



M Ű E G Y E T E M 1 7 8 2

Diplomamunka

**Mikro-CT készülék rekonstrukciós és kalibrációs környezetének
létrehozása**

Marinovszki Árpád

Témavezető: Dr. Légrády Dávid
egyetemi docens
BME Nukleáris technika Intézet

BME

2016

Diplomamunka feladat a Fizikus mesterképzési (MSc) szak hallgatói számára

A hallgató neve: Marinovszki Árpád	szakiránya: orvosi fizika	<input type="button" value="▼"/>
A diplomamunkát gondozó (a záróvizsgát szervező) tanszék:		Nukleáris Technika Tanszék <input type="button" value="▼"/>

A diplomamunka készítésének helye: Budapesti M^o szaki és Gazdaságtudományi Egyetem	
A témavezető neve: Dr. Légrády Dávid	A konzulens neve:
– munkahelye: BME NTI	(külső témavezető esetén kijelölt tanszéki munkatárs)
– beosztása: docens	– beosztása:
– e-mail címe: legrady@reak.bme.hu	– e-mail címe:

A diplomamunka címe: Mikro-CT készülék rekonstrukciós és kalibrációs környezetének létrehozása	azonosítója: DM-2016-45
<p>A téma rövid leírása, a megoldandó legfontosabb feladatok felsorolása:</p> <p>A BME Nukleáris Technikai Intézetében rendelkezésre álló mikro-CT készülék kalibrációs és rekonstrukciós szoftverrendszere alkalmas tomográfias felvételek készítésére és a képrekonstrukció elvégzésére. A vonatkozó algoritmusok érettségi foka azonban még alacsony szint^o, az eddig megvalósított eljárások a leghi^ogeegyszer^obb és olykor instabil eljárások. A hallgató feladata egységes, grafikus kártyát (GPU) is igénybe vevő kalibrációs és rekonstrukciós szoftver- és algoritmuskönyezet létrehozása. A hallgató feladatai:</p> <ul style="list-style-type: none"> -A geometriai kalibráció képfelismerési algoritmusának kidolgozása, robosztus algoritmus létrehozása és implementálása, melynél a geometriai kalibráció felhasználói beavatkozás nélkül, automatikusan elvégezhető. A hallgató feladata a geometriai kalibráció robosztusságának vizsgálata a becsült geometriai paraméterekre nézve és a rekonstruált kép tekintetében is. -A gain kalibráció robosztus algoritmusának kialakítása és implementálása a halott pixelek és csillanó pontok felhasználói beavatkozást nem igénylő korrekciója. -Az új szoftverkönyezettel készített felvételek miniségi elemzése, a készülék felbontásának vizsgálata. -A determinisztikus rekonstrukció felkeményedés-korrekciójának implementálása és összehasonlítása Monte Carlo alapú rekonstrukcióval. 	

A feladat kiadásának időpontja: 2016.02.19.
--

Témavezető vagy tanszéki konzulens aláírása:	A diplomamunka témakiírását jóváhagyom (tanszékvezető aláírása):

Önállósági nyilatkozat

Alulírott Marinovszki Árpád a Budapesti Műszaki és Gazdaságtudományi Egyetem fizikus MSc szakos hallgatója kijelentem, hogy ezt a diplomamunkát meg nem engedett segédeszközök nélkül, önállóan, a témavezető irányításával készítettem, és csak a megadott forrásokat használtam fel.

Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból vettem, a forrás megadásával jelöltem.

Budapest, 2016. június 6.

aláírás

Tartalomjegyzék

1. Bevezetés	4
1.1. A Cone-Beam CT működése	4
1.2. A tanszéken rendelkezésre álló eszközök	5
1.3. A CT vizsgálathoz rendelkezésre álló szoftverek	6
2. Célkitűzés	7
3. Az alkalmazott szoftveres megoldások	8
3.1. CUDA	8
3.2. Qt	9
4. Gain korrekció	9
4.1. A gain korrekció által javított artefaktumok bemutatása	9
4.2. A gain korrekció algoritmusai	13
4.3. A gain kalibráció használata	15
4.4. A gain kalibráció implementálása és a szükséges mérések leírása	17
5. Geometria kalibráció	20
5.1. A síkpaneles CBCT geometriája	21
5.2. A geometriai paraméterek meghatározásának elmélete	22
5.2.1. A paraméterek meghatározása Noo [9] cikke alapján	22
5.2.2. A paraméterek meghatározása Wu [10] cikke alapján	25
5.3. A geometriai kalibráló mérés kivitelezése	27
5.4. A geometriai kalibráció megvalósítása	28
5.4.1. A folyamat áttekintése és a felhasználtól kért adatok.	29
5.4.2. Teljes körbefordulás meghatározása	31
5.4.3. A golyók középpontjának meghatározása	32
5.4.4. Ellipszis illesztés	39
5.4.5. A geometriai paraméterek számítása	40
6. Eredmények	42
6.1. Gain korrekció	42
6.2. Geometriai kalibráció	43
7. Összefoglalás és kitekintés	51
Hivatkozások	52

1. Bevezetés

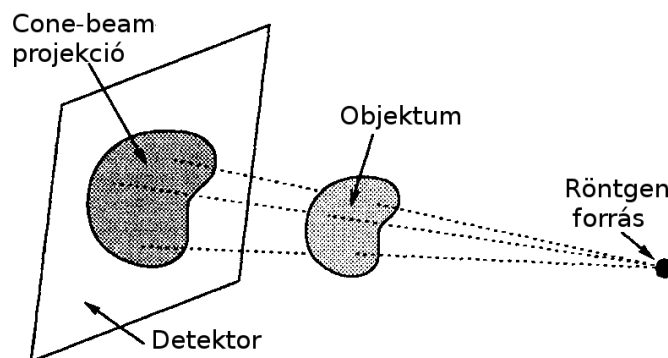
A CT, avagy Computed Tomography – magyarul komputertomográfia vagy számítógépes tomográfia – a modern orvosi képalkotó eljárások aktívan kutatott, esetenként megkerülhetetlen eszköze, amely a vizsgált páciens testének belső szerkezetéről ad információt. Az orvosi alkalmazások mellett gyakran használják a technikát állatok vizsgálatához, valamint élettelen tárgyak roncsolásmentes vizsgálatához is. A BME Nukleáris Technika Intézetében 2013–2014 között került sor egy kis méretű kúpsugaras – ún. Cone-Beam, vagy röviden CB– CT berendezés beüzemelésére, a hallgatók aktív közreműködése mellett. Jelen diplomamunkában ezt a munkafolyamatot folytatva tökéletesítem a megalkotott eszközöket, illetve dolgozok ki új metódusokat az eddig nem orvosolt problémák megoldására. Diplomamunkám elején röviden összefoglalom a CT működési hátterét, ismertetem a tanszéken rendelkezésre álló eszközöket és a rendszer eddig megtervezett részeit. Ez után összegzem a jelen diplomamunkában kitűzött feladatokat és ismertetem a megvalósításukhoz használt eszközöket.

1.1. A Cone-Beam CT működése

A CT felvételek elkészítésekor röntgensugárzással világítjuk át a vizsgált objektumot, hogy információt szerezzünk annak belső szerkezetéről. A Cone-Beam jelző arra utal, hogy a pontszerű röntgenforrás mellett síkdetektort használunk, így a detektált nyaláb kúp alakú. Egy ilyen elrendezés vázlatos rajza látható az 1. ábrán. A teljes vizsgálat során a detektort és a forrást 360° -ban körbeforgatják a vizsgált objektum körül, vagy, mint ahogy az általunk használt berendezésnél történik, az objektumot forgatják körbe, miközben a többi berendezés nem mozdul. A detektorból kiolvasott képeket hívjuk az adott elfordulás mellett kapott projekciónak. Ezen képen a kibocsátott sugárzás azon hányadával arányos intenzitásérték jelenik meg, amely hányad az objektumban nem nyelődött el. Az objektum elnyelési képességét jellemző $\mu(\underline{r})$ helyfüggő lineáris gyengítési együttható segítségével a detektort érő sugárzás I intenzitása felírható a Beer-Lambert (1) törvény segítségével, ahol I_0 jelöli a röntgenforrásból az adott pixel térszögébe irányuló sugárzás intenzitását, az integrálás pedig a forrást a detektor adott pixelével összekötő L egyenes mentén történik. A képletben, és az analitikus Cone-Beam rekonstrukció során általában, elhanyagolásra kerül többek között a gyengítési együttható és a sugárzás intenzitásának energiafüggése, valamint a különböző szórás folyamatok.

$$I = I_0 \cdot e^{-\int_L \mu(\underline{r}) d\underline{r}} \quad (1)$$

Mivel a gyengülés nélkül kapott I_0 intenzitás meghatározható, az átvilágítás során



1. ábra. A Cone-Beam CT vázlatos rajza [9].

fizikai információt a fenti egyenlet átrendezéséből kapható (2) kifejezés adja, amely az objektum lineáris gyengítési együtthatója egy adott egyenes mentén integrálva.

$$\int_L \mu(x) dx = -\ln\left(\frac{I}{I_0}\right) \quad (2)$$

A CT vizsgálat feladata a gyengítési együttható különböző egyenesek mentén integrált értékeiből visszaállítani annak 3 dimenziós eloszlását. Ezt a lépést visszavetítésnek hívjuk, és megvalósítása a projekciók inverz Radon-transzformálásával történik.

1.2. A tanszéken rendelkezésre álló eszközök

Az egyetem reaktorépületében beüzemelt CT képalkotó berendezés a következő elemekből áll.

- Röntgenforrásként egy *Source Ray SB-80-1k* típusú mikrofókuszos röntgensöveget használunk, amely kis méretű fókuszfolttal (min. $33\mu\text{m}$) rendelkezik, 35–80 kV közötti gyorsítófeszültséggel és 20–1000 mA közötti áramerősséggel üzemeltethető.
- Detektorként egy *Dexela 1207* síkdetektort használunk, amely $114,9 \times 64,6 \text{ mm}^2$ érzékeny felülettel rendelkezik, amellyel 12–130 kV közötti Röntgen-energiákra érzékeny. Pixeleinek élhossza $74,8 \mu\text{m}$, így a felbontása 1536×864 pixel.
- A vizsgált objektum forgatásáért egy *PHI Instruments* gyártmányú rotációs asztal felel, amelyet két *RDM 564/50*, *BERGER LAHR* típusú léptetőmotor hajt, minimálisan $0,72^\circ$ -os lépésközzel.

A rendszer által leképezhető objektum nagyságát a detektor méretei behatárolják, hiszen az objektum képe teljes szélességében a detektorra kell, hogy essen. Így maximálisan $\approx 10 \text{ cm}$ átmérőjű tárgyak vizsgálhatók a berendezéssel. Az egyes elemek azonban

nincsenek egymáshoz képest fixen rögzítve, azok szabadon mozgathatóak, így kisebb objektumokról is megfelelő minőségű felvételeket tudunk készíteni: minél közelebb helyezzük a tárgyat a forráshoz, annál inkább nagyított képet kapunk a detektoron. Éppen az efféle átrendezhetőség miatt fontos elvárás, hogy a geometria adatait rutinszerűen, gyorsan és pontosan lehessen meghatározni, hiszen a visszavetítéshez szükséges ezen adatok pontos ismerete. Ezt a műveletet, azaz a tárgy forgástengelyének, a forrásnak és a detektornak az egymástól való távolságának a meghatározását nevezzük geometriai kalibrációnak, amelyről részletesen az 5. fejezetben írok.

1.3. A CT vizsgálathoz rendelkezésre álló szoftverek

A teljes rendszer összeszerelését és a szükséges szoftverek megalkotását is korábbi hallgatók kivitelezték. Így rendelkezésre áll egy *C#* nyelven írt, grafikus felülettel rendelkező mérésvezérlő szoftver, amely Kleizer Gábor készítette, részben korábbi szagdolgozata [6] alapján. Ez a szoftver felelős a röntgenső vezérléséért és az ehhez kapcsolódó biztonsági berendezések működtetéséért (fényjelzés a röntgenső működése közben, stb.). Ugyancsak ez a szoftver vezérli a tárgy forgatásáért felelős léptetőmotorokat, és kommunikál a detektorral. A szoftver tehát teljes körűen elvégzi minden, a CT felvételek elkészítéséhez szükséges feladatot, a képeket pedig bináris formátumban a merevlemezre menti. Az elkészült képekről tudni érdemes, hogy a pixelek értékei 0 és $(2^{14} - 1) = 16\,383$ közötti egész számot vehetnek fel, amely szám a detektor adott pixelére eső sugárzás intenzitásával arányos a detektor mérési tartományában.

A képek feldolgozására két, az előbbtől és egymástól is független szoftver áll rendelkezésre. A vizsgált tárgy 3 dimenziós képének visszaállításáért felelős szoftvert Deli Gábor [5] írta meg, diplomamunkája során. Ez a program *C++* és *CUDA* nyelven íródott. Utóbbi egy olyan platformot nyújt, amely segítségével a visszavetítésért felelős képműveletek párhuzamosan és igen gyorsan végezhetőek el. Erről a 3.1. fejezeten írok részletesen.

Másrészt a geometriai kalibráció elvégzéséért, valamint a detektorpixelek eltérő tulajdonságaiból származó képhibák eltüntetéséért felel a Tölgyesi Botond által, diplomamunkája [2] keretei közt készített szoftver. A *MATLAB* nyelven íródott program képes a 4. fejezetben bemutatott gain hibák kezelésére, valamint az 5. fejezetben részletezett geometriai kalibráció elvégzésére is. Ezeket a funkciókat azonban viszonylag lassan végzi el, erőforrásigénye igen nagy (akár 10–20 GB RAM). Ezen kívül a képfeldolgozás során alkalmazott módszerei nem a leghatásosabbak, eredményességük nagyban függ a felhasználó által megadott paramétereiktől és a kalibrációs adatsor minőségétől. Ugyanakkor Tölgyesi megmutatta, hogy optimális körülmények között a szoftver képes a feladatok jó minőségű elvégzésére. A képek gain korrigálás igen jó minőségben

képes kivitelezni, és megfelelő paraméterek megválasztásával a geometriai kalibrációs is helyes eredményt szolgáltat. Az alkalmazott algoritmusai tehát alapvetően helyesek, amelyeket Tölgyesi részletes tesztekkel is alátámasztott.

Diplomamunkámban ezeket a funkciókat fejlesztettem tovább, hogy azok gyorsan, hatékonyan és pontosan legyenek képesek feladatuk elvégzésére. A részben instabil működése miatt ezen korábban megvalósított szoftverre, illetve annak algoritmusára a továbbiakban *kísérleti algoritmus*ként hivatkozom.

2. Célkitűzés

Jelen diplomamunkámban a Tölgyesi Botond által vizsgált gain korrekciós és geometria kalibrációs módszereket ültetem át *MATLAB* környezetből grafikus felülettel rendelkező, felhasználóbarát környezetbe. A feladat elvégzése során szem előtt tartva, hogy a megvalósított funkciók minél gyorsabban, alacsony erőforrást igényelve, a felhasználó által a lehető legkevesebb beavatkozást igényelve fussanak le. Ennek érdekében a képek feldolgozását *CUDA* nyelven valósítom meg, az elkészítendő program pedig *Qt* grafikus felületet kap. A geometriai kalibráció esetében új képfeldolgozási algoritmust vezetek be, amelyek pontosabb eredményt szolgáltat, mint az eddigiekben megvalósított. Ezen kívül a geometriai kalibrációs adatok meghatározásánál számolom és számon tartom a kalibrációs mérések hibáit. A hibákat figyelembe veszem a geometriai adatok számolásánál, így a képek feldolgozása során vétett pontatlanságokat a szoftver a felhasználó beavatkozása nélkül kezeli. A párhuzamos architektúra használatával az új szoftver lényegesen gyorsabban lesz képes a feladatok elvégzésére, amely a program rutinszerű használatát nagyban megkönnyíti.

Az új algoritmusok, illetve megvalósításuk helyességét a CT-vel készült képek alapján tesztelem. Ilyen képekből azonban viszonylag kevés áll rendelkezésre, mivel a reaktor épületét időközben felújítás miatt lezárták, így az ott összeszerelt CT berendezés is hozzáférhetlenné vált. A felvételek elkészítésére így limitált idő állt rendelkezésemre. Emiatt megfelelő minőségű geometriai kalibrációs felvételek csak egyféle geometria mellett készültek, kétféle fantommal. Az eredményeimet ezek használatával fogom bemutatni.

3. Az alkalmazott szoftveres megoldások

3.1. CUDA

A *CUDA* [4] programnyelv az *NVIDIA* nevű, grafikus kártyákat fejlesztő cég saját, *C++*-hoz hasonló szerkezetű programnyelve, amellyel az általuk megalkotott videokártyákon lehet általános célú, párhuzamos programozást igénylő feladatokat megvalósítani. A képfeldolgozásban általában igen sok művelet alkalmas a párhuzamosításra, ezért is alkalmazzák előszeretettel ezen a területen is. A képműveletek gyors elvégzése érdekében én is *CUDA* nyelven írtam meg a képek feldolgozását végző kódrészleteket.

A grafikus kártyán való párhuzamos feldolgozást illetően elmondható, hogy egy adott munkafolyamat több szinten szerveződő struktúrába foglalható. A legalacsonyabb szintű működésért az ún. *thread*-ek, vagyis *szálak* felelnek, amik a tulajdonképpeni utasításokat végrehajtó egységek. Ezeknek saját, gyors elérésű, de limitált mennyiségű memória áll rendelkezésre. A *thread*-ek *blokkokba* csoportosíthatók. Az egy blokkon belüli *szálak* egymással kommunikálhatnak, a blokk minden szála által használható, közepes sebességű *megosztott*, vagyis úgynevezett *shared* memória segítségével. Míg a *szálak* saját memóriája, illetve a blokkon belüli *shared* memória csak ideiglenes tárolóként szolgál, a videokártya *globális* memóriájában lehet a különböző műveletek között eltárolni a feldolgozandó adatokat. Szintén a globális memória szolgál a számítógép virtuális, ún. *host* memóriájával. A globális memória mérete általában igen nagy, elérési sebessége viszont az előbb említett alacsonyabb szintű memóriaterületekhez képest lassú. A memóriaterületek ilyen szintű szerveződése igen sok lehetőséget ad a *CUDA* programok futási sebességének további optimalizációjához.

A nyelv további tulajdonsága, hogy a megírt programkód sebessége skálázódik a számítógépben lévő grafikus kártya sebességével. Vagyis ugyan az a programkód minden féle módosítás nélkül is gyorsabban fut egy nagyobb teljesítményű grafikus kártyán. Ez viszont némi megkötésekkel jár, így a *CUDA* programok futtatásakor a rendszer a következő garanciákat teszi. Egy adott párhuzamos feladat, vagyis *kernel* futtatása során a *szálak* 32 elemű csoportokban, úgynevezett *warp*-okban futnak – magyarul „üstben”, de gyakran a magyar szakirodalom itt gyakran az angol megfelelőt használja. Egy *warp*-on belül minden *szál* pontosan ugyan azt a műveletet hajtja végre, párhuzamosan. A *kernel*nek megadhatjuk, hogy hány blokkot, és azokon belül hány *szálat* indítson az adott feladat elvégzéséhez, de nincs garancia arra, hogy ezek milyen sorrendben fognak lefutni. Ezek kissé laza feltételeknek látszanak, amiket a program tervezése során valóban szem előtt kell tartani, azonban így megoldható az, hogy egyszerre annyi párhuzamos *szálon* fusson a kívánt művelet, amennyi rendelkezésre áll.

3.2. Qt

A reaktor lezárása miatt elszállításra került a mérésvezérlő, nagy teljesítményű számítógép is, amelyen a jövőben futtatni szeretnénk az általam megalkotott szoftvert. Ezen a számítógépen Windows operációs rendszer fut, minthogy a mérésvezérlő szoftver csak ilyen platformra lett megírva. A kísérleti algoritmussal való összehasonlításához azonban fontosnak tartottam, hogy a saját programomat ugyan azon a számítógépen lehessen futtatni, amelyiken Tölgyesi programját is tesztelem. A tanszéken rendelkezésre álló másik nagy teljesítményű számítógép azonban Linux operációs rendszert futtat. A feladatom megoldásához grafikus felületként így egy platformfüggetlen megoldást választottam, a Qt-t [3]. A C++ nyelven programozható, széles körben ismert keretrendszer egyéb szolgáltatásai mellett alkalmas grafikus felületű programok fejlesztésére, így esett a választásom erre a megoldásra.

4. Gain korrekció

A gain korrekció az elkészült felvételeken elsődlegesen elvégzendő korrekció, amely felőlős egyrészt azért, hogy a detektorpixelek eltérő érzékenysége és zaja következtében fellépő képhibákat eltüntesse. Továbbá, hogy korrigálja a röntgennyaláb mért intenzitásában történő azon gyengülést, amely pusztán azért következik be, mert a detektorpixelek más–más távolságra vannak a forrástól.

A fejezet elején megmutatom, hogy milyen hibákat eredményez a gain korrekció elhanyagolása. Ez után ismertetem a korrekció elvégzéséhez szükséges algoritmusokat és méréseket, majd bemutatom, hogy implementáltam a gain korrekciót az elkészült programban.

4.1. A gain korrekció által javított artefaktumok bemutatása

A gain korrekció által kiszűrt hibák három részre oszthatóak. Egyrészt belátható, hogy a sík detektoron egy pontszerű forrásból származó fluxus értéke nem egyenletes. Ha olyan felvételt tekintünk, ahol nincs leképezendő objektum, a legnagyobb fluxus a detektor azon pixelét fogja érni, amely a legközelebb van a röntgenforrás fókuszpontjához. Az ettől távolabbi pixeleken mérhető intenzitás pusztán azért is kisebb lesz, mert a forrásponttól távolabb vannak, hiszen a mérhető intenzitás a fókuszponttól számított távolság négyzetének inverzével arányos. A jelenség következtében tehát a detektorpixelek intenzitása a detektor szélei felé csökken, és a csökkenés mértéke az expozíciós beállításoktól – úgy mint az expozíciós idő, valamint a röntgenső feszültsége és árama – független. A kialakult képtorzulást *flatness hibának* nevezzük, utalva arra, hogy ez

azért következik be, mert a detektorunk sík – gömbfelületű detektornál ez a hiba nem lépne fel.

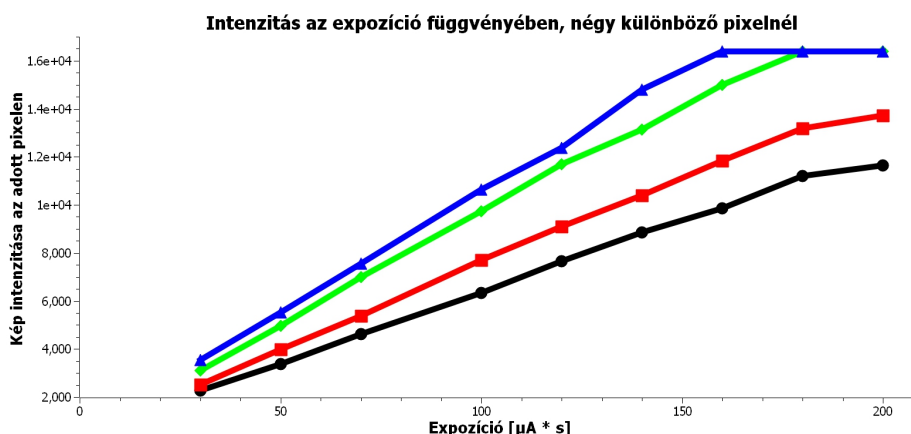
Megállapítható továbbá, hogy a detektorpixelek eltérő érzékenysége miatt is fellépnek képhibák. Ezen hibákat két csoportra lehet bontani: *offset hibára*, valamint *gain hibára*. Az elnevezés igen szemléletes, amelynek megértéséhez a következőket kell végig gondolnunk.

Válasszunk ki a detektoron egy adott pixelt, és vizsgáljuk ennek a pixelnek az mért intenzitását, miközben változtatjuk az expozíciós időt! Az adott pixelen a mért intenzitás – expozíciós idő függvényt ábrázolva olyan görbét kapunk, amely kezdetben igen jó közelítéssel egyenest mutat, mígnem az intenzitás egy adott értéket elérve nem nő tovább. Utóbbi esetet, azaz, amikor az intenzitásérték tovább nem nő, szaturációnak hívjuk. Ez a jelenség többek között hobbifotózásból is ismeretes: túlexponált képeknél a kép kiég, mivel egy bizonyos fénymennyiség felett a detektor már telítésbe megy és a valós fénymennyiségtől függetlenül a lehető legmagasabb intenzitásértéket fogja szolgáltatni.

Az előbbihez hasonlóan, ha állandó expozíciós idő mellett a röntgenső áramerősségét változtatjuk, és így nem a felvétel idejét növeljük, hanem a detektorra eső nyalábintenzitást, szintén kezdetben lineáris, majd telítődő összefüggést kapunk a pixel által mért intenzitás – alkalmazott áramerősség függvény tekintetében. Éppen ezért az expozíciós idő és a röntgenső áramerősségének szorzatának – a továbbiakban *expozíció* függvényében szokás vizsgálni az adott pixel intenzitását. Az így kapott görbe, azaz az intenzitás az expozíció függvényében, adja meg az adott pixel érzékenységet.

Ez az érzékenység azonban pixelről pixelre változik. Különböző pixelek érzékenységét megvizsgálva észrevehetjük, hogy az egyes érzékenységi görbék lineáris szakaszának meredeksége és tengelymetszete is változik. Változik továbbá a szaturációs expozíció is, azaz az az expozíció érték, amelynél az adott detektor telítésbe megy. Ezt szemlélteti a 2. ábra, hülönböző pixel által mért intenzitást ábrázoltam, az expozíció függvényében. Az ábrán jól látszódik, hogy a különböző pixelek érzékenysége eltérő meredekségű és tengelymetszetű egyenessel jellemezhető, valamint a szaturációs expozíciójuk is eltérő. Az *offset* és *gain* korrekciók a detektorpixelek ezen érzékenységbeli különbségeit korrigálják. A *gain* hibák javítása jelenti az érzékenységek lineáris szakaszainak eltérő meredekségéből, tengelymetszetéből, valamint az eltérő szaturációs expozíciókból származó hibák javítását. Ezek tehát az expozíciós beállításoktól függő hibák. Az *offset* hibák javítása jelenti a mérés során a háttérből – tehát nem a méréshez szükséges röntgen forrásból – , valamint a zajokból származó mért intenzitásokat. Az *offset* hibák nagysága nem függ tehát az expozíciótól, de függ az expozíciós időtől. Ugyanis kikapcsolt röntgenforrás mellett is detektálunk valamekkora háttérsugárzást, illetve a

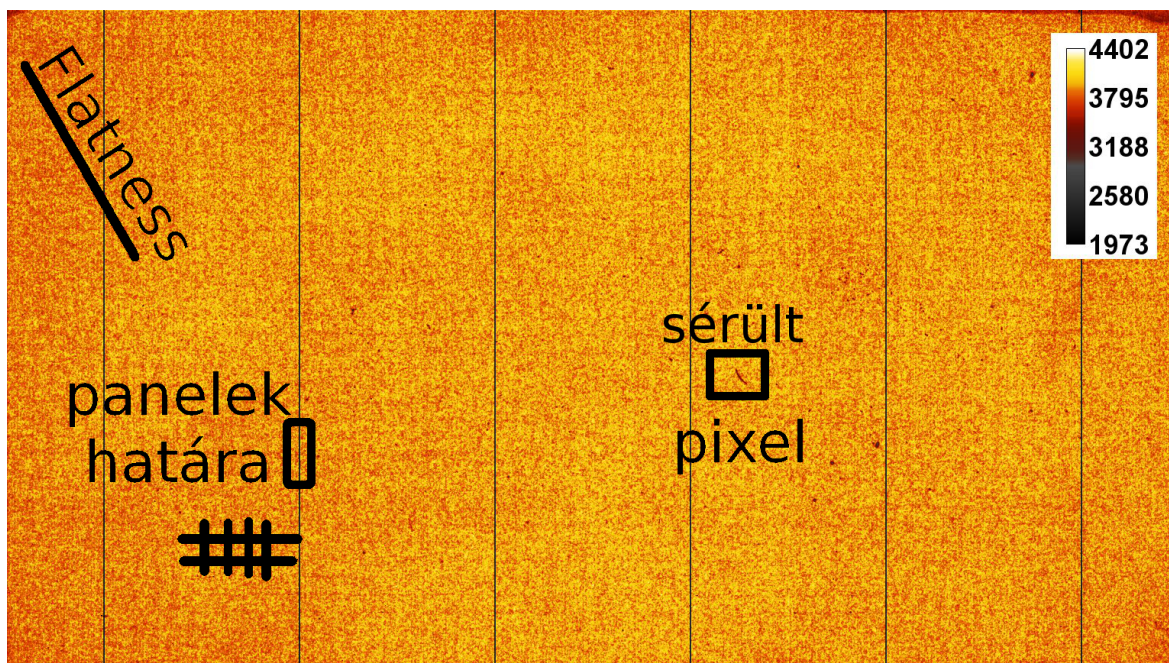
detektorok zaja is mérhető intenzitást fog eredményezni a felvételen. A teljes, röntgenforrás nélkül mért jel arányos lesz az expozíciós idővel, hiszen minél tovább mérjük a detektor zaját, az annál nagyobb értéket fog képviselni. Továbbá a mért háttér függ a pixelek érzékenységétől is: egy jobban erősítő pixel a zajt is jobban erősíti. Az offset hibákat tehát a pixelenkénti mért intenzitás – expozíciós idő görbét felvéve tudjuk javítani.



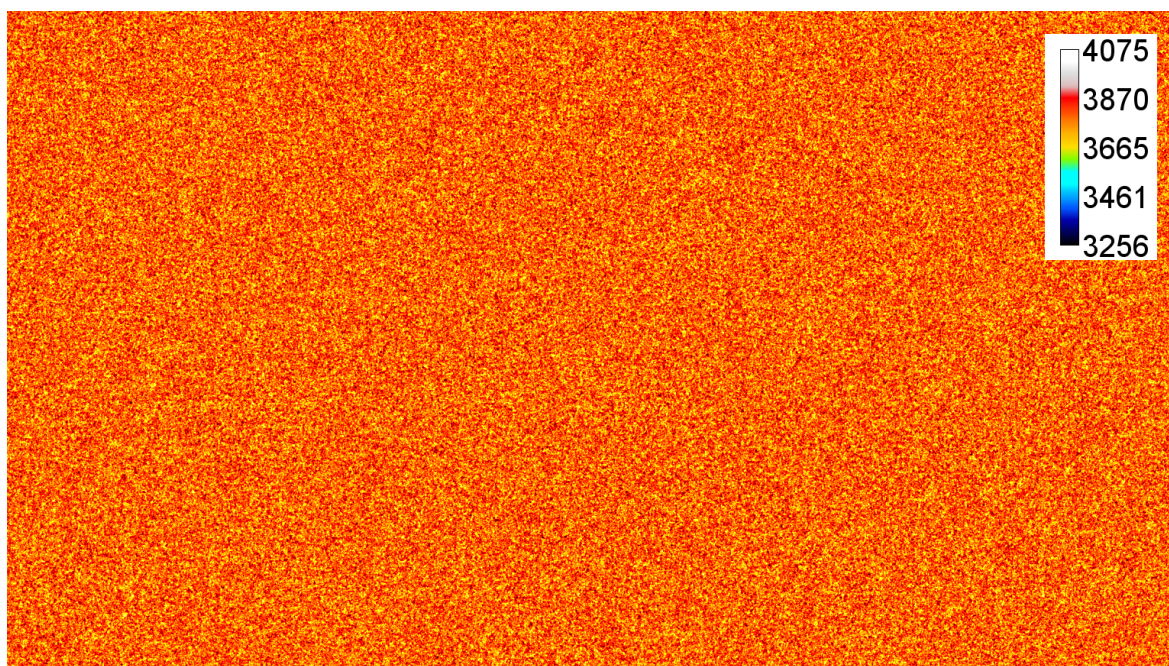
2. ábra. Négy kiszemelt pixel intenzitásának változása az expozíció függvényében.

Az egyes hibák hatását az elkészült felvételeken a 3. ábrán szemléltetem. Az ábrán egy olyan felvétel látható, amely készítésekor a detektor és a forrás között leképezendő objektum nem volt, továbbá a felvétel mindenféle korrekciótól mentes, azaz a mérőszoftver ezt a képet olvasta ki a detektorból. Az ábrán látható, hogy a leképezendő objektum hiányában homogénnek várt kép közel sem homogén. Általánosan megfigyelhető, hogy az egyes pixelek értékei jócskán szórnak, továbbá látható, hogy a képen négyzetrácsos struktúra rajzolódik ki, amelynek oka az, hogy detektorpanelek határainál megváltozik a pixelek érzékenysége. Különösen határozott ez az eltérés a képen látható hat függőleges egyenes mentén. Ezen pixelek az összes felvételen alacsonyabb intenzitással rendelkeznek társaiknál és értékük is kevésbé változik az expozíció növelésével, mint a többi pixel. Megfigyelhető továbbá, hogy a kép egyre sötétebb a sarkok felé, amely a flatness hibának tudható be.

A gain korrekció során tehát ezen hibákat távolítjuk el a képről. Gain korrekció után a fenti képből a 4. ábrán látható képet kapjuk. Megfigyelhetjük, hogy itt nem látszódnak az előbb jelölt artefaktumok. Bár a kép továbbra sem teljesen homogén – ezt nem is várjuk el, valamekkora zaj mindenképp terhelné fogja a méréseinket –, a pixelek intenzitásának szórása jóval lecsökkent, megszűnt a gain korrekció nélküli képet átható négyzetes struktúra, és a kevésbé érzékeny pixelek sem lógnak ki a többi közül. A következőkben ezen gain korrekcióhoz szükséges algoritmusokat ismertetem.



3. ábra. Tárgy nélkül készített felvétel, gain korrigálás nélkül.



4. ábra. Tárgy nélkül készített felvétel, gain korrigálás után.

4.2. A gain korrekció algoritmusai

A felvételek gain korrigálásához először is meg kell határozni az egyes detektorpixel-
lekhez tartozó érzékenységeket, azaz azt, hogy az expozíció növelésével milyen gyorsan
változik a pixel intenzitása, valamint azt, hogy az adott pixel milyen expozíció mellett
telítődik. ezen kívül meg kell határozni, hogy megvilágítás nélkül hogyan változik az
adott pixel intenzitása az expozíciós idő függvényében. Ehhez először kalibráló képsor-
okat kell készíteni. Az offset hibák javításához a röntgenforrás bekapcsolása nélkül,
különböző expozíciós időkkel, a gain és flatness hibák javításához bekapcsolt forrással,
különböző expozíció értékek mellett.

A képeket először az offset hibáktól kell megszabadítani, mivel azok a leképezendő
objektumtól és a forrás beállításaitól függetlenül vannak jelen. Az ehhez szükséges
korrekciós faktorok meghatározásához tehát különböző expozíciós időkkel, kikapcsolt
forrással készítünk képeket. Az egyes pixelek értékét az expozíció idő függvényében
ábrázolva olyan görbét kapunk, amelyre jól illeszthető egyenes. Ez alapján az i . pixel
által, forrás nélkül mért I_i^{offset} intenzitás a (3) képlettel írható le, ahol a_i^{offset} és b_i^{offset}
a fenti módon előállított képsorozatból, egyenes illesztéssel megkapható és pixelenként
változó paraméterek.

$$I_i^{\text{offset}} = a_i^{\text{offset}} \cdot t_{\text{exp}} + b_i^{\text{offset}} \quad (3)$$

Az offset hibák korrigálása után az i . pixel intenzitása a (4) képlet szerint alakul,
ahol $I_i^{\text{mért}}$ az i . pixel eredeti, korrekció nélküli intenzitása, $I_i^{\text{offset korrigált}}$ pedig az offset
hibától mentes érték.

$$\begin{aligned} I_i^{\text{offset korrigált}} &= I_i^{\text{mért}} - I_i^{\text{offset}} \\ &= I_i^{\text{mért}} - \left(a_i^{\text{offset}} \cdot t_{\text{exp}} + b_i^{\text{offset}} \right) \end{aligned} \quad (4)$$

Ez után javíthatjuk ki a gain és flatness hibákat. A két hiba valójában egy algo-
ritmussal javítható, hiszen a flatness hiba felfogható úgy, mintha a detektor pixeleinek
érzékenysége lenne más a detektor más-más területein. A szükséges korrekciós fakto-
rok meghatározásához olyan kalibráló képsorokat kell készítenünk, amelyek különböző
expozícióval készülnek. A 2. ábrán látottak alapján megállapítható, hogy az egyes de-
tektorpixelek intenzitása kezdetben az expozícióval lineárisan változik, vagyis egyenest
tudunk rá illeszteni. Amennyiben tehát nincs tárgy a detektor és a forrás között, az
 i . pixel által, a detektort érő sugárzás, mint hasznos jel által okozott (tehát offset
hibától mentes) $I_i^{\text{offset korrigált}}$ intenzitás értéke az (5) egyenlet szerint számolható az
expozícióból. Itt a_i^{gain} és b_i^{gain} jelöli a különböző expozíciós értékek mentén felvett in-

tenzitásértékekre illesztett egyenes paramétereit, E pedig az expozíció értékét, amely a fent leírtak alapján az expozíciós idő és a röntgencső áramának szorzata.

$$\begin{aligned} I_i^{\text{offset korrigált}} &= I_i^{\text{mért}} - I_i^{\text{offset}} \\ &= a_i^{\text{gain}} \cdot E + b_i^{\text{gain}} \end{aligned} \quad (5)$$

Egy olyan felvételen, amely készítésekor valamilyen tárgyat helyezünk a detektor és a forrás közé, a mért intenzitásértékek nyilván el fognak térni a detektor különböző részein, ahogy a leképezni kívánt tárgy lineáris gyengítési együtthatója is eltér a különböző térfogatokban. Ezt úgy vehetjük figyelembe, hogy az (5) képletben az E expozíció érték helyére egy helyfüggő, E_i^{virt} virtuális expozíciót vezetünk be. Ezen érték úgy módosul az eredeti E expozícióhoz képest, hogy figyelembe veszi a röntgen nyaláb gyengülését, miközben az áthalad a leképezni kívánt tárgyon. Vagyis egy adott mérés során az E_i^{virt} virtuális expozíció nem lesz más, mint az a (valós) expozíció érték, amely esetén ugyan akkora intenzitást mértünk volna tárgy nélküli elrendezésben, mint jelen esetben a leképezendő tárggyal. Ezzel a jelöléssel kapjuk a (6) egyenletet, amely immáron általánosan igaz, egy tetszőleges felvételre – míg az (5) egyenlet csak arra az esetre érvényes, amikor nincs semmi a detektor és a forrás között.

$$\begin{aligned} I_i^{\text{offset korrigált}} &= I_i^{\text{mért}} - I_i^{\text{offset}} \\ &= a_i^{\text{gain}} \cdot E_i^{\text{virt}} + b_i^{\text{gain}} \end{aligned} \quad (6)$$

A gain hibák javítása során azt szeretnénk elérni, hogy az egyes pixelek intenzitása ne függjön az adott pixel érzékenységtől, azaz az a_i^{gain} és b_i^{gain} együtthatóktól, csak az adott pixelt ért – virtuális – expozíciótól. Vagyis azt, hogy ha két detektorpixelt egy felvétel során ugyanolyan intenzitású sugárzás ér, és az expozíciós idő is megegyezik, akkor a két pixel intenzitása az elkészült képen is azonos legyen. Ezt úgy végezzük el, hogy először meghatározzuk az egyes detektorpixelek virtuális expozícióját, majd ezekből úgy alakítjuk ki a gain korrigált intenzitásértéket, mintha minden pixel érzékenysége megegyezne.

Először tehát kiszámoljuk az adott offset korrigált intenzitáshoz tartozó virtuális expozíció értékét a (6) egyenlet segítségével, amelyből egyszerű átrendezéssel kapjuk a virtuális expozícióra a (7) egyenletet.

$$E_i^{\text{virt}} = \frac{I_i^{\text{offset korrigált}} - b_i^{\text{gain}}}{a_i^{\text{gain}}} \quad (7)$$

A virtuális expozíció tehát azzal van összefüggésben, hogy mekkora valós fluxus érte az adott pixelt, és mentes az adott pixel érzékenységek torzító hatásától. Ebből

úgy határozzuk meg a gain korrigált intenzitás értéket, mintha minden pixel érzékenysége, azaz intenzitás – virtuális expozíció függvénye egy origón átmenő egyenes lenne, mindnek azonos meredekséggel. A szóban forgó meredekséget pedig úgy határozzuk meg, hogy éppen annál az expozíciós értéknél telítődjön, amelynél a legérzékenyebb pixel már telítődik. Azaz meg kell vizsgálni, hogy az expozíciót növelve mikor kapunk olyan képet, amin már van szaturált pixel. Ezen expozíciót hívjuk szaturációs expozíciónak (E^{szat}). Ezen érték a kalibráló adatsorból szintén meghatározható paraméter. Ezen paraméter ismeretében a gain korrigált $I_i^{\text{gainkorrigalt}}$ pixelintenzitás a (8) egyenlettel számolható, ahol $(2^{14} - 1)$ a detektor által mérhető maximális intenzitásérték.

$$I_i^{\text{gainkorrigalt}} = E_i^{\text{virt}} \cdot \frac{2^{14} - 1}{E^{\text{szat}}} \quad (8)$$

A módszer hatékonyságát – azaz azt, hogy a megfelelő adatsorokra valóban jól illeszkedik egyenes, valamint, hogy az így meghatározott korrekciós faktorokkal valóban hatékonyan lehet javítani a képek minőségét – a kísérleti algoritmus létrehozása során Tölgyesi részletes tesztekkel vizsgálta [2], ezért én ettől eltekintek. Ugyanakkor megjegyzem, hogy a korrekciós faktorokat külön meg kell határozni minden feszültségértékre, amin a röntgenforrást működtetjük. A feszültség módosításával ugyanis megváltozik a detektorpixeleken a sugárzás elnyelődésének valószínűsége, azaz változik a pixel érzékenysége.

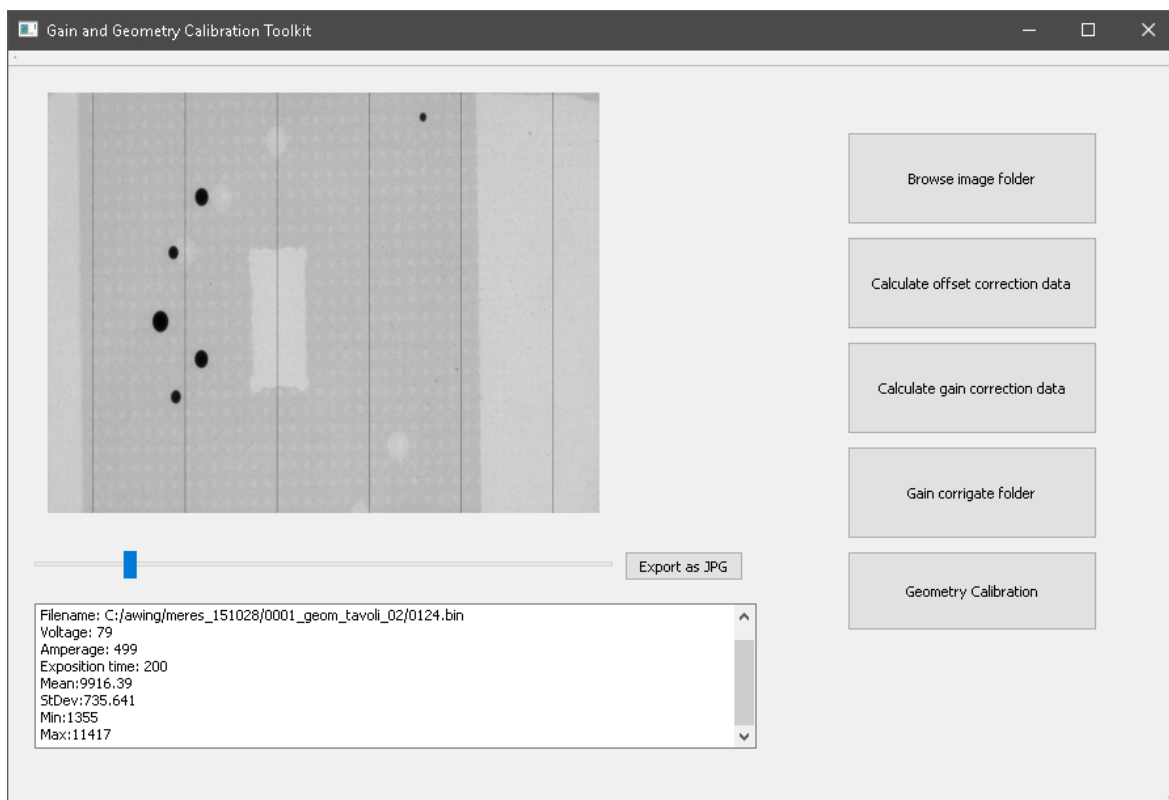
A továbbiakban azt mutatom be, hogy ezeket az algoritmusokat hogyan építem bele az elkészült szoftverbe, illetve milyen mérési utasítást ajánlok a korrekciók elkészítéséhez.

4.3. A gain kalibráció használata

A kalibrációs szoftver létrehozásakor szempont volt, hogy annak sebessége gyors, kezelése egyszerű legyen, így létrehoztam egy grafikus felülettel rendelkező programot, amelyben a felhasználó nyomógombokkal tud választani az elérhető funkciók között. A felhasználói felület képe az 5. ábrán látható.

A gain korrekcióval kapcsolatos funkciók (offset és gain korrekciós faktorok meghatározása, mappa gain korrigálása), elérhető a geometriai kalibrációs, valamint a képnézegető mód. Az ábrán ez utóbbi aktív, a felhasználó épp egy geometriai kalibrációs képeket tartalmazó mappát böngész. Az alsó részen a csúszka segítségével lehet lapozni a képek között, alul a képekre vonatkozó adatok láthatóak.

Az offset vagy gain korrekciós faktorok meghatározására szolgáló funkciót választva a szoftver egy új ablakban kéri a felhasználót, hogy válassza ki az elkészített kalibrációs mérések mappáját, valamint azt a mappát, ahova a kimeneti fájlokat menteni szeretné.

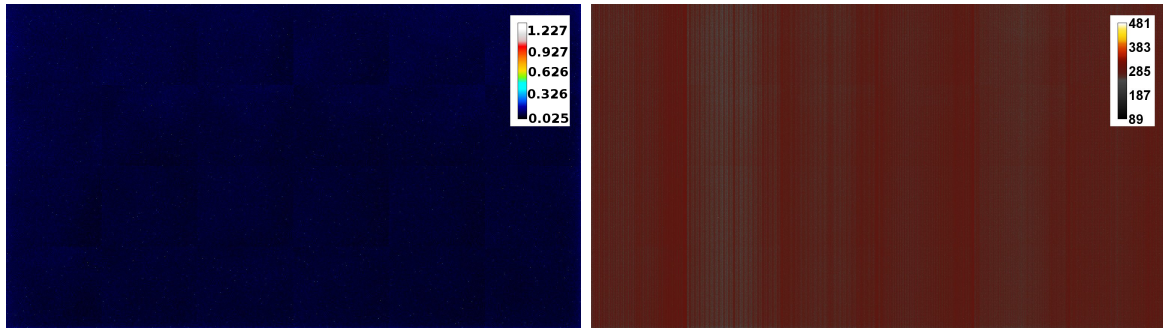


5. ábra. A létrehozott szoftver grafikus felülete. Az ábrán épp a képnézegető mód aktív.

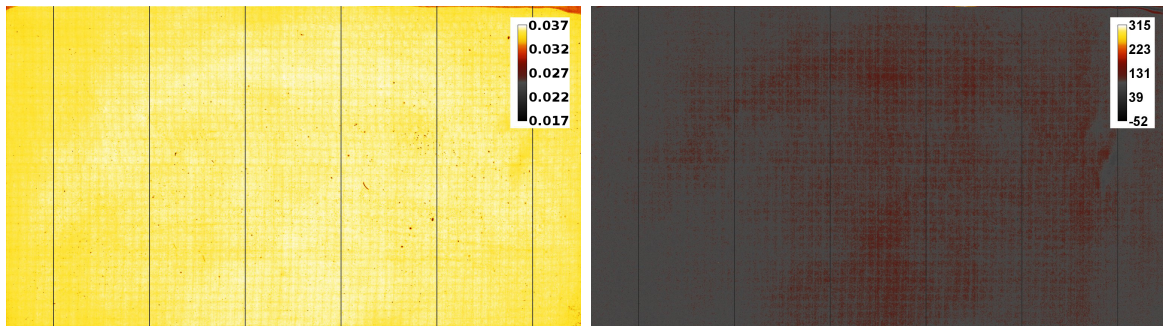
Ez után a korrekciós faktorok számolása automatikusan megtörténik, minden olyan feszültségértékre, amely feszültség értékekkel megfelelő mennyiségű és minőségű kép található az adott mappastruktúrában (lásd alább). Az esetleges hibákról a felhasználó értesítést kap a konzolon.

A korrekciós faktorok pixelenként kerülnek meghatározásra, így azok, mint egy együttható-térkép, képként kerülnek mentésre. A diplomamunkához használt gain korrekciós faktorok 50kV-os feszültségen készült képekhez tartozó együttható térképét a (6). és a (7). ábrán láthatóak. A struktúrák jobb láthatósága érdekében az egyes képek ábrázolásához különböző színtérképeket használtam. Az ábrákon látjuk azt, hogy a detektor struktúrája valóban nagy hatással van az egyes pixelek érzékenységére.

Ezekkel a korrekciós faktorokkal a felhasználónak lehetősége nyílik az elkészített projekcióit offset és gain korigálni, az erre vonatkozó (4) és (8) képleteket felhasználva. A felhasználónak ennél a funkciónál is grafikus felületen van lehetősége kijelölni a korigálni kívánt mappát, amelyből a program az összes képet gain korigálja és elmenti egy kívánt, kijelölt mappába – akár ugyan abba.



6. ábra. Offset hibák korrekciójára szolgáló együttható-térkép. Balra a korrekciós egyenes tengelymetszete, jobbra a meredeksége [ms^{-1}].



7. ábra. Gain és flatness hibák korrekciójára szolgáló együttható-térkép. Balra a korrekciós egyenes tengelymetszete, jobbra a meredeksége [$(\mu\text{A} \cdot \text{ms})^{-1}$].

4.4. A gain kalibráció implementálása és a szükséges mérések leírása

A bemeneti fájlok elkészítéséhez, azaz a mérési utasításhoz szükséges megjegyezmem, hogy a mérések során észrevehető volt, hogy a detektor által készített képek egy része valamiért jóval sötétebb, mint az azonos körülmények között készült egyéb képek. Ez a hiba a képek kb. 5%-át érinti, és okát nem sikerült felderíteni. Más felhasználók is panaszkodtak hasonló hibára ennél a detektortípusnál. A korrekciós faktorok számításánál azonban kimondottan fontos, hogy ilyen képeket ne használjunk fel a számításokhoz – továbbá az ilyen képek kezelése egyéb helyzetekben is szükségszerűnek tűnik. Ennek érdekében a gain kalibrációhoz szükséges méréseket úgy végeztük el, hogy egy adott beállítással több képet készítettünk (átlagosan ≈ 100 képet beállításonként). Az azonos beállítással készült képeket külön mappákba mentettük, így a szoftver is ilyen formában várja azokat az adatok feldolgozása során. Ez a megoldás azért szerencsés, mert egy mappán belül átlagolva a képek intenzitásértékét, könnyen és egyértelműen kiszűrhető, ha valamelyik kép jóval sötétebb, a fent említett hiba miatt. Továbbá az azonos mappákban lévő, tehát azonos beállítással készült képeket átlagolva a képeket terhelő zaj is csökken. Így amikor a korrekciós faktorokat számolom, mappánként

átlagolom azokat a képeket, amelyek átlagos intenzitása nem tér el 10%-nál jobban a mappában lévő képek átlagos intenzitásának átlagától. Az program robusztusságát tovább növelve, a képek beolvasásakor vizsgálom, hogy azok milyen expozíciós beállításokkal készültek, és ez alapján is szelektálom a képeket, kiszűrve a mérés során elkövetett emberi hibákat. A mérésvezérlő szoftverről ugyanis tudni érdemes, hogy a felvételek készítésének elindítása a röntgen forrás bekapcsolásától függetlenül elindítható, és az indítás után sorozatosan készít képeket, amíg le nem állítják. Mérés közben a forrás feszültsége és áramerőssége szabadon változtatható, illetve a forrás ki- és bekapcsolható. A röntgenső független kezelése részben biztonságvédelmi célokat szolgál, azonban lehetőséget ad a felhasználónak, hogy a képek sorozatos kimentését elindítsa úgy, hogy a forrás nincs bekapcsolva, vagy a beállított paraméterei nem megfelelők. Ekkor gyakran előfordul, hogy a felhasználó a paramétereket menet közben állítja, és az addigi eredményeket nem törli ki. Ezt kiküszöbölendő, a gain korrekcióhoz készített kalibrációs képek feldolgozása során a képek átlagos intenzitásán kívül a program vizsgálja az átlagos áramerősség, feszültség és exponálási idő paramétereket is, és a kilógó értékkel készült képeket figyelmen kívül hagyja. Továbbá figyelembe veszi azt is, hogy offset korrekcióhoz készült kalibráló képek között ne legyen olyan, ahol a forrás be volt kapcsolva – vagy ha van, akkor hagyja azt is figyelmen kívül. Hasonlóan a gain és flatness hibák faktorai meghatározásánál az algoritmus ignorálja azokat a képeket, ahol a forrás ki volt kapcsolva. Amennyiben egy mappában kevés kép van, amely a fenti szűrőknek megfelel (5-nél nem több), akkor a mappa tartalmát a szoftver nem használja fel a korrekciós faktorok meghatározásához. Továbbá, amennyiben túl kevés mappát talál a szoftver (5-nél nem többet), tehát túl kevés a különböző expozíciós beállítások száma a korrekciós faktorok meghatározásához – akár összesen, akár túl kevés megy át a fenti szűrükön –, a kalibrációs faktorok nem kerülnek meghatározásra. Így elkerülhető, hogy túl kevés pontra illesszünk egyenest, és ezért hibás eredményeket kapjunk, vagyis a képkészlet ellenőrzése szintén a szoftver robusztusságát növeli. A felvételek elkészítésekor alkalmazott feszültségértékeket az program szintén a képek beolvasásakor határozza meg, így azt külön megadni nem szükséges, mint ahogy a képeket feszültség szerint sem kell szétválogatni – ellentétben a kísérleti algoritmussal.

A képek beolvasása és mappánkénti átlagolása után a pixelenkénti egyenes illesztést a szokásos, egyenes illesztésére vonatkozó (9) képleteket felhasználva oldottam meg, ahol az x és y mérési pontokra illesztjük az α és β paraméterekkel jellemzett egyenest. A képletben a felülvonás az átlagolást jelenti.

$$\beta = \frac{\overline{xy} - \bar{x} \cdot \bar{y}}{\overline{x^2} - \bar{x}^2} \quad (9)$$

$$\alpha = \bar{y} - \beta \cdot \bar{x}$$

Jelen esetben x értéke az expozíciós idő (offset hibáknál), vagy az expozíció (gain és flatness hibáknál), y pedig az adott pixel intenzitása. Az egyenes illesztésére egyszerű és gyors megoldásnak találtam, hogy az illeszteni kívánt képeket pixelemként összeadom, valamint ugyanezt megteszem az adott expozícióval – vagy expozíciós idővel – megszorozva is, majd ezeket a képeket pixelenként elosztom az összeadott képek számával. Így képek összeadásával és skalárral való osztásával az \bar{x} és \overline{xy} értékek pixelenként előállnak, mint egy-egy kép. Ez után a kettő különbségét – mint képet – leosztva az $(\overline{x^2} - \bar{x}^2)$ skalárral, közvetlenül előáll a pixelekre illesztett egyenes meredeksége, és a tengelymetszet további képkivonással kapható. Mivel a szükséges képműveletek – összeadás, skalárral szorzás, kivonás – könnyen párhuzamosíthatóak, gyakorlatilag az egyenes illesztése is párhuzamosan történik, így jelentősen lerövidítve a számoláshoz szükséges időt. A módszer egyébként a memóriával is takarékosabban bánik, mint a kísérleti algoritmus, amely először beolvassa az összes képet a memóriába, majd pixelenként illeszt egyenest. Az ottani megoldással ellentétben, ahol a szükséges memória lineárisan nő a képek számával, nálam a szükséges memória a képek számától független és alacsony.

A fenti megoldáshoz a grafikus felületet és a mappakezelést a Qt biztosítja, míg a gyors képműveletek CUDA kód alkalmazásával valósulnak meg. A képfeldolgozáshoz létrehoztam olyan kép osztályt, amely képes magát fájlból a GPU memóriájába olvasni, és alkalmas a képeken végzett műveletek gyors végrehajtására a grafikus kártyán. A pixelenkénti összeadás, kivonás, skalárral szorzás megoldása viszonylag triviális, itt egy szál egy pixel értékét számolja. Ezzel ellenben a maximum, minimum és átlagos intenzitás számolása első ránézésre nem triviális, ezért néhány mondatban bemutatom.

Az előbb felsorolt műveletek, amelyek eredménye előáll a kép pixeleiből úgy, hogy minden egyes pixel értékét egyszer használjuk fel, egy kommutatív, asszociatív, két elem között ható – programozási értelemben vett – operátorral, az angol irodalom *reduce* műveletnek hívja. Ilyen az előbb említett összeadás, minimum és maximum számítás is. Például három elem, a , b és c összegét megkapjuk, ha a és b összegéhez hozzáadjuk c -t: $\text{sum} = (a+b)+c$. Az összeadást kicserélhetjük maximum számításra is, az algoritmus ekkor csak a számok közé rakott operátorban tér el: $\text{maximum} = \max(\max(a, b), c)$. Az ilyen, *reduce* típusú műveletek optimális használatát az összeadás példáján keresztül írom le, de a fentiek értelmében a minimum és a maximum számolás is hasonlóan kivitelezhető.

Az alkalmazásomban is használt *reduce* műveletet a CUDA hivatalos dokumentációjához mellékelt példa [7] alapján állítottam össze. Az ilyen műveletek általános megoldása – az összeadás példáján –, hogy a képet két részre osztjuk, majd az egyik részt hozzáadjuk a másikhoz. A műveletet folytatjuk, mindig felezve a fennmaradó kép-

részletet, míg végül a kép egyik pixelében előáll az összes pixel összege. A már említett példa ezen túl olyan, CUDA specifikus technikákat is ajánl a művelet gyorsításához, mint a *shared memory*, azaz az összes szál által hozzáférhető (de kívülről nem hozzáférhető), gyors elérésű memória használata – a lassú elérésű, globális memória helyett, ahol a képek eleve tárolva vannak. Továbbá figyelembe veszi, hogy a párhuzamos futtatás során a szálak 32 darabos „csomagokban”, úgynevezett *warpokban* futnak, ezzel tovább optimalizálva az algoritmust. Abban az esetben ugyanis, ha több, mint 32 szál futtatunk egyszerre, semmi nem garantálja, hogy azok egyszerre fognak futni. Ilyenkor időről időre be kell várnia a szálaknak egymást. Például ha a kép egyik felét hozzáadjuk a másikhoz, csak akkor felelhetjük tovább az elkészült képet, ha minden összeadás megtörtént. Amikor már 32-nél kevesebb szál használunk, akkor a szálaknak nem kell egymást bevárni, hiszen biztosan nem fut több. Továbbá a példa alapján az összeadáshoz vonatkozó kerneleket C++ template-ekkel írtam meg, amely segítségével már fordítási időben ismertté válik, hogy adott blokkmérettel futtatva a műveletet, hol kell a képet elfelezni – nem pedig futási időben kell *if* műveletekkel vizsgálnia, hogy a blokk mekkora méretű. A hivatkozott példa alapján a *reduce* műveletek ilyen technikákkal való megvalósítása akár 6–30-szor gyorsabbak, mintha ezek nélkül valósítanánk meg a műveletet. Ez a program futási sebességére igen jótékony hatással van, hiszen a kép pixeleinek összeadására – átlag számoláshoz – szinte minden megnyitás után szükség van, hogy eldöntsük, hogy az az említett ismeretlen hiba miatt elsötétült e, vagy sem.

5. Geometria kalibráció

A diplomaunkám során a másik fő feladatomból volt a az úgynevezett *geometria kalibráció* robusztus algoritmusainak megalkotása, amely segítségével a CT-vel elkészített képek alapján állapíthatjuk meg, hogy milyen távol van egymástól a detektor, a forgatható asztal forgástengelye és a röntgenső fókuszpontja. A geometriai adatokra igen pontosan szükség van, amikor az elkészült projekciók alapján meghatározzuk a felvett tárgy 3 dimenziós képét, vagyis a visszavetítéskor. Ugyanakkor a geometriai paramétereket kézzel nem tudjuk a szükséges pontossággal lemérni, már csak azért sem, mert a röntgenső fókuszpontjának a helye nem hozzáférhető és nem ismert, továbbá a mérési pontatlanság is viszonylag nagy lenne. Ezért alkalmazzuk azt a módszert, hogy a berendezéssel készített képek alapján határozzuk meg a szóban forgó paramétereket.

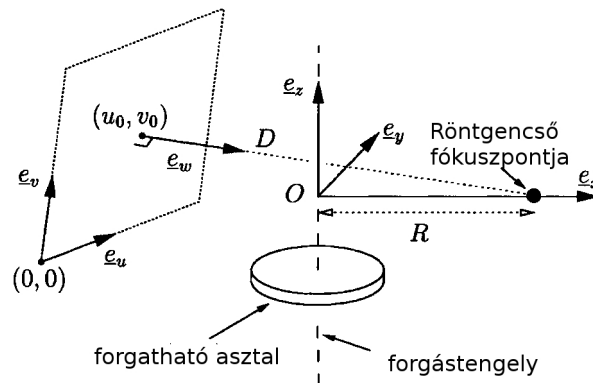
A kalibrációt két cikk alapján végeztem el. Noo cikke [9] valamivel bonyolultabb módon, kevés adatból határozza meg a geometriai paramétereket, míg Wu [10] cikke részben az előbbire támaszkodva, azonban azt leegyszerűsítve, és több adatsorból határozza meg a távolságokat, a szerzők szerint pontosabban, mint a Noo féle megoldással. A kí-

sérleti algoritmus ezért is csupán utóbbival dolgozik. Én azonban megvizsgáltam, hogy hibaszámítással alkalmazható e hatékonyan a Noo által leírt, kevesebb elhanyagolást tartalmazó módszer, valamint hibaszámítással javítottam a Wu által vázolt módszeren is, továbbá a kísérleti algoritmushoz képest fundamentálisan más megoldást vezettem be a képek feldolgozásához.

A továbbiakban bemutatom a geometriai kalibráció meghatározásának elméletét a cikkek alapján, majd bemutatom, hogy a geometriai adatok meghatározásához milyen módszereket használt a kísérleti algoritmus, valamint az én algoritmusaim, továbbá ismeretem az elért eredményeimet.

5.1. A síkpaneles CBCT geometriája

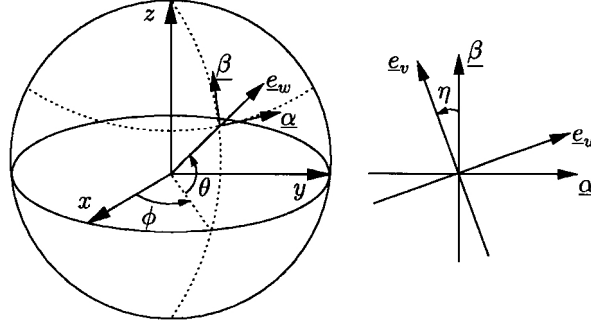
Hogy a geometriai kalibráció során milyen adatok kerülnek meghatározásra, és milyen adatok kerülnek elhanyagolásra, azt jól szemlélteti a (8). és a (9). ábra.



8. ábra. A Cone-beam CT elrendezésének vázlata [9]. A z tengellyel párhuzamos forgástengely R távolságra van az x irányban lévő forrásponttól. A detektor és a forrás közötti legrövidebb távolság D . A forrásoz legközelebb eső pontja a detektornak az u_0, v_0 koordinátával jellemzett pixel.

A 8. ábrán látható paraméterek jelentése a következő. A forgatható asztal forgástengelye és a röntgenső fókuszpontjának távolsága R . Konvenció szerint a forgástengely a z tengellyel párhuzamos és az x tengely irányában található a forrás fókuszpontja. A detektor az (u_0, v_0) koordinátákkal jellemzett pixelétől mért távolsága a legkisebb a forrástól, és ez a távolság D .

A 9. ábrán azt láthatjuk, hogy míg a forgástengely és a forrás iránya a modell szerint rögzített, a detektor irányát és állását a ϕ, θ és η szögeket jellemzik. Ezek közül az η szöget, vagyis a detektornak a saját síkjában való elfordulását mindkét cikk figyelembe veszi és számolja. A ϕ szöget, vagyis a detektor kifordulását, az asztal forgástengelyével párhuzamos tengely mentén, már csak Noo [9] cikke számolja ki. A θ szöggel jellemzett



9. ábra. A Cone-beam CT elrendezésében jellemző irányok [9]. e_w a detektor síkjára merőleges irány, α és β a detektor síkjában fekvő, egymásra merőleges vektorok. A detektor dőlését a ϕ , θ , és η szögek jellemzik.

elfordulást pedig mindkét cikk elhanyagolja. Az elhanyagolásokra az ad lehetőséget, hogy az nem befolyásolja nagy mértékben a rekonstruált kép minőségét, illetve a többi paraméter mérési pontosságát [10].

Összefoglalva tehát meghatározandó paraméter a detektor és a forrás távolsága (D), a forgástengely és a forrás távolsága (R), a detektor pixelei közül a forráshoz legközelebbi (u_0, v_0) , valamint a detektor elfordulása a saját síkjában (η). Ezen kívül Noo módszerével meghatározható a detektor ferdeségét jellemző ϕ szög is, de ez egyrészt a kísérleti algoritmusban sem került meghatározásra, másrészt az elkészült rekonstrukciós kód sem veszi figyelembe.

5.2. A geometriai paraméterek meghatározásának elmélete

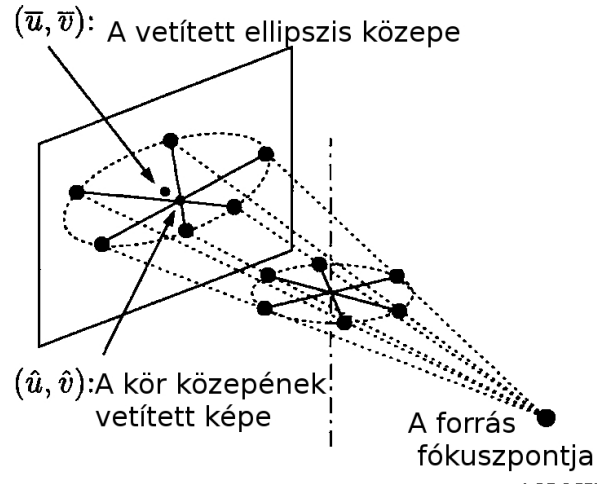
A fent összegzett paraméterek az említett cikkek alapján meghatározhatóak úgy, hogy a forgatható asztalra egy plexi – vagy egyéb, alacsony sűrűségű – lapra fémből – vagy egyéb, nagy sűrűségű anyagból – készült gömb alakú tárgyakat helyezünk el, egymástól jól meghatározott távolságra, majd az asztalt körbeforgatva felvételeket készítünk. Belátható [9], hogy a forgatás során körbeforduló gömbök a detektoron ellipszis pályát írnak le. A geometriai paramétereket ezen ellipszis pályákból tudjuk meghatározni. Ennek módját először Noo cikke alapján mutatom be, majd ismertetem, hogy ahhoz képest Wu cikke milyen változásokat tartalmaz.

5.2.1. A paraméterek meghatározása Noo [9] cikke alapján

Az adatok feldolgozása mindkét esetben azzal kezdődik, hogy az egyes projekciókon meg kell keresni, hogy a detektoron hol jelenik meg a golyók képe. A sorrendben k . golyó helyzetét az i . projekción (u_k^i, v_k^i) -vel jelölöm. Noo cikkében két golyót tartalmazó fantomot használnak, így k értéke 1 vagy 2 lehet. i értéke 1 és N között változik,

ahol N a teljes körbeforduláshoz alatt elkészített felvételek száma. Az egyes projekción az említett koordináták megkeresése nem egyértelmű probléma, és az említett cikkek sem adnak részletes segítséget a feladat elvégzéséhez. Így a golyók középpontjának megkeresésére nem itt, hanem a geometriai kalibráció megvalósítást tárgyaló az 5.4.3 fejezetben térek ki.

Miután a golyók középpontja az összes projekción meghatározásra került, elsőként a detektor saját síkjában való elfordulását kell korrigálni, azaz meghatározni az η szöget. Ehhez meg kell határozni a forgástengely detektorra vetített (\hat{u}_k, \hat{v}_k) képét a két ellipszis esetében. A 10. ábrán látható, ez a pont az épp ellentétes fázisban lévő pontok közé húzott szakaszok közös metszéspontjában van, amely nem feltétlenül esik egybe az ellipszis (\bar{u}_k, \bar{v}_k) középpontjával. A kereset (\hat{u}_k, \hat{v}_k) pont így a (10) egyenletrendszer alapján lineáris regresszióval kapható meg. A pontok ismeretében pedig a detektor elfordulása a (11) egyenlettel kapható.



10. ábra. A körbeforduló golyók vetített képe. A forgástengely vetített képe nem feltétlenül esik egybe a vetített ellipszis középpontjával. [9]

$$\left(u_k^j - u_k^i\right) \hat{v}_k - \left(v_k^j - v_k^i\right) \hat{u}_k = v_k^i u_k^j - v_k^j u_k^i, \quad j = i + \frac{N}{2}, \quad i = 1 \dots \frac{N}{2} \quad (10)$$

$$\eta = \arctan \left(\frac{\hat{u}_1 - \hat{u}_2}{\hat{v}_1 - \hat{v}_2} \right) \quad (11)$$

Az η szög meghatározása után az összes projekción a golyók vetített képeinek koordinátáját el kell forgatni ezzel a szöggel. Ez után a képek úgy kezelhetők, mintha egy ferdeségtől mentes detektorral készültek volna. Vagyis egy ellipszis (u_k^i, v_k^i) pontja helyett a (12) egyenletekkel számolt (u_k^{i*}, v_k^{i*}) pontot vesszük figyelembe a számolás további részében. Csupán arra kell odafigyelni, hogy a detektoron keresett (u_0, v_0) koordinátákat végül vissza kell forgatni .

$$\begin{pmatrix} u_k^{i*} \\ v_k^{i*} \end{pmatrix} = \begin{pmatrix} \cos \eta & -\sin \eta \\ \sin \eta & \cos \eta \end{pmatrix} \cdot \begin{pmatrix} u_k^i \\ v_k^i \end{pmatrix} \quad (12)$$

Ez után az elforgatott pontokra ellipszist kell illeszteni, ahol az ellipszis pontjai a (13) egyenletet elégítik ki.

$$a \left(u^{i*} - \bar{u} \right)^2 + b \left(v^{i*} - \bar{v} \right)^2 + 2c \left(u^{i*} - \bar{u} \right) \left(v^{i*} - \bar{v} \right) = 1 \quad (13)$$

A fenti egyenletből lineáris regresszióval meghatározhatóak az ellipszis (\bar{u}, \bar{v}) középpontja, az a és b paraméterek, amelyek a tengelyek hosszával vannak kapcsolatban, valamint c , amely paraméter az ellipszis ferdeségével hozható összefüggésbe. Ezekből a paraméterekből először a forrás és a tengely távolsága (D) a meghatározható, a (14) egyenletek alapján. Az újonnan meghatározott D távolság ismeretében a további meghatározandó paraméterek a (15) egyenletekkel számolhatóak. Az egyenletekben z_1 és z_2 a golyók z koordinátájára utal a fantomon. Amennyiben az adott golyó a fantomon a röntgenforrás vonala felett helyezkedik el, úgy előjele pozitív, amennyiben alatta, úgy előjele negatív. Ez az adat sem feltétlenül mérhető, így ezt is az elkészült felvételek alapján kell meghatározni. Továbbá a (14). egyenletekből az elsónél látható ϵ tényező értéke ± 1 : pozitív előjellel, ha a két golyó a röntgenforrás két oldalán helyezkedik el, vagyis ha $z_1 z_2 < 0$. Amikor a golyók ugyan azon az oldalon vannak, vagyis $z_1 z_2 > 0$, akkor ϵ értéke lehet $+1$, valamint -1 is: ilyen esetben az előjel megállapításához szükséges a felhasználó döntése a távolság mért értéke alapján. Az egyenletekben jelen lévő $n_0, n_1, m_0, m_1, \rho_k, \zeta_k$ változók segédváltozók a többi paraméter számolásához.

$$\left. \begin{aligned} D^2 &= \frac{(a_1 - 2n_0 n_1) - \epsilon \sqrt{a_1^2 + 4n_1^2 - 4n_0 n_1 a_1}}{2n_1^2} \\ n_0 &= \frac{1 - m_0^2 - m_1^2}{2m_0 m_1} \\ n_1 &= \frac{a_2 - a_1 m_1^2}{2m_0 m_1} \\ m_0 &= (\bar{v}_2 - \bar{v}_1) \sqrt{b_2 - \frac{c_2^2}{a_2}} \\ m_1 &= \sqrt{b_2 - \frac{c_2^2}{a_2}} \sqrt{b_1 - \frac{c_1^2}{a_1}} \end{aligned} \right\} \quad (14)$$

$$\left. \begin{aligned}
v_0^* &= \bar{v}_1 - \text{sign}(z_1) \frac{\sqrt{a_1 + a_1^2 D^2}}{\sqrt{a_1 b_1 - c_1^2}} \\
u_0^* &= \frac{1}{2} \bar{u}_1 + \frac{1}{2} \bar{u}_2 + \frac{c_1}{2a_1} (\bar{v}_1 - v_0^*) + \frac{c_2}{2a_2} (\bar{v}_2 - v_0^*) \\
\rho_k &= \frac{\sqrt{a_k b_k - c_k^2}}{\sqrt{a_k b_k + a_k^2 b_k D^2 - c_k^2}} \quad k = 1, 2 \\
\zeta_k &= D \cdot \text{sign}(z_k) a_k \frac{\sqrt{a_k}}{\sqrt{a_k b_k + a_k^2 b_k D^2 - c_k^2}} \\
\sin \phi &= -\frac{c_1}{2a_1} \zeta_1 - \frac{c_2}{2a_2} \zeta_2
\end{aligned} \right\} \quad (15)$$

Miután a fenti egyenletekkel meghatároztuk a geometriára jellemző D, v_0^*, u_0^* és ϕ paramétereket, végül már csak a forrás és forgástengely közötti R távolságot kell meghatároznunk. Ez a (16) egyenlet lapján történik, ahol d a két golyó – valós – távolsága a fantomon.

$$\frac{d^2}{R^2} = \frac{1}{N} \sum_{i=1}^N \left\{ \left(\zeta_1 \frac{u_1^i - u_0^*}{v_1^i - v_0^*} - \zeta_2 \frac{u_2^i - u_0^*}{v_2^i - v_0^*} \right)^2 + \left(\frac{D\zeta_1}{v_1^i - v_0^*} - \frac{D\zeta_2}{v_2^i - v_0^*} \right)^2 + (\zeta_1 - \zeta_2)^2 \right\} \quad (16)$$

5.2.2. A paraméterek meghatározása Wu [10] cikke alapján

A fentiekhez képest a legnagyobb eltérést az jelenti Noo és Wu cikke között, hogy utóbbi esetben a fantomon nem két golyóval történik a kalibráló mérés, hanem többel, ezzel növelve a módszer megbízhatóságát. Ezen kívül apróbb elhanyagolásokat tesz a szerző, amellyel a képletek jócskán leegyszerűsödnek.

A paraméterek meghatározása ebben az esetben is az η szög meghatározásával és a vele való korrekcióval történik, ám ebben az esetben – mivel több golyó adata áll rendelkezésre – egyenes illesztéssel tudjuk meghatározni $\tan \eta$ értékét, a (17) egyenlet szerint.

$$\hat{u}_k = c + \tan \eta \cdot \hat{v}_k \quad (17)$$

A mérési pontokra ez után szintén ellipszist kell illeszteni, azonban Wu cikkében a (18) egyenlet írja le a ellipszist. El van hanyagolva tehát a (13) egyenletben még meglévő c paraméter, amely akkor tehető meg, ha az ellipszis nagytengelye éppen vízszintes állású. Ez jó közelítéssel igaz is, hiszen az ellipszisen el vannak forgatva az előzetesen meghatározott η szöggel, amely következtében a forgástengely képe a detektoron egy függőleges egyenes. Mivel az asztal körbeforgatása során a golyók merőlegesen mozog-

nak a forgástengelyre, a vetített képükre is igaz lesz, hogy a pályájuk nagytengelye merőleges a forgástengely vetületére, azaz éppen vízszintes.

$$a \left(u^{i*} - \bar{u} \right)^2 + b \left(v^{i*} - \bar{v} \right)^2 = 1 \quad (18)$$

Ezzel az egyszerűsítéssel a keresett geometriai paramétereket leíró egyenletek is jócskán leegyszerűsödnek. A keresett v_0^* és D paraméterek a (19) egyenletekből lineáris regresszióval megkaphatóak, amely egyenletek a k és k' sorszámú ellipszisek között teremtenek kapcsolatot.

$$\frac{\bar{v}_k + \bar{v}_{k'}}{2} - \frac{1}{2(\bar{v}_k - \bar{v}_{k'})} \left(\frac{1}{b_k} - \frac{1}{b_{k'}} \right) = v_0^* + D^2 \frac{1}{2(\bar{v}_k - \bar{v}_{k'})} \left(\frac{a_k}{b_k} - \frac{a_{k'}}{b_{k'}} \right) \quad (19)$$

Vegyük észre, hogy Noo módszerével ellentétben itt nem szükséges a golyók függőleges pozíciójának (z_1, z_2) vizsgálata. Így ez a módszer minden esetben egyértelműen számolja a kérdéses paramétereket, míg Noo módszerénél az összes többi paraméter értékét meghatározó D paraméter bizonyos esetekben – amikor a kiválasztott golyók függőlegesen a forrás azonos oldalain voltak – nem volt egyértelmű. Továbbá ez a képlet jóval egyszerűbb, mint a (14) egyenletekben leírtak.

Ugyancsak egyszerű képletet kapunk u_0^* értékére: mivel a c paraméterek értékét elhanyagoljuk, a Noo által használt (15) képletek megfelelő sorára ránézve látszik, hogy u_0^* értéke az ellipszis illesztésnél kapott \bar{u}_k értékek átlagolásával kapható, vagyis a (20), ahol E az alkalmazott golyók – és így az ellipszisek – száma.

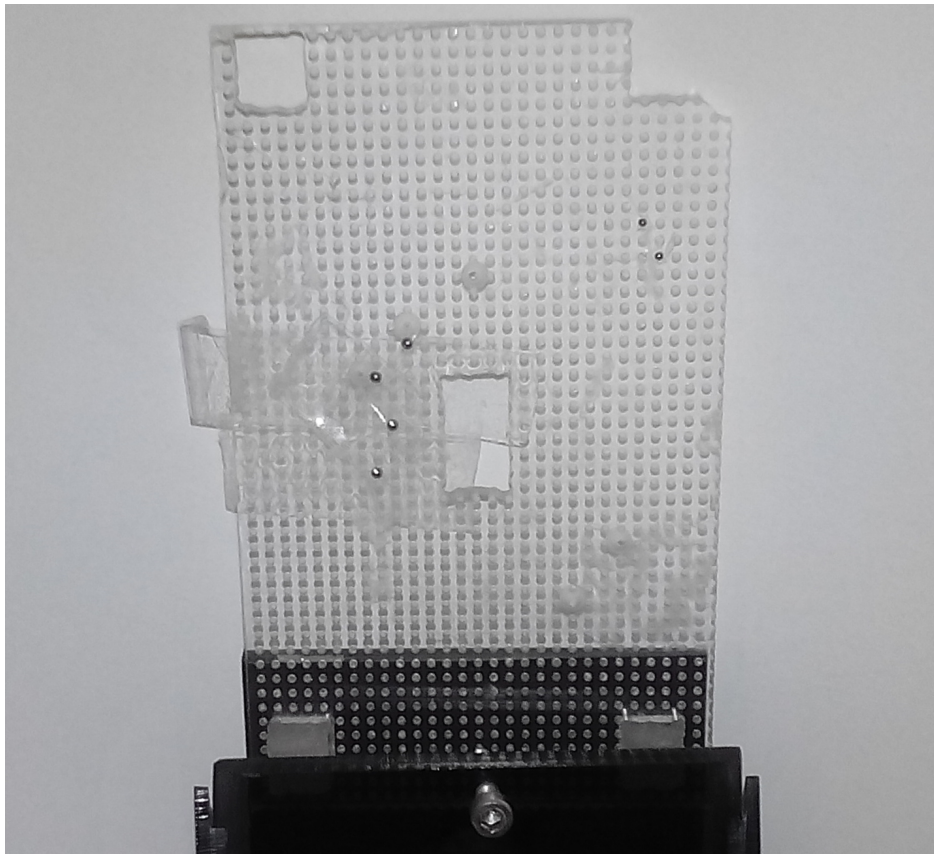
$$u_0^* = \frac{1}{E} \sum_{k=1}^E \bar{u}_k \quad (20)$$

Ez után az egyetlen meghatározandó paraméter a forgástengely és a forrás R távolsága. Ezt a távolságot most a (21) egyenletekkel tudjuk meghatározni, amelyek egy k és egy k' sorszámú ellipszis között teremtenek kapcsolatot, és ahol $d^{kk'}$ a két ellipszishöz tartozó golyók távolsága a fantomon. A képletben továbbá $(\alpha_k - \alpha_{k'})$ jelöli annak az elfordulásnak a szögét, amely elfordulással a két golyó az ellipszisük azonos fázisába forgathatóak. Ez a szög tehát 0° , amennyiben a golyók a plexi fantomon a forgástengely azonos oldalára vannak felerősítve, és 180° , amennyiben a fantom ellentétes oldalain vannak.

$$\left. \begin{aligned} \left(\frac{d^{kk'}}{R}\right)^2 &= (\zeta_k - \zeta_{k'})^2 + (\rho_k)^2 + (\rho_{k'})^2 - 2\rho_k\rho_{k'} \cos(\alpha_k - \alpha_{k'}) \\ \rho_k &= \frac{1}{\sqrt{(\bar{v}_k - v_0^*)^2 \cdot b_k}} \\ \zeta_k &= (\bar{v}_k - v_0^*) \frac{1 - (\rho_k)^2}{D} \end{aligned} \right\} \quad (21)$$

5.3. A geometriai kalibráló mérés kivitelezése

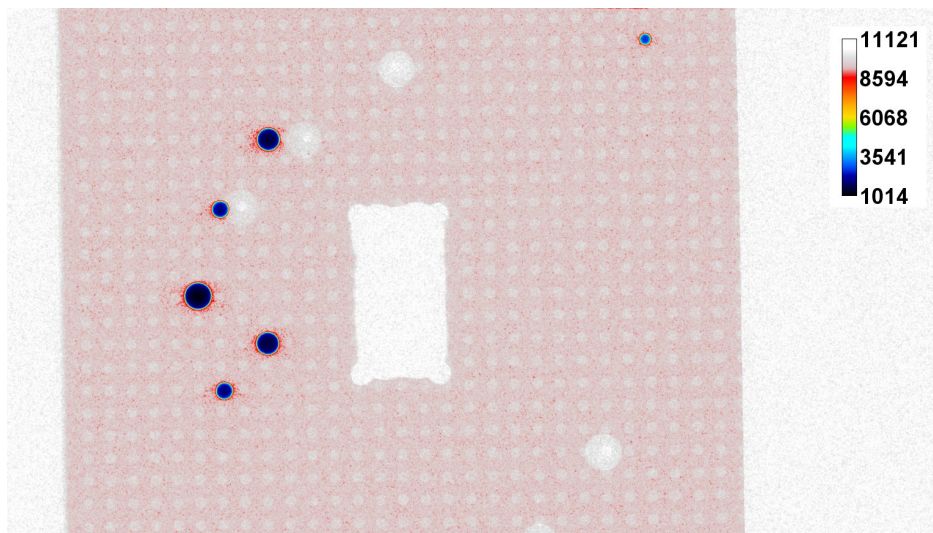
A geometriai kalibrációs mérésekhez különböző méretű csapágygolyókat használtunk, amelyeket cellux ragasztóval rögzítettünk egy plexi lapra, a 11. ábrán látható módon. Bár a megoldás nem a legprecízebb, lehetőséget ad rá, hogy a fantomot igény szerint átrendezzük. Erre akkor lehet szükség, ha például más nagyítás mellett a golyók nagy része kilógna a detektált képből.



11. ábra. A geometriai kalibrációhoz használt fantom

Az alkalmazott csapágygolyók átmérői 1 mm és 2,5 mm közöttiek voltak. A különböző méretű golyók alkalmazásával a kiértékelés robusztusságát kívántuk vizsgálni, azaz azt, hogy a golyók különböző méretei nehézséget okoznak-e a középpontjuk megtalálásakor. A plexin 2 mm-enkénti bemélyedés segítette a pontos pozicionálást, a golyókat

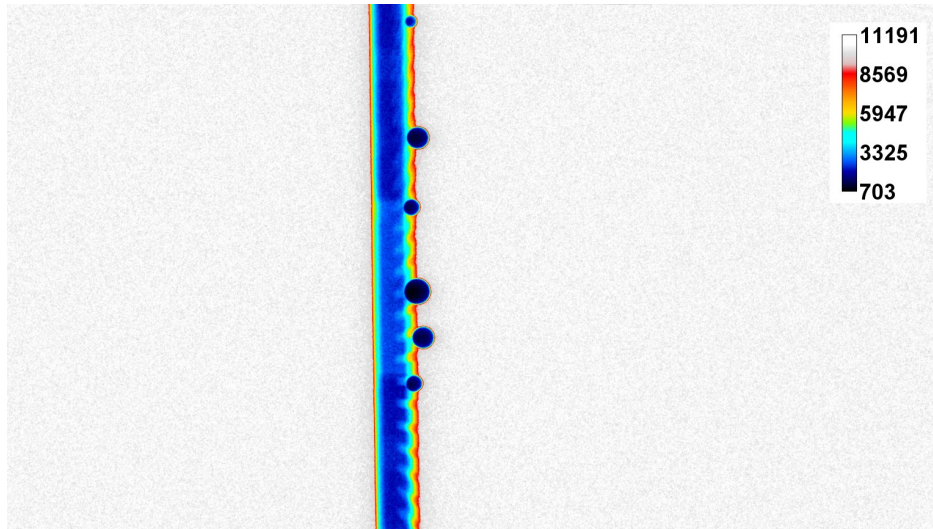
ezekbe a lyukakba ragasztottuk bele. Továbbá ezen barázdarendszer a golyók távolságának leolvasását is megkönnyítette. A méréseket a lehető legmagasabb csőfeszültség mellett, 80 kV-on végeztük, a fantomot körbeforgatva. A magas feszültségre azért volt szükség, hogy a plexi minél kevésbé legyen látható az elkészült felvételeken, ezzel segítve a golyók megtalálását a képen. Ugyanakkor a magas feszültség ellenére is, amikor a lemez síkja épp párhuzamos volt a fősugárral – tehát a detektoron a 11. ábrán látható elrendezés 90° -kal való elforgatottja jelent meg, a plexi lemez a fém golyókkal összemérhető intenzitást adott, vagyis a golyók elkülönítése a plexitől ekkor nehezebb feladat, mintha a golyók intenzitása jelentősen eltérne a plexiétől. Egy ilyen helyzetben és egy 90° -kal elforgatott helyzetben készült ábrázol a 12. és a 13. ábra. Láthatjuk rajta, hogy a fősugárra merőleges állásban a golyók intenzitására 1000–2000 körüli érték adódik, míg a plexinél ez az érték 10 000 körüli. Elforgatott állásban a plexi és a golyók intenzitása is 1000–2000 körüli érték, vagyis ekkor már a plexi elsötétülése megegyezik a golyókéval. Ez a jelenség az algoritmusok tervezése során egy megoldandó problémát jelentett, amelyre az 5.4.3. fejezetben térek ki.



12. ábra. A geometriai kalibrációhoz használt fantomról készült felvétel, miközben a fantom síkja merőleges a fősugárra.

5.4. A geometriai kalibráció megvalósítása

A geometriai kalibráció robusztus algoritmusainak megalkotása során rendelkezésemre állt a kísérleti algoritmus során eredményesnek bizonyult megvalósítás [2], ami lényegében a [10] cikken alapult, valamint felhasználtam a [9] cikkben leírtakat is. Míg utóbbi csak két golyó adataival dolgozik, kevesebb elhanyagolást alkalmaz. Előbbi azonban egyszerűbb képleteket használ, eleve több golyós fantomot feltételezve, amely gyakran lineáris regresszióvá alakítja a képleteket, felhasználva egyszerre az összes golyó



13. ábra. A geometriai kalibrációhoz használt fantomról készült felvétel, miközben a fantom síkja párhuzamos a fősugárral.

adatait. Mindkét cikk algoritmusát implementáltam a programomban, megvizsgáltam, hogy vajon a kevesebb elhanyagolást eredményező változat robusztussá tehető e, ha a több golyó adatait megfelelő hibaszámítással súlyozva átlagolom. A hibaszámítást a másik algoritmus megvalósítása során is bevezettem, így mindkét módszernél kezelem az esetleges hibákat a golyók keresése és az ellipszisek illesztése során. Továbbá új algoritmust vezettem be a golyók középpontjának megkeresésére, amely a felhasználótól kevesebb információt kér, pontosabban eredményt ad, valamint a videokártyán való megvalósításnak köszönhetően gyors működésre képes.

5.4.1. A folyamat áttekintése és a felhasználótól kért adatok.

Az általam megvalósított kalibrációs szoftver újabb funkciója a geometriai korrekciós mérések kiértékelése. A funkció kiválasztásakor először beolvasásra kerülnek a gain kalibráció adatai, amely helyének kiválasztására a felhasználónak grafikus felületen van lehetősége. Szintén grafikus felületen választható ki az, hogy hol keresse a program a geometriai kalibrációhoz szükséges képkészletet. A mappában lévő képek ezek után egy csúszka mozgatásával átnézhetőek, továbbá szintén grafikus felületen várja a program a szükséges felhasználótól bekérendő adatok megadását. Ezek az adatok a következők:

- Első felhasználni kívánt kép sorszáma.
- Azon golyók száma, amelyek az összes képen rajta vannak.
- Azon golyók száma, amik nincsenek rajta az összes képen.
- Az első képpel azonos állású, 360°-kal elforgatott kép sorszáma.

- A forrás és a detektor becsült távolsága mm-ben.
- A fantomon a golyók egymástól vett távolságát.

Az első kép kiválasztására azért van lehetőség, mert a felhasználó hibázhat a mérés elindításakor: nem kapcsolja be a röntgen forrást, nem indítja el a motort, stb. A keresendő golyók száma szintén fontos, a golyók keresésénél felhasznált paraméter. Az általam megvalósított algoritmusnak segítséget ad az is, hogy tudja, hogy bizonyos képeken több golyó lehet a képen, mint az elsón – például mert a kép alsó és felső szélén lévő golyók vetülete a detektoron kívülre eset és így a képről eltűnhet a forgatás hatására. Ennek megállapításához a felhasználónak eleve át kell néznie azon képeket, amik egy teljes körbefordulás alatt készültek, így arra is megkéri a program, hogy a csúszkát ez után hagyja egy olyan állapotban, amely nagyjából 360° -kal van elforgatva az első képhez képest. Itt nem szükséges pontosan a teljes körbeforduláshoz tartozó projekciót megadni, a felhasználó ettől eltérhet $\pm 45^\circ$ -kal. A cél itt az, hogy a felhasználó jelölje ki, körül-belül hol kell keresni a teljes körbefordulást, hogy annak megkeresése rövidebb időt vegyen igénybe, mintha az összes projekciót át kellene nézni. Szükséges továbbá a Noo féle számításához megadni a detektor és a forrás távolságát. Erre azért van szükség, mert ahogy az 5.2.1 fejezetben említettem, a Noo féle módszernél ezen távolság kiszámítása azon esetekben, amikor a két kiszemelt golyó a forgástengely azonos oldalán vannak, nem egyértelmű. Ennek eldöntését segíti a felhasználó által megadott paraméter, amely szintén nagyságrendi becslést kell, hogy adjon, tehát pár centiméteres hiba még megengedett. Ez után egy újabb grafikus felületen kell megadni a fantomon a golyók helyzetét, amelyből a távolságuk kiszámításra kerülnek.

Amennyiben ez megtörtént, a felhasználó ellenőrizheti, hogy az algoritmus jó helyen találta-e meg a golyókat a kezdeti képen, illetve, hogy a teljes körbefordulás utáni kép egyezik-e az elsőnek választottal. Amennyiben bármelyikben hibázott a program, lehetőség van visszalépni, és újabb kezdeti projekciót választani. A grafikus felület egy ilyen állapotát mutatja meg a 14. ábra.

A kijelölt projekció módosítására egyébként a diplomamunkám elkészítése során nem kellett sort kerítenem, az első képen is biztosan felismerte a golyók helyzetét a program és a körbefordulás utáni képet is stabilan megtalálta. Az esetleges hibás felismerések elkerülése végett ugyanakkor érdemes olyan projekcióról indítani a kalibrációt, amelyen a plexi elsötétedése alacsony mértékű.

A kísérleti algoritmusnál megvalósítottakhoz képest ez a felület jóval felhasználóbarátabb, kevesebb információt kér a felhasználtól, kisebb a felhasználó hibázási lehetősége is – ahol a program számot vár, ott szöveget nem tud megadni, stb.

A program a felhasználó jóváhagyása után az alább ismertetett algoritmusokkal ki-



14. ábra. A geometriai kalibrációs funkció felhasználói felülete. A felhasználó a fenti mappát választotta ki a képek helyéül. A bal oldali képet választotta első képnek, az alsó részen pedig megadta a szükséges paramétereket. A szoftver meghatározta a teljes körbefordulás utáni projekciót és megkereste a két projekción a gömbök középpontját.

számolja a kalibrációs faktorokat, majd az eredményt közli a felhasználóval. A folyamat jó részét a képek feldolgozása, azaz a golyók középpontjának megkeresése teszi ki. A feldolgozás állapotáról a felhasználó egy folyamatjelző sávon tudja követni.

5.4.2. Teljes körbefordulás meghatározása

A szoftver első feladata, még a felhasználótól való adatbekérés során, hogy meghatározza, melyik projekció felel meg az elsőként kiválasztott projekció 360°-kal való elforgatottjának. Ehhez a felhasználó megad egy viszonyítási alapot: egy olyan képet, amely nagyjából megfelel a fenti kritériumnak. A szoftver ez után a két kép sorszámának különbségét elosztja 8-cal. A teljes körbefordulás képét a felhasználó által megadott kép körül ilyen távolságban keresi, a képek sorszámainak tekintetében. Így nem kell megvizsgálni a teljes mappa tartalmát, csupán annak egy részét, amely következtében a körbefordult kép megkeresése kevesebb időt vesz igénybe.

A teljes körbefordulást vizsgálatot úgy végzem, hogy kiszámolom az első kép és egy kiválasztott másik kép korrelációs együtthatóját, és megvizsgálom, hogy ez az együttható melyik kép választása esetén a legnagyobb. Ez a kép lesz a teljes körbefordulás utáni-

nak választott kép. A kiválasztott A és B kép C korrelációs együtthatóját a (22) képlet alapján számolom, ahol az összegzés a kép összes pixelére vonatkozik, a felülvonás az átlagolást jelenti, σ pedig a képek szórását jelöli. A műveletet grafikus kártyán, párhuzamosan végzem. Az átlagokat és a szórásokat pedig a művelet előtt, szintén grafikus kártyán párhuzamosítva végzem: az átlagoláshoz a pixelek értékeit össze kell adni, a szórás pedig a pixel értékeinek négyzetösszegéből származtatható. Így valójában mindkét művelet a 4.4 fejezetben bemutatott, könnyen párhuzamosítható *reduce* műveletet testesíti meg.

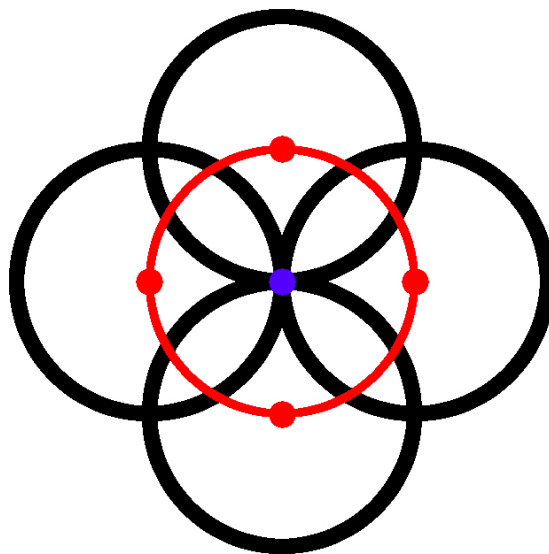
$$C = \frac{\sum_{i=1}^N \left\{ (A_i - \bar{A}) \cdot (B_i - \bar{B}) \right\}}{N \cdot \sigma_A \cdot \sigma_B} \quad (22)$$

A fenti módon számolt korrelációs érték 0 és 1 közötti értéket vehet fel: a kép önmagával való korrelációs együtthatója éppen 1. Ez alapján a szoftver vizsgálhatja azt is, hogy a megtalált kép eléggé hasonlít-e az elsőként kiválasztott képre. Amennyiben a korrelációs együttható 0,9 alatt van, vagyis az elsőként kijelölt képhez talált legnagyobb korrelációjú párja sem hasonlít eléggé az eredetire, a szoftver figyelmezteti a felhasználót.

5.4.3. A golyók középpontjának meghatározása

Amennyiben a felhasználó minden adatot megadott, megkezdődik a képek feldolgozása, amelynek során először meghatározásra kerülnek az egyes projekciókon a golyók koordinátái. Tölgyesi [2] munkája során erre két algoritmust implementált. Az egyiknél a kép kis részleteinek egy körlap képével való korreláció eredményét számolja, majd a kép azon pozícióban határozza meg a keresett kört, amely pozícióba helyezve a körlap képét, a korrelációs együttható maximális. Ez a módszer bizonytalannak bizonyult és ezért már a korábbi munka során el lett vetve. A másik módszere szerint a képeket először invertálja (hogy a nagy elnyelésű golyók helyén a kép intenzitása nagy legyen), majd golyók helyét a kép egy részletén belül az ott látható képpontok intenzitással súlyozott súlypontjába becsüli. Ennek az algoritmusnak is van hátulütője. Egyrészt megfelelő méretű ablakot kell választani, hogy a golyó épp beleessen a vizsgált tartományba és ennek az ablaknak a helyét minden projekciónál számon kell tartani. Kifejezetten hibás eredményhez vezethet ez az algoritmus akkor, amikor a golyókat épp a fősugárral párhuzamosan álló fantom képén próbáljuk megtalálni. Ekkor a plexi lap okozta gyengülés a 13. ábrán látottak alapján összemérhető, vagy akár nagyobb, mint a golyó által okozott. A plexi képe ilyenkor jelentős mértékben befolyásolni tudja a súlypontkeresést, akár téves útra térítve a golyók keresését. Ez ellen a vizsgált ablak csökkentésével lehet, de teljesen nem kiküszöbölhető.

A fenti hibák elkerülése érdekében a körök középpontjainak megkereséséhez egy új fajta algoritmust használtam fel: a Hough-transzformációt. Ez az igen széles körben használt módszer körök, egyenesek, illetve akár tetszőleges alakzatok lokalizálására használható. Körök keresése esetében először a képeken található éleket kell megkeresni, vagyis a képen található körök szélén lévő pixeleket. Majd a képre olyan körvonalakat kell rajzolni – ezeket a továbbiakban másodlagos köröknek nevezem –, amely köröknek középpontjai az eredeti körvonalról kerülnek kiválasztásra. Amennyiben a keresett kör sugara ismert, a másodlagos köröket is ilyen sugárral kell megrajzolni. Ekkor a rajzolt körök közös metszete adja meg a keresett, eredeti kör középpontját. Egy ilyen eljárást szemléltet a 15. ábra. Az ábrán lévő piros kör középpontját úgy találjuk meg, hogy a körvonal kijelölt pontjaiból újabb köröket rajzolunk – az ábrán fekete színnel. A legtöbb másodlagos kör által érintett – kék – pont adja meg a keresett kör középpontját. Természetesen érdemes a keresett kör összes pontja – vagyis a képen az összes él-pont – köré másodlagos köröket rajzolni, hogy minél pontosabb eredményt kapjunk. A módszer használható akkor is, ha a keresett kör sugara nem ismert. Ekkor ugyan azon pontokból különböző sugárral húzunk másodlagos körvonalakat. Mivel az eredeti körnek csak a középpontja van egyenlő távolságra a körvonal pontjaitól, a különböző sugarú másodlagos körök csak akkor fogják egymást egy pontban metszeni, ha a sugaruk éppen a keresett kör sugarával egyezik meg. Vagyis ismeretlen sugarú kör keresése esetén több féle sugárral kell másodlagos köröket rajzolni, és meg kell keresni, hogy melyik sugár alkalmazása esetén metszi egymást a legtöbb kör. Ekkor megtaláltuk a keresett kör sugarát, a metszések helye pedig megadja a keresett kör középpontját.



15. ábra. Körkeresés Hough-transzformációval.

A Hough-transzformációval tehát kimondottan a kör alakú objektumok keresésére

alkalmas, és emellett az algoritmus jól párhuzamosítható is, hiszen az élkeresés egy konvolúciós művelettel elvégezhető, amely után a képre az él pixelek közepéből húzott körvonalak szintén párhuzamosan rajzolhatóak. A továbbiakban bemutatom, hogy a körkeresést pontosan hogyan implementáltam az elkészült programban.

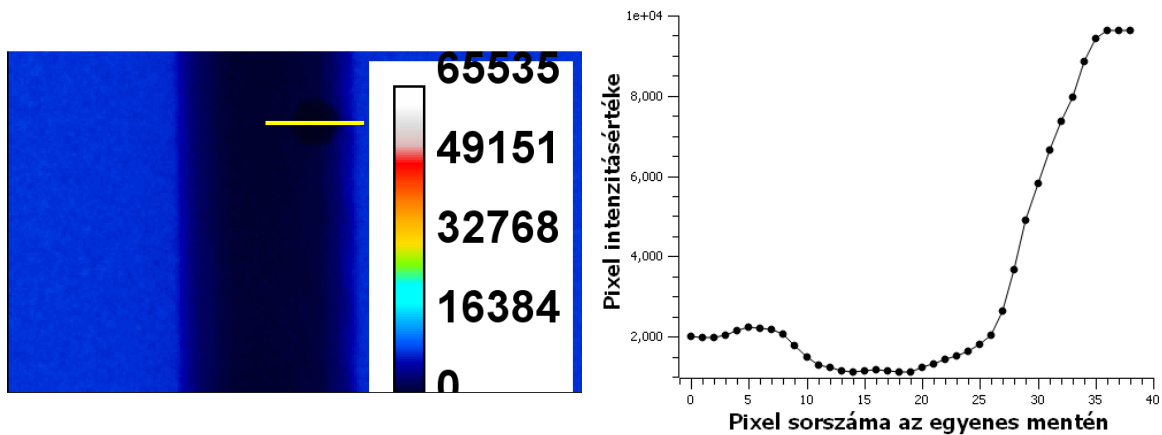
A körkeresés első lépés az élek megkeresése. Ez a művelet több féleképp is kivitelezhető, azonban a módszerek közös kiindulási alapja, hogy első lépésben a képet simítani szükséges. Ezzel el tudjuk kerülni, hogy a zajok okozta hirtelen intenzitás változások befolyásolják az élkeresést. Ez után kerülnek kiválasztásra az él-pixelek, amelyek kiválasztására két gyakori megoldás a következő.

- Kiszámoljuk a kép gradiensét, azaz azt, hogy egy adott pontban milyen gyorsan változik a kép intenzitása. Ez után különféle módokon határozhatjuk meg az él-pixelet. A legegyszerűbb esetben választhatjuk azokat a pontokat, ahol a gradiens értéke egy adott küszöbérték feletti.
- Hattatjuk a képre a Laplace-operátort, vagyis kiszámoljuk a kép második deriváltját. Ott jelöljük ki az éleket, ahol a kép Laplace-ának értéke előjelet vált, vagyis ahol szélsőértéke van a gradiensnek.

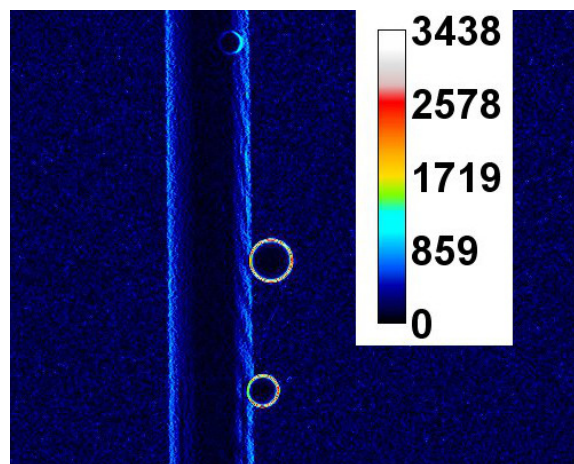
Előbbi jóval egyszerűbben számolható, hiszen a gradiens képen az adott pixel értéke eldönti, hogy él-pixel e vagy sem, hiszen csak azt kell vizsgálni, hogy értéke a kijelölt küszöbérték feletti, vagy sem. Az utóbbi módszernél vizsgálni kell az egyes pixelek környezetét, hogy kiderüljön, hogy az adott helyen a kép Laplace-a előjelet vált e. Vagyis ez a módszer számításigényesebb, ugyanakkor megbízhatóbb eredményt ad.

A helyes algoritmus megválasztása során annak minőségét tartottam szem előtt, a futási sebességgel szemben, így végül a számításigényesebb, Laplace értékekkel való élkeresést választottam. Ugyanis amikor az elkészült projekción a legkisebb csapágygolyó épp a plexi takarásában van, a gradiens módszer nem bizonyult hatékonynak. Ekkor ez a módszer nem tudta megkülönböztetni a golyó éleit a plexi élétől. Ezt a 16. ábrán szemléltetem. A bal oldali ábrán a golyó képét láthatjuk, jobb oldalon pedig a sárga vonallal kijelölt szakasz mentén vett intenzitásértékek grafikonját. A golyó helyén, azaz a szakasz 10–20 pixelei között az intenzitásértékek alig változnak. A 17. ábrán, amelyen ugyanannak a felvételnek a gradiens képe látható, észrevehetjük, hogy a felső golyó körvonala mentén a gradiens érték valóban nem kiemelkedően magas a plexi lemez határán mérhetőhöz képest, míg a nagyobb golyók határainál jóval magasabb gradiens érték figyelhető meg.

Gradiens alapon tehát nem sikerül jól elválasztani a golyól szélét a plexi lemez szélétől, ami a Hough-transzformáció során rossz eredményeket produkálhat, ugyanis így a plexi szélében lévő kisebb görbületek miatt a keresett kis csapágygolyó közepén lévő



16. ábra. A plexi takarásában lévő legkisebb golyó képe, valamint az intenzitás értékek változása a golyó vízszintes metszetében – a bal oldali képen sárga egyenessel kijelölve.



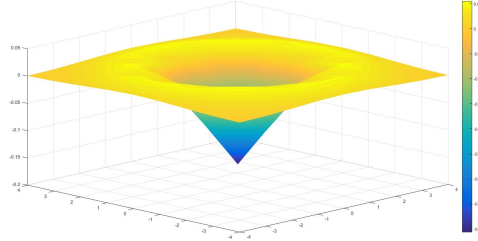
17. ábra. A plexi takarásában lévő és más golyók, a kép gradiensén.

Hough-transzformált érték gyakran alulmúlja a plexi lemez miatt máshol megjelenő értékeket. Így ezen speciális esetekben a Hough-transzformált nem a golyó középpontjánál lesz a legmagasabb értékű.

A gradiens módszer helyett ezért a Laplace-operátorral kapott kép null-átmenetei alapján választottam ki az él-pixelet. Ez alapján a végül megvalósított algoritmus a következő. A felvételeket gain korrigálás után egy $\sigma = 1,4$ paraméterű, Gauss-eloszlású mátrixszal konvolválva elmostam a zajok kiszűrése érdekében, valamint meghatároztam a kép Laplace-át. A két művelet egyszerre lehet elvégezni, hiszen a konvolúció tulajdonságai alapján elég a képet egy olyan mátrixszal konvolválni, amely a Gauss-eloszlású mátrix Laplace-operátort szimbolizáló mátrixával konvolválva kapjuk meg. Tehát a mátrix elemeit a Gauss-görbe második deriváltjából kell mintavételezni, amely függvényt a (23) egyenlet írja le. Az egyenletben, és a továbbiakban az angol irodalomban használatos *LoG* jelölést használtam a Gauss görbével simított kép Laplace-ának jelölésére, σ pedig a Gauss görbe szórása. Az egyenletből egy 9×9 méretű mátrixot al-

kottam meg, mint a LoG művelet konvolúciós mátrixát. Az elkészült mátrix grafikusan ábrázolva a 18. ábrán látható.

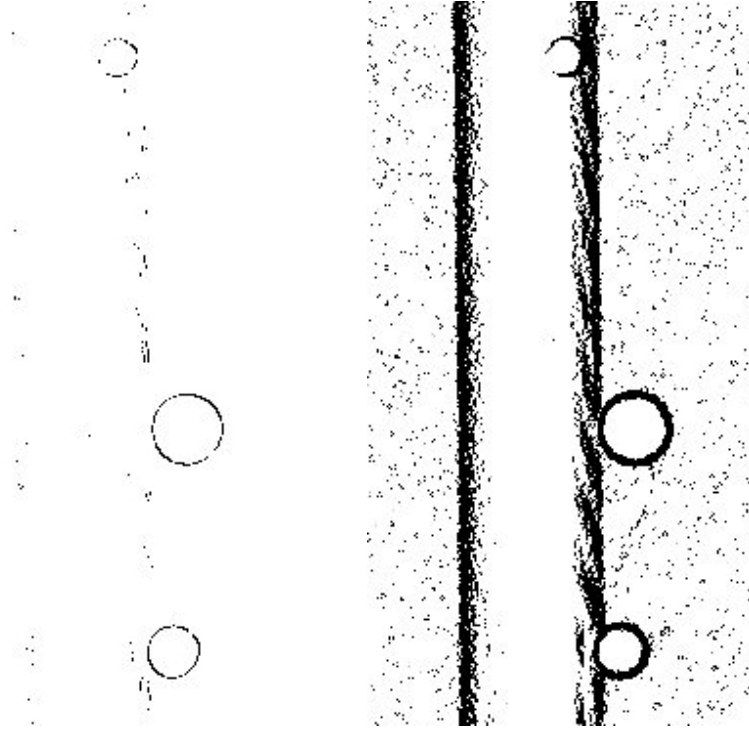
$$LoG = \frac{-1}{\pi\sigma^2} \cdot \left(1 - \frac{x^2 + y^2}{2\sigma^2}\right) \cdot e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (23)$$



18. ábra. $\sigma = 1,4$ paraméterű Gauss görbe Laplace-a.

Az élkeresésnél a gain korrigált felvételeket ezzel a mátrixszal konvolválva kaptam meg a simított kép Laplace-át, amelyen ez után a nullátmenetek helyét kerestem meg. Ennél a lépésnél minden pixelre megvizsgáltam, hogy az adott pixel függőleges, illetve vízszintes szomszédjainak előjele eltér e. Továbbá bevezettem egy újabb kritériumot is az él-pixelek keresésénél, mégpedig azt, hogy a szomszédos pixeleknek ne csak az előjele térjen el, hanem a nulla átmenet legyen elég erős, azaz az átmenet olyan ellentétes előjelű értékek között valósuljon meg, amelyek abszolút értéke nagyobb, mint egy kritikus érték. Ezt a kritikus értéket a LoG művelettel kapott kép szórásának a felének határoztam meg. Az ilyen módon meghatározott él-pixelek nagy része a golyók határát jelölte ki, mégpedig úgy, hogy a kisebb golyók szélének a nagy részét is megtalálta az algoritmus. Az eddigi felvételen személtetve ez a módszera 19. ábra bal oldali képén feltüntetett pixeleket találja meg élként. Ha a gradiens módszerrel szeretnénk olyan élképet kapni, amely a felső, kis méretű golyó területének nagy részén felismeri az éleket, akkor olyan küszöbértéket kell választani a gradiens képen, amellyel a jobb oldali képén látható pixeleket kapjuk élekként. Vegyük észre, hogy a gradiens módszer jelentős mennyiségű pixelt észlel élekként ott, ahol a plexi határa található, és a kép háttérében is zajos képet ad. Ezek a tényezők rontják a Hough-transzformált minőségét és növelik a futási idejét. Az eredeti képet jobban elmosva, azaz nagyobb szórású Gauss görbével konvolválva kaphatnánk ennél kevésbé zajos él-képet a gradiens módszerrel is, de ekkor azt tapasztaltam, hogy a kisebb golyók éle is kevésbé válik felismerhetővé.

A gradiens alapú élkeresést tehát elvettem, és így a Laplace alapú algoritmus-sal állítom elő az él-képet, amelyet utána Hough-transzformálok. Mivel a felvételeken a körök sugara nem ismert, és projekciónként és mérésenként is változóak, a Hough-transzformációt a következőképp valósítottam meg. Az él-pixelek köre különböző sugarú



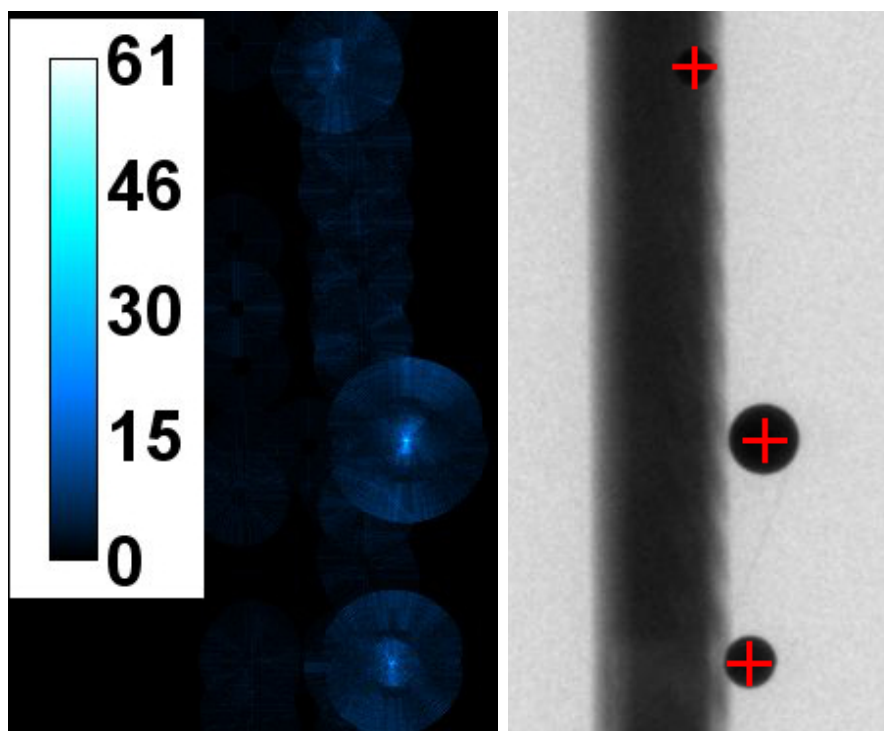
19. ábra. Élpixelek a szoftverben megvalósított módon Laplace-transzformálttal (balra), valamint gradiens küszöböléssel (jobbra).

köröket rajzolok, vagyis az eredeti képpel megegyező méretű képeket készítek, ahol a pixelek intenzitása helyett most a pixelek azt a számot tárolják, hogy hány, adott sugarú másodlagos kör metszi egymást azon a helyen. Ezeket a képeket a továbbiakban másodlagos képeknek nevezem. Többféle sugárral megrajzolva a másodlagos köröket, a kapott másodlagos képekből úgy alakítottam ki a végső Hough-transzformált képét, hogy minden pixelnek a másodlagos képeken felvett legnagyobb értékét adtam végső értékül. Vagyis a végső Hough-transzformált képen (HT) az i pixel értéke az r sugarú másodlagos körök rajzolatából kapott HT_i^r képekből a (24) egyenlet alapján keletkezik. A másodlagos képeket $r_{min} = 5$ és $r_{max} = 25$ közötti sugarú körök rajzolásával készítettem, amely a projekciós adatsoron elégséges tartománynak bizonyult. Amennyiben más nagyítás mellett készülnek a kalibrációs felvételek, vagy más a csapágygolyók mérete, a keresett körök sugara változhat, így ezen paraméterek esetlegesen módosításra szorulhatnak. A keresett körök sugarai a projekciós képekből azonban egyértelműen meghatározhatóak, és a megfelelő értékre állíthatóak.

$$HT_i^{vegso} = \max_{r_{min} \leq r \leq r_{max}} (HT_i^r) \quad (24)$$

Az így kapott Hough-transzformált kép egy részlete a 20. ábra bal oldalán látható. A képen a maximális intenzitásokat megkeresve meghatározhatóak a körök középpontjai. A megtalált középpontokat az eredeti képen piros kereszttel jelölve a jobb oldali ábrán

tüntetem fel. Jól látszik, hogy a köröket igen pontosan megtalálta az algoritmus.



20. ábra. A kép Hough-transzformáltja (balra) és a megtalált körök középpontjai az eredeti felvételen jelölve (jobbra).

A középpontok kereséséhez használt összes képtranszformációt grafikus kártyán valószínűsítettem meg, így azok párhuzamosan futva, gyorsan dolgozzák fel a képeket. A feldolgozás során kiszűrésre kerülnek az egyes projekción a transzformált kép maximum értékeinek helyei. Szám szerint annyi ilyen pont kerül meghatározásra, amennyit a felhasználó előzetesen megadott, beleértve azon golyók számát is, amiket a felhasználó megjelölt, mint a felvételek valamelyikén a detektor látóteréből kilépő golyó. Így a felvételek azon részén, ahol nem esik minden golyó képe a detektorra, olyan pixeleket is azonosíthat az algoritmus golyó középpontnak, ahol valójában nincs is golyó. Ez a képek feldolgozását nem zavarja, ugyanis az egyes ellipszisekhez tartozó koordinátákat a következőképp gyűjtöm össze. A felhasználó által kijelölt első projekción meghatározásra kerülnek azon golyók középpontjai, amelyek azon a képen láthatóak. Így fontos kérés a felhasználó felé, hogy kezdetben olyan képet adjon meg, amelyen nincs olyan golyó, amely bármikor is eltűnik a detektorról. Ezen golyók középpontjai – amelyet a felhasználó a további feldolgozás előtt ellenőriz és jóváhagy – mellé fogja gyűjteni a szoftver az ellipszispálya további pontjait, úgy, hogy minden projekcióról választ egy pontot, mint az adott ellipszishez tartozó pont. Alapvetően a Hough-transzformált képen található maximális intenzitású pixeleket rendeli a szoftver az egyes ellipszisekhez, úgy, hogy egy adott ellipszis adatsorához az előző projekción megtalált pixelhez leg-

közelebb eső Hough-maximumot rendeli. Amennyiben valamely ellipszis adatsorához olyan pozíciót rendelnénk, amely az előzőtől jócskán eltér – önkényesen megadva: függőlegesen több, mint 15 pixellel, illetve vízszintesen több, mint 100 pixellel –, úgy az előző projekción meghatározott pozíció közvetlen környezetéből választok koordinátát az adott ellipszishoz, aszerint, hogy az adott környezetben az új képen hol a legnagyobb a Hough-transzformált értéke. Ilyen eset akkor fordulhat elő, ha valamiért nem sikerült megtalálni egy golyót az adott felvételen. Az eljárás meggátolja a szoftvert, hogy ilyenkor túlságosan kiszóró pontot adjon hozzá valamelyik ellipszis koordináta-halmazához.

5.4.4. Ellipszis illesztés

Az egyes ellipszisekhez tartozó adatsorokhoz meg kell határozni a rájuk illeszthető ellipszis paramétereit. Ezt a [9] cikk alapján úgy végeztem el, hogy az egyes ellipszisekhez kerestem azon p_0, p_1, p_2, p_3, p_4 együtthatókat, amelyekkel a (25) egyenletrendszer bal oldala a legkisebb négyzetes eltérést eredményezi a jobb oldalhoz képest.

$$p_0(u^i)^2 - 2p_1u^i - 2p_2v^i + 2p_3u^iv^i + p_4 = -(v^i)^2 \quad i = 1..N \quad (25)$$

A legkisebb négyzetek probléma megoldását a CUDA *cusolverSpScsrlsqvqrHost* függvényével állítottam elő. Ezen paraméterekből az a további feldolgozás szempontjából lényeges, (13) egyenletben bevezetett $a, b, c, \bar{u}, \bar{v}$ együtthatókat a (26) egyenletekkel lehet megkapni. Bár a paraméterek meghatározása a két vizsgált cikkben eltérő módon van megvalósítva, az ellipszis illesztést a fent leírtak szerint, Noo [9] cikke alapján végeztem el. A Wu [10] cikkben leírtak alapján csak az ellipszis c paramétere van elhanyagolva, amely igen jó közelítéssel már a fenti illesztésből is következik.

$$\left. \begin{aligned} \bar{u} &= \frac{p_1 - p_2p_3}{p_0 - p_3^2} \\ \bar{v} &= \frac{p_0p_2 - p_1p_3}{p_0 - p_3^2} \\ a &= \frac{p_0}{p_0\bar{u}^2 + \bar{v}^2 + 2p_3\bar{u}\bar{v} - p_4} \\ b &= \frac{a}{p_0} \\ c &= p_3b \end{aligned} \right\} \quad (26)$$

A paraméterek kiszámolásán túl kiszámoltam azok hibáját is. Ehhez a [8] forrást használtam fel, amely alapján a fenti egyenlettel definiált ellipszis adott p_i paraméterének hibáját a (27) képlettel lehet becsülni, ahol N a teljes körbefordulás során felvett projekciók száma, z_{ij} pedig a (26) egyenlet alapján a p_i . paraméterhez tartozó geometriai érték: p_0 esetén $(u^i)^2$, stb. A kifejezés számlálójában így az illesztés során kapott

négyzetes eltérés látható, míg a nevezőben az illesztett paraméterhez tartozó geometriai adatok négyzetösszege, szorozva a rendszer szabadsági fokaival, ami 5 paraméter keresése esetén az összes adatsor számánál 5-tel kisebb.

$$\sqrt{\frac{\sum_{j=1}^N \left(\sum_{i=0}^4 p_i z_{ij} + (v^j)^2 \right)^2}{(N-5) \sum_{j=1}^N z_{ij}^2}} \quad (27)$$

5.4.5. A geometriai paraméterek számítása

Az illesztett ellipszis $a, b, c, \bar{u}, \bar{v}$ paramétereit alapján a geometriai paramétereket az 5.2.1 és az 5.2.2 fejezetben bemutatott képletek segítségével számoltam.

Először kiszámításra kerül a detektor ferdeségét jellemző η paraméter. A két cikk hasonló módszerrel számolja ezt, az egyetlen különbség, hogy Wu cikke elve több golyóval számol, így egyenest illeszt a tengely vetített képeire. Épp ezért a számítások során az η paramétert egyféleképpen, egyenes illesztéssel határoztam meg. A k . golyó forgástengelyének vetített \hat{u}_k, \hat{v}_k koordinátáit a (10) képlet szerint lineáris regresszióval határoztam meg, szintén a CUDA *cusolverSpScsrlsqvqrHost* függvényével. Mivel az egyes koordináták hibája nem ismert, itt az illesztés négyzetes hibájával tudom jellemezni a kapott \hat{u}_k, \hat{v}_k paraméterek hibáját, amit a továbbiakban $\hat{\sigma}_k$ -val jelölök. Ezen paraméterek segítségével tan η értékét a hibával terhelt mérésekre vonatkozó egyenes illesztés képleteivel számolom. Általános esetben a felhasznált egyenletek az $y = a \cdot x + b$ egyenletű egyenes a, b paramétereit határozzák meg az x_i, y_i mért pontokból, ahol σ_i az y_i paraméter hibája. A keresett paraméterek és hibáik a (28) jelöléseket használva a (29) egyenletek alapján kaphatóak meg.

$$\left. \begin{aligned} S &= \sum \frac{1}{\sigma_i^2}, & S_x &= \sum \frac{x_i}{\sigma_i^2}, & S_y &= \sum \frac{y_i}{\sigma_i^2} \\ S_{xx} &= \sum \frac{x_i^2}{\sigma_i^2}, & S_{xy} &= \frac{x_i y_i}{\sigma_i^2}, & \Delta &= S \cdot S_{xx} - (S_x)^2 \end{aligned} \right\} \quad (28)$$

$$\left. \begin{aligned} a &= \frac{S_{xx} \cdot S_y - S_x \cdot S_{xy}}{\Delta} \\ b &= \frac{S \cdot S_{xy} - S_x \cdot S_y}{\Delta} \\ \sigma_a^2 &= \frac{S_{xx}}{\Delta}, & \sigma_b^2 &= \frac{S}{\Delta} \end{aligned} \right\} \quad (29)$$

Bár a fenti módszer matematikailag csak abban az esetben korrekt, ha csak az y_i paramétereket terheli a σ_i hiba, jelen esetben a módszer fizikailag releváns eredményt ad, hiszen valójában az egyes mérési pontok súlyozása történik a mért hiba inverzének négyzetével. A képletekben x, y és σ helyére az \hat{u}_k, \hat{v}_k és $\hat{\sigma}_k$ értékeket helyettesítve az

illesztett egyenes meredeksége megadja $\tan \eta$ értékét, annak hibájával együtt. Ebből számolom a program futása során az η paramétert, annak hibájával együtt. Minthogy mind az η szög, mind annak hibája igen kicsi érték ($\approx 1 \pm 0,1$ fok), a hibát a további számolás során elhanyagolom, azt csupán a felhasználó számára írom ki. Az η paramétert pedig felhasználom arra, hogy a mért koordinátákat a detektor ferdeségével korrigáljam, azaz elforgassam azokat.

A további paramétereket a program a két cikk alapján külön –külön számolja.

Noo metódusa szerint a fennmaradó paraméterek egy-egy ellipszis (v_0^*), illetve egy-egy ellipszis pár (D, u_0^*, ϕ, R) alapján határozhatók meg. Mindegyik paraméter esetében úgy jártam el, hogy kiszámoltam a paraméter értékét az adott ellipszisre vagy ellipszispárra, valamint kiszámoltam a paraméter hibáját is, az ellipszis illesztése során meghatározott hibák alapján kvadratikusan súlyoztam. Majd a geometriai paraméter végső értékét úgy számoltam ki, hogy az egyes részeredményeket azok hibájának inverzének négyzetével súlyoztam, és átlagoltam az értékeket. Ellipszispárok esetében a párok alapján számolt részeredmények közötti korrelációt nem vettem figyelembe a számítások során, mindenhol az összes ellipszispárból számolt eredményeket átlagoltam, a hibájuk alapján súlyozva. Ahol lehetett, ott az egyenletbe a már kiszámolt paraméterek összes ellipszis adataiból becsült értékét helyettesítettem be.

Wu metódusa eleve több ellipszist feltételez, így például D és v_0^* értékét is egyenes illesztéssel határozza meg, ahogy azt a (19) egyenlet leírja. Itt szintén az η meghatározása során használt módszert alkalmaztam, azaz a hibával terhelt mérésekre vonatkozó (28) és (29) egyenletek alapján illesztettem egyenest. Az illesztés során figyelembe vett hibát a (19) egyenlet függő változójának (bal oldal) hibájával becsültem. Mivel u_0^* ennél a módszernél az illesztett \bar{u}_k paraméterek átlagolásával történik, a hibaszámítással figyelembe vett, hiba inverzének négyzetével való súlyozással kapott átlag itt teljesen korrekt módszerként használtam. A Wu módszer egyetlen ellipszispáronként számolt paramétere A (21) egyenlet alapján számolt R paraméter, vagyis a forrás és a forgástengely távolsága. Itt szintén ellipszis páronként határoztam meg az R értéket és az adott ellipszispárra kapott hiba négyzetének inverzével súlyozva kaptam a paraméter becslését az összes ellipszis adatainak felhasználásával.

A geometriai paraméterek számolása során tehát mindvégig számoltam a paraméterek hibáját, és a végső paraméterek számolásához az összes ellipszis adatát felhasználva kaptam meg. A kísérleti algoritmusban az átlagolás az ellipszispárokra súlyozás nélkül történt, én azonban a köztes paraméterek hibáját felhasználva, az alapján súlyozva határoztam meg a keresett geometriai paramétereket. Így amennyiben valamely ellipszis pontjai pontatlanul kerülnek meghatározásra, az a végeredményt nem torzítja el, hiszen ez esetben az ellipszis illesztésekor annak paraméterei nagy hibával kerülnek meghatá-

rozásra, így az adott ellipszis kis súllyal fogja befolyásolni a kimeneti paramétereket. A szoftver tehát felhasználói beavatkozás nélkül figyelembe veszi az adatsorok jóságát és azokat figyelembe véve számítja ki a geometriai adatokat.

6. Eredmények

Diplomamunkám megírása során elkészítettem egy platformfüggetlen grafikus szoftver-környezetet, amely a gain korrekciót és a geometriai kalibrációt hatékonyan és gyorsan képes véghez vinni. Ehhez megterveztem a képek grafikus kártyán való tárolásáért és feldolgozásáért felelős *C++* osztályokat és függvényeit, a képfeldolgozásért felelős részeket *CUDA*-n megvalósítva. Az egyes funkciók erőforrásigénye is lényegesen alacsonyabb, így a szoftver alkalmassá vált egy átlagos személyi számítógépen való használatra is.

6.1. Gain korrekció

A gain korrekciós faktorok meghatározása során használatos algoritmusok lényegi részén a kísérleti algoritmusokhoz képest nem változtattam, viszont a feldolgozást grafikus kártyát használva párhuzamosítottam. A kalibrációs képek feldolgozásakor bevezettem az azonos expozíciós beállítással készült átlagolását, az esetleges zajok kiszűrése végett. A képek feldolgozásakor azok feszültségértékének meghatározását az eddigi megvalósítással ellentétben nem a felhasználóra bízom, hanem a projekciós képek adataiból olvasom ki, amely kényelmesebb használatot biztosít a felhasználó számára. Vizsgálom továbbá a képek átlagos intenzitásbeli különbségeket és az esetleges hibás felvételeket kiszűröm.

A futási időben elért növekedést meghatározása érdekében megmértem a két algoritmus számára szükséges időt, egy adott kalibrációs képsorozat feldolgozásakor. A kalibrációs faktorok meghatározásához 15 féle beállítás mellett készült offset korrekció, használtam. Ezek voltak tehát azok, ahol csupán az expozíciós időt változtatva mértjük a detektált háttérrel. A forrás bekapcsolásával készült kalibrációs képek 36 féle expozíciós érték mellett készültek. Beállításoként több képet készítettem, a fenti 15 + 31 féle beállítás mellett összesen 4 742 kép készült készült, beállításoként tehát átlagosan 103.

A gain korrekciós faktorok meghatározását egy nagy teljesítményű számítógépen végeztem, amely egy 4 magos, *Intel i7-3770* típusú, 3,40 GHz sebességű processzorral, egy *NVIDIA GeForce GTX 690* típusú videokártyával, valamint nagy mennyiségű, 32 Gb RAM-mal van rendelkezésre. A futási időket az 1. táblázatban tüntettem fel.

A táblázatból láthatjuk, hogy a párhuzamos feldolgozásnak köszönhetően a programom lényegesen gyorsabban végzi el a gain korrekciós faktorok meghatározását.

Algoritmus	Futási idő
Kísérleti algoritmus	95,6 perc
Általam használt, párhuzamosított feldolgozás	3,2 perc

1. táblázat. A gain korrekciós faktorok meghatározásához szükséges futási idő az egyes algoritmusok esetében, 4 742 darab kép feldolgozása során.

Az párhuzamos megvalósítással a felhasználó igen hamar meg tudja határozni a szükséges faktorokat. Ezen kívül a futáshoz használt erőforrásigény is csökkent a kísérleti algoritmushoz képest. Utóbbi ugyanis időként 25–30 Gb memóriaterülete is lefoglalt a számítógép virtuális memóriájából. (A képek a merevlemezen kb. 12 Gb tárhelyet foglalnak el.) Mivel az én megvalósításom nem a virtuális memóriát használja, hanem a grafikus kártya memóriáját, ezért annak RAM igénye csupán 10–20 MB. A grafikus kártyán szükséges memória nagysága kb. 500 Mb, ennyi tárhelyen fér el ugyanis az egy mappából beolvasott kb. 100 darab kép. Ekkora memóriával már az alsó kategóriás videokártyák is rendelkeznek.

A kalibrációs faktorok meghatározásához szükséges futási időn kívül mértem a projekciók gain korrigálásához szükséges futási időket is. 1 000 kép gain korrigálását végeztem el mindkét algoritmussal és mértem az eközben eltelt időt. Az eredmények a 2. táblázatban láthatóak.

Algoritmus	Futási idő
Kísérleti algoritmus	70 sec
Általam használt, párhuzamosított feldolgozás	37 sec

2. táblázat. 1 000 kép gain korrigálásához szükséges futási idő az egyes algoritmusok esetében.

A különbség még mindig jelentős, de az előzőekhez képest kevésbé látványos. Ennek oka, hogy ezen művelet kevesebb számolást igényel, így a párhuzamosítás okozta előny is csökken az előzőekhez képest.

6.2. Geometriai kalibráció

A geometriai kalibráció megvalósítása során szintén párhuzamosítottam a képfeldolgozó eljárásokat. Ezen kívül új módszert vezettem be a körök megkereséséhez a kalibrációs felvételeken és hibaszámítást figyelembe véve valósítottam meg a geometriai adatok számolását.

A kísérleti algoritmussal való összehasonlításhoz lefuttattam a rendelkezésemre ál-

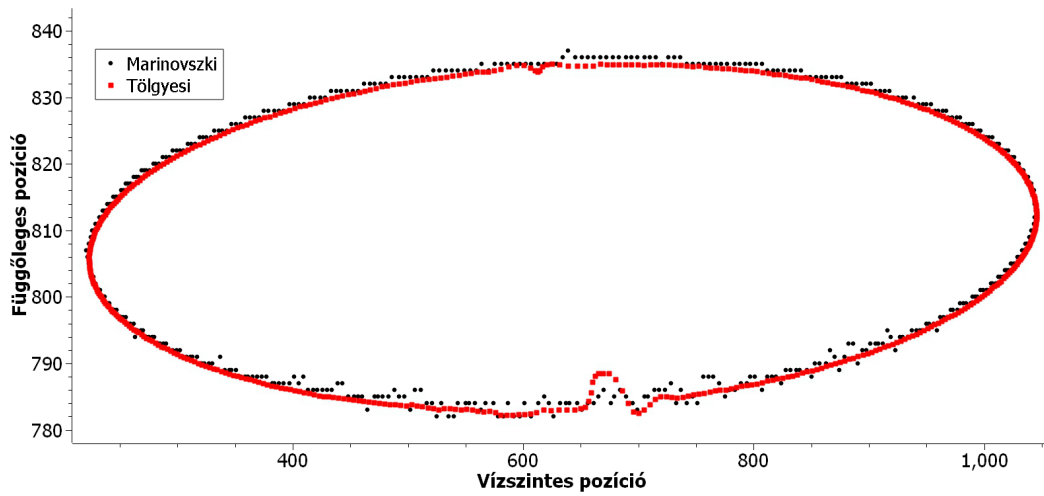
ló két mérési adatsoron mindkét algoritmust és vizsgáltam eredményüket. A kísérleti algoritmus esetében a felhasználónak először ki kell jelölnie az összes golyó helyzetét az első két projekción, majd a szoftver megkeresi azokat a további projekciókon. A kereséshez szükséges egy további paramétert, mégpedig az, hogy az adott golyó feltételezett középpontja körül milyen sugarú területen történjen a kép intenzitásértékeinek a súlyozott átlagolása. Ezen paraméter optimális beállítása nem bizonyult könnyű feladatnak. Geometriai kalibrációs mérésből ugyan azon geometria mellett készítettem két sorozatot, kétféle fantommal. A fantomok között a használt golyók mérete adta a különbséget. Egyik esetben 1 mm és 1,5 mm átmérőjű csapágygolyókat rögzítettünk a fantomon, a másik esetben a legnagyobb csapágygolyó átmérője 2,5 mm volt. A kalibrációs képsorozatban a legkisebb golyó képe 20 pixel átmérőjű volt, míg a legnagyobb, 2,5 mm átmérőjű csapágygolyó 50 pixel átmérőjű árnyékot adott. Ezek alapján a középpont körüli 25–30 pixel sugarú körben tűnt relevánsnak a golyók súlypontjának a keresése. Ilyen paraméter mellett azonban a kisebb golyókat a kísérleti algoritmus nem tudta megtalálni a képeken, így az illesztett ellipszisei és a számolt geometriai adatok is nagyságrendileg hibások voltak. A súlypontkeresés által vizsgálandó tartomány sugarának 10 pixelre való csökkentésével sikerült kiértékelhető eredményt kapnom. Ugyan erre a megállapításra jutottam a másik fantom kiértékelésénél is. Ott a csapágygolyók árnyéke 20 és 30 pixel között változott, azonban a súlypontkeresés mezőjét 16 pixel sugarúra állítva itt is helytelen eredményeket kaptam. Itt is a 10 pixeles érték bizonyult hatékonynak. Minthogy az így nyert geometriai adatok, amiket alább ismertetek, nem bizonyultak helyesnek, megállapítható, hogy a kísérleti algoritmus az alkalmazott csapágygolyók közötti kis eltéréseket sem tudja jól kezelni.

A körkeresés minőségét egyrészt a megtalált középpontok ábráját megtekintve ítélni tudjuk meg. A legkisebb golyó középpontjának mozgását a 21. ábrán láthatjuk. Piros színnel ábrázoltam a kísérleti algoritmus által meghatározott pozíciókat, fekete színnel az általam megvalósított algoritmus által becsült koordinátákat.

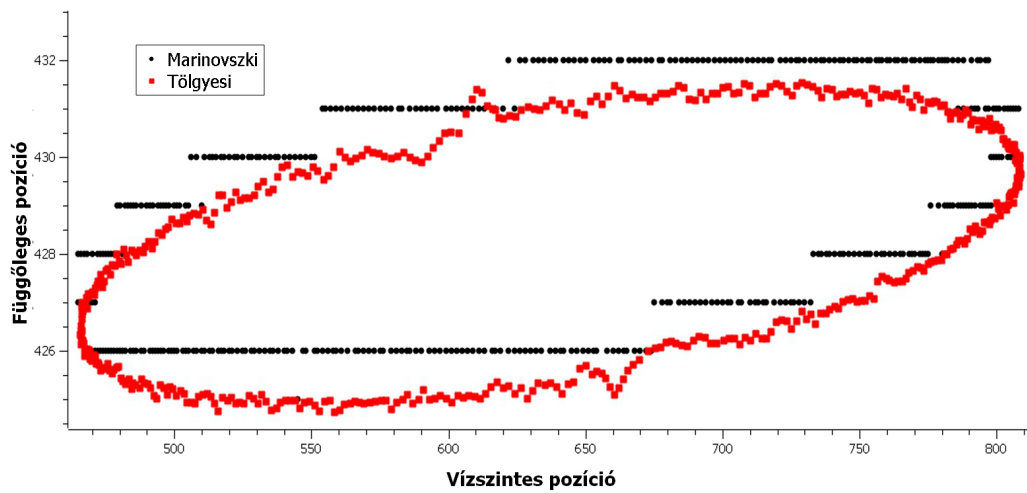
Az ábrán láthatjuk, hogy a pálya közepén, a kép alsó, középső részén a kísérleti algoritmus által meghatározott koordináták eltérnek a golyó pályájától. Épp ebben a pozícióban fordul a fantom olyan állásba, amely állásban a plexi lemez jelentősen befolyásolhatja a súlypontkeresést minőségét. Az általam megvalósított körkereső algoritmus ezen a részen nem téved, ezen a részen is képes a csapágygolyó képét pontosan megtalálni.

Megvizsgáltam továbbá, hogyan alakulnak a felismert középpontok, a többi, nagyobb méretű golyó esetében. A 1,5 mm átmérőjű csapágygolyó árnyékának detektált középpontjait a 22. ábrán tüntetem fel.

Az ábrán egy kisebb pályát leíró golyó képe látható, ezért is ennyire észrevehető az



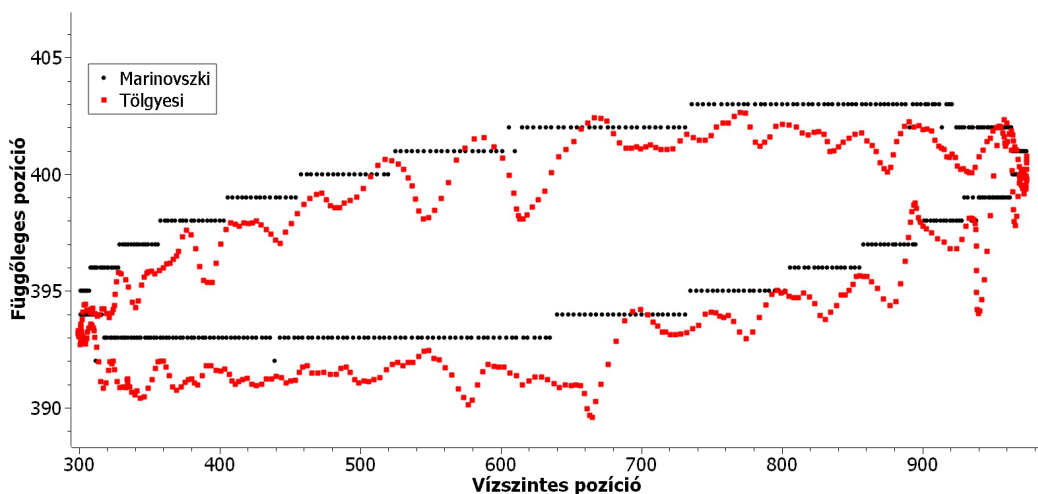
21. ábra. Teljes körbefordulás alatt meghatározott golyó középpontok, a legkisebb golyó pályája mentén. Fekete színnel ábrázolva az általam implementált algoritmus által meghatározott koordinátákat, piros színnel a kísérleti algoritmus által meghatározott koordinátákat.



22. ábra. Teljes körbefordulás alatt meghatározott golyó középpontok, közepes méretű golyónál. Fekete színnel ábrázolva az általam implementált algoritmus által meghatározott koordinátákat, piros színnel a kísérleti algoritmus által meghatározott koordinátákat.

általam generált adatsor kvantáltsága. A Hough-transzformáció során ugyanis a körök középpontját egész koordinátákban határozom meg. Bár a kísérleti algoritmus által meghatározott koordinátákban látható némi szórás, annak mértéke ± 1 pixelen belüli. Ugyanakkor a plexi által elsötétített projekciókon itt az átlagosnál nagyobb kiszórás figyelhető meg.

Szemügyre vettem továbbá a még nagyobb, 2,5 mm átmérőjű golyó két algoritmus által meghatározott pályáját is. Ez látható a 23. ábrán



23. ábra. Teljes körbefordulás alatt meghatározott golyó középpontok, nagy méretű golyónál. Fekete színnel ábrázolva az általam implementált algoritmus által meghatározott koordinátákat, piros színnel a kísérleti algoritmus által meghatározott koordinátákat.

Itt már szembetűnő, hogy a kísérleti algoritmus igen nagy szórással követi a golyót a pályáján. A középpontok egyfajta periodikus ingadozást mutatnak, a pálya szélén pedig jelentős mennyiségű kiszóró pontot láthatunk. Ez tulajdonképpen nem meglepő, hiszen itt a kísérleti algoritmus az adott golyót annak méreténél jelentősen kisebb ablakban kereste.

A fentiek alapján megállapítható, hogy míg a kísérleti algoritmus által megvalósított körkereső algoritmust minőségét valóban befolyásolja a plexi lemez okozta elsötétlés. Az általam megvalósított körkereső algoritmus a golyókat úgy találja meg, hogy ez nincs befolyással a folyamatra. A koordináták ebben az esetben is megfelelő minőségben állnak elő. Megállapítható továbbá, hogy a kísérleti algoritmus hatékonysága nagy mértékben csökken, amennyiben különböző méretű golyókat használunk a kalibrációs fantomon. Ugyanakkor ez az általam megvalósított algoritmusra ez nincs hatással, az a különböző méretű golyók középpontjait is pontosan találja meg.

Az egyes algoritmusok által megállapított geometriai adatok helyességét is megvizsgáltam. Mivel ezek helyessége numerikusan közvetlenül nem ellenőrizhető, az adott

geometriai mellett elkészített felvételek rekonstrukciója során előálló 3 dimenziós kép minőségét vizsgáltam, a különböző kapott geometriai paraméterek mellett.

Az egyes algoritmusok által meghatározott paraméterek a 3. táblázatban láthatóak. A táblázatban látható adatokkal rekonstruáltam egy plexi hasábról készült felvételsorozatot, amely rekonstrukciók keresztmetszetei képei a 24. ábrán láthatóak. A képek közepén a plexi fantom közepén lévő vékony furatot látjuk inhomogenitásként.

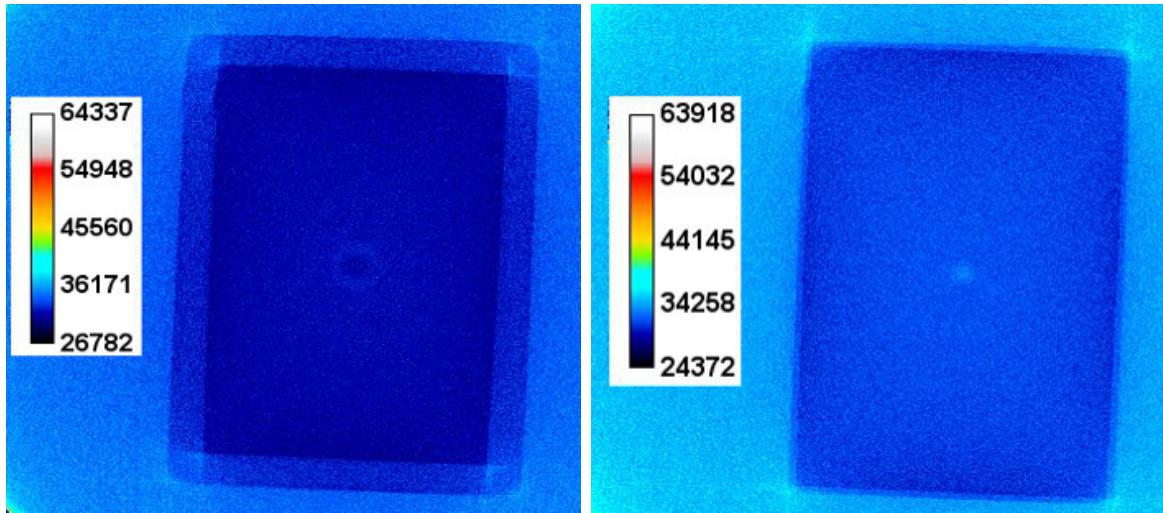
Algoritmus	η	u_0	v_0	D	R
Kísérleti algoritmus változó golyómérettel	0.117334	616.654	584.596	632.234	440.933
Kísérleti algoritmus hasonló méretű golyókkal	0.153843	635.381	582.007	632.416	425.902
Új algoritmus Noo egyenleteivel	1.21522	643.451	656.5	671.671	359.402
Új algoritmus Wu egyenleteivel	1.21522	641.646	589.027	632.072	410.122

3. táblázat. Az egyes algoritmusok által meghatározott geometriai adatok

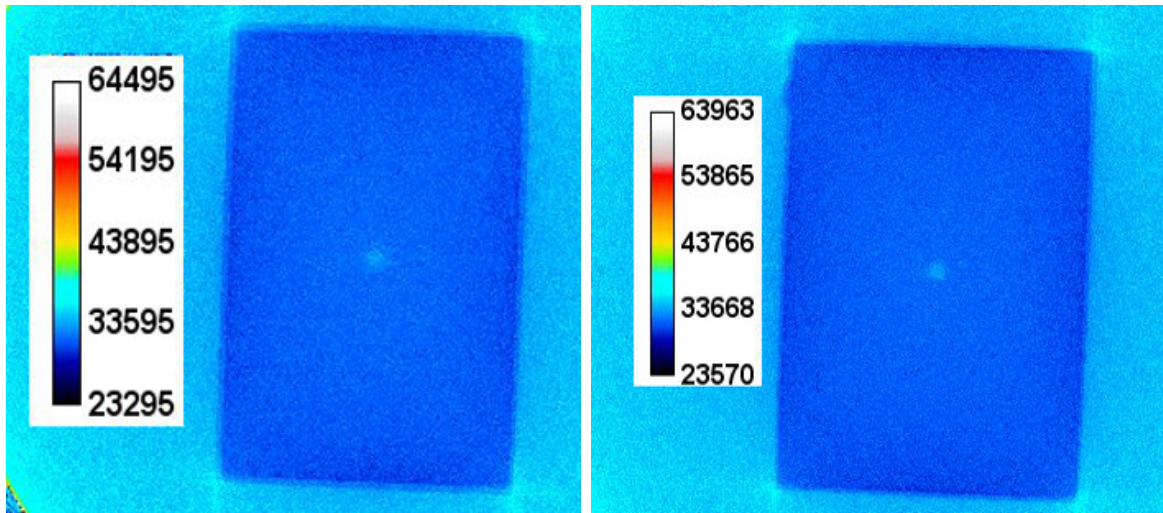
Az ábrákra ránézve szembeötlő, hogy a kísérleti algoritmus által számolt adatokkal a rekonstrukció minősége igen rossz, amennyiben a kalibráló fantomon a golyók változó méretűek (24a. ábra). A többi képen a képek szélén, illetve közepén figyelhetjük meg a változó paraméterek okozta hatást. A közepén lévő furat elmosódottsága, valamint a széleken az intenzitásés meredeksége utal a képek minőségére: minél élesebbek ezek a határok, annál jobb a rekonstrukció minősége, és így a geometriai paraméterek is. A maradék három kép között a különbség már nem olyan nagy, de az általam használt algoritmusok és a Wu féle egyenletekkel való számolás eredményeképp előálló kép minősége még szemmel láthatóan jobb a többinél.

A képminőségek objektív megítélése érdekében a képeket további vizsgálatnak vettem alá és kiszámoltam a képek élválasz-függvényének félértékszélességét[1]. Az élválasz-függvény egy képen látható él menti intenzitásátmenet élességét vizsgálja. Valós képeken az ilyen átmenetek nem azonnaliak, hanem az él mentén egy véges tartományban folytonosan változnak az intenzitás értékei, mint ahogy azt láthatjuk a plexi szélét megvizsgálva. Az élre merőleges egyenest állítva, az a mentén megfigyelt intenzitásértékek deriváltja Gauss-görbével írható le. Ezt a görbét hívjuk élválasz-függvénynek, amely tehát kis félértékszélesség esetén hirtelen változó intenzitásra utal, nagy félértékszélesség esetén pedig elmosódott élt jellemez.

A 24. ábrán látható képeknek meghatároztam az élválasz-függvényeit. Ehhez a kép



(a) Kísérleti algoritmus, változó golyómérettel (b) Kísérleti algoritmus, hasonló méretű golyókkal



(c) Új algoritmus, Noo egyenleteivel

(d) Új algoritmus, Wu egyenleteivel

24. ábra. A különböző algoritmusok által számolt geometriai adatokat felhasználva készült rekonstrukciók egy-egy keresztmetszeti képe.

éleire merőlegesen állított egyenes mentén felvettem az intenzitásértékeket, majd azokat mozgóablakos módszerrel simítottam. Az így kapott adatsort numerikusan deriváltam és Gauss-görbét illesztettem rá. Ezt képenként 5–5 helyen tettem meg. Egy ilyen műveletsor lépései láthatóak a 25. ábrán. Az illesztéssel kapott Gauss görbe az élválasz-görbe, amelynek félértékszélességét a fentiek szerint lemérve és átlagolva a 4. táblázatban tüntettem fel.

A táblázat értékei alapján objektíven is megállapítható, hogy az általam bevezetett új képfelismerő algoritmusok és a számolások során a hibák figyelembevételével kapott eredményekkel jobb minőségű kép rekonstruálható, mint az eddig használt algoritlussal. A kétféle számítási módszer közül a Wu féle bizonyult előnyösebbnek, így a továbbiakban annak használata javasolt. Ez az algoritmus az általam kipróbált

Algoritmus	Élválasz-függvény félértékszélessége (pixel)
Kísérleti algoritmus hasonló méretű golyókkal	12.4
Új algoritmus Noo egyenleteivel	12.2
Új algoritmus Wu egyenleteivel	10.0

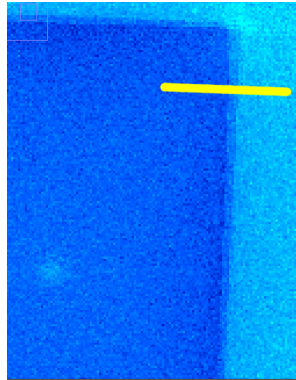
4. táblázat. Élválasz-függvény félértékszélesség értékei a különböző algoritmusok által számolt geometriai paraméterekkel való visszavetítés után.

másik, Noo féle módszerrel ellentétben nem igényel semmilyen előzetes információt a geometria adatairól, így megbízhatósága is nagyobb.

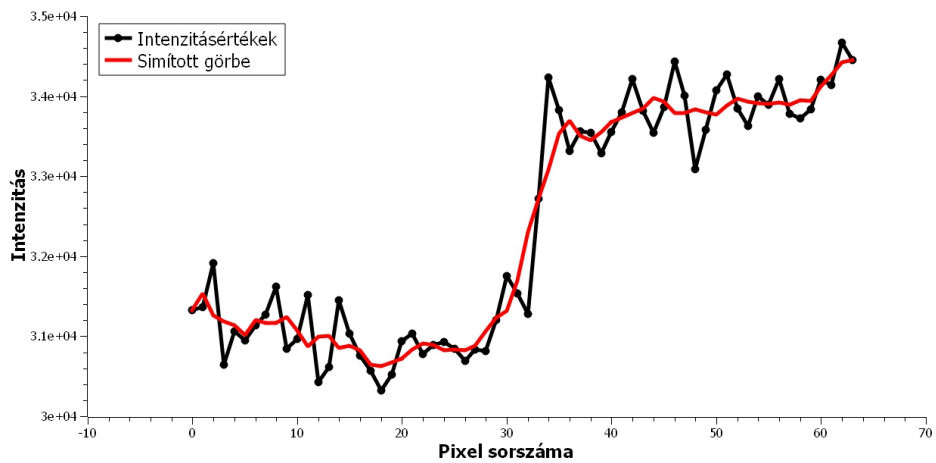
Figyelemre méltó változás ezeken túl a futási idő csökkenése. A párhuzamos feldolgozás miatt az általam használt, több számolást igénylő algoritmusok is hamarabb futnak le, mint a kísérleti algoritmus kódja. A futási idők az 5. táblázatban láthatóak. Az általam megalkotott szoftver tehát nem csak pontosabban eredményeket ad, hanem a képeket – a nagyobb számítási igény ellenére is – gyorsabban dolgozza fel. Az adatok megítéléséhez még megjegyzendő, hogy a saját algoritmusom futása során a képek gain korrigálása is megtörtént, míg a kísérleti algoritmust előre gain korrigált képeken futtattam.

Algoritmus	Futási idő
Kísérleti algoritmus	130 sec
Általam használt, párhuzamosított feldolgozás	450 sec

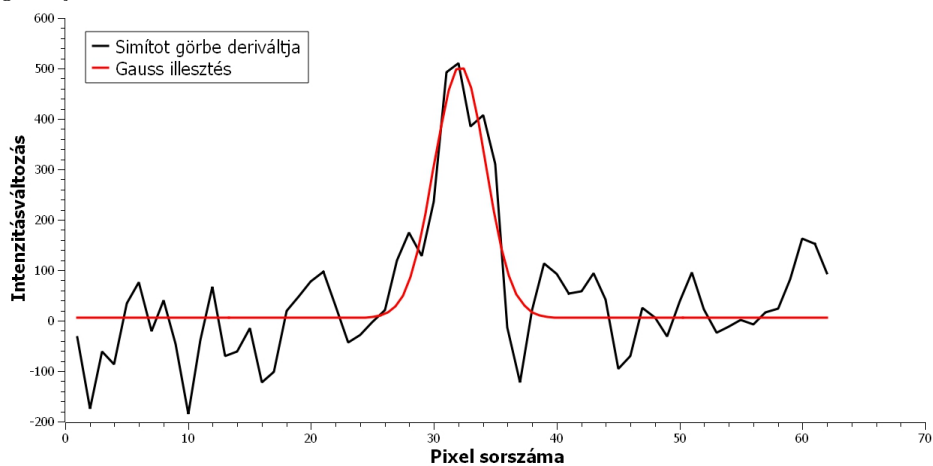
5. táblázat. Geometriai korrekciók meghatározásának ideje a két algoritmussal, 503 képet tartalmazó kalibrációs adatsorból.



(a) A sárga egyenes mentén vizsgálom az intenzitásértékek változását



(b) Intenzitásértékek az egyenes mentén – feketével –, valamint annak simított görbéje – pirossal



(c) A simított görbe deriváltja, valamint az arra illesztett Gauss görbe

25. ábra. Az élválasz-függvény meghatározásának lépései.

7. Összefoglalás és kitekintés

Diplomamunkám során a Tölgyesi Botond által [2] tesztelt gain korrekciós és geometriai kalibrációs algoritmusokat ültettem át grafikus kártyára, valamint a funkciók könnyű használata érdekében grafikus felülettel láttam el az elkészült programot. Az új szoftver lényegesen gyorsabb és kevésbé erőforrás igényes, mint elődje. Továbbfejlesztettem a geometriai kalibrációban használt körkereső algoritmust is, amely így pontosabb eredményeket ad, mint az eddigiek. Ezen kívül a geometriai adatok meghatározása során figyelembe vettem a kiértékelés során felmerülő hibákat, amely segítségével a rosszul illeszkedő, tehát potenciálisan kevesebb információ hordozó ellipszisek kisebb hatással vannak a meghatározott paraméterekre, így azok pontossága javul. Ezekkel az eljárásokkal a geometria kalibráció a minőséget nagyban befolyásoló paraméter megadása nélkül, gyorsan és pontosan végzi. Végeredményben pedig jobb eredményt ad, mint az elődje.

Az algoritmusok gyorsasága a grafikus kártyára való átültetéssel megérett arra, hogy ezeket a korrekciókat mérés közben végezzük el. Hasznos lenne ezeket a funkciókat a mérésvezérlő szoftver részévé tenni, hogy ne utólag kelljen a képek korrekcióját elvégezni.

További feladat maradt még a spektrumfelkeményedés korrekció megvalósítása, amire jelen diplomamunka elkészítése során nem valósítottam meg, mivel érdekesebbnek tűnt a megvalósított részekre több időt szentelni. A korrekció megvalósításával az elkészült felvételek minőségét tovább lehetne javítani.

Ugyancsak javíthatna a geometriai kalibráció minőségén egy pontos geometriai fantom, vagy fantom sorozat és a kalibrációs mérés általános, robusztus protokolljának megtervezése. A kis tengelyekkel rendelkező ellipszisek hibája ugyanis igen nagy, megfelelő fantom tervezéssel jobb geometriai adatokat lehetne becsülni. A fantomtervezést megkönnyítheti az általam megalkotott gyors adatfeldolgozó rendszer, amellyel akár a mérés közben lehetne a felhasználónak ajánlást tenni, hogy egy nagy hibával rendelkező ellipszis csapágygolyóját a fantom melyik területére helyezze át.

További felhasználási módot jelent a megalkotott szoftverkörnyezetnek a tanszéken tervezett optikai CT rendszer, amely beüzemelésében várhatóan szintén nagy szerepet fog játszani a most megalkotott, gyors és megbízható feldolgozórendszer.

Hivatkozások

- [1] S. M. Bentzen: Evaluation of the spatial resolution of a ct scanner by direct analysis of the edge response function. *Medical Physics*, 10. évf. (1983) 5. sz., 579–581. p.
- [2] Tölgyesi Botond: Röntgen ct készülék építése. Diplomamunka (BME NTI). 2014.
- [3] The Qt Company: Qt documentation. <https://doc.qt.io/>. [Online; utolsó hozzáférés: 2016.05.28. 14:00].
- [4] NVIDIA Corporation: Cuda toolkit documentation. <http://docs.nvidia.com/cuda/>. [Online; utolsó hozzáférés: 2016.05.28. 14:00].
- [5] Deli Gábor: Általános tomográfiás rekonstrukciós szoftver fejlesztése gpu-ra. Diplomamunka (BME NTI). 2014.
- [6] Kleizer Gábor: Mechatronikai rendszer összekapcsolása nukleáris adatgyűjtő rendszerekkel. Diplomamunka (BME NTI). 2011.
- [7] Mark Harris: Optimizing Parallel Reduction in CUDA. http://docs.nvidia.com/cuda/samples/6_Advanced/reduction/doc/reduction.pdf. [Online; utolsó hozzáférés: 2016.05.28. 14:00].
- [8] Kirk T. McDonald: Error estimation in fitting of ellipses. <http://www.physics.princeton.edu/mumu/target/ellipse.pdf>. [Online; utolsó hozzáférés: 2016.05.28. 14:00].
- [9] Frédéric Noo–Rolf Clackdoyle–Catherine Mennessier–Timothy A White–Timothy J Roney: Analytic method based on identification of ellipse parameters for scanner calibration in cone-beam tomography. *Physics in Medicine and Biology*, 45. évf. (2000) 11. sz., 3489. p.
- [10] D. Wu–L. Li–L. Zhang–Y. Xing–Z. Chen–Y. Xiao: Geometric calibration of cone-beam ct with a flat-panel detector. In *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2011 IEEE* (konferenciaanyag). 2011. Oct, 2952–2955. p. ISSN 1082-3654.