontása 720x576 pixel. Annak ellenére, hogy a kamera felbontása jóval alacsonyabb, mint az alkalmazott webkameráé, mégis sokkal jobb minőségű képeket készített, és így a modellek felülete és a *textúrája* egyaránt jobb lett. Fontos még odafigyelni arra, hogy az automatikus fókuszálás ki legyen kapcsolva, mert kis megvilágításnál a kamerák elkezdenek fókuszálni, ezzel tönkretéve a képszekvencia egyes részeit.

3.3 A hardver és a szoftver használata

Az alábbiakban a hardver és a szoftver használatával kapcsolatos legfontosabb gyakorlati tapasztalatainkat ismertetjük, melyekben kitérünk a tárgy szkennelhetőségére, függően annak anyagától és formájától. Továbbá szót ejtünk a megfelelő fényviszonyok kialakításáról, valamint a hardver és a szoftver megfelelő beállításairól.

3.4 A tárgy anyaga

A szkennelés megkezdése előtt fontos meggyőződni arról, hogy az adott tárgy teljes felületén jól látható a lézerfény. Legjobb eredményt a matt felszínű tárgyakkal lehet elérni, mint például fa, agyag vagy műanyagszobrok. Azonban nem elég a szabad szemmel való meggyőződésre hagyatkozni, mert előfordulhat, hogy a kamera nem úgy látja a tárgyra vetített

lézer fényt, mint az emberi szem. Erre egyik példa a 3.5 ábrán látható festetlen viasz, melynél az általunk használt kamerával, a lézerrel megvilágított viasz a kamera képén erősen szóródott, ezzel lehetetlenné téve annak középpontjának megállapítását. Ezek a tényezők más fényviszonyokkal, és nagyobb felbontású kamerával egyes esetekben kiküszöbölhetőek.



3.5 ábra. A viasz festetlen felületein a fény teljesen elnyelődik.

3.5 A tárgy felülete

A ma használt legmodernebb 3D szkennerek több tengely mentén vesznek mintákat, és az úgy gyűjtött adatok alapján tudják háromdimenziós mintaillesztő algoritmusokkal összeállítani az elérhető legjobb minőségű modellt.

Az általunk használt módszer azonban csak egy tengely mentén készít mintavételezést, ezért nem alkalmas minden tárgy digitalizálására. Az egy tengely mentén történő mintavételezés hátránya, hogy bonyolultabb formák szkennelésénél a generált objektum nem teljesen fog hasonlítani az eredeti tárgyra, például, ha egy olyan szobrot szkennelünk aminek kettő vagy négy lába van, akkor azokat nem lehet elkülöníteni, és azok a középpontban "összekötve" fognak megjelenni a modellen. Egy másik példa erre a hibajelenségre, ha például emberi fej szkennelésekor a fülek mögötti rész "összenőve" jelennek meg a modellen.

3.6 Tárgy pozicionálása

A képsorozatok készítése előtt pontosan el kell helyezni a tárgyat a forgó tálcán. Erre a legjobb módszer megkeresni a tárgy alján a középpontját, ha szükséges meg is lehet jelölni azt, majd ennek megfelelően elhelyezni a forgótálcán. Ha a tárgy rosszul van elhelyezve, akkor egyik oldala összenyomott, másik oldalán pedig elnyújtott lesz a modell. Kivétel lehet ez alól, ha a tárgynak van egy vagy több olyan része, melyek jóval kintebb vannak, mint a tárgy többi pontja.

3.7 Fényviszonyok

A képek begyűjtése előtt meg kell győződni arról hogy a fények megfelelőek-e az adott tárgy megvilágításához. Ehhez mi mesterséges fényeket használtunk. A megvilágítás mellett gondoskodni kell arról is hogy minél sötétebb legyen a háttér, hogy jobban kiemelje a szkennelt tárgyat.

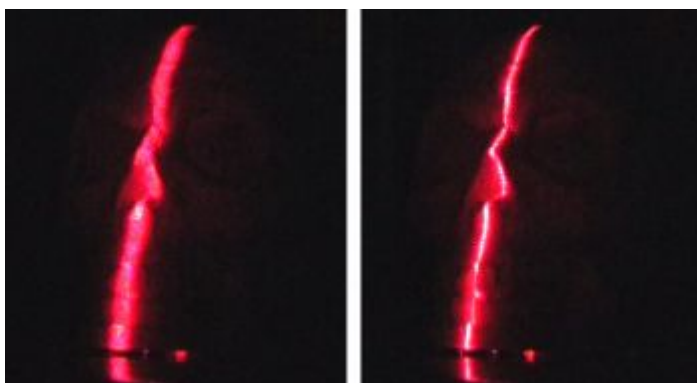
3.7.1 Vonallézer beállítása

A vonallézer precíz beállítása kézzel elég nehéz feladat, mert egyaránt fontos az, hogy a vetített fény a tárgy forgástengelyével párhuzamos (függőleges) legyen, és az is, hogy olyan távolságban ahova a szkennelendő tárgy kerül, oda jól fókuszált vékony lézerfény kerüljön. Ennek megkönnyítésére egy olyan eszközt készítettünk, melyre felszerelhető a vonallézer, és két ponton rögzíthető rajta.

A kamerától, tárgytól és a fényviszonyoktól függően két hibalehetőség fordulhat elő:

- Lehet, hogy túl világos a lézerfény, és emiatt az nem kellően vékony, túlexponált, így annak középpontjának megkeresése nem lesz optimális. Ez a jelenség kiküszöbölhető, ha a háttérvilágítást növeljük.
- Másik hibalehetőség, ha a lézerfény annyira gyenge, hogy nem jelenik meg a kamera képén. Ebben az esetben szintén megoldást jelent a háttérvilágítás növelése.

A vonallézert úgy kell fókuszálni a szkennelés megkezdése előtt, hogy a lehető legvékonyabb legyen a fénye a tárgy felületén. Az általunk használt vonallézer fókuszálhatósága a tárgyakon átlagosan, körülbelül 4-5 mm-es vonalvastagságot eredményezett. Legutóbbi méréseink során egy 3 dioptriás segédlelencsét használtunk, amivel sikerült 2 mm alá csökkenteni a lézervonal vastagságát. Ahogy a 3.6 ábrán látható az így készült képek is szembetűnően jobbak lettek, és ezáltal a generált modell felülete is.



3.6 ábra. Műanyag tárgy vonallézerrel megvilágítva, és optikai lencsével fókuszálva.

3.7.2 Fények beállítása a textúra készítéshez

A textúra készítéséhez elég a tárgyat egy oldalról megvilágítani. A fényforrást a kamerával egy vonalban érdemes elhelyezni, mert csak annak a résznek kell jól megvilágítva lenni, amelyre előzőleg a vonallézer fénye került. Fehér fényt kell használni, hogy a tárgy eredeti színei ne módosuljanak. Erre alkalmas lehet egy nagyobb teljesítményű (150W) halogén lámpa, ami nem közvetlenül a tárgyra világít, hanem szórt fényét használjuk. Fontos, hogy a színhű leképezéshez a kamera szűrője is a megvilágítás színhőmérsékletének megfelelően legyen beállítva. A pontos fényerő beállításához érdemes egyszer körbeforgatni a tárgyat és közben a kamera képén ellenőrizni hogy nem túl sötét, vagy világos-e.

3.8 Kalibrálás

Az térbeli pontok koordinátáinak kiszámításához elengedhetetlenül fontos a pontos távolság meghatározása. Különböző méretű, alakú tárgyak szkennelésénél gyakran előfordult, hogy változtattunk a kamera és forgótálca távolságán, valamint a vonallézer és a kamera által bezárt szögön is. Ilyen változtatásoknál a rendszert újra kell kalibrálni.

3.8.1 Saját kalibrálási és sugár meghatározási módszereink

Az első módszerünk a kalibrálásra nem a háromszögeléses módszeren alapult, hanem egy saját módszert alkalmaztunk. A kalibrálás abból állt, hogy minden szkennelés megkezdése előtt egy képet kellett készíteni úgy, hogy a forgótálca forgástengelyének középpontjában egy fehér papírlapot helyeztünk el függőlegesen, amit a vonallézer megvilágított. Ezen kép alapján a szoftverrel azt közöltük, hogy a készült kép x tengelyén hova kerül a lézervonal középpontja a kalibráláskor.

Az így bekerült adat segítségével az egyes pontokhoz tartozó sugár értéket számoltuk. A sugár érték alatt a forgástengely középpontjától való térbeli távolságot értjük. A módszerünk egyszerű volt, de működőképes. A feldolgozott képeken minden egyes lézervonal középpontjához tartozó érték és a kalibrálás során tárolt érték közti különbséget hatványoztuk, és egy konstanssal szoroztuk.

Ennek a kalibrálási módszernek a hátránya az, hogy a szoftverben ezeket a statikus adatokat változtatni kellett, ha a kamera és lézer által bezárt szög változott.

A legjobb eredményt akkor értük el, amikor a bezárt szög 20°-os volt, ekkor a következő képlet adta a legalakhűbb modellt:

$$R = \frac{(K - L)^{1.4}}{3}$$

ahol K a kalibrációval megadott érték, L a lézervonal középpontjához tartozó x tengely menti pixel sorszáma, R pedig az így számított sugár.

További távolság meghatározásra használt módszereinket a 4.7 fejezetben fejtjük ki bővebben.

3.8.2 Távolság meghatározás háromszögeléssel

A távolság meghatározást a háromszögelés (*trianguláció*) elvén alapuló módszerrel pontosabbá tudtuk tenni. Ehhez egy kartondobozból hajtott sablon formát használtunk, melyet a 3.7 ábra mutat. A szoftver automatikusan felismeri a kartonforma alján és a tetején látható lézervonalat, és a hozzájuk tartozó pixel értékeket letárolja, amit később használ fel a térbeli pontok koordinátáinak kiszámításához. Ennek a kalibráló eszköznek a méretei ismertek, és a szoftverben is statikus adatként szerepelnek milliméterben megadva. Ennek a jelentősége az, hogy a kamera és a vonallézer által bezárt szögtől és a forgótálca távolságától függően, más-más kép jelenik meg a kamera képén. Ezért fontos minden szkennelés előtt újra kalibrálni a rendszert. Ennek a módszernek nagy előnye, hogy nem fontos tudni a forgótálca és a kamera közti távolságot, sem a kamera és a lézer által bezárt szöget.



3.7 ábra. Kartonlappból készült kalibrációs segédeszköz

Az általunk használt kalibrálási módszerrel a következő adatokat tudatjuk a szoftverrel:

- A kamera és a forgótálca középpontjának távolsága
- A kalibráló doboz magassága
- A kalibráló doboz mélysége

Ezekből az adatokból a szoftver a pixel értékeket valós távolság adatokhoz tudja társítani, és ezáltal a képek feldolgozása során a pontfelhő *vertexeinek* pontos térbeli elhelyezkedését tudja számolni.

3.9 Képek készítése

A szoftver használatának első lépése a képek begyűjtése. A képek letárolása előtt lehetőség van arra, hogy előre megadjuk, hogy egy teljes körbefordulás alatt hány kép készüljön. Az általunk használt forgótálcát szinkronmotor mozgatja, és egy körbefordulásához 136 másodperc szükséges. Ez elég sok idő ahhoz, hogy akár több száz mintavételt készítsen a szoftver, melyeket ".bmp" formátumban tömörítetlenül tárol.

Letárolt képeink állóképek, melyeket PAL szabványú videokamerával készítünk. A 25 képkocka/másodperces videó egy képkockája 40ms időtartamot jelent. A forgóasztal folyamatos forgómozgást végez, így a képalkotás ideje alatt a tárgy kismértékben, de elfordul. A hiba a következőképpen számítható:

$$\alpha_h = \frac{360}{T} \cdot t$$

ahol a t az egy képkocka ideje (másodpercben), α_h a képkocka ideje alatti elfordulás szöge (fokban).

Esetünkben ez a hiba 0,105°-ra adódott, ami jelenleg elhanyagolhatónak tekintünk. A rendszer továbbfejlesztésénél azonban ez problémaként kezelendő, hiszen a szkennelés sebességének növelésével (a körbefordulás idejének csökkentésével) ez a hiba is nő.

A képsorozat előállítását és letárolását valamint a képek feldolgozását időben elkülönülten végeztük. Ennek praktikus okai voltak:

- A szoftver fejlesztésének bizonyos fázisai függetlenné válhattak a hardvertől
- Így lehetőségünk volt a szoftver fejlesztése közben többször is ugyanazokat képsorozatokat használnunk a teszteléshez, hibakereséshez (debugoláshoz).
- Amennyiben az első beállításokkal készített modell nem megfelelő, akkor bármikor újra lehet próbálni más beállításokkal, és nem kell újra mindent beállítani a szkenneléshez.

Felmerülhet az a kérdés is, hogy miért kell tömörítetlenül tárolni a képeket. Azért döntöttünk a .bmp fájlformátum mellett, mert nagyon fontos, hogy veszteségmentes jó minőségű képekkel dolgozzunk. A mai gyors merevlemez sebességnél ez nem jelent akadályt, még akkor sem, ha a képek több mint egy-két *gigabyte*-ot foglalnak el. Másrészt a képek rögzítése során esetleges hibalehetőség lenne, ha nem csak letárolnia kellene a szoftvernek a képeket, hanem még tömörítenie is azokat.

Nagyon sűrű mintavételezésnél kipróbáltuk *ramdrive*-ra tárolni a képeket, de az ilyen sűrű mintavételezés a legtöbb modellnél elhanyagolható minőség javuláshoz vezetett, viszont az objektum fájl mérete jelentősen megnőtt. Sűrű mintavételezés alatt azt értjük, ha több mint ezer képet készít a szoftver. Mivel a modell elkészítéséhez a legtöbb esetben elegendő 400 kép ezért a továbbiakban valószínűleg nem erre fogjuk használni. Azonban hasznos lehet a sűrű mintavételezés a textúra összeállításához, melyről az 5. fejezetben írunk.

A szoftverben a szkennelés megkezdése előtt négy dolgot kell beállítania a felhasználónak:

- A szkennelt tárgy nevét
- Egy mappát ahova a program letárolja a képeket
- Mintavételezés gyakoriságát
- Textúrázás engedélyezését vagy kikapcsolását

A szoftver használatához - praktikus kényelmi szolgáltatásként - előre ki lehet jelölni a képek tárolására szolgáló mappát, melyhez automatikusan létrejön két al-mappa. Ezek egyikébe a vonallézerrel megvilágított képek, a másikba pedig a textúra összeállításához szükséges képek kerülnek.

Ha például ez a mappa a "D:\SCAN\", akkor egy "Beethoven" nevű szkennelés, amin a textúrázás be van kapcsolva, és a mintavételek száma 300-ra van állítva az a következő mappákat fogja létrehozni:

- D:\SCAN\Beethoven\Laser\
- D:\SCAN\Beethoven\Texture\

A fenti mappák mindegyikében 300 darab bmp fájl lesz. A felsőben találhatóak a vonallézerrel megvilágítottak, az alsóban pedig szintén ugyanennyi fájl melyek jó fényviszonyok mellett készültek. Ez a modell fájl elkészítésénél is előnyös, mert nem kell külön két mappát beállítani, elég csak a szkennelés nevét megadni, és az alapján a szoftver már tudja az elérési utakat.

3.10 ROI (Range Of Interest)

A képek feldolgozása szempontjából előnyös dolog kijelölni a hasznos tartományt, így a modell fájl mérete kisebb, és az előállítás is kevesebb ideig tart. A hasznos tartományt a képfeldolgozásban idegen szóval *Range Of Interest*-nek hívjuk. A képek készítésénél is fontos hogy a kamera a lehető legközelebb legyen a tárgyhoz, mert így jobban ki tudjuk használni a kamera felbontását, ezáltal jobb minőségű modellt lehet készíteni. Azonban az esetek többségében nem használjuk fel a teljes tartományt, ezért van szükség a *ROI* használatára. A képek letárolásánál is előnyös beállítani a *ROI*-t mert így kevesebb helyet foglalnak a képek, másrészt pedig a modellfájl készítésének idejét rövidíti.

A kamerák többségének vízszintes felbontása nagyobb mint a függőleges tengely menti. A vonallézerrel történő szkenneléskor viszont főként a függőleges tartományt használjuk ki. Éppen ezért előnyös a hardver kialakítása amiben egy 4:3 képarányú webkamerát 90 fokkal elforgatva használunk, így a képarány 3:4-re módosul. A közeljövőben nagyobb felbontású *Full HD* kamerát szeretnénk szintén elforgatva használni, amivel a nagy felbontásnak és a szélesvásznú képarálynak köszönhetően még jobb minőségű modelleket tudnánk előállítani.

3.11 Méretarányosság

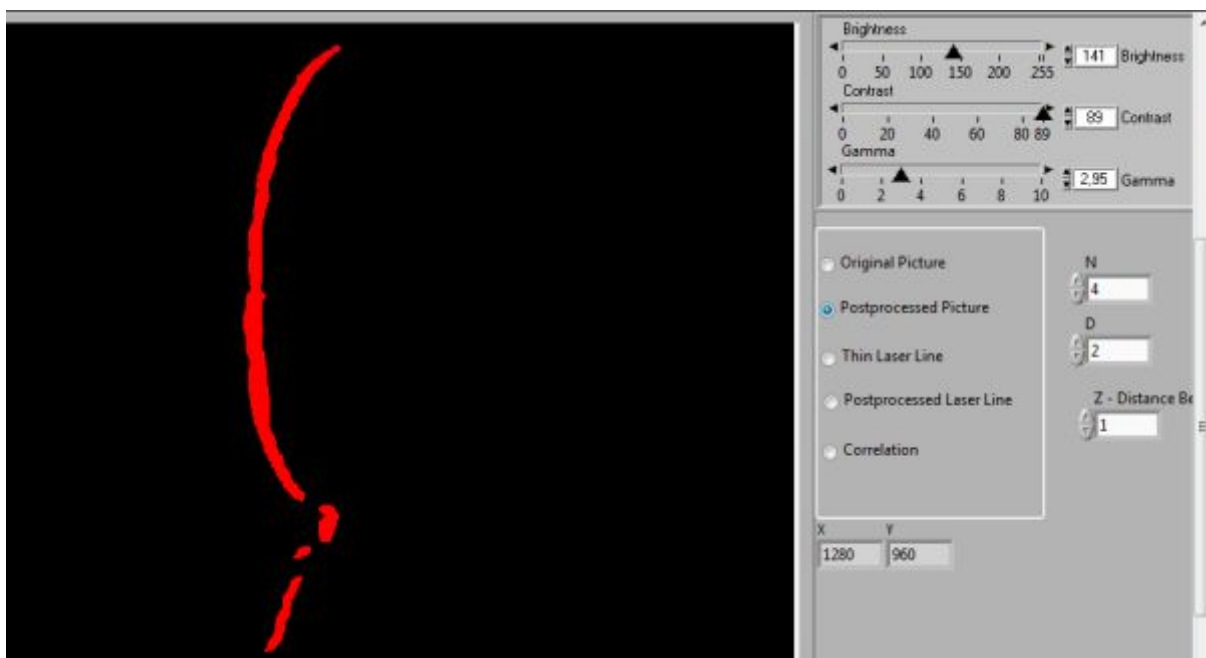
A jó minőségű alakhű modellek előállításán kívül nagyon fontos szerepe van még a méretarányosságnak is. Az objektumok méretarányosságára azért van szükség, hogy egy három dimenziós térben elhelyezve is arányos legyen a környezetében lévő dolgokhoz. Tehát például egy 3D szerkesztő programot használó animátornak, vagy modellezőnek, ne kelljen külön beállítani a méreteket, hanem egyből méretarányosan jelenjen meg a virtuális térben.

A modell exportálása előtt a felhasználónak lehetősége van arra hogy megadja a szkennelt tárgy valódi magasságát milliméterben. Azért döntöttünk úgy hogy a magasságot lehessen csak megadni, mert a modell úgy is méretarányos lesz, és a legtöbb esetben a tárgy magasságát a legegyszerűbb lemérni.

3.12 Optimális képfeldolgozási beállítások

3.12.1 Brightness, Contrast, Gamma Correction

A *Brightness, Contrast, Gamma* korrekció (továbbiakban *BCG*) az első része az elkészített képszekvencia feldolgozásának. A programnak ezen funkciója követeli meg a legnagyobb odafigyelést, mivel nagyon fontos a megfelelő értékek beállítása. Ezen a három csúszkán kell olyan optimális értékeket megadni, amikor szinte csak a lézervonal látszik, minden más fekete, vagy nagyon sötét. Mindhárom érték nagyban függ az adott szkennelés körülményeitől. Ide tartozik a szkennelendő tárgy anyaga, a lézervízszórása és tükröződése a felületen, a fényviszonyok, valamint a kamera megfelelő beállítása. Tehát szinte minden szkennelésnél be kell állítani ezeket az értékeket, hogy a megfelelő eredményt kapjuk. A 3.8 ábra a *BCG* értékek beállításra szolgáló képernyőt mutatja.



3.8 ábra. BCG értékek beállítása

Alapértelmezett értékek: Brightness: 128 Contrast: 45 Gamma: 1.0

Az első csúszkával állítható a kép világossága, az alapértelmezett beállítás 128, ami általában minden szkenneléshez az optimális érték. Amennyiben növeljük az értéket, az egész kép világosabb lesz, ha csökkentjük, akkor sötétebb, a világosság lineárisan változik. Ezt akkor érdemes használni, ha fényviszonyok miatt olyan képek készültek amiken a lézervonal inkább sötétebb piros. Megfelelő fények beállításával tehát ezt az értéket ritkán kell állítani, ha mégis szükséges, akkor a 80 - 150 közötti tartomány lesz megfelelő.

A második csúszkával a kontraszt állítható, ez az egyik legfontosabb része a *BCG* beállításának. Itt a fényerők közötti arány adható meg, tehát a legvilágosabb és a legsötétebb részek közötti érték. Az alapértelmezett érték a 45, ha növeljük az értéket, akkor a világos és sötét részek közötti különbség kevesebb lesz.

A kontrasztot minden esetben 45-től nagyobb értékre kell állítani, hiszen a programnak csak a lézervonalra van szüksége, tehát egy olyan értéket kell találni, amikor ez nagyon jól látszódik, viszont minden más sötét marad. Természetesen az esetek nagy részében ez még nem elég, hogy tökéletes végeredményt kapjunk, a világosabb részek a fenti beállításoktól általában beégnek és több helyen foltok, és kósza pixelek maradhatnak. A másik véglet pedig az, hogy a lézervonal elvékonyodik, eltűnik egy-két helyen, ezzel rontva a szkennelés minőségén, ezért kell még egy gamma-korrekciót is alkalmaznunk, amelyet az utolsó csúszkával tehetünk meg.

A harmadik csúszkával a gamma-korrekció erősségét adhatjuk meg, melynek lényege az, hogy a kis és nagy intenzitású pixelek világosságát kis mértékben, a közepes intenzitásúak világosságát pedig nagymértékben növeli meg. Így lehet a kontraszton és a világosságon javítani, ezáltal telítettebb lesz a lézervonal, tehát nem vékonyodik el, nem lesznek szakadások, és beégett részek sem a képen. Az alapértelmezett 1-es érték már egy egészen jó végeredményt hoz, amennyiben a képek megfelelően sötét fényviszonyok között készültek. Ugyanis ekkor a két felső csúszkán szinte alig kell változtatni, tehát sem beégés, sem pedig a megszakadt lézervonal effektus egyaránt nem jelentkezik. A gamma-korrekció erősségének a növelésével a közepes világosságú pixelek egyre sötétebbek lesznek, tehát a gyengén megvilágított háttér, és szkennelendő objektum képe eltűnik, míg a sötét részek, és az erős lézervonal változatlan marad. Az optimális érték minden esetben 1.0 fölött van. A gamma-korrekció a világosság szabályzással szemben egy nemlineáris transzformációként írható le. [5]

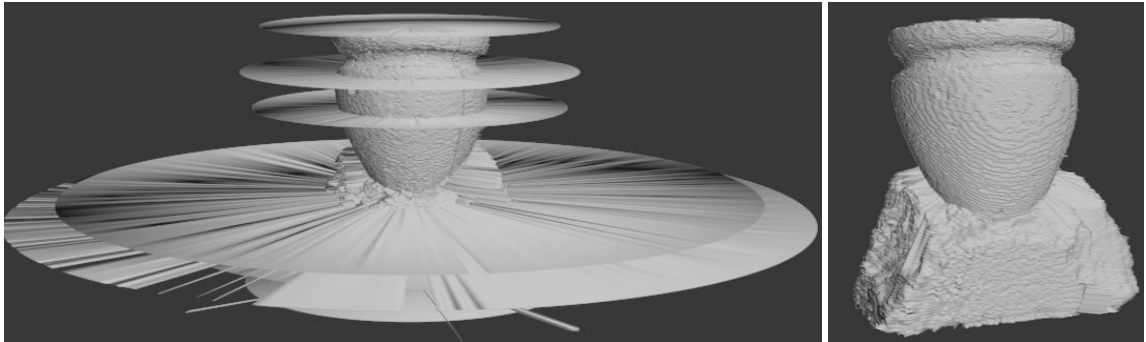
3.12.2 Kiugró élek eltávolítása

A képszekvencia akármennyire is készült jó fényviszonyok mellett, és akármilyen jók is a *BCG* beállítások hibajelenségek szinte mindig előfordulnak.

Kísérleteink során tapasztalt hibák:

- A képfeldolgozás ezen fázisában gyakran előfordul az a hibajelenség, hogy a lézervonal szoftver által meghatározott középpontja rossz helyre kerül, néha teljesen máshova, mint ahol az eredetileg lenne. Ennek több oka is lehet, például a szkennelt tárgy felületéről visszaverődik valahova máshova a fény, és az így szóródott fényt dolgozza fel a szoftver.

- A másik esetleges hiba, ha a kamerának *CCD* hibája van és néhol világosabb pixeleket rögzít. Ezeket a hibákat kiugró adatoknak nevezzük, mert követve a lézervonal alakját, egyértelműen látható azokon a pontokon, hogy nem odavaló hibás adatként kerültek be. A további feldolgozás előtt mindenképpen el kell tüntetni a kiugró éleket, mert amennyiben ezeket figyelmen kívül hagyjuk az úgy exportált modell fájl felülete nagyon rücskös, tüskés, lesz. Ha minden kép hibás, akkor nem odaillő korong formák jelennek meg a modellen.

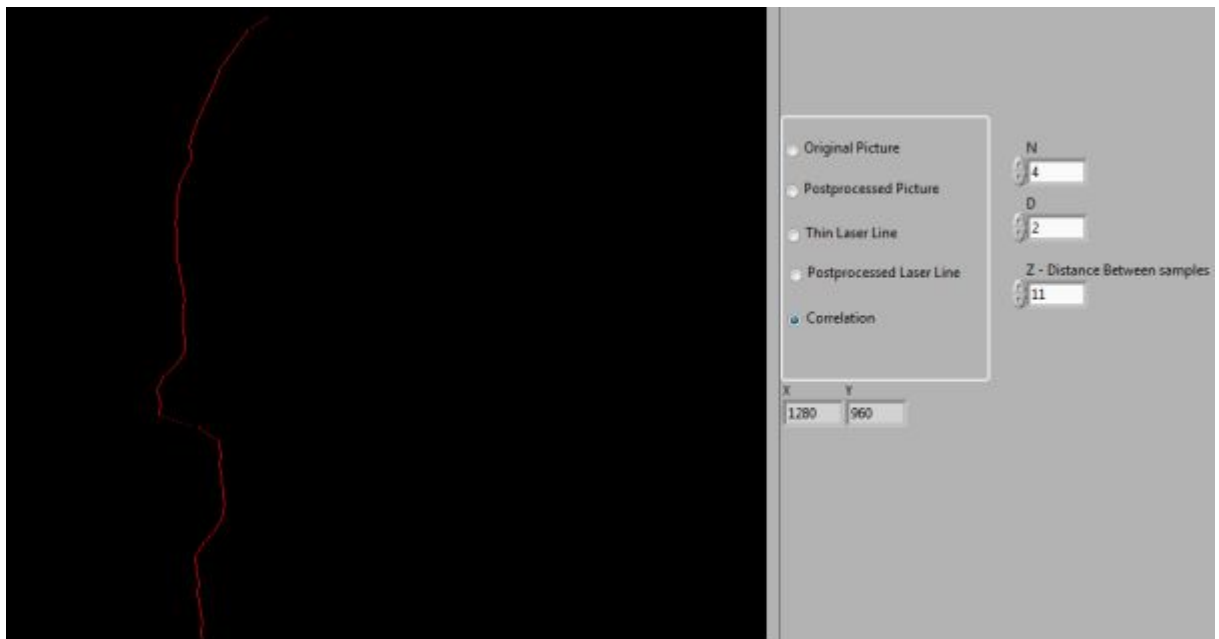


3.9 ábra. Modell kiugró élekkel, és azok nélkül

Ahogy a 3.9 ábrán látható, nagyon fontos, hogy a kiugró adatokat eltávolítsuk a modell készítése előtt. Az itt bemutatott modell jó fényviszonyok mellett készült, viszont a webkamera *CCD*-jén pixelhibák voltak, de mint ahogy a jobb oldali képen is látszik a hibás adatok eltávolításával az ilyen problémák teljesen kiküszöbölhetőek.

3.12.3 Hézagok összekötése

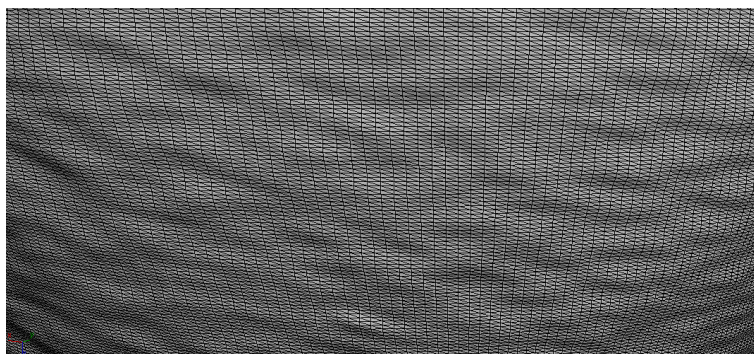
Fent említett módszer azt eredményezi, hogy lyukak keletkeznek a modell felületén, tehát ezzel még nincs teljesen megoldva a probléma. A kiugró adatok el lettek távolítva, és mivel a képek alapján semmilyen információval nem rendelkezünk afelől, hogy valójában oda milyen értékek illenének, ezért az így keletkezett hézagokat egyenesekkel hidaltuk át. Ez a legtöbb általunk vizsgált esetben nagyon célravezető módszer volt, egyáltalán nem okozta a modell torzulását, sőt így, hogy a felület folytonos maradt sokkal jobban néztek ki a modellek.



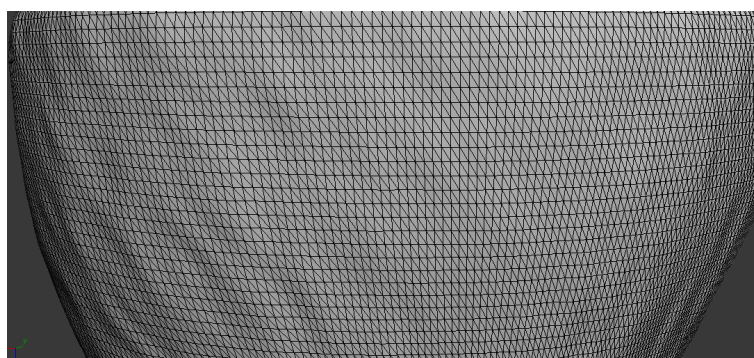
3.10 ábra. A szoftver által utófeldolgozással meghatározott lézervonal középpontok

3.12.4 Mintavételek átlagolása és polygonok számának csökkentése

A mintavételek ritkítására azért van szükség, mert a legtöbb 3D szoftver csak lassan tudja kezelni a nagyon magas *polygonszámú* objektumokat, ezért ki kellett fejlesztenünk olyan algoritmusokat, melyekkel a modell még alakhű marad, és nem romlik jelentősen a felbontása. Egyes esetekben a térbeli pontok számának csökkentése sokkal jobb minőségű modelleket eredményezett. A módszerünk a *vertex*-ek számának csökkentésére az a mintavételek átlagolása. A szoftverben a fájl generálása előtt be lehet állítani egy *z* tengely szerinti átlagolást. Ez azt jelenti, hogy ha ez az érték például 5-re van állítva, akkor öt darab pont értékét átlagolja, amiből egyetlen pontot készít, és nem használja fel az összes adatot a modellgeneráláshoz, csak az átlagolással kapott értékeket. Ezzel a módszerrel a pontfelhő számossága, és a fájl mérete is megközelítőleg ötödére csökken, ezáltal a feldolgozhatósága más szoftverek számára is gyorsabb lesz.



3.11 ábra. Modell felszíne az összes adat felhasználásával



3.12 ábra. Modell felszíne csökkentett *polygonszámmal*

A 3.11 ábrán látható modellhez az összes adatot felhasználta a szoftver a *polygon*-ok készítéséhez, míg a 3.12 ábrán lévők a térbeli felbontása ötöd akkora. A képeken látható, hogy a modell minősége nem romlik amiatt, hogy kevesebb *polygon*-ból épül fel.

3.12.5 Moduláris, Pipe-Line szerű kép elő- és utófeldolgozás

A fejlesztés során sok saját fejlesztésű, tetszőlegesen paraméterezhető algoritmust használtunk annak érdekében, hogy a lehető legjobb kinézetű modell fájlt kapjuk eredményül. Azonban a képfeldolgozás szempontjából nem mindegy, hogy milyen sorrendben történnek meg ezek a módosítások és milyen paraméterekkel. Ennek érdekében egy olyan programmodulon dolgozunk, melynek a segítségével teljesen modulárisra lehetne tenni ezen algoritmusok alkalmazását. Ez alatt azt értjük, hogy az egyes elő- és utófeldolgozó módszereket akár többször is le lehessen futtatni, és a végrehajtásuk sorrendjét és azok paramétereit is a felhasználó adhatta meg.

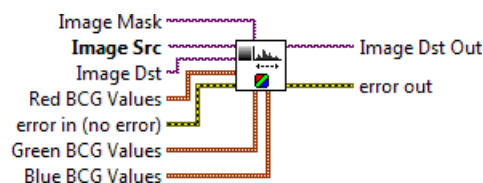
(megjegyzés: A dolgozat írásakor ez a modul még fejlesztés alatt áll.)

4. Képek feldolgozására használt algoritmusok

Ebben a fejezetben találhatóak a főbb algoritmusok működésének leírásai, melyek a programrészek futásának sorrendjében kerülnek bemutatásra. A képszekvenciákból modell fájl előállítása több munkafázisból áll, melyek megvalósítására saját megoldásainkat használtuk. A fejlesztés során használtunk a LabView IMAQ képfeldolgozó modulok mellett más programnyelveket is, mint például C és C++.

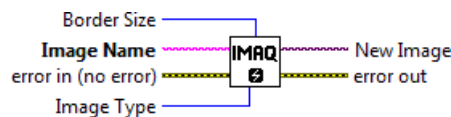
4.1 Brightness, Contrast, Gamma Correction

A lementett képszekvencia utófeldolgozásának az első lépése a BCG (Brightness, Contrast, Gamma Correction). A programban ezt az NI-IMAQ ColorBCGLookup VI segítségével valósítottuk meg. Ez egy beépített VI, melyet az NI Vison képfeldolgozáshoz használt bővítménye tartalmaz.



4.1. ábra. IMAQ ColorBCGLookup VI

A VI a bemenetére (Red, Green, Blue BCG Values) kapott adatok alapján, egy kép fényintenzitását, a pixelek közötti fényerő arányát, és a közepes intenzitású pixelek erősségét szabályozza. A VI az RGB színmodellt (red, green, blue) használja, tehát mindhárom színsíkra külön BCG korrekció alkalmazható. A bemeneti kép és a BCG paraméterek mellett biztosítani kell a javított kép számára egy ideiglenes memória területet is, amit az Image Dst bemenetre kell kötni.



4.2. ábra. IMAQ Create VI

Fontos, hogy a memória foglalásánál megfelelő típus legyen beállítva, ami egy színes kép esetén az RGB U32, tehát a kép minden pixele egy előjel nélküli 32 bites egész számként kerül letárolásra a memóriában.

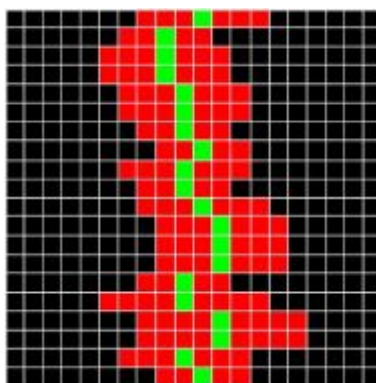
4.2 Lézervonal középpontjainak meghatározása és letárolása 2D tömbbe

Miután a megfelelő beállításokkal a program elvégezte a BCG korrekciót, a lézervonal középpontjainak a meghatározására van szükségünk. Először a feldolgozandó képet egy kétdimenziós tömbbé alakítja a program az IMAQ ImageToArray VI segítségével. A tömb elemei a kép pixeleinek RGB értéke, amelyek 0 és 16,777,216-en közötti számok lehetnek. A középpontok megtalálásához azonban elegendő, ha csak két értéket használunk, ezért a nem nullaértékű elemeket megváltoztatja a program, az alábbiak szerint:

- ahol nulla a BCG feldolgozás után az adott pixel RGB értéke, ott nulla marad az elem értéke
- ahol a BCG feldolgozás után az adott pixel RGB értéke nem nulla, ott egyre módosul az érték.

Ezután az így kapott kétdimenziós tömböt a program sorfolytonosan vizsgálja, az adott sorban balról jobbra pixelenként haladva, először egy olyan elemet keres, aminek az értéke 1, tehát a több képpont széles lézervonal bal széle. Ha talál ilyen elemet a program, akkor a lézervonal jobb széle értelemszerűen azon pozíciójú elem lesz, ami után 0 értékű képpont következik. Tehát elindul egy újabb keresés az adott sorban az elsőként talált nem nullaértékű elemtől, és addig vizsgálja a tömb elemeit, amíg meg nem találja az első nullaértékű elemet.

Majd a keresés során a feltételeknek eleget tevő két elem indexének, tehát a lézervonal két szélének koordinátájának átlagából meghatározásra kerül az adott sor lézerközepontja. Ez az algoritmus a kétdimenziós tömb minden során lefut, kimenete egy egydimenziós tömb, amely tartalmazza a lézervonal középpontjainak a pixelben mért távolságát a kép bal szélétől.



4.3. ábra. Lézervonal középpontjainak meghatározása

4.3 Kiugró élek eltávolítására használt algoritmusok

Hibásnak tekintjük azokat az értékeket melyek jóval eltérnek a környezetében lévő adatoktól. Miután már a lézervonal középpontját megtalálta a szoftver előző modulja, mindössze egy olyan algoritmusra van szükség, ami eltávolít minden egyes olyan pixelt, aminek nincsen egyetlen közvetlen szomszédja sem. Tehát ezzel a módszerrel csak az egy pixel méretű hibákat tudtuk kiszűrni. Kezdetekben ezzel a módszerrel már jelentős javulást értünk el, de ez természetesen nem jelentett teljes megoldást. Később egy olyan paraméterezhető függvényt dolgoztunk ki, amivel közel az összes hibát el lehet távolítani a megfelelő beállítások megadásával.

A nagyobb kiterjedésű hibás pixeleket is el lehet távolítani. Ennek az algoritmusnak két paramétert kell megadni:

- N (mint Neighbour)
- D (mint Distance)

A módszer egyszerű, a szoftver minden egyes pixelt megvizsgál, megszámlálja, hogy az adott pixeltől D távolságra hány szomszédos pixel van. Ha D távolságban a szomszédos pixelek száma kevesebb, mint N, akkor az adott tartomány összes pixelét törli.

Az algoritmus C nyelven:

```
void kill_lonely_pixels(int **matrix) {
    int i,j;
    for (i = lowPixel; i < highPixel; i++)
        for (j = 0; j < width; j++)
            if (matrix[i][j] && (!(matrix[i+1][j-1] || matrix[i+1][j] || matrix[i+1][j+1] ||
matrix[i-1][j-1] || matrix[i-1][j] || matrix[i-1][j+1])))
                matrix[i][j] = 0;
}

void kill_lonely_pixels_2(int **matrix, int minneighbours , int maxdistance) {
    int i,j,k,l,neighbours;
    for (i = lowPixel; i < highPixel; i++) {
        for (j = 0; j < width; j++) {
            neighbours=0;
            if (matrix[i][j]) {
                for (k=i-maxdistance ; k<i+maxdistance ; k++) {
                    for (l=j-maxdistance ; l<j+maxdistance ; l++) {
                        if (matrix[k][l]) {
                            neighbours++;
                        }
                    }
                }
                if (neighbours<minneighbours) {
                    for (k=i-maxdistance ; k<i+maxdistance ; k++) {
                        for (l=j-maxdistance ; l<j+maxdistance ; l++) {
                            matrix[k][l] = 3; // törölhető
                        }
                    }
                }
            }
        }
    }
    for (i = lowPixel; i < highPixel; i++) {
        for (j = 0; j < width; j++) {
            if (matrix[i][j] == 3) {
                matrix[i][j] = 0;
            }
        }
    }
}
```

4.4 Hézagok kitöltése

A szoftver első működőképes verziójának használata közben szembesültünk azzal a hibával, ami akkor tapasztalható, ha a tárgy felületének egyes részein nem látszik a lézervonal, ezt a jelenséget nevezik *okklúzió*nak. Ez két okból történhet, az egyik ha a kamera nem látja a lézervényt, a másik pedig az, ha a lézervény meg sem világítja a tárgyat. Ugyanezt eredményezi a kiugró élek eltávolítására használt módszer is. A térbeli pontok hiánya miatt megszakad a modell felületének folytonossága. Ennek kiküszöbölésére azt a megoldást találtuk ki, hogy a hézagokat egy egyenes vonallal áthidaljuk. Viszonylag kis felületeken szokott előfordulni az, hogy optikai okok miatt nem látszik a lézervény, ezért az egyenes vonallal való összekötés egyáltalán nem okoz torzulást a modellen. [6]

A hézagok összekötésére használt C nyelven írt kódrészlet:

```
//////// ÖSSZEKÖTÉS
for(gap=0; gap<gapcnt; gap++) {
    // GAP X1,Y1
    for (i = lowPixel; i < highPixel; i++) {
        rowHasPixel = 0;
        for (j = 0; j < width; j++) {
            if (matrix[i][j]) {
                rowHasPixel = 1;
                break;
            }
        }
        if (rowHasPixel == 1) {
            x1 = j;
            y1 = i;
        }
        else {
            break;
        }
    }

    // GAP X2,Y2
    for (i = y1+1; i <= highPixel; i++) {
        rowHasPixel = 0;
        for (j = 0; j < width; j++) {
            if (matrix[i][j]) {
                rowHasPixel = 1;
                break;
            }
        }
        if (rowHasPixel == 1) {
            x2 = j;
            y2 = i;
            break;
        }
    }

    // FILL GAP
    step = (float)(x2 - x1) / (float)(y2 - y1);
    stepPos = 0;
    for (i = y1+1; i < y2; i++) {
        matrix[i][(int)(x1+stepPos)] = 3;
        stepPos += step;
    }
    for (i = lowPixel; i < highPixel; i++) {
        rowHasPixel = 0;
        for (j = 0; j < width; j++) {
            if (matrix[i][j]) {
                rowHasPixel = 1;
                break;
            }
        }
    }
    lowPixel = y2;
}
```

4.5 Mesh optimalizálás

A háromdimenziós pontfelhő utólagos feldolgozás nélkül túlzottan sok pontot tartalmaz, ami nem feltétlenül jelent minőségjavulást, de a szoftverek viszont lassabban kezelik a nagyobb állományokat, ezért kell csökkenteni a *vertex*-ek számát.

A pontok számának redukálását átlagolással oldottuk meg. Az algoritmus egy kétdimenziós tömbbel dolgozik, amiből a fent leírt módszerrel már eltávolítottuk a hibás adatokat. A modellfájl generálásának megkezdése előtt meg lehet adni egy egész számot, amit a *mesh* pontjainak a csökkentésére használ fel a szoftver. Ez a program modul azt csinálja, hogy minden egyes két dimenziós tömböt az előbb említett értékenként átlagolja. Tehát ha például ezt a számot 5-re állítjuk, akkor a térbeli pontok száma is ötödére fog csökkenni, de mivel az újonnan készült pontokat átlagolással számítottuk, így a modell nem torzul jelentősen. Sok képszekvenciával teszteltük ezt az algoritmust, és úgy vettük észre, hogy a megfelelő beállításokkal, nemhogy rontja, de sokkal inkább javítja a modell felületét.

Az átlagoláson alapuló *mesh* optimalizálásra használt kódrészlet:

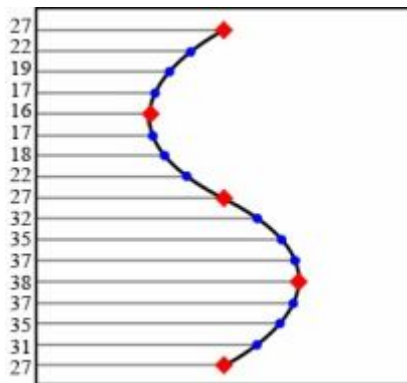
```
////////// ATLAG:
void atlag (int** matrix, int dist) {
    int i=0,c,j,k,l,avg;
    for (c=0 ; c<dist ; c++)
        for (j = 0; j < width; j++)
            if (matrix[i][j])
                matrix[i][j] = 1;

    while (i<lowPixel) {
        avg = 0;
        for (c=0 ; c<dist ; c++) {
            for (j = 0; j < width; j++) {
                if (matrix[i][j]) {
                    matrix[i][j] = 0;
                    avg += j;
                    break;
                }
            }
            i++;
            avg = avg / dist;
        }
        matrix[i][avg] = 1;
    }
}
```

4.6 Vonalsimítás B-Spline görbével

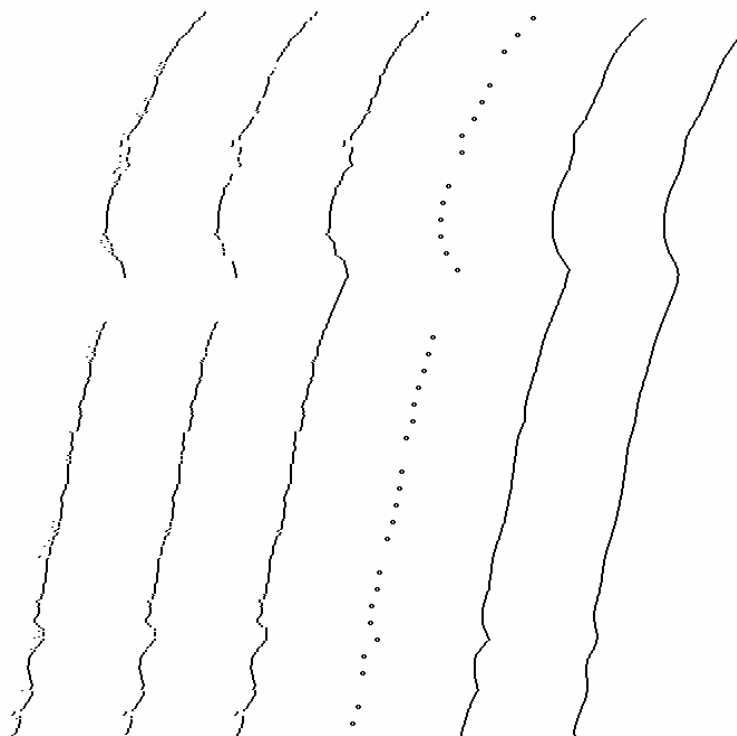
A fent említett módszerek kidolgozásával egyre jobb és jobb minőségű modell fájlokat tudtunk létrehozni, de törekszünk a további fejlesztésre is ezért egy olyan módszert használtunk, amivel még simábbá tehető a modell felszíne. A z (magasság) tengely mentén lévő adatokat átlagoljuk és az így kapott pontokat nem egy egyenessel kötjük össze, mint a fentebb leírt módszernél, hanem *B-Spline* görbét rajzol a szoftver a pontok közé, így még jobban fog kinézni a modell felülete. Ennél a módszernél azonban a pontok számát nem lehet olyan mértékben csökkenteni, mint amikor csak egy egyenes vonallal kötjük össze az átlagolással nyert pontokat. Ezt a módszert a számítógépes grafika más területein *resampling*-nek is nevezik. Az általunk használt módszer is hasonló, csak esetünkben kétdimenziós tömbbel végzünk műveleteket. Az erre kifejlesztett módszerünk úgy működik, hogy meghatározott pontonként átlagolja a szoftver a pixelek x tengely menti értékeit, majd az így kapott pontok alapján *B-Spline* görbét rajzol, és az így készült kép egyes pontjainak koordinátáit tárolja le, végül ezen adatok felhasználásával készíti el a modell fájlt. [7]

A *B-Spline* görbe megrajzolására nem írtunk saját algoritmust, egy már meglévőt használtunk fel, melyet 1994-ben Keith Vertanen publikált. [10]



4.4. ábra. Mintavételezés a *B-Spline* görbe pontjairól

Az 4.4 ábrán a piros négyzetek a kontrollpontjai a görbének, melyről bizonyos távolságonként mintát veszünk.

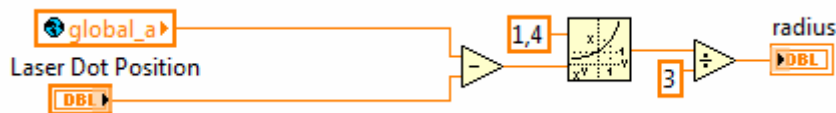


4.5. ábra. A lézervonal középpontjának optimális meghatározása

A fent leírt algoritmusokat jól szemlélteti az 4.5. ábra, amin látható, hogy az eredetileg megtalált középpontok gyakran tartalmaznak kiugró pixeleket, az ilyen adatok alapján készített modell felülete pedig nem alakhű. A második vonalnál már nem látszanak a hibás pixelek, a harmadik vonalnál pedig látható, hogy a hézagokat egyenes vonallal kötöttük össze. Ettől jobbra pedig az öt pixelenként átlagolt pixel pozíciók láthatóak, majd a következő vonal az átlagolt pixeleket összekötve ábrázolja, melyet felhasználva már sokkal simább felületű modell készíthető, mint az eredeti adatok alapján. A jobb szélső vonal esetében *B-Spline* görbét alkalmaztunk. Összevetve az egyenesekkel összekötött vonallal (jobbról a második) látható, hogy a vonal sokkal simább, így a végeredmény sokkal szebb.

4.7 A tárgy forgástengelyétől való távolság meghatározására használt módszerek

A szkenneléshez először a saját kalibrációs módszerünket használtuk, miszerint egy fehér lapot tartottunk a forgótálca közepéhez, meghatározva így a tárgy forgástengelyét. Ez a modell z tengelye. A képeken látható lézervonal középpontok x koordinátáit kivontuk a referenciaképen látható lézer középpontjának x koordinátájából, és az így kapott számot konstanssal szoroztuk, majd később hatványoztuk. A szorzótényező illetve a hatványkitevő értékét a *trial and error* módszerrel határoztuk meg úgy, hogy az elkészült modellek teljesen alakhűek legyenek. Természetesen ez a módszer nem alkalmas mérethelyes koordináták meghatározására.



4.6. ábra. A sugár számítására használt első módszerünk

A távolság meghatározására használt második módszerünk azt feltételezi, hogy a kamera pontosan a forgástengely felé néz, tehát a kép közepén lévő függőleges tengelyt használjuk referencia vonalként. A lézervonal pontjai, és a középvonal közötti távolságot egy megfelelő számmal szorozva kapjuk meg a megfelelő sugár értékeket. Ez a szorzó a kamera és a lézer által bezárt szögtől függően változik. Minél kisebb a beesési szög annál nagyobb a szorzó. Ez a módszer azt követeli meg, hogy a felhasználó tudja a kamera és a vonallézer által bezárt szöget, amit a szoftverben be kell állítani.

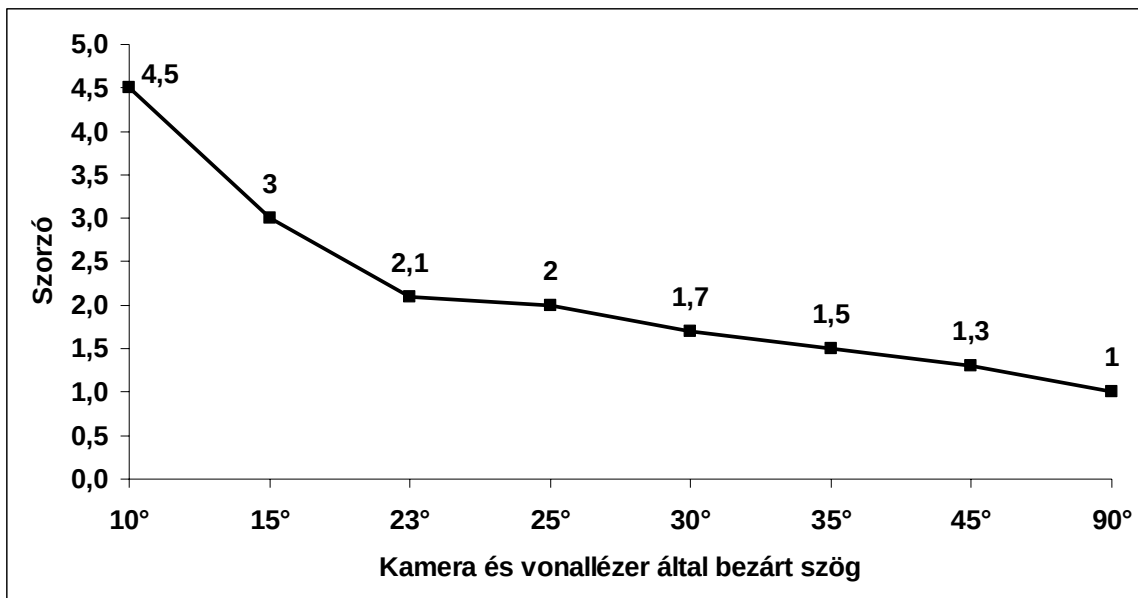
Lehetőséget nyújt a szoftver arra is, hogy a rendelkezésre álló képterület nagyobb részét kihasználjuk, a pontosság érdekében. Ebben az esetben készíteni kell egy referenciaképet, amin egy függőlegesen elhelyezett papír lapot világít meg a lézerfény, ami a forgástengely vonalában van elhelyezve. Ezen funkció használatakor a szoftverben be kell állítani, hogy az előbb elkészített referenciavonalat automatikusan ismerje fel.

A kalibrációs görbe elkészítéséhez több mérést is végeztünk, úgy, hogy a kamera és a lézer más-más szöget zárt be, majd megállapítottuk a megfelelő szorzó értékeket a különböző beállítások esetén. Ezeknek a referencia adatoknak az ismeretében a szoftver képes kiszámítani bármilyen 10 és 90 fok közötti szöghöz megfelelő szorzót, amivel a legpontosabb sugár értéket tudjuk számítani, és így az elérhető leginkább mérethelyes modellt előállítani.

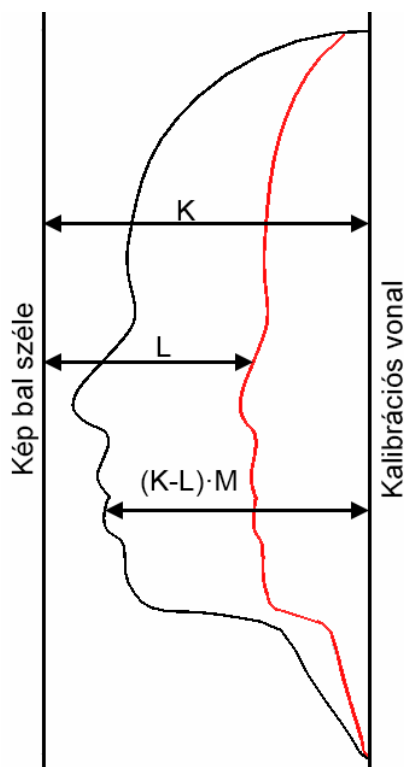
A megfelelő szorzó számítására az alábbi kódrészletet készítettük:

```
double datas[8][2] = { {10, 4.5},
                       {15, 3},
                       {22.5, 2.1},
                       {25, 2},
                       {30, 1.7},
                       {35, 1.5},
                       {45, 1.3},
                       {90, 1} };

float x, prc;
int range, i, j;
for (j=0 ; j<8 ; j++) {
    if (szog>=datas[0][j] && szog<=datas[0][j+1]) {
        range = j;
        break;
    }
}
prc = ( (szog-datas[0][range]) / (datas[0][range+1]-datas[0][range]) );
x = datas[1][range] - (datas[1][range] - datas[1][range+1]) * prc;
```



4.7. ábra. Különböző beállításokhoz tartozó kalibrációs görbe

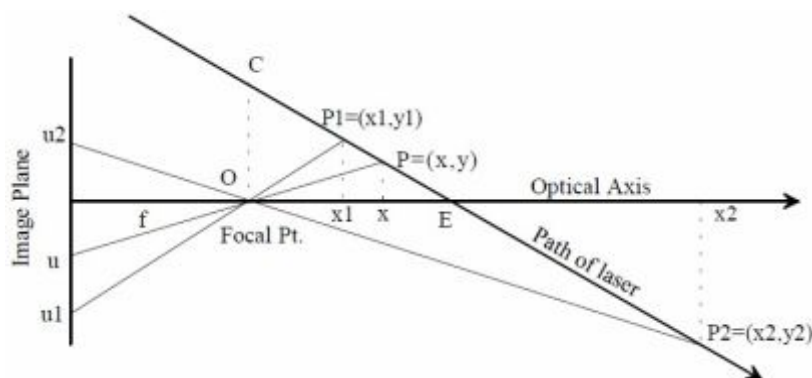


4.8. ábra. Sugár értékek meghatározása a lézervonal pontjai alapján

- $r = (K - L) \cdot M$
- K: a kép bal szélétől mért távolság pixelben a kalibrációs vonalig
- L: a lézervonal koordinátja az x tenelyen
- M: a megadott szög alapján számított szorzó

Mivel a módszer nem teljesen pontos, ezért kezdtünk el dolgozni egy harmadik módszeren a távolság meghatározására, a háromszögelés (*trianguláció*) elvén működő távolság meghatározáson.

Mivel a kalibráló referencia doboznak minden mérete ismert, így a generált modell magasságát a szoftver könnyen ki tudja számítani. A térbeli pontok koordinátáinak kiszámítását is segíti ez a módszer. Ha ismert az origó mélységében egy felületen látható lézerfényhez tartozó pixel érték, és egy másik az origótól adott távolságra lévő felületen látható lézerfényhez tartozó pixel érték, akkor bármilyen pixelértékhez tartozó távolságot könnyen kiszámolható.



4.9. ábra. Háromszögelés működésének elvi rajza

Az 4.7. ábrán látható a háromszögelés alapú távolság meghatározás elvi rajza. Az x értéket ki lehet számítani a kalibrálás során megállapított $P1$ és $P2$ pontok, valamint a már ismert $x1$, $x2$, és $u1$, $u2$ pontok alapján. [4]

4.8 Sorfolytonos adattárolás előnyei

A *vertex*-eket összekötő algoritmust C nyelven kezdtük el írni, és egy olyan módszert találtunk ki, ami a lehető legoptimálisabb az általunk alkalmazott módszernél. A sorfolytonos adattárolás alatt azt értjük, hogy a képfeldolgozás során egy kép alapján egy egydimenziós tömböt készítünk, melyben felülről lefelé haladva tároljuk azt a szám adatot ami az adott sorban a lézervonal középpontjának x tengely menti pozíciója. A szoftver az összes kép feldolgozása során az így készült egy dimenziós tömbökből egy kétdimenziós tömböt készít, melyben az összes nyers adatot tároljuk. Az így tárolt adathalmaz ezáltal könnyen feldolgozható, mert a tömb minden eleméről kiszámítható az indexei alapján, hogy milyen magassághoz, és elfordulási szöghöz tartozik.

Amint a szoftver végzett a képek feldolgozásával és feltöltötte adatokkal a kétdimenziós tömböt, a következő lépésben a pixel értékekből ki kell számolni, hogy mennyi az origótól számított sugár. Az origó alatt a generálandó modell fájl geometriai középpontjának tengelyét értjük.

4.9 Pontfelhő generálása

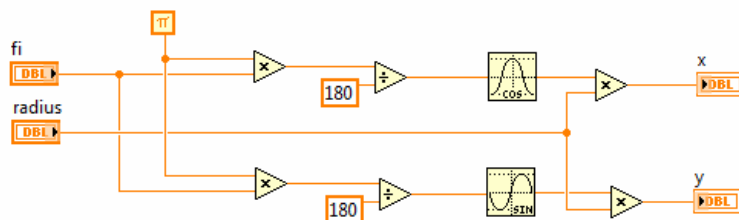
A fent leírt adathalmazt feldolgozó algoritmusnak az a feladata, hogy egy pontfelhőt készítsen azon adatok alapján. Tehát az előbb említett kétdimenziós tömb minden egyes adatából három darab számot kell előállítani, a térbeli pont három koordinátáját. A magasság, azaz a z tengely menti koordináta értéke a feldolgozott adat oszlopindexe. A másik két adat, amire még szükség van, az az x és y koordináták, melyek a szinusz és koszinusz szögfüggvényekkel meghatározhatóak, ugyanis a tömb sorindexéből megtudhatjuk, hogy milyen elfordulási szöghöz tartozik az adat. Az összes adatot feldolgozva így egy olyan pontfelhőt kapunk melyben a modell térbeli pontjainak koordinátái vannak.

Az x és y koordináták a következőképpen számíthatók ki:

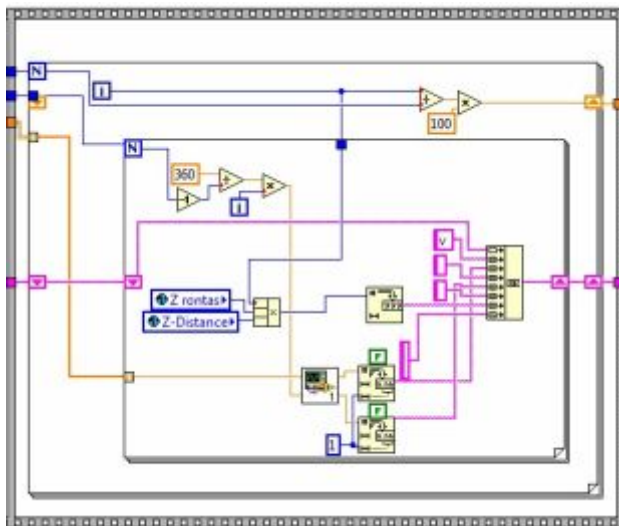
$$x = \cos \phi \cdot r$$

$$y = \sin \phi \cdot r$$

Ahol ϕ az elfordulás szöge fokban, és az r a sugár. Mivel azonban a legtöbb programozási nyelv radiánban számol így a ϕ értéket meg kell szorozni π -vel és el kell osztani 180-nal.



4.10. ábra. Az x és y koordináták meghatározására használt LabView modul.



4.11. ábra A pontfelhő készítéséhez használt LabView modul

4.10 Szomszédos pontok összekötése

Az elkészített pontfelhő szövegesen kerül tárolásra, melynél szintén fontos a sorfolytonos adattárolás, mert a pontokra sorszámuk alapján tudunk hivatkozni. A pontok összekötésére a legjobb módszer, ha három pontot összekötünk, az így kapott felületet *face*-nek nevezzük. Két *face*-ből pedig már egy négyszöget kapunk, amit *polygon*-nak hívunk.

A pontok összekötésre használt algoritmus a következő:

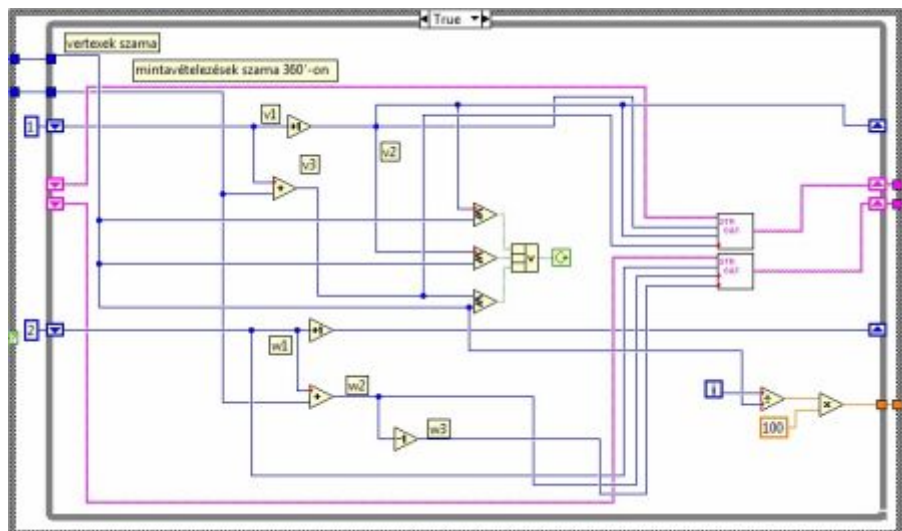
Az egyik háromszöget úgy kapjuk, hogy a legelső pontot össze kell kötni a másodikkal, és az első pont felett lévő ponttal.

A második háromszöget pedig úgy kapjuk, ha a második pontot, a második pont feletti pontot, és a második pont feletti pont előtt lévő pontot kötjük össze.

Ezt a módszert az összes térbeli pontra alkalmazzuk, így minden szomszédos pontot összekötünk a megfelelő pontokkal, vagyis *polygon*-okat hozunk létre, melyek a modell felületét alkotják.

A pontokat összekötő algoritmus C nyelven:

```
v1=1 , w1=2;  
while (v1<vertexcnt && v2<vertexcnt && v3<vertexcnt) {  
    v2 = v1+1;  
    v3 = v1+samples;  
    fprintf(fp, "f %d %d %d\n" , v1, v2, v3 );  
    w2 = w1+samples;  
    w3 = w1+samples-1;  
    fprintf(fp, "f %d %d %d\n" , w1, w2, w3 );  
    v1 += 1;  
    w1 += 1;  
}
```

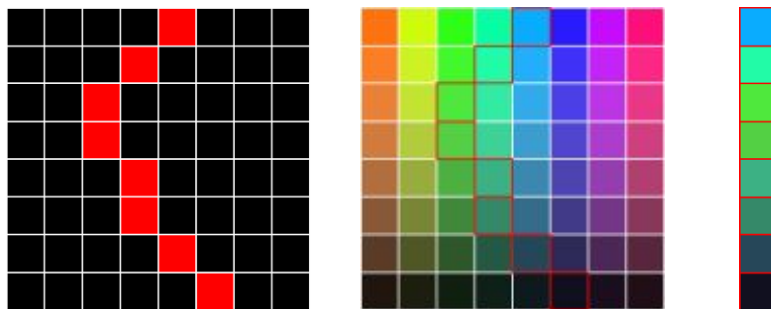


4.12. ábra A pontokat összekötő algoritmus LabView-ban megvalósítva

4.11 Textúrázáshoz használt algoritmusok

A textúra elkészítése több modulból áll. Az algoritmus, amit alkalmaztunk a lézerrel megvilágított képek alapján számolt vonalközéppontok értékeit használja fel újra. Minden egyes térbeli ponthoz egy pixelnyi színinformációt társít. A jó fényviszonyok mellett készült képszekvenciára azért van szükség, mert a szoftver a textúrázáskor azokat dolgozza fel.

Az algoritmus úgy működik, hogy az adott elfordulási szögnél a szoftver a lézervonal középpontjainak koordinátáinak tárolására használt tömb adatai alapján, és a megvilágított képről pontosan azokat a pixeleket válogatja össze, ahol azelőtt a lézervonal látszott. Magyarul ahol az első képszekvencián a lézervonal volt, most a második képszekvenciából pontosan ugyanonnan veszünk mintát, ami már színinformációkat tartalmaz. Az így kinyert adatokat egy vonalba rendezzük, tehát egy függőleges kép lesz belőle, ami egy pixel széles, és ennek a tartalma egy mintavételhez tartozó színinformáció. Ezeket az egy pixel széles képeket egymás mellé illesztve áll össze a textúra, melyet a modellre kiterítünk.



4.13. ábra. Egy mintavételhez tartozó textúra összeállításának szemléltetése egy 8x8 pixeles képen

A textúra összeállítása csak az egyik lépés, ezen kívül még minden egyes *vertex*-hez hozzá kell rendelni egy képpontot a kétdimenziós képen, a textúrán. A képpontokra nem valódi pixel koordinátájukkal kell hivatkozni, hanem két 0 és 1 közötti értékekkel.

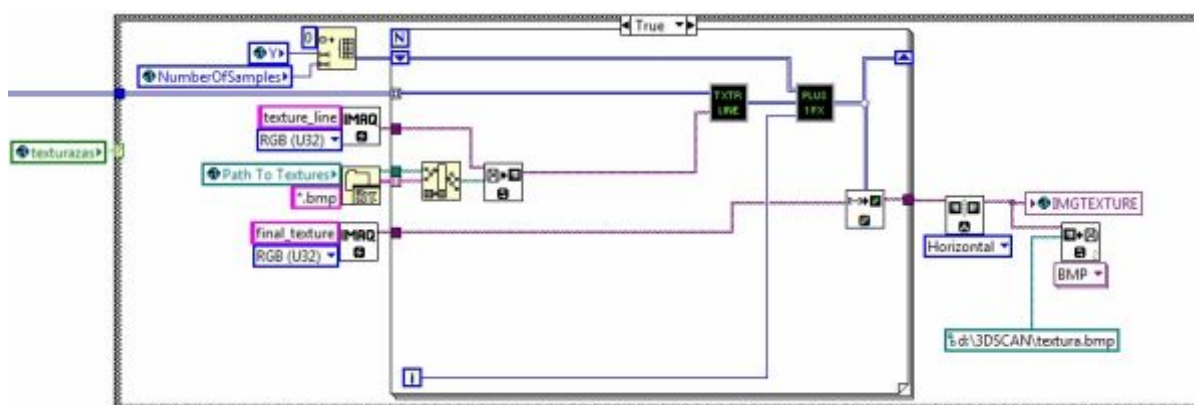
Ehhez azonban tudnunk kell minden egyes pontról, hogy milyen magasan van, és az adott magasságban hányadik mintavételhez tartozik. Ezt nagyban megkönnyíti a sorfolytonos adattárolás.

A textúra készítés és a *mesh* optimalizálás eleinte nem voltak kompatibilisek a szoftverben, mert amennyivel csökkent a térbeli pontok száma, úgy romlott az összeállított textúra felbontása is. Ezt úgy oldottuk meg, hogy a textúra összeillesztéséhez a szoftver a nyers adatokkal dolgozik, figyelmen kívül hagyva minden *mesh* optimalizáláshoz szükséges adatot. Majd amint készen van a textúra fájl, utána kezdi el a *mesh* optimalizálást, és a térbeli pontokhoz a textúrán lévő képpontok hozzárendelését.

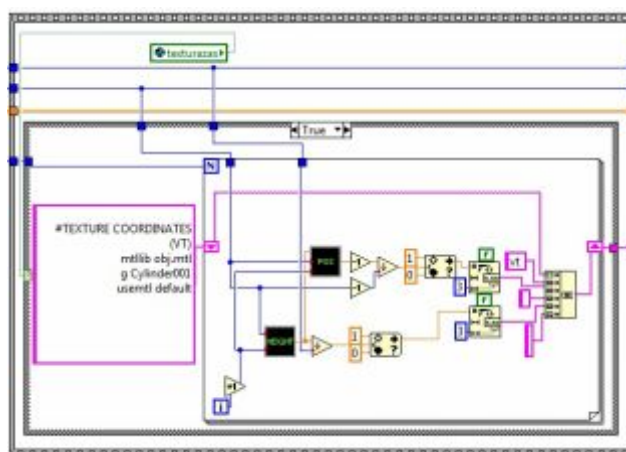
A mintával ellátott modell még alacsony *vertex*szám esetén is jobban néz ki, mint anélkül. Néhány játékprogramban használt modellt megvizsgáltunk 3D szerkesztő programokkal, és láttuk, hogy viszonylag kevés *polygon*-t használnak, inkább a textúra részletességével érik el a valósághű hatást.



4.14. ábra. A szoftver által készített textúrák



4.15. ábra. A textúra összeállításához készített LabView modul



4.16. ábra. A textúra koordináták számításához használt LabView modul

4.12 A szoftver által generált fájlok

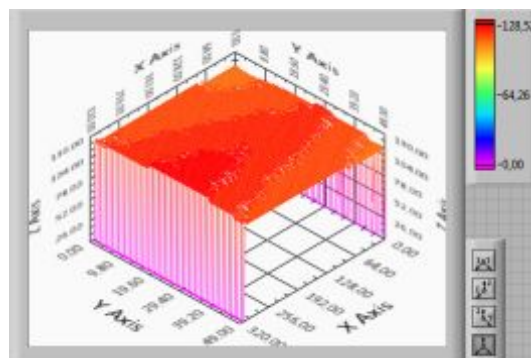
Miután a szoftver az összes fent leírt algoritmust végrehajtotta, tehát összeállította a textúrát, a pontfelhőt, és a megfelelő pontokat össze is kötötte már csak egy dolog van hátra, hogy letároljuk azokat. A program három darab fájlt ment le, éppen ezért, hogy megkönnyítse a felhasználó dolgát, csak egy elérési utat kell kijelölni és a szoftver készít oda egy mappát aminek a neve ugyanaz lesz mint a szkennelése neve, és abba tárolja le mindhárom fájlt.

Az legfontosabb fájl az maga az ".obj" fájl amiben a modell formája van leírva. A másik fájl egy "texture.bmp" nevű képfájl, ami a textúrát tartalmazza. A harmadik pedig az object.mtl fájl, erre a fájlra azért van szükség, mert az .obj fájl relatív elérési útvonallal hivatkozik rá, hogy ez alapján tölthesse be a modell anyagi paramétereit, *material*-ját. Az mtl a *material* rövidítése. Ez a *material* leíró fájl pedig a texture.bmp-re hivatkozik, megkönnyítve ezzel a legtöbb szoftver számára a textúra betöltését.

Így, hogy mindhárom fájlt egy mappában tároljuk, és relatív címezést használunk a fájlnevekhez, megspórolhatjuk azt az időt, amit amúgy egy 3D szoftverben meg kellene csinálni, hogy kikeressük a modellhez tartozó textúrát, és a modell felületére kiterítsük azt. Ezzel a módszerrel a fájl importálásakor automatikusan betöltődik a textúra is.

4.13 Pontosság

A legpontosabb referenciaadatok érdekében több mérést is elvégeztünk valódi szkennelésekkel, és 3D Studio Max-al renderelt képszekvenciák feldolgozásával is. Az értékek meghatározásához különböző átmérőjű hengerekről készült képszekvenciákat használtunk. Méréseink során arra törekedtünk hogy a sugár értékek szórása a lehető legkisebb legyen. A hengerek után különböző kockák és hasáb alakzatokról készült képsorozatokot dolgoztunk fel, melyeknél legfőbb következtetéseinket az exportált modellek alakhűségéből vontuk le. A hasáb oldalainak egyenességét, és folytonosságát vizsgáltuk, valamint az éleket. A mérési adatoknál a szoftverbe implementált eloszlási görbén, egy három dimenziós diagrammon, és szórásnégyzet számítással állapítottuk meg hogy a pontosságra vonatkozó értékek mikor a legmegfelelőbbek. Ezekkel a kijelzett adatokkal megállapíthatóak az olyan hibák, mint például ha nem a forgástengelyre helyeztük, vagy nem megfelelő szögben áll a tárgy.



4.17 ábra. A szoftverbe implementált három dimenziós grafikon a sugár értékek megjelenítésére

Az alábbi táblázatban két szkennelt hengert és egy 3D Studio Max-ból renderelt henger alakzat képszekvenciáinak pontosságára vonatkozó adatait szemléltetjük. Az értékekből látható hogy a renderelt képeknél nem tapasztaltunk hibát. A valódi hengerek szkennelésekor a legfőbb hibaforrás a szabadkézi pozicionálásból, és a webkamera képén látható lézervonal detektálásából adódott. A szoftverben a szkennelt tárgy magasságát adtuk meg milliméterben, és ezen adat alapján számítja ki a sugár értékeket is. Ehhez azonban szükséges az is hogy a modell generálása előtt a képszekvencián kijelöljük a hasznos tartományt, vagyis a ROI-t.

	Henger #1		Henger #2		Renderelt henger	
Valós sugárérték	36 mm		23,5 mm		50 mm	
Szög	23°	35°	23°	35°	23°	35°
Sugárértékek átlaga	36,16 mm	35,98 mm	23,58 mm	23,59 mm	50 mm	50 mm
Legkisebb sugárérték	33,84 mm	33,67 mm	21,75 mm	20,86 mm	50 mm	50 mm
Legnagyobb sugárérték	38,35 mm	38,69 mm	26,25 mm	25,95 mm	50 mm	50 mm
Átlagtól való negatív eltérés	2,32 mm	2,31 mm	1,83 mm	2,73 mm	0 mm	0 mm
Átlagtól való pozitív eltérés	2,19 mm	2,71 mm	2,67 mm	2,36 mm	0 mm	0 mm
Szórásnégyzet	0,673	0,669	0,933	0,776	0	0

4.18 ábra. Pontosságra vonatkozó adatok több mérés alapján

5. Összehasonlítás más eszközökkel

Több szkennelési módszert és számos eszköz specifikációját tanulmányoztuk át, hogy az általunk megfizethető alkatrészek felhasználásával a lehető legalakhibb, és legpontosabb modellt tudjuk rekonstruálni. Sok csúcskategóriás, mérnöki eszköz létezik, melyekkel nagyon pontos méréseket lehet végezni. Összehasonlítás során a pontosságon kívül figyelembe vettük még az adott eszköz felhasználási területeit, és az árát. Az alsó középkategóriás eszközök, amiket hobbi célú felhasználásra szánnak is százezres nagyságrendű árral rendelkeznek. A fejlesztéseink során az összköltség nem haladta meg a tízezer forintot, ennek ellenére a geometriai korlátokat leszámítva, az ár és pontosság arányát tekintve jó eredményeket értünk el.

Az alábbi táblázatban egy csúcskategóriás, egy alsó középkategóriás és a saját fejlesztésű szkennерünk legfontosabb jellemzőinek összevetése látható:

	NextEngine 3D Scanner HD	DAVID Laserscanner	Saját
Pontosság	0,1 mm	0,2 mm	1-2 mm
Lézer	2db 4x4 lézer, fókuszálható, Class 1M	1db fókuszálható, Class 1	1db fókuszálható, Class 3b
Szenzor	2db CMOS szenzor	Logitech Quickcam 9000 Pro	Genius Look 1320 V2
Szenzor képfelbontása	2048 x 1536	1600 x 1200	1280 x 960
Technológia	MLT - MultiStripe Laser Triangulation	Továbbfejlesztett háromszögelés	Saját módszer
Textúra készítés	igen	igen	igen
Szoftver	NextEngine Scan Studio HD	DAVID Laserscanner Pro Edition 3	Saját szoftver
Ár	\$3000 ~700000 Ft	\$530 ~ 130000 Ft	8000 Ft (költség)

5.1 ábra. 3D szkennerek összehasonlítása a saját fejlesztésű 3D szkennерünkkel

6. Továbbfejlesztési lehetőségek

Látva a fejlesztés közben elért előrelépéseket, szeretnénk még tovább fejleszteni a szoftvert, amire több ötletünk és szerencsére lehetőségünk is van. Ebben a fejezetben a fejlesztési lehetőségeket fogjuk ismertetni, melyek közül reméljük minél többet meg fogunk oldani a közeljövőben.

6.1 Mechanika automatizálása National Instruments eszközökkel

A most használt forgótálca működőképes megoldás, de mivel a szoftver nem szabályozza a forgás irányát, és azt, hogy mikor induljon el a forgás, a textúrázás pontatlanságát eredményezheti. A fények ki és bekapcsolása is jelenleg kézi működtetésű, ami szintén megnehezíti a felhasználó dolgát.

Ezen nehézség kiküszöbölése érdekében a közeljövőben, NI hardver eszközökkel (pl: CompactRIO, vagy MyDAQ, vagy USB-6212) szeretnénk megoldani a fények ki és bekapcsolását, valamint a forgatás vezérlését is. A forgótálca szinkronmotorja helyett egy *enkóderes* motort fogunk használni. Az *enkóderes* motor használatának legnagyobb előnye az, hogy pontosan ugyanabba az elfordulási helyzetbe tud visszaállni, mint a legelső körbefordulás során, így a textúra készítéshez szükséges képekhez pontosan ugyanaz az elfordulási szög tartozik majd, mint a lézervonallal megvilágított képekhez. Ezzel a módszerrel nem csak a felhasználónak lenne könnyebb az eszköz üzemeltetése, de a textúra elcsúszását is minimalizálni tudnánk.

6.2 Lézervonal középpontjának automatikus megkeresése

A szoftver képfeldolgozó moduljainak fejlesztése közben sok képszekvenciával dolgoztunk, melyek más és más fényviszonyok között készültek, ezért szükség volt egy olyan kezelőfelületre is amivel a képek fényerejét, kontrasztját, és gammakorrekcióját lehet beállítani. Ezeket az értékeket a modell generálásának megkezdése előtt kézzel állítjuk be a lehető legoptimálisabb értékekre. A szoftver későbbiekben lehetőséget nyújt majd ezen adatok automatikus beállítására is. Az erre használt módszerünk lényege az lesz, hogy a szoftver törekedjen a lehető legvékonyabbra állítani a megtalált lézervonalat, de úgy, hogy közben a folytonossága se szakadjon meg.

6.3 Más módszerek a textúra készítéséhez

Egy működő algoritmust már sikerült kifejlesztenünk, amivel megoldottuk a textúrázást. A jelenleg alkalmazott módszerünk szerint minden egyes térbeli ponthoz pontosan egy pixelnyi színinformáció tartozik, amit egy tömörítetlen kép fájlban tárol a szoftver. Azonban van két másik ötletünk is arra, hogy hogyan lehetne még jobb minőségű textúrát készíteni.

6.3.1 Több mintavétel a textúrázáshoz

Az egyik módszer, amit szeretnénk kipróbálni az úgy működne, hogy körülbelül 300 képet készítenénk a tárgyról a modell felépítéséhez, viszont a textúra összerakásához nagyjából 900 képet készítenénk, ezáltal a textúra fájl felbontása is nagyobb lenne. Így vízszintesen a szomszédos térbeli pontok között több pixelnyi színinformációt is meg tudnánk jeleníteni, ami biztos, hogy még élethűbbé tenné a modellt.

6.3.2 Kevesebb mintavétel a textúrázáshoz

A másik módszer nem biztos, hogy jobb eredményt hoz majd mint a most használt, vagy a fent leírt módszer, de más háromdimenziós szkennerek is működnek olyan elven, hogy viszonylag kevés képet készítenek a tárgyról, és az alapján készítik el a textúrát. Ennek a módszernek persze hátrányai is lehetnek abban az esetben, ha a szkennelendő tárgy formája miatt egyes részek takarásban vannak, és így a kevesebb mintavétel miatt nem a megfelelő színek kerülnek az adott helyre.

6.4 Realtime képfeldolgozás NI Smart Camera használatával

A National Instruments által gyártott Smart Camerák olyan eszközök, melyek nem csak kameraként funkcionálnak, de saját processzoruk is van, így alkalmasak Labview programok futtatására is. A szoftvernek a képfeldolgozó részét fel lehet tölteni a kamerára és a lézervonal középpontjának megkeresését, így nem a számítógép oldaná meg, hanem a Smart Camera. Ezzel a módszerrel jelentősen fel lehetne gyorsítani a szkennelés menetét. A Smart Camera már csak a feldolgozott adatokat küldené tovább a számítógépre, ahol a képernyőn egyből látható lenne a lézervonal középpontja, és amennyiben azokat a felhasználó optimálisnak ítéli meg, egyből kezdheti a modellfájl generálását, ami így sokkal kevesebb ideig tartana, főleg textúrázott modellek esetén. Ezek a kamerák hálózaton kapcsolódnak a számítógéphez, így akár távoli adatátvitelt is meg lehet oldani velük, ha éppen arra van igény.

6.5 Kompatibilitás 3D Printer eszközökkel

Manapság egyre elterjedtebbek a 3D nyomtatók, széles körben használják ezeket az eszközöket a legkülönbözőbb tárgyak készítésére, és tömeggyártásra. Sajnos eddig még nem volt alkalmunk ilyen eszközöket megismerni, de szeretnénk olyan szoftvert is kifejleszteni, ami olyan fájlokat készít, ami szintaktikailag kompatibilis valamilyen 3D nyomtatóval. Tehát célunk egy konverter program elkészítése is, ami az ".obj" fájlformátumból képes létrehozni 3D Printerekkel kompatibilis fájlformátumot, vagy közvetlen vezérelni a nyomtatót.

6.6 Arc szkennelés

A szoftver már most képes lenne jó minőségű, textúrával ellátott modellt készíteni akár emberi fejről is, azonban a hardver ezt nem teszi lehetővé.

Terveink közt szerepel egy olyan eszköz létrehozása is, amivel egy helyben, mozdulatlanul ülő emberről tudunk képszekvenciákat készíteni. A módszer ugyanaz, mint az eddigi, annyi különbséggel, hogy így a szkennelendő dolog körül kellene forгатni a kamerát és a lézert. Ennek kialakítása azonban sokkal bonyolultabb, mint a tárgy forгатása. Meg kell oldani, hogy az szkennelendő ember feje pontosan a kamera forgástengelyének közepére kerüljön. A másik nehézsége pedig az, hogy a kamera és a lézer távolsága végig állandó kell, hogy maradjon a középponttól, és a magassága sem változhat. Ennek a kivitelezéséhez valamilyen kör alakú sín rendszeren való mozgatásra lenne szükség, amin a kamera és a lézer állandó sebességgel tud mozogni. Az emberi fej szkennelésekor az egyénnek teljesen mozdulatlanul kell ülnie, ha minden oldalról alakhű modellt szeretnénk készíteni. A vonallézeres szkennelés hátránya ebben az esetben az, hogy meg kell oldani azt is, hogy amikor a lézer az ember szeméhez ér, akkor abban a vonalban egy arra alkalmas eszköz a megfelelő helyen kitakarja a lézervonalat. Amennyiben ez a kitakart rész megfelelően kicsi, úgy nem okozna jelentős torzulást a modellen.

6.7 Optimalizálás

A szoftver fejlesztése során számos újítást eszközöltünk, amik az exportált modell felületének javítását szolgálták. A folyamatos fejlesztésekhez fontosnak tartjuk azt is, hogy a már meglévő módszereinket optimalizáljuk, ezáltal lerövidítve a modellek exportálásához szükséges időt. Az egyes modulokat külön-külön optimalizáljuk, tehát minden egyes részfeladat lefutását mérjük időben, amiket egy *logfile*-ban rögzítünk. A modulok különböző működő verzióinak lefutási ideit megvizsgálva ki tudjuk választani a lehető leghatékosabb programrészeket. A feldolgozandó pixelek számát a ROI-val kijelölt kép magasságának, szélességének valamint a mintavételek számának szorzata adja.

		Intel Core 2 Q6600 (2.40 GHz)	Intel Core i5 2430M (2.40 GHz)
Teszt szken #1	Mintavételek száma	200	
	Képfelbontás	800 x 600	
	Feldolgozandó pixelek száma	96000000	
	Feldolgozás időtartama	40 másodperc	33 másodperc
Teszt szken #2	Mintavételek száma	400	
	Képfelbontás	720 x 576	
	Feldolgozandó pixelek száma	165473000	
	Feldolgozás időtartama	75 másodperc	60 másodperc
Teszt szken #3	Mintavételek száma	300	
	Képfelbontás	1280 x 1024	
	Feldolgozandó pixelek száma	391905000	
	Feldolgozás időtartama	179 másodperc	141 másodperc

6.1 ábra.

7. Összefoglalás

Dolgozatunkban egy saját fejlesztésű 3D szkennerek hardver és szoftverének elkészítési tapasztalatait írtuk le. Az általunk bemutatott hardver minimális költséggel – kis műszaki érzékkel – előállítható, reprodukálható. Nem kereskedelmi célú eszközt akartunk gyártani, hanem lehetőséget teremteni magunk számára, hogy kísérletezni tudjunk azzal, hogy lehet méret és alakhűen valós tárgyakat úgy leképezni, hogy az további számítógépes alkalmazások számára használható legyen.

A munka során egy olyan aktív nonkontakt típusú rendszert állítottunk össze, mely egy forgótálcára helyezett és körbeforgatott tárgyról készít mérethelyes háromdimenziós digitális objektumot. Ezt a mérés végeztével a szoftver textúrával is ellátja, ha a felhasználó igényt tart rá. A rendszer elkészítése során törekedtünk arra, hogy a felmerült problémák megoldásához saját algoritmusokat készítsünk, és minden programrészlet saját fejlesztésű legyen (3. és 4. fejezet). A jó minőségű digitális objektum létrehozásának első feltétele a minta felületére vetített lézervonal ideális detektálása, közepének megtalálása és az eredeti tárgy méreteivel arányos skálázás megállapítása volt. Az így létrehozott pontfelhő felhasználásával előállítottuk a minta felületét reprezentáló térhálót (*mesh*) úgy, hogy optimalizáltuk a *mesh* létrehozásához szükséges pontok számát, valamint a rácsháló rajzolásához szükséges egyenes vonalak létrehozásának algoritmusát. Fontosnak tartottuk, hogy a felület lehetőleg tükrözze az eredeti felület emberi szem számára látható anyagi tulajdonságait, ezért az eredeti tárgy textúráját egy újabb mérés során meghatároztuk, ezután pedig leképeztük a korábban létrehozott térhálóra (*mesh*). A szoftver blokkdiagramszerűen a következő műveleteket végzi:

- a képek begyűjtése és tárolása
- a lézervonal középpontjának meghatározása
- az adatmennyiség jelentős csökkentése
- a modell felületének optimalizálása
- a térbeli pontok meghatározása
- a megfelelő pontok összekötése
- a textúra összeállítása és modellre való kiterítése
- a jelenleg használatos 3D szoftverekkel kompatibilis fájlok előállítása

A háromdimenziós ábrázolásra ma is van, és a jövőben is lesz igény, az élet egyre több területén. A hardver közvetlen továbbfejlesztési lehetőségeket a NI eszközökkel való integrálásban látjuk (5. fejezet). Szoftveres tekintetben a legfontosabb továbblépés a háromszögelés adaptálása, hogy minél mérethelyesebb modelleket alkothassunk, valamint a textúrázás továbbfejlesztése. Kitűzött célunk hogy munkánkat tovább folytassuk, más mérőrendszerek működési elveit is megismerve, alkalmazni azokat, és a gyakorlatban mások által is használható olyan egységes szoftvermodulokat létrehozni, melyek kompatibilitása egyaránt megkönnyíti a fejlesztők és felhasználók dolgát.

Irodalomjegyzék

Szakirodalmi hivatkozások

- [1] **Dr. Mostafa Abdel-Bary Ebrahim:** 3D Laser Scanners: History, Applications, and Future - Civil Engineering Department, Faculty of Engineering, Assiut University (2011)
- [2] **Dr. Szirmay-Kalos László - Antal György - Csonka Ferenc:** Háromdimenziós grafika, animáció és játékfejlesztés - ComputerBooks, Budapest (2004)
- [4] **Hoa G. Nguyen Michael R. Blackburn:** A Simple Method for Range Finding via Laser Triangulation - Naval Command, Control and Ocean Surveillance Center RDT&E Division, San Diego CA (1995)
- [5] **Rubén Usamentiaga - Julio Molleda - Daniel F. García:** Fast and robust laser stripe extraction for 3D reconstruction in industrial environments - Machine Vision and Applications, DOI 10.1007/s00138-010-0288-6 (2012)
- [6] **SICK** - Ranger E/D Reference Manual, SICK IVP (2008)

Internetes hivatkozások

- [7] **Jonathan Sachs** - Image Resampling (2001)
<http://ftp2.bmtmicro.com/dlc/Resampling.pdf>
[Letöltve: 2012. március]
- [8] **Martinreddy.net** - Object files (.obj)
<http://www.martinreddy.net/gfx/3d/OBJ.spec>
[Letöltve: 2012. február]
- [9] **MTL material format** (Lightwave, OBJ)
<http://local.wasp.uwa.edu.au/~pbourke/dataformats/mtl/>
[Letöltve: 2012. február]
- [10] **Keith Vertanen** - Simple B-Spline curve algorithm
<http://www.keithv.com/software/3dpath/spline.cpp>
[Letöltve: 2012. április]

Köszönetnyilvánítás

Ezúton szeretnénk köszönetet mondani:

- Témavezetőnknek Dr. Cserhádi Csaba egyetemi docensnek, hogy elvállalta a témavezetői szerepet és, hogy szakmai tanácsokkal segítette munkánkat.
- Dr. Katona Gábornak, hogy szakmai tanácsokkal látott el, és biztatásával tovább ösztönözte munkánkat.

A publikáció/prezentáció/poszter elkészítését a TÁMOP-4.2.2/B-10/1-2010-0024 számú projekt támogatta.

A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.



SZÉCHENYI TERV

Függelék

Az alábbiakban bemutatásra kerül néhány szkennelt tárgy, és az azok alapján készített textúrával ellátott modellek, valamint internetes hivatkozások, melyeken az általunk készített videók láthatóak.

Projektünk weboldala:

<http://3dscanner.site50.net/>

Videók a szkennelés menetéről és a modellekről:

<http://www.youtube.com/watch?v=AYAigbCvpmY>

<http://www.youtube.com/watch?v=QJFTAOnAQWo>

<http://www.youtube.com/watch?v=ytu1KKGDUSc>









