DIPLOMAMUNKA

Statisztikus képszámítási eljárás kidolgozása gamma kamerához

Meszlényi Regina Júlia

Témavezető: Dr. Bükki Tamás
Tudományos Kutató
Mediso Kft.

Tanszéki konzulens: Dr. Czifrus Szabolcs
Egyetemi Docens
BME- Nukleáris Technikai Intézet

BME
2014

Önállósági Nyilatkozat

Alulírott **Meszlényi Regina Júlia** 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,

aláírás

Köszönetnyilvánítás

Mindenekelőtt köszönöm a támogatást témavezetőmnek, Bükki Tamásnak, aki az elmúlt közel másfél év alatt mindig segítségemre volt szakértelmével, és kérdéseivel, amiken elgondolkozva sikerült a munkámat végig a helyes mederben tartani.

Hálás vagyok Kovács Ferencnek, akivel az első méréseseket együtt végeztük, és Légrádi Gábornak, aki a diplomamunkában végül felhasznált adatokat kimérte, és Major Péterrel együtt számos hasznos tanáccsal látott el a munka során.

Emellett köszönöm Szedlacsek Endrének, aki telepítette és beállította a laptopomon a Nucline szoftver egy olyan verzióját, ami statisztikus képszámítást is lehetővé tesz, így nekem már csupán az átadott kódokon kellett módosítanom a mérések kiértékeléséhez.

Végezetül köszönöm a Mediso Kft.-nek, hogy lehetőséget biztosított arra, hogy a diplomamunkámat valódi ipari fejlesztésben releváns témában írhassam meg.

Azonosító: DM-2013-93	
Diplomatéma címe:	Statisztikus képszámítási eljárás kidolgozása gamma kamerához
Melyik szakiránynak ajánlott?	*"Orvosi fizika"*
A jelentkezővel szemben támasztott elvárások:	szakirodalom olvasása angol nyelven, programozási ismeretek
Témavezető neve:	Dr. Bükki Tamás
Témavezető Intézménye:	Mediso Kft.
Témavezető Intézményének a címe:	1047 Budapest, Baross utca 91-95.
Témavezető e-mail címe:	bukki.tamas@mediso.hu
Témavezető telefonszáma:	06303751960
Leírása:	A gamma kamerák szcintillációs eseményeinek pozíció-meghatározására általában a klasszikus Anger, azaz súlyponti megoldást alkalmazzuk, illetve az NLCC metódust használjuk, mely átlagos módon figyelembe veszi a PMT jelválaszát, azt izotrópnak feltételezve. Ennek megfelelően egy speciális nemlineáris zajvágást alkalmaz, mely az Anger képszámításhoz képest jobb intrinsic felbontást eredményez. Korszerű adatgyűjtő elektronika esetén, ahol a kamerafej összes PMT jele ismert, lehetőség van olyan, ún. statisztikus képszámítási eljárások kidolgozására, melyek ki tudják aknázni a többletinformációt és így a kamera hasznos látóterében nagyobb átlagos intrinsic felbontást és nagyobb hasznos látóteret eredményezhetnek. A diplomamunkás feladata a statisztikus képszámítási eljárások irodalmi áttekintése után ezek gyakorlati megvalósítása lesz, továbbá az eljárások szabad paramétereinek optimalizálása, beleértve egy nemlineáris zajszűrési eljárás kidolgozását, annak érdekében, hogy a lehető legjobb legyen a hasznos látómezőben mérhető intrinsic felbontás és legnagyobb a hasznos látómező mérete. A diplomamunkás a Mediso Kft. segítségével valós, mért adatokon dolgozhat. A képszámítást végző magfüggvényt C++, az optimalizációs feladatot C++ vagy MATLAB nyelven kell implementálnia; a képszámítást végző magfüggvénynek illeszkednie kell a Mediso Kft. Nucline adatgyűjtő keretrendszeréhez, mely a különböző primer képkorrekciókat (linearitás, uniformitás...) végzi.

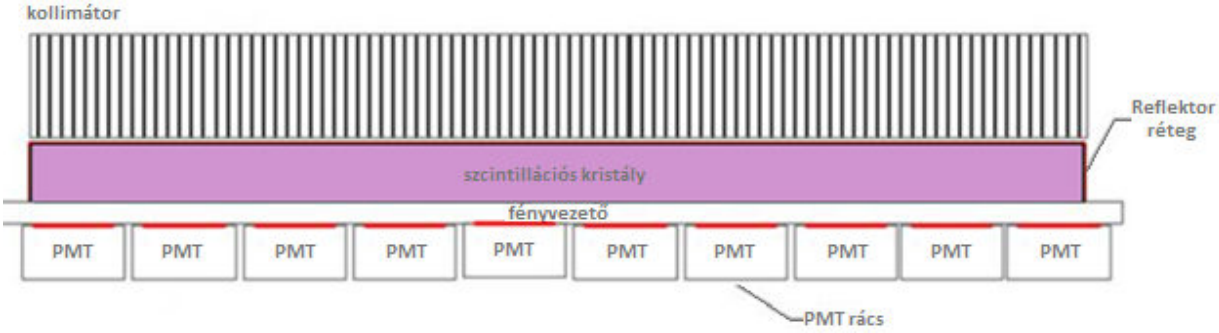
TARTALOMJEGYZÉK

I. Elméleti bevezető

I.1. Gammakamerák felépítése és működése

Az első gammakamerát Hal Anger építette meg 1957-ben [1.]. Az általa használt konstrukció, amit Anger-kamerának [2.] is nevezünk, napjainkban is széles körben elterjedt, ezért a gammakamerák működési elvét egy ilyen készülék leírásával mutatom be.

Ahogy az összes nukleáris medicina területéhez tartozó képalkotó eljárásnál, a gammakameráknál is a célunk az, hogy egy radioaktív izotóp, esetünkben gammasugárzó radiofarmakon, pozícióját minél pontosabban meghatározzuk. A detektor elvi felépítése igen egyszerű, ezt mutatja az 1. ábra is:



1. ábra: A gammakamera detektorának felépítése [1.]

A gammasugárzás elsőként a kollimátort éri el, majd ezen áthaladva, és részben elnyelődve a szcintillátor kristályba jut, itt a nagyenergiás gamma-fotonok látható fényfelvillanásokat okoznak. Azt, hogy egy ilyen felvillanásban hány foton keletkezik elsősorban a szcintillátor anyaga, és a gamma-foton energiája határozza meg (NaI kristály esetében ez 38 foton/keV [1.]). A reflektor réteg a kristályból rossz irányba távozó fotonokat veri vissza, így a fényvezető rétegen keresztül az egy-egy gamma-foton elnyelődéséből keletkező több ezer foton nagyobb hányada (5-20%) jut el a fotoelektronsokszorozókhoz (PMT, photo multiplier tube).

A gammakamerákban található szcintillátorok általában egykristályok, míg mögöttük a fotoelektronsokszorozókat rácsban helyezik el. A PMT-k feladata, hogy a beérkező látható fényt elektromos jellé alakítsák át, ily módon a beérkező fény intenzitásának mérését feszültségmérésre vezethetjük vissza.

Az Anger-kamerákban maga a pozíció-meghatározás is igen egyszerű elven zajlik [1.]. Legnagyobb fényintenzitást az a PMT kap, amelyik a legközelebb esik a szcintillátor kristályban keletkezett felvillanás helyéhez, míg a távolabbi fotoelektronsokszorozókhoz jóval

kevesebb foton jut el. A legegyszerűbb pozíciószámítási eljárás tehát egyszerű súlypontszámítással adódik (1. egyenlet):

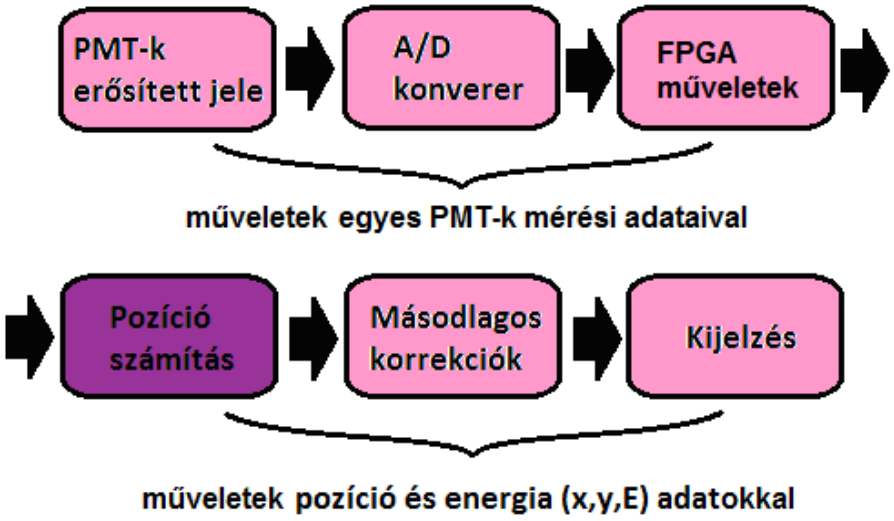
$$X = \frac{\sum_i x_i \cdot S_i}{\sum_i S_i}, Y = \frac{\sum_i y_i \cdot S_i}{\sum_i S_i} \qquad \textbf{(1.)}$$

Ahol S_i az i. PMT-n mért fényintenzitás, x_i és y_i pedig az i. PMT koordinátáit jelölik.

Az Anger képszámítási módszer nagy előnye, hogy akár analóg módon, feszültségosztókkal is elvégezhető, így nagyon gyorsan megkaphatóak a koordináták. A súlyponti eljárás számításigénye még digitalizálva is nagyon kicsi, ugyanakkor megfelelő korrekciók elvégzésével jól használható képet ad.

I.2. A gammakamera jelfeldolgozási lépései

Napjainkban a fotoelektronsokszorozók jeleit már többnyire nem analóg módon összegzik, hanem megfelelő előerősítés és szűrés után rögtön digitalizálják. Az ilyen kamerákat digitálisnak nevezzük, és jelfeldolgozásuk lépéseit foglalja össze a 2. ábra.



2. ábra: Digitális gammakamera jelfeldolgozási lépései

A digitalizálás során keletkező adatfolyamot FPGA-k (field-programmable gate array) segítségével összegezzük; valamilyen idő-ablakon belül integrálva kaphatjuk meg a súlypont-számításokhoz szükséges S_i értékeket. Ezeket az értékeket list-módú adatnak nevezzük, ilyenkor minden egyes PMT jelét külön tároljuk az idő függvényében mintavételezve. A list-módú adatokon egyéb elsődleges korrekciókat is végezhetünk, mint a pile-up hatások kiküszöbölése, illetve az esetleges alapvonal-korrekciók.

A list-módú adatokból valódi képet a pozíciószámítási eljárás segítségével kaphatunk. Ez a legtöbb ma piacon levő készülékben a már ismertetett egyszerű súlypont-számítási algoritmus, vagy annak egy továbbfejlesztett változata, de fontos kutatási területet képeznek a statisztikus képszámítási módszerek is.

A pozíciószámítás után az adataink már kép formájában megjeleníthetőek ugyan, de többnyire további, úgynevezett másodlagos korrekciókra szorulnak. Ide tartozik a linearitás és a homogenitás korrekciója. A javított, torzítatlan kép pedig már valóban kijelezhető a felhasználók felé.

II. Pozíciószámítási eljárások

Meggondolandó, hogy pontosan mit is várunk el a pozíciószámolási algoritmusoktól. Természetesen fontos, hogy az eljárás minél hatékonyabb és gyorsabb legyen, de a leglényegesebb szempontok mégis a létrehozott kép jellemzői között keresendők, hiszen mindent nem lehet utólagos korrekciókkal kijavítani. Vizsgálható a keletkezett képek intrinsic linearitása és homogenitása, illetve nagyon fontos a kép elsődleges felbontása, és az úgynevezett hasznos látómező (UFOV, useful field of view) mérete, mivel ez utóbbi két paraméter később nem javítható jelentősen.

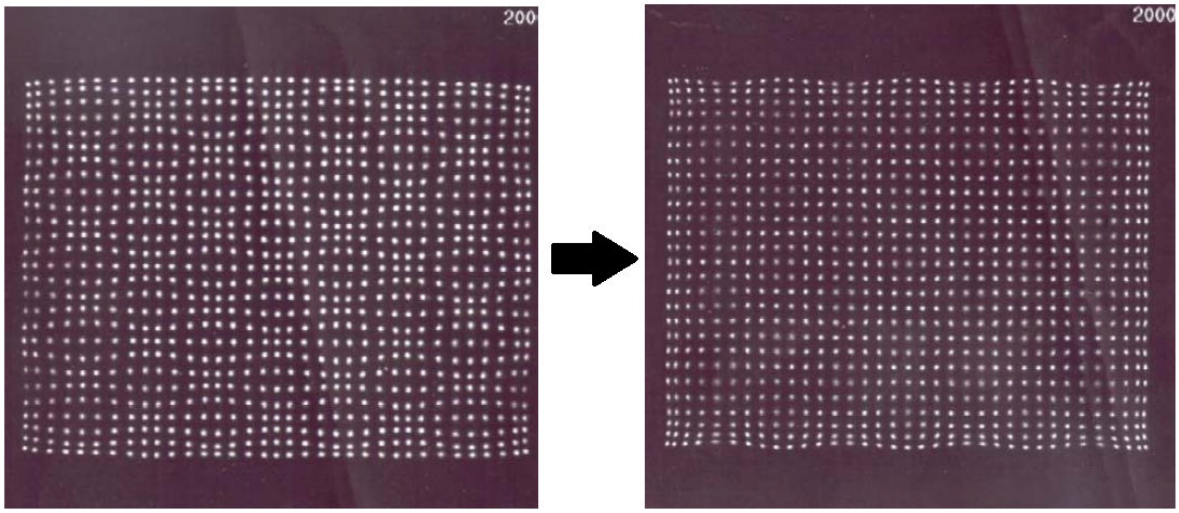
II.1. Súlyponti módszerek

Az összes determinisztikus, súlyponti elven működő számítás alapját az Anger-féle képlet (1. egyenlet) képezi. Bár ez az eljárás rendkívül gyors és kis számításigényű, számos hátránya is van. Ezek közül a legjelentősebb, hogy a kamera hasznos látómezejét az eljárás jelentősen csökkenti, ugyanis a súlypontszámítási képletből adódik, hogy a legszélső fotoelektronsokszorozók középvonalánál kijjebbről érkező fotonokat mindenképp a középvonalon belül látjuk megjelenni. Ez amellett, hogy az UFOV méretét nemkívánatos módon csökkenti, jelentős torzításokat is eredményez a kamera látómezejének szélein.

Az Anger-pozícionálásból adódó torzításokon sokat lehet javítani az úgynevezett NLCC (Non-Linear Core Correction) [3] módszer segítségével. Ez az eljárás továbbra is súlypont-számításon alapszik, de a koordinátákat már nem egyszerűen a mért PMT jelekkel való súlyozással kapjuk meg, hanem, még az Anger-képlet (1. egyenlet) alkalmazása előtt egy jelnagyság függő nemlineáris korrekciót alkalmazunk. A korrigált értékekkel számított súlypont lesz az NLCC módszerrel becsült koordináta.

A nemlineáris korrekció paramétereinek választása többnyire tapasztalati alapon történik, a paraméterek rossz beállítása esetében az NLCC módszer az eredeti Anger képalkotásnál is rosszabbul teljesít. Ugyanakkor megfelelő paraméterek mellett az NLCC eljárás alkalmazása sokat segíthet a primer kép linearitásán és homogenitásán, valamint a felbontáson is, mivel a korrekciós eljárás miatt a szcintillációtól távoli, alacsony jel-zaj arányú csövek jelét nem vesszük figyelembe a pozícióbecslésben.

Azonban mivel továbbra is súlypont-számítási képletet alkalmazunk, a módszer a hasznos látómező méretét nem növeli jelentősen, így teljes megoldást nem jelent az Anger-képalkotás problémáira. Az eljárás hatását a képalkotás minőségére a 3. ábra szemlélteti:



3. ábra: Nemlineáris korrekció hatása a primer képre, négyzetrácsban elhelyezett pontforrásokról készített felvétel esetében [3.]

II.2. Statisztikus módszerek

A statisztikus képszámítási módszerek kidolgozására éppen azért van igény, mert a súlyponti eljárásokkal bizonyos határon túl a hasznos látómező elvileg sem növelhető. A statisztikai alapú módszerek számításigénye jelentős, így futásidejük is számottevően hosszabb az előző eljárásokénál, ugyanakkor segítségükkel lényegében az összes vizsgált képjellemző (linearitás, homogenitás, felbontás és UFOV méret) javítható.

A statisztikus módszerek ismertetéséhez szükséges bevezetnünk az LRF (light response function) [4.] fogalmát. Ezek a függvények egy-egy adott PMT válaszát adják meg a helykoordináták (x,y) függvényében. Ezeket a függvényeket kimérhetjük kollimált pontforrás léptetésével a detektor előtt, vagy létrehozhatjuk akár Monte-Carlo szimulációk segítségével is.

Ha az LRF-ek kellő felbontással rendelkezésünkre állnak, és van egy mért jelünk egy ismeretlen helyzetű felvillanásból, akkor számos eljárással megvizsgálhatjuk, hogy mi volt a szcintilláció legvalószínűbb helye. A módszerek közös tulajdonsága, hogy egy olyan függvényt definiálnak, ami függ az x,y koordinátáktól, és szélsőértéke van a becsült helynél. Ez a függvény általában valamilyen metrikában távolságot mér az LRF-ek adott helyen felvett értéke és a mérési adataink között, és a metrikába a mérési adataink és az LRF-ek statisztikus tulajdonságait (Poisson-eloszlású valószínűségi változók) is beépíthetjük.

Az egyik leggyakrabban használt távolságmérték az L2 metrika. Két statisztikus módszer is felhasználja, ezek közül az egyszerűbb eljárás a legkisebb négyzetek módszere (LS, Least Square). Ez esetben a minimalizálandó függvény (2. egyenlet):

$$\varepsilon_{LS}(x,y)=\sum_i\left(S_i-LRF_i(x,y)\right)^2 \quad \textbf{(2.)}$$

Ahol S_i az i. PMT-n mért jel, $LRF_i(x,y)$ az i. PMT várható válasza egy x,y pontban történt felvillanásra.

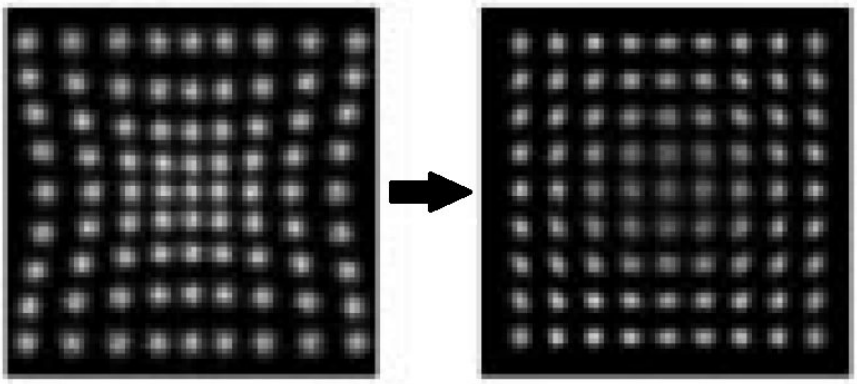
A másik eljárás a khi-négyzet módszer (CSE, Chi Square Error) [4.]. Ebben az esetben a függvény, amit minimalizálnunk kell (3. egyenlet) tartalmazza az adataink szórására (σ_{LRF_i}) vonatkozó adatokat is:

$$\varepsilon_{CSE}(x,y)=\sum_i\frac{\left(S_i-LRF_i(x,y)\right)^2}{\sigma_{LRF_i}{}^2(x,y)} \quad \textbf{(3.)}$$

A becsült koordinátákat mindkét esetben a 4. egyenlet segítségével számíthatjuk ki:

$$(\hat{x},\hat{y})=\underset{x,y}{\operatorname{argmin}}\varepsilon(x,y) \quad \textbf{(4.)}$$

A 4. ábra bemutatja a CSE módszer hatását az Anger-képalkotáshoz képest:



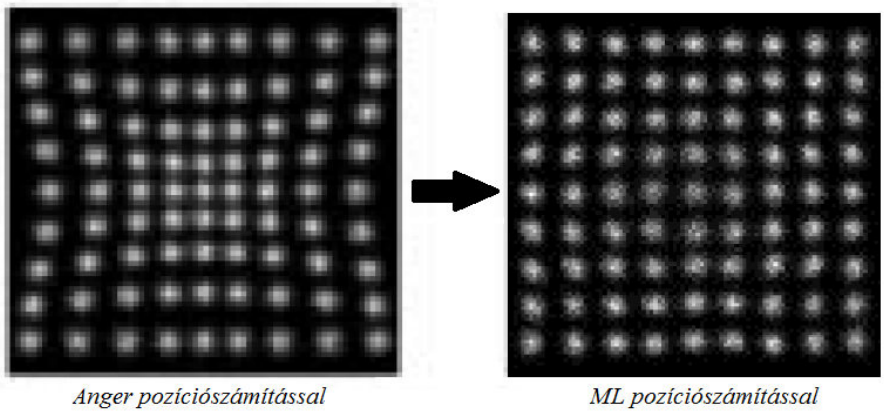
Anger pozíciószámítással *CSE pozíciószámítással*

4. ábra: A CSE eljárás pozitív hatásai a primer kép minőségére [4.]

A khi-négyzet módszer mellett kézenfekvőnek látszik a különféle maximum-likelihood [5.] eljárások bevetése is. Ez a módszer, ahogy a neve is mutatja, azon alapszik, hogy azt a helyet keressük, ahonnan egy esetleges felvillanás a lehető legvalószínűbben eredményezné a ténylegesen mért jelet. Ebben az esetben a maximalizálandó függvényünk az úgynevezett likelihood függvény, ami Poisson-eloszlást feltételezve (5. egyenlet) [5.]:

$$P(S_1,S_2..\,|\,x,y)=\pi_i\frac{LRF_i(x,y)^{S_i}\cdot e^{LRF_i(x,y)}}{S_i!} \quad \textbf{(5.)}$$

A maximum-likelihood módszerrel készített primer kép minőségét mutatja az 5. ábra:



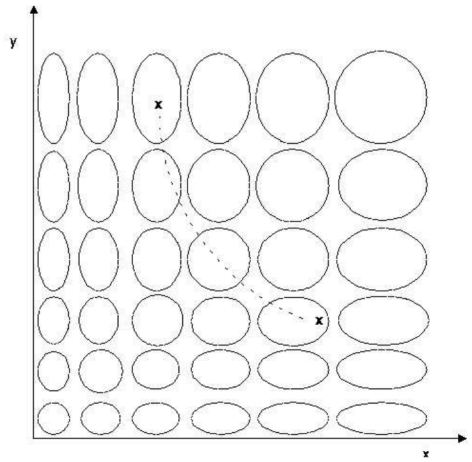
5. ábra: ML képalkotás hatása az Anger pozíciószámításhoz képest [5.]

Ahogy az 5. ábra is mutatja, a Poisson-eloszlást feltételező maximum-likelihood módszer már nagyon jó minőségű képalkotást tesz lehetővé, ugyanakkor a többi, távolságmérésen alapuló statisztikus képalkotási módszernél is jelentősen nagyobb számításigényű, így segítségével valós idejű képalkotást megvalósítani igen nagy kihívást jelenthet.

II.3. Bhattacharyya-mérték

A fentiekben bemutatott két távolságmérésen alapuló statisztikus módszert, az LS és a CSE eljárást nem kimondottan Poisson-eloszlású valószínűségi változók vizsgálatára vezették be, ugyanakkor a problémára tökéletesen illeszkedő maximum-likelihood módszer nagyon számításigényes, így kézenfekvőnek látszik valamilyen új eljárás kifejlesztése, ami távolságmérésen alapszik ugyan, de alkalmazkodik a méréseink statisztikájához.

Tudjuk, hogy Poisson-eloszlás esetén a várhatóérték és a szórás négyzete megegyezik, vagyis például két megfelelő eloszlású valószínűségi változó összehasonlítása esetén a kétdimenziós síkon a szórás nagysága folytonosan változik, így két pont közt a legrövidebb távolság nem szükségszerűen az egyenes [6.], amit az L2 metrikát használó LS és CSE módszer feltételez. Ezt szemlélteti a 6. ábra:



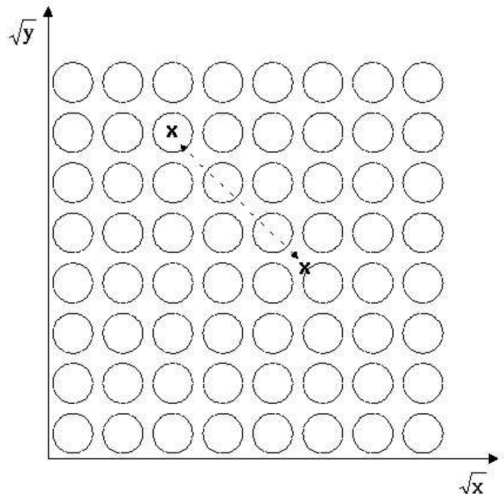
6. ábra: Legrövidebb távolság két Poisson-eloszlású valószínűségi változóból vett mérési pont között [6.]

Keressünk egy olyan transzformáló függvényt, ami ebből a változó szórású térből a valószínűségi változóinkat egy konstans szórású térbe vetíti. A levezetés igen egyszerű: legyen g a keresett függvény, ami a konstans (k) szórású térbe vetíti a Poisson-eloszlású x valószínűségi változó terét. Az x valószínűségi változó szórása $\delta x = \sqrt{x}$, így a transzformált szórás (6. és 7. egyenlet [6.]):

$$\delta g = k = \frac{\delta g}{\delta x}\delta x \approx \frac{dg}{dx}\delta x = \frac{dg}{dx}\sqrt{x} \qquad \textbf{(6.)}$$

$$\frac{dg}{dx} = \frac{k}{\sqrt{x}} \quad \rightarrow \quad g(x) = \int \frac{k}{\sqrt{x}}dx = k \cdot \sqrt{x} + c \qquad \textbf{(7.)}$$

Könnyen belátható, hogy $k = 1,\ c = 0$ esetén a keresett g függvény épp a gyökvonás, ami a Poisson-eloszlású valószínűségi változó vektorokból konstans egy szórású valószínűségi változókat generál. Ebben a térben pedig már valóban az egyenes a legrövidebb út két pont között, ahogy ezt a 7. ábra is szemlélteti:



7. ábra: Legrövidebb távolság két Poisson-eloszlású valószínűségi változó gyökéből vett mérési pont között [6.]

Ez alapján a távolság, amit definiálni érdemes, az úgynevezett Matusita-mérték, illetve az azzal egyenértékű Bhattacharyya-mérték lesz. A metrikák legegyszerűbb definíciója (8. egyenlet) n dimenziós, L1 metrikában 1-re normált vektorokra értelmezhető [6.]:

$$p, q \in R^n_{0,+} : \quad \sum_{i=1}^{n} p_i = \sum_{i=1}^{n} q_i = 1$$

$$d_{Matusita} = \sum_{i=1}^{n}\left(\sqrt{p_i} - \sqrt{q_i}\right)^2 \qquad \textbf{(8.)}$$

$$d_{Bhattacharyya} = 1 - \sum_{i=1}^{n}\sqrt{p_i q_i}$$

A két mérték között egyértelmű megfeleltetés létesíthető (9. egyenlet [6.]). Az előbbi jelölésekkel:

$$d_{Matusita} = 2 - 2\sum_{i=1}^{n}\sqrt{p_i q_i} = 2 \cdot d_{Bhattacharyya} \qquad \textbf{(9.)}$$

Könnyen belátható, hogy megfelelően normálva a mérési adatainkat és az LRF értékeket, a Bhattacharyya-mérték segítségével is definiálható minimalizálandó $\varepsilon(x, y)$ függvény egy statisztikus képalkotási eljáráshoz (10. egyenlet [6.]):

$$\varepsilon_{Bhattacharyya}(x, y) = 1 - \sum_{i}\sqrt{S_i \cdot LRF_i(x, y)} \qquad \textbf{(10.)}$$

Figyelembe véve, hogy a Bhattacharyya-mérték kezeli a Poisson-eloszlású valószínűségi változóink folytonosan változó szórásának problémáját, feltehető, hogy a Bhattacharyya-mérték segítségével előállított statisztikus eljárás mindenképpen jól használható, sőt akár pontosabb is lehet, mint a gyakran felhasznált khi-négyzet módszer.

III. Szimulációs vizsgálatok

Az egyes metrikák hatékonyságának összehasonlításához a legkézenfekvőbb, ha szimulációs vizsgálatokat, és méréseket is alkalmazunk. A következő fejezet az L2 metrikát alkalmazó statisztikus módszerek, az LS és a CSE metódus, valamint az eredeti Anger-elvű képalkotás, és az általam javasolt Bhattacharyya-mértéken alapuló statisztikus képszámítási módszer hatékonyságának összevetésére írt szimulációk leírását tartalmazza.

III.1. Szimulált gamma kamera tulajdonságai

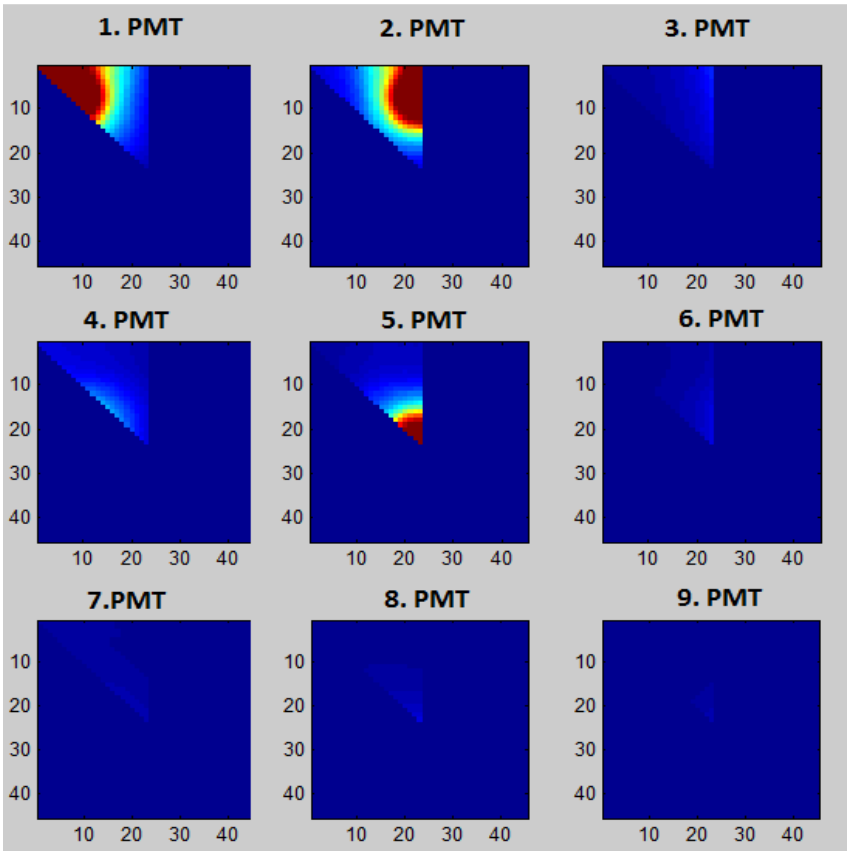
Az egyes statisztikus módszerek, illetve az eredeti Anger-képalkotás teszteléséhez egy szimulált gamma kamera mérési adatait kaptam kézhez. A hipotetikus kamera 3x3 fotoelektronsokszorozóból áll, amik négyzetrács alakban vannak elhelyezve a NaI szcintillációs kristály és a fényosztó üveg alatt. A PMT-k aktív átmérője 70 mm, egymástól pedig 76 mm-re vannak elhelyezve, a látómező pedig egy összesen 225,44mm x 225,44mm-es négyzet, 45x45 pixelre osztva.

A mérési eredmények PetDetSim [7.] optikai Monte Carlo szimulációval készültek, a gerjesztés, egy hipotetikus pontforrás a látómező 1/8-án haladt végig, egy felső háromszögben (szimmetriai okok miatt ebből már a teljes látómező adatai rekonstruálhatóak). A szimulációk során minden vizsgált pixelre 1000 szcintillációs esemény lett lefuttatva, a szcintillációk mélységét a kód exponenciális eloszlás alapján sorsolta, 4 mm várhatóértékkel, így modellezve a gamma-foton elnyelődését. Más hatások, mint a Compton-szórás, valamint az átszóródás szomszédos pixelekbe, nem voltak modellezve.

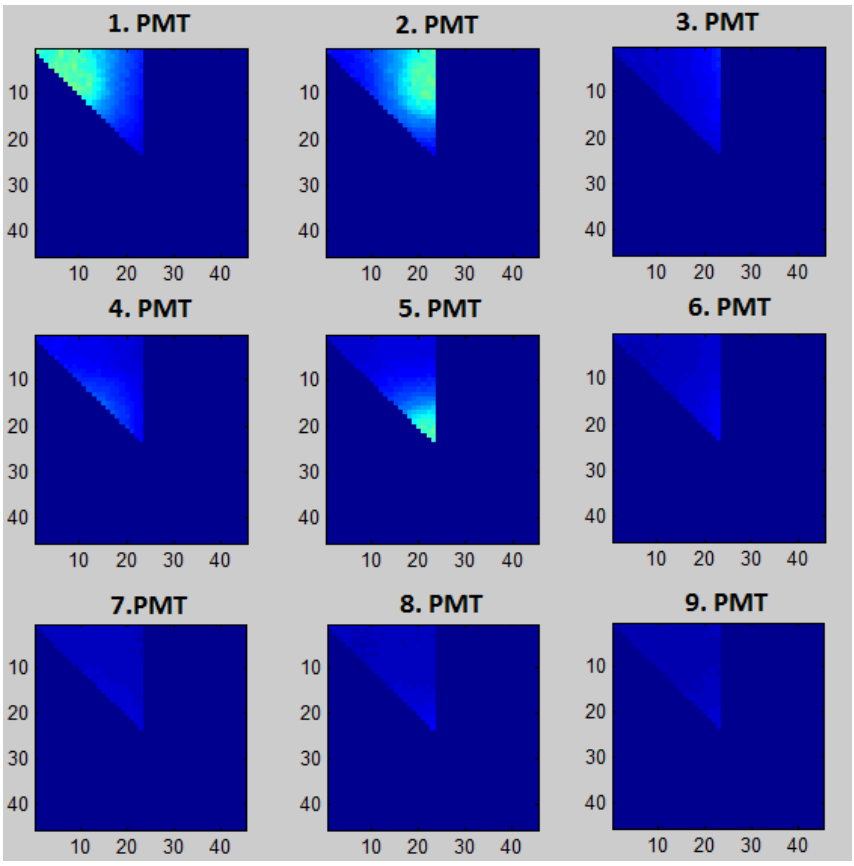
A kézhez kapott adatokból a 9 PMT LRF és VAR adatai könnyen megkaphatóak, hiszen minden pozícióban és minden szcintillációs eseményhez adott külön-külön a 9 PMT időintegrált jele. Így minden pozícióban megkaphatjuk 1000 eseményből a jelek átlagértékét (LRF) és szórásnégyzetét (VAR).

A mérési adatokat tartalmazó .dat fájlok feldolgozását MATLAB segítségével végeztem el. Az lrferedeti.m fájl beolvassa az eredeti .dat fájlokat, és azokból először egy 276x9x1000-es adattömböt készít (276 pozíció, 9 PMT, 1000 mérés/pozíció), majd ebből egyszerű átlagolással megkapjuk a 276x9-es LRFdata tömböt, illetve a négyzetek átlagából kivonva az LRF adatok négyzetét, a szintén 276x9-es Var tömböt. Ezeket az adatokat érdemes átrendezni. Tudjuk, hogy a mérési adatok mely pixelekhez tartoznak, így létrehozhatunk az eredeti látómezőnek megfelelő 45x45x9-es tömböket is az egyes PMT-khez tartozó LRF és VAR adatok tárolására.

Az így kapott LRF és VAR adatokat mutatja a 8. ábra és a 9. ábra:

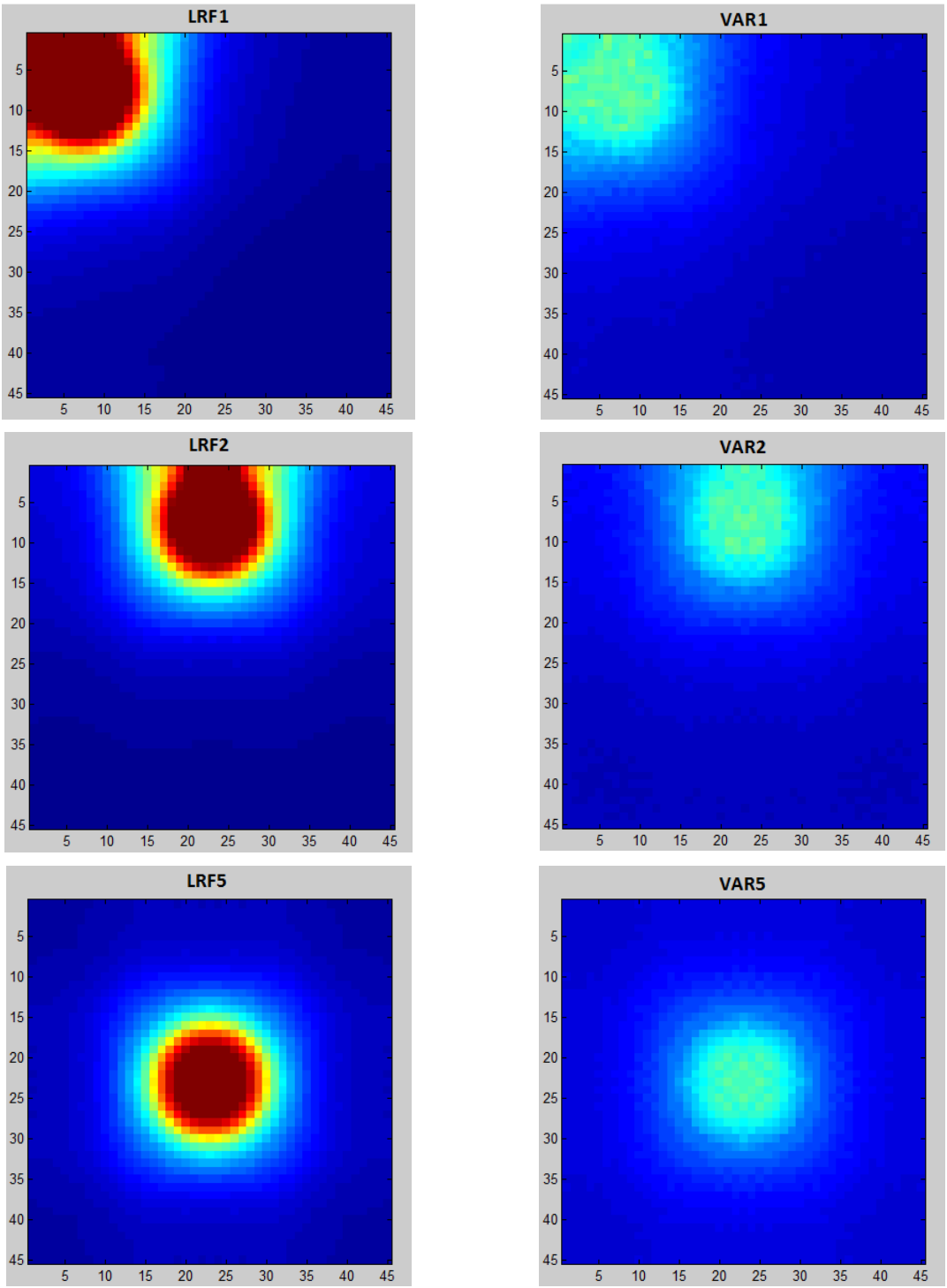


8. ábra: A szimulált mérési adatokból kapható LRF adatok



9. ábra: A szimulált mérési adatokból kapható VAR adatok

A fenti adatokból szimmetriai megfontolások segítségével minden egyes PMT teljes LRF és VAR táblázata megkapható, ezt a rendezést szintén az lrferedeti.m fájl végzi el. Az 1., 2. és 5. PMT teljes táblázatait mutatja a 10. ábra:



10. ábra: 1.,2., és 5., vagyis egy sarok, egy szélső és a középső PMT teljes LRF és VAR táblázatai

Az egyes PMT-k adatainak ismeretében már lehetőségem nyílt újabb mérések szimulációjára is, amiket az egyes képszámítási algoritmusok összehasonlítására alkalmazhattam.

III.2. Mérés-szimulációk működési elve

Saját szimulációim futtatásánál elsősorban arra voltam kíváncsi, hogy az egyes közismert statisztikus módszerek hogyan teljesítenek a hagyományos Anger-elvű képalkotáshoz, illetve az általam javasolt Bhattacharyya-mértéket felhasználó eljáráshoz képest.

Mivel az LRF és VAR táblázatok 45x45 pixeles felbontásban álltak rendelkezésemre, vizsgálataim során képeket is csak ilyen felbontásban állítottam elő. Az összehasonlítás elve a következő volt: kijelöltem egy x,y koordinátát, ahol a szimulált felvillanás történik. Ezután a program beolvassa a 9 LRF és VAR táblázatból a megfelelő helyről érkező fény által keltett jel várható értékét és annak szórását az egyes PMT-kre.

A várhatóértékekből képzett 9 hosszú vektor körül generálhatunk Poisson-eloszlású randomszám vektort, ezek lesznek a szimulált mérési értékeink. A szimulált mérési adatok ismeretében aztán az egyes algoritmusok segítségével kiszámíthatjuk a felvillanás általunk érzékelt koordinátáit is, amiből képet is alkothatunk.

III.3. MATLAB szimulációk

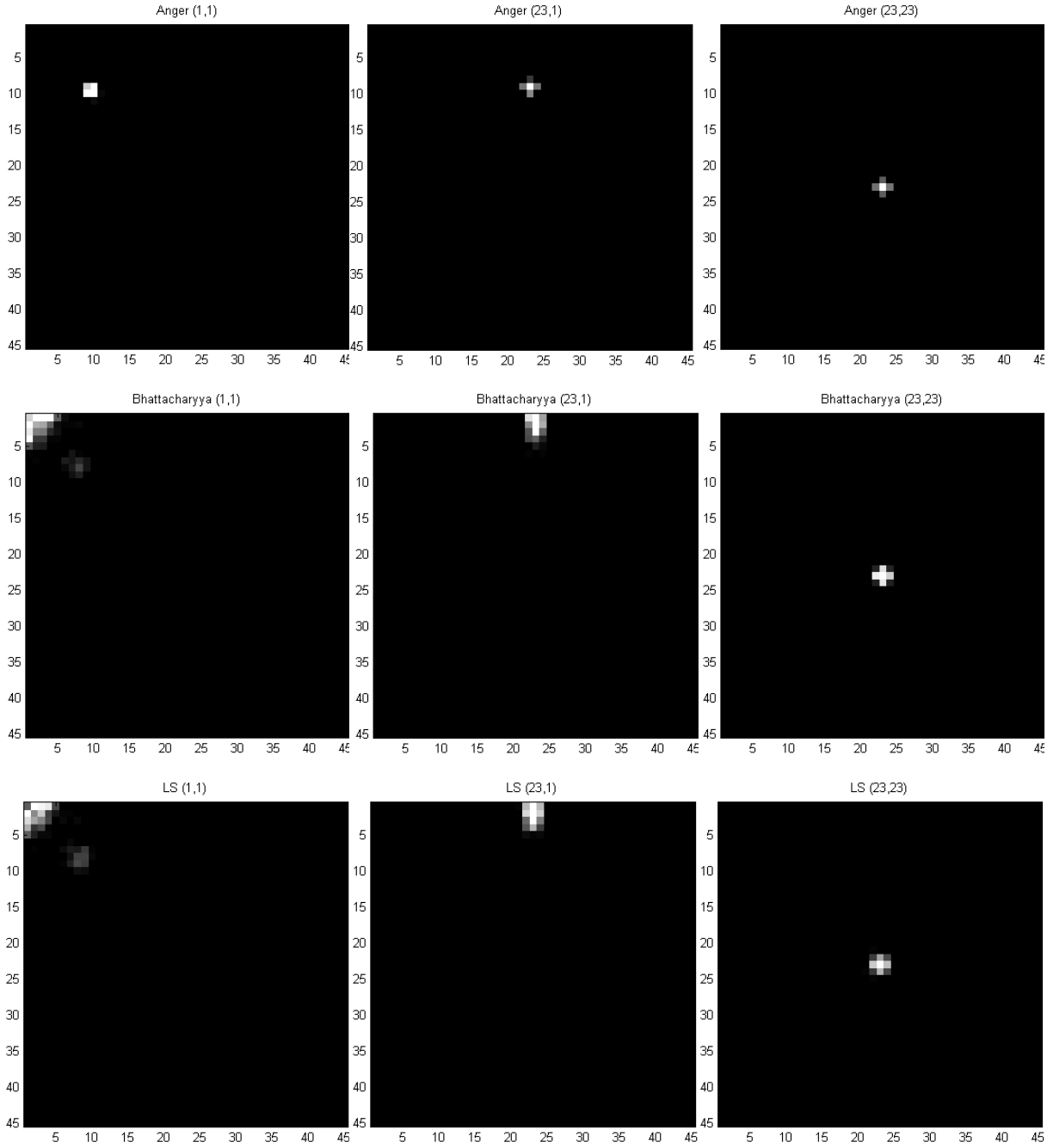
A mérések szimulációját az adatfájlok kiértékeléséhez hasonlóan MATLAB segítségével végeztem el. A szimulaciok.m fájl egyetlen futása során 46 különböző pozícióban, a látómező átlója mentén, illetve a négyzet egyik oldalfelezője mentén a szélétől a FOV közepe felé haladva 23-23 pontban szimulált darabonként 10000 szcintillációs eseményt. Vagyis a fenti leírásnak megfelelően mind a 46 pozícióban megkeressük a várhatóérték vektort, azaz a 45x45x9-es LRF tömb megfelelő helyzetű 9 elemű oszlopát, és ekörül a MATLAB beépített Poisson-randomszám generáló függvényével generáljuk a 10000 „mérési” eredményt.

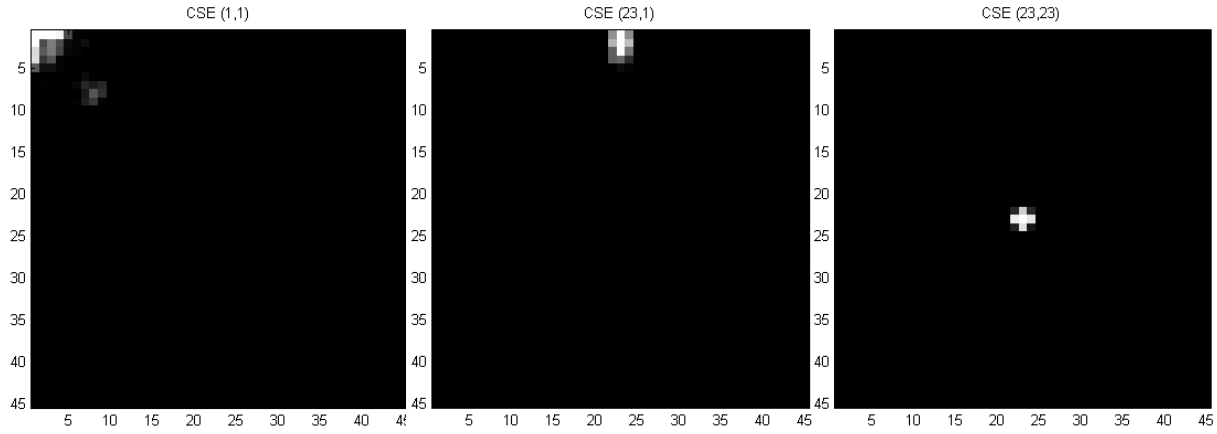
A szimulaciok.m fájl elvégzi a képalkotást is, a vizsgált négy metódussal, az Anger-módszerrel, és az LS-, CSE- illetve Bhattacharyya-eljárással is. Anger-elvű képalkotás esetén egyszerűen minden mérési eredményre a súlypont-számítási képletet alkalmazva megkapható a felvillanás helyének becsült koordinátája, ezt mind a 10000 mérésre kiszámíthatjuk, és a 46 pozíciónak megfelelően 46 képen eltárolhatjuk, hogy az egyes pixelekben hány eseményt észleltünk.

A statisztikus képszámítási módszerek esetén az egyes mérési adatokból a minimalizálandó $\varepsilon(x, y)$ függvény értékét kiszámítjuk minden egyes pixelre, az így nyert $\varepsilon(x, y)$ mátrix abszolút minimumát pedig a MATLAB beépített függvényei segítségével könnyen megkaphatjuk. A minimumhely koordinátái adják a becsült pozíciót, így a 46 képen

ezeket ábrázolhatjuk. A három statisztikus módszer, ahogy ezt a bevezetőben is leírtam, csupán $\varepsilon(x, y)$ számítási módjában tér el egymástól.

Az .m fájl végül a négy eljárással készített 45x45x46-os (46 képből álló) tömböket elmenti. Az egyes képek ábrázolhatóak, illetve különböző szempontok szerint elemezhetőek. Az (1,1), (23,1) és (23,23) pozícióban elhelyezett pontforrásról az egyes képalkotási eljárásokkal készített felvételeket mutatja a 11. ábra:





11. ábra: MATLAB szimulációkból készített képek a hipotetikus pontforrás egyes helyzeteiben

A 11. ábra képein jól látható, hogy az Anger képalkotás a látómező szélén látható felvillanásokat rossz helyre képezi le, ezzel szemben a statisztikus módszerek meglehetősen jól teljesítenek, bár a jobb leképezés a félértékszélesség rovására megy.

III.4. Felbontás vizsgálata az egyes képalkotási módszerek esetén

Definiáljuk a képek felbontását a következőképpen (11. egyenlet):

$$R(x) = w(x) \cdot \left(\frac{dX_{\text{érzékelt}}(x)}{dx} \right)^{-1} \qquad \textbf{(11.)}$$

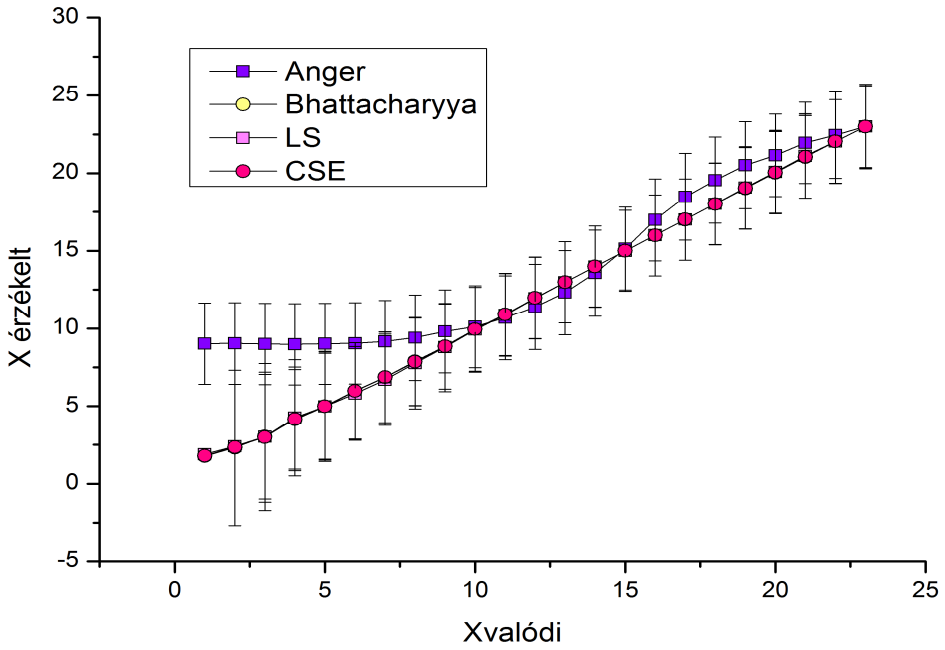
Ahol $w(x) = FWHM(x)$, egy pontforrás képének félértékszélessége az adott x koordinátájú pontban, $X_{\text{érzékelt}}(x)$ pedig azt adja meg, hogy az adott képalkotási módszer az x pontban elhelyezett pontforrás képét hol jeleníti meg. Optimális képalkotás esetében $X_{\text{érzékelt}}(x) = x$, vagyis a fenti derivált, és annak inverze is 1 minden pontban, valamint egy jó módszertől elvárjuk azt is, hogy a kiszélesedés mértéke a lehető legkisebb legyen.

A felbontás méréséhez szükséges adatok kiszámítására a kiertekelo.m fájl szolgál. Az érzékelt koordinátákat a megfelelő pozíciókhoz és képalkotási módszerekhez tartozó képek esetében a súlypont koordinátáival közelítettem, míg a FWHM értékeket NEMA protokoll [8.] alapján kaptam meg, úgy, hogy a középpont körüli öt mérési pontra illetett parabola félértékszélességét számítottam ki. Ez az eljárás a statisztikus módszerek legszélső egy-két pozícióban készített képeitől eltekintve működőképesnek bizonyult. Becsült koordinátákat és FWHM értékeket (x és y irányút) minden pozícióban számítottam.

Az Anger-elvű képalkotás FWHM értékei közel azonosak abban a tartományban, ahol a módszer leképezésre képes. Ezzel szemben érdekes megfigyelés, hogy statisztikus képalkotási módszerek esetén a FWHM értékek jelentősen változhatnak. Ugyanakkor fontos észrevétel, hogy adott irányú félértékszélesség jelentősen csak akkor változik, ha arra merőleges irányban

elmozdulás történt a pontforrás pozíciójában, vagyis például a négyzet oldalfelező merőlegesére eső 23 pozíció esetében jelentősen csak az elmozdulással párhuzamos, y irányú FWHM érték változott, x irányban a kiszélesedés gyakorlatilag változatlan volt. Ezzel szemben a négyzet átlója mentén elhelyezett 23 pontforrás képén a két különböző irányú félértékszélesség közel azonos, és a látómező széle felé növekszik.

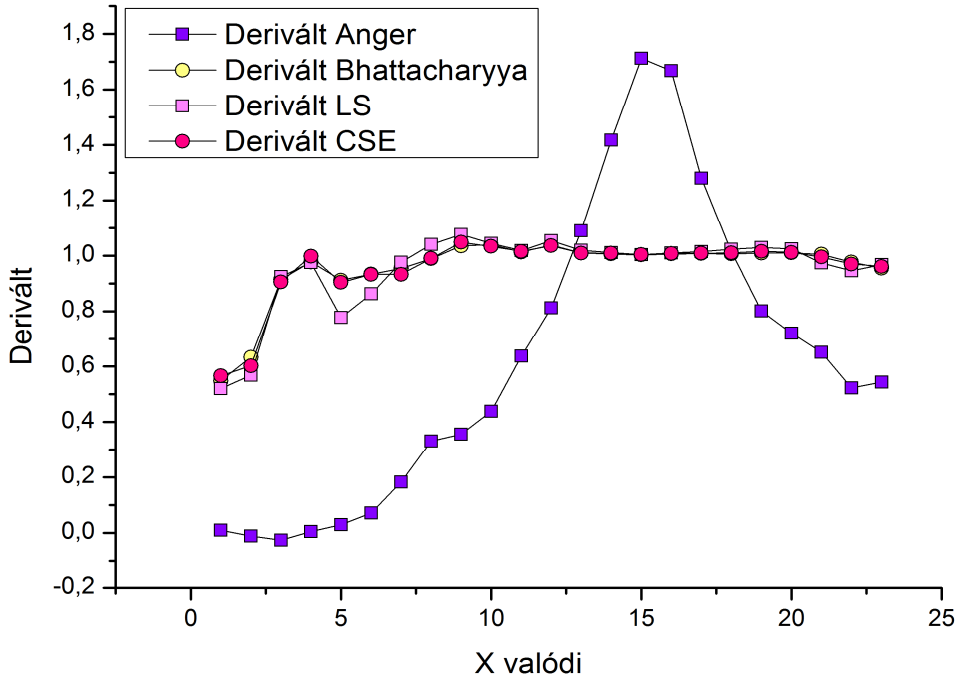
A felbontás vizsgálatához a négyzet oldalfelezője mentén elhelyezett 23 pozíció adatait vizsgáltam meg. Az $X_{érzékelt}(x)$ függvényeket, és a hozzájuk tartozó FWHM értékeket ez esetben nem a kétdimenziós helyzet, hanem a pozíció y koordinátái, illetve a becsült y koordináták, valamint az y irányú félértékszélesség jelenti, ezek viselkedését figyelhetjük meg a 12. ábrán:



12. ábra: $X_{érzékelt}(x)$ függvények az egyes képalkotási módok esetében, a mérési hiba az y irányú FWHM

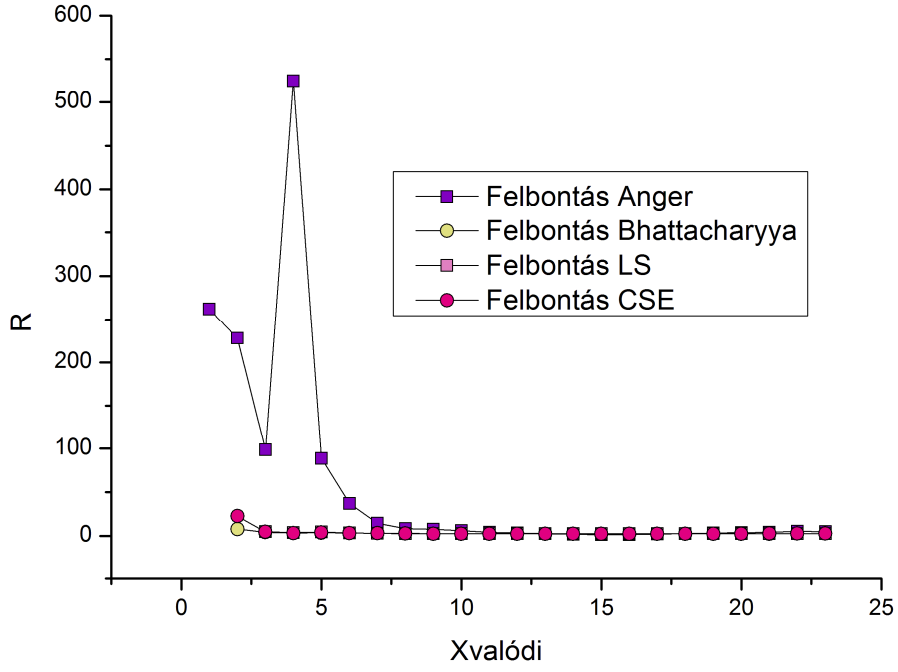
Ahogy az a 12. ábrán látható, a három statisztikus képszámítási eljárás leképezése ebben a felbontásban közel azonos, az eredeti Anger-elvű képalkotáshoz viszonyítva mindhárom jelentős előrelépést mutat, hiszen az $X_{érzékelt}(x)$ függvényeik jóval közelebb esnek az identitásfüggvényhez.

Az egyes statisztikus módszerek közti különbség sokkal jobban látható, ha az $X_{érzékelt}(x)$ függvényeik deriváltjait ábrázoljuk (13. ábra):



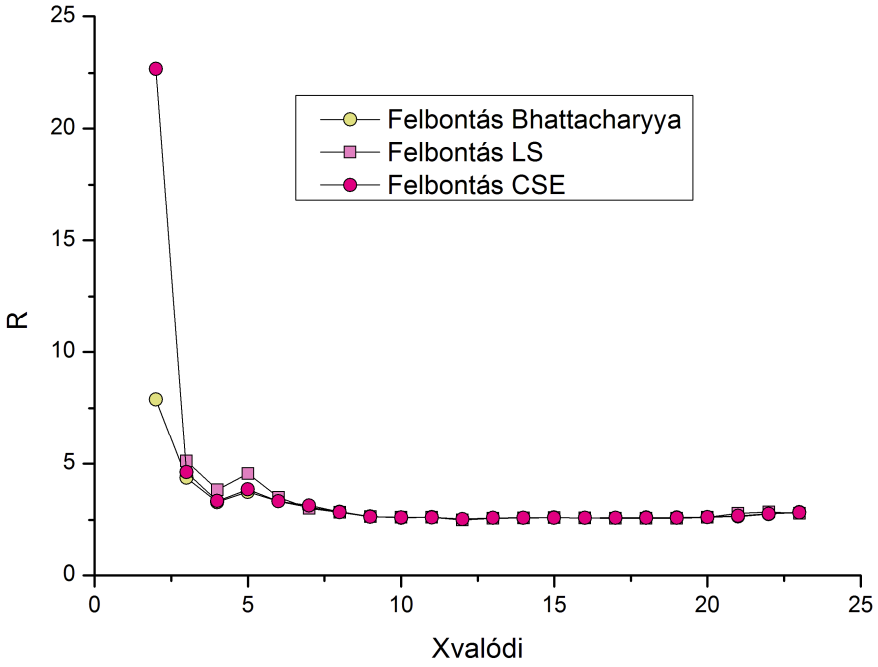
13. ábra: $X_{érzékelt}(x)$ *függvények deriváltjai az egyes képalkotási módszerek esetén*

A deriváltak és az FWHM értékek ismeretében a módszerek felbontás adatai meghatározhatóak. Mind a négy módszer felbontását egy grafikonon mutatja a 14. ábra:



14. ábra: Felbontás a négy képalkotási módszer esetében

Mivel az Anger-módszer esetében a $\left(\frac{dX_{érzékelt}}{dx}\right)^{-1}$ értékek a látómező szélén igen nagyra nőnek, az Anger képalkotás felbontása nagyságrendekkel nagyobb a statisztikus módszereknél. Hogy a statisztikus módszerek közti különbségeket jobban láthassuk azok felbontását külön is ábrázoltam:



15. ábra: A három statisztikus képszámítási módszer felbontása

Látható, hogy a három statisztikus képalkotási módszer közül, a számunkra legrelevánsabb tartományban, a látómező szélén, a Bhattacharyya-metrikát használó módszer felbontása a legjobb. Néhány szélső pozícióhoz tartozó felbontás-értéket tartalmaz az 1. táblázat:

	Anger	**Bhattacharyya**	**LS**	**CSE**
(23,2)	228,15	7,88		22,69
(23,3)	98,77	4,37	5,11	4,62
(23,4)	524,66	3,29	3,83	3,34
(23,5)	88,64	3,74	4,55	3,85

1. táblázat: Felbontás értékek az egyes képalkotási módok esetén (pixelekben)

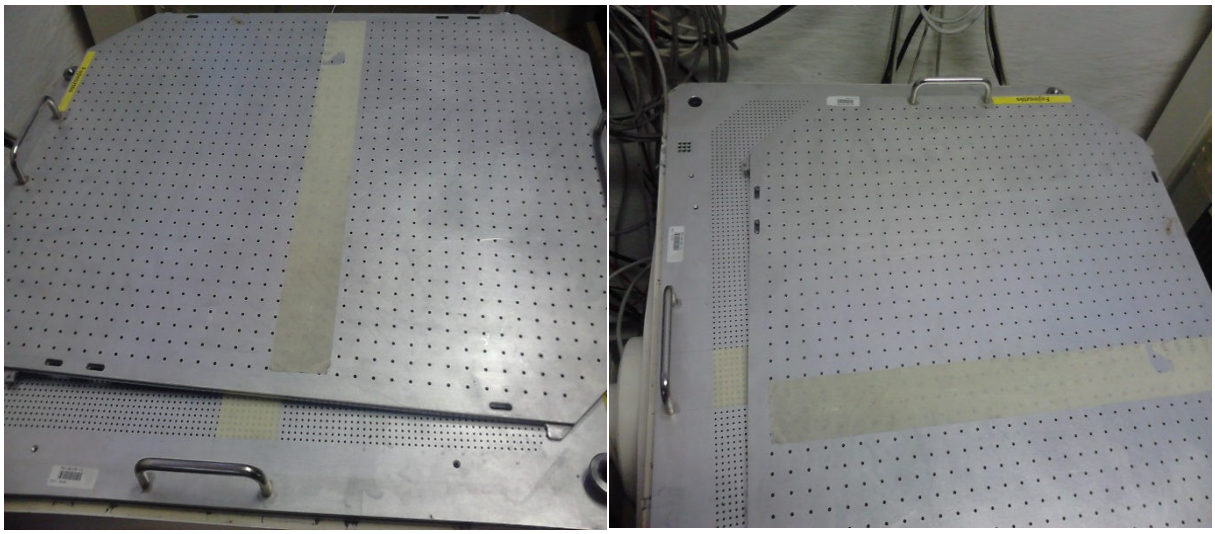
A számszerű értékekből még a 15. ábra grafikonjainál is jobban látszik, hogy a szimulációs vizsgálatokon a Bhattacharyya-módszer teljesített legjobban. Ezek alapján az általam javasolt metrika mindenképpen érdemes további, valós mérések általi vizsgálatokra is.

IV. LRF és VAR függvények kimérése

Ahhoz, hogy a különböző képszámítási eljárásokat mérések alapján is összehasonlíthassuk, és megvizsgálhassuk a Bhattacharyya-metrika használhatóságát, szükség volt egy valódi gamma kamera LRF és VAR adatainak kimérésére is. A mérési módszerek leírását tartalmazza a következő fejezet.

IV.1. A mérési összeállítás

A válaszfüggvények kimérését a Mediso Kft. egy, 60 PMT-t tartalmazó gamma kameráján végeztük el. A mérésekhez speciális pontrács fantom állt rendelkezésre, ami két különböző furatokkal rendelkező ólomlemezből állítható össze. Az alsó lemez egymástól 5 mm-re négyzetrácsban elhelyezett furatokkal rendelkezik, míg a felső lemez 15 mm-es rácstávolsággal készült és 31x41 furatot tartalmaz (a sarkok kihagyásával). A két ólomlap úgy van kialakítva, hogy a rögzítés során a felső lemez az alsóhoz képest kilenc pozícióban rögzíthető, egy 3x3-mas négyzetrács mentén, úgy, hogy az alsó lemez kilenc furatából mindig csak egy látszódjon. A pontrács fantom látható a 16. ábrán:



16. ábra: A mérésekhez használt pontrács fantom

A méréseink során használt forrás ^{99m}Tc izotóp volt, a mérési adatokat pedig Nucline szoftver segítségével gyűjtöttük be, ami a további feldolgozások előtt alkalmas arra is, hogy a list-módú adatokat .dat formátumban elmentse, így a fájlból más lementett adatok, például az órajel mellett az egyes PMT-k beütésszámai is kinyerhetőek.

Az LRF és VAR függvények kiméréséhez tizennyolc mérést végeztünk, a pontrács fantom minden lehetséges helyzetében kettőt. Egy mérést végeztünk az UFOV közepén található furatokról, egyet pedig a rács szélén elhelyezkedő sorokról és oszlopokról külön. A

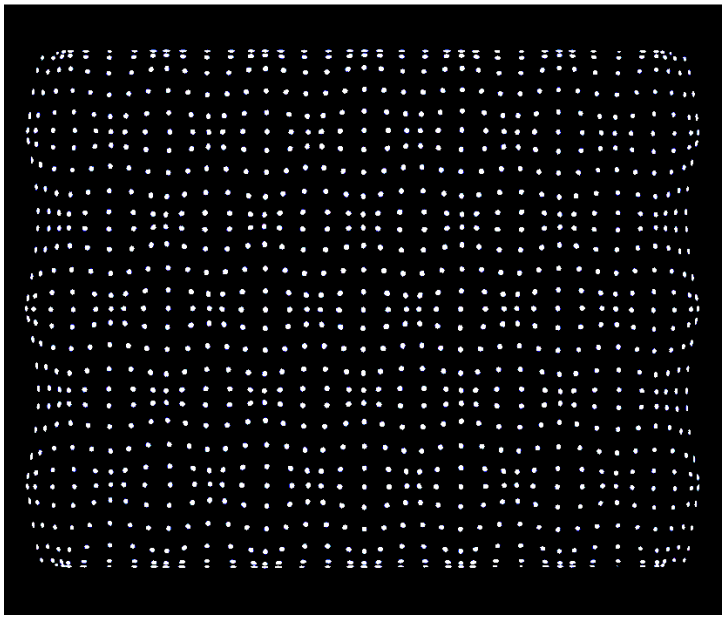
méréseink hosszát úgy választottuk meg, hogy a .dat fájlok mérete a középső helyzetben 4 GB körül alakuljon, ez 30 millió beütésnek megfelelő adatmennyiség, szélső helyzetben 1 GB adatot vettünk fel, így itt nagyjából 8 millió beütést számoltunk le. Az így készült felvételeken így a fantom helyzetétől függően egy 29x39-es vagy 30x39-es pontrács képe került rögzítésre.

IV.2. A mérési eredmények kiértékelése

IV.2.1. .dat fájlok feldolgozása

A mérési adatok feldolgozását legegyszerűbben a Nucline szoftver segítségével lehetett megvalósítani. A további számításokhoz az adatainkra olyan formában volt szükség, hogy az egyes beütések esetében rendelkezésre álljon a Nucline által jelenleg használt NLCC módszerrel kiszámított koordináta, és mind a 60 fotoelektronsokszorozó beütésszáma is. Ezeket az adatokat könnyen ki tudtam íratni a Nucline statisztikus képszámításhoz használt függvényének pár soros kiegészítésével. A statisztikus koordináták kiszámításához használható függvény ugyanis inputként megkapja az NLCC koordinátákat és a PMT-k jelét is. Így a kód lefuttatásával olyan 30 illetve 8 millió soros .txt fájlokat hoztam létre, amelyeknek minden sora 62 elemű, az első két tag az x és az y koordináta, a következő 60 szám pedig a PMT-k beütésszámát tartalmazza meghatározott sorrendben.

Az így feldolgozott mérési adatokból NLCC képeket is létrehoztam az imagemaker.m MATLAB script segítségével, így hozzájuthattam az eredeti felvételekhez is, vagyis megtudhattam, hogy NLCC módszer segítségével a kamera milyennek látja a pontrács fantomot az adott helyzetben. A középső, 5. mérés adataival készített képet mutatja a 17. ábra:



17. ábra: Pontrács fantom képe a középső helyzetben, a felvételen 29x37 pont látható

IV.2.2. Képfeldolgozás MATLAB segítségével

Ahhoz, hogy a rendelkezésre álló adatokból LRF és VAR függvényeket hozhassak létre, elsőként a tizennyolc elkészült felvételt kellett kiértékelni. Mindegyik felvételen meg kell határoznunk ugyanis az egyes furatok képének helyét, erre a legjobb megoldás, ha a felvételeken lokális maximumhelyeket keresünk.

MATLAB segítségével a munka nagyja automatikusan elvégezhető, az ingyenesen letölthető extrema2.m [9.] függvény segítségével. Az extrema2 lokális szélsőértékeket keres, egy megadott mátrixban, és szintén megadható bemenő paraméternek az is, hogy hány szélsőértékhelyet keresünk. Ha ezt a paramétert is megadjuk a függvénynek, akkor az egyes rácspontok középpontját meglehetősen hatékonyan találja meg, bár ellenőrzésre szorul, mivel egyes rácspontokon belül néha két maximumhelyet is talál.

Az így kapott adatokat két további MATLAB függvény segítségével dolgoztam fel. A racslinearis.m függvény a gamma kamera linearitás tábláinak felhasználásával áttranszformálja a maximumhelyek koordinátáit, hogy azok könnyebben kezelhetőek legyenek a rendezést végző racsrendezes.m (illetve a racsrendezes_szel.m) függvény számára. Végeredményként a maximumhelyek x és y koordinátáit kapjuk meg, 29x39 illetve 30x39-es mátrix formában.

Miután mind a kilenc mérés során készített felvételeket kiértékeltem, megállapítható lett, hogy csupán a 12., 22. és 32. mérési elrendezésben volt 29x39 furat képe látható (a sarkokat leszámítva), az összes többi esetben, ahol a felső ólomlemez a rács rövidebb oldala mentén is el volt tolva, 30x39 furat látható. Ezek koordinátáit .xls formátumban, illetve x és y koordinátákra szétszedve is elmentettem mind a kilenc mérés kiértékelésekor. Ezekből a mátrixokból összefésülhető egy, a teljes rács adatait tartalmazó 89x117-es táblázat is. Az LRF és VAR táblázatokat a furatok koordinátáinak felhasználásával készítettem el.

IV.2.3. Zajszűrés lokális teljesenergia-csúcs értékének meghatározásával

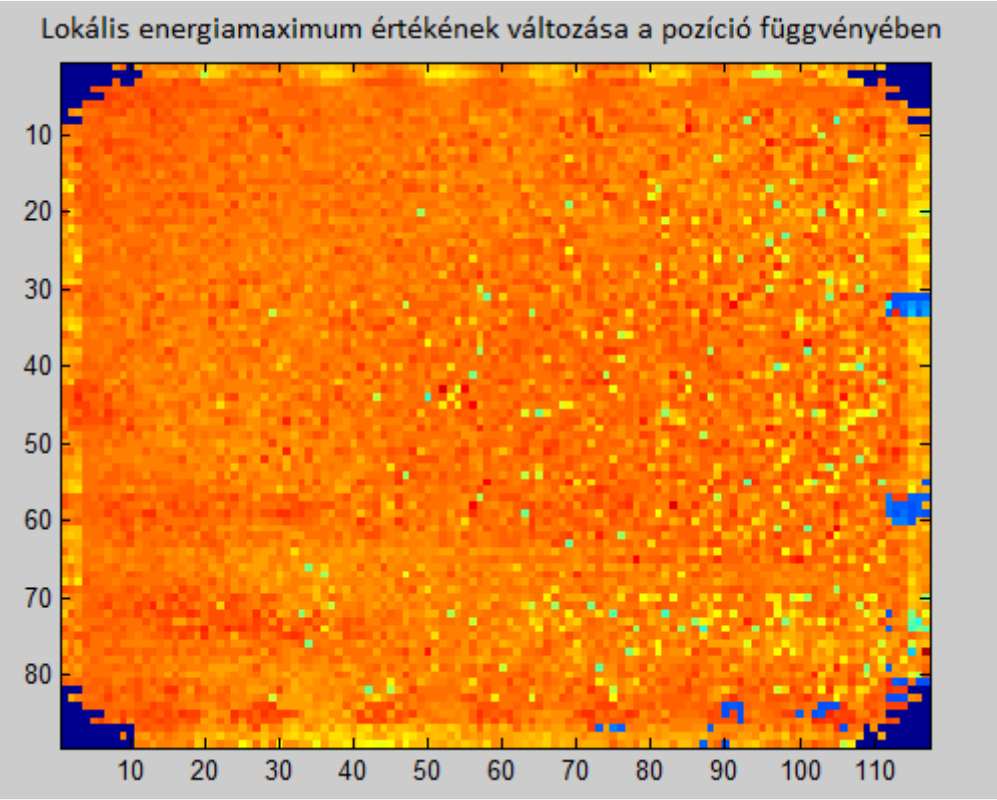
Az LRF és VAR értékek meghatározásához azon detektált szcintillációkat kell felhasználnunk, amik a ^{99m}Tc izotóp spektrumának fotocsúcsából származnak, hiszen a szórt fotonok, illetve az elektronikus zajból származó jelek hamisítanák az eredményeket.

Egy szcintillációból származó energiának a PMT jelek összegét tekintjük, ezek átlagos viselkedését ismerve a spektrumot 5-25 ezer között vizsgáltam (a jelnagyságot csatornaszámban értve) minden rácspontnál 50-es felbontásban. A 0 és 5000 közti tartományt azért nem vettem figyelembe, mert ott biztosra vehető, hogy az elektronikus zaj dominál a mérhető jelünk helyett.

A spektrumokat a mérési adatokból a lok_energiamax.m file segítségével készítettem el. A függvény soronként beolvassa a .txt adat fájlokat és először megvizsgálja, hogy a beolvasott x,y koordináták melyik furat középpontjához esnek legközelebb, majd a koordináták mögött tárolt 60 érték összegét veszi, és elmenti az adott furat spektrumának megfelelő tartományában. Természetesen külön kell beolvasnunk a látómező közepéről készített felvétel adatait, és a szélekről készült felvételeket is, különben a legszélső furatok adatait nem tudnánk szétválasztani a belső rácshelyekétől.

A fotocsúcs helye egyszerű közelítésben úgy kapható meg, hogy az így nyert spektrumok maximumhelyét keressük, esetleg ha a maximum értéket több helyen is felveszi a függvény, akkor ezek között átlagolunk. Ezen az elven kiszámítottam a fotocsúcs helyét minden mérési pontban a lokmax.m file segítségével.

Ez a módszer a mérési pontok igen nagy részénél jól működik, ugyanakkor semmiképp sem tekinthető pontosnak, sőt egyes pontok esetében teljesen kudarcot vall. A lokális fotocsúcs energia értékeket mutatja a 18. ábra:

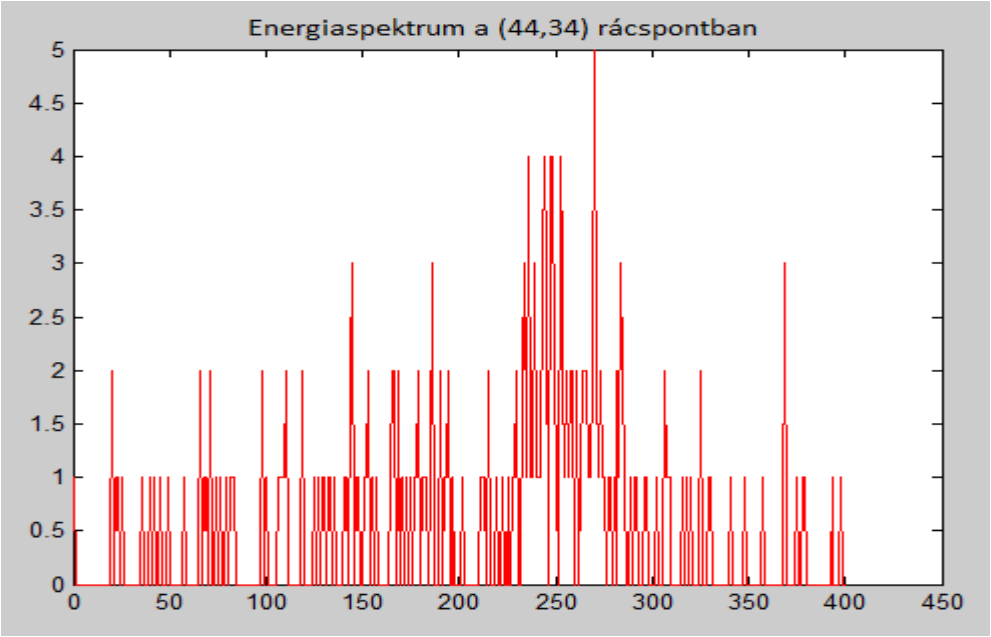


18. ábra: A lokális fotocsúcs energia értékek az automatikus maximumkereséssel

Látható, hogy a fotocsúcs a legtöbb rácsponban közel azonos helyen található (a mátrixban az átlagos fotocsúcs érték és annak szórása: 17460 ± 2662), ugyanakkor a sarkokat leszámítva is találhatóak olyan tartományok, például a kép jobb szélén, de akár a kép középső tartományában is, ahol a fenti algoritmus feltehetően rosszul működött. A módszer

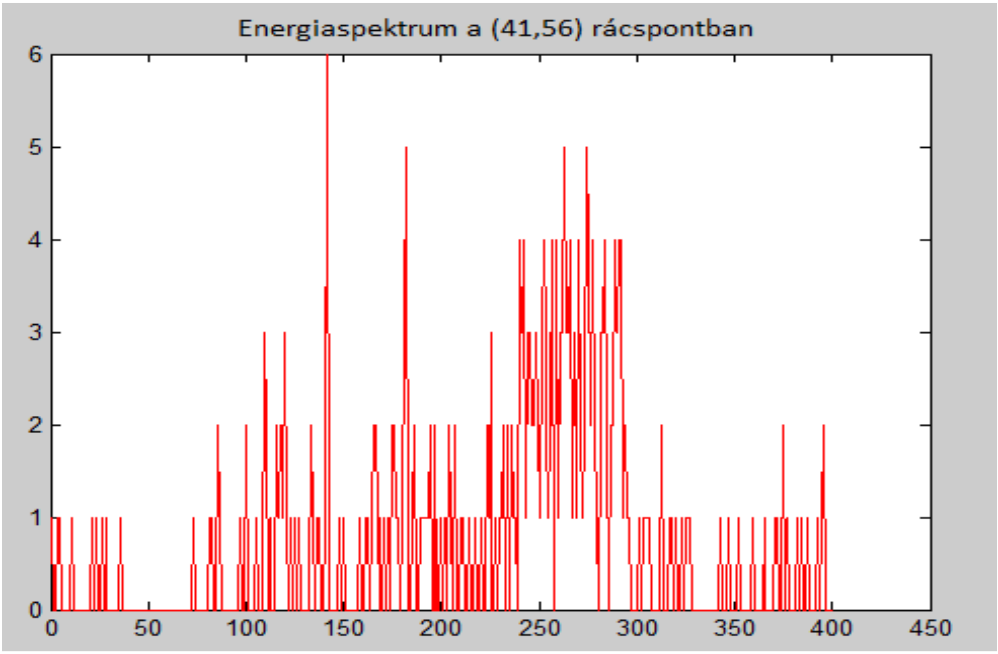
alulteljesítésének okát megérthetjük, ha megvizsgáljuk bizonyos tipikus rácspontok spektrumának alakját.

A detektor középső tartományában nehezíti a feladatot az is, hogy egyes pontokba relatíve kevés összbeütésszámot kapunk, így a spektrumok minden esetben elég zajosak, ennek ellenére a fotocsúcs legtöbbször így is kiemelkedik az adatokból (19. ábra).



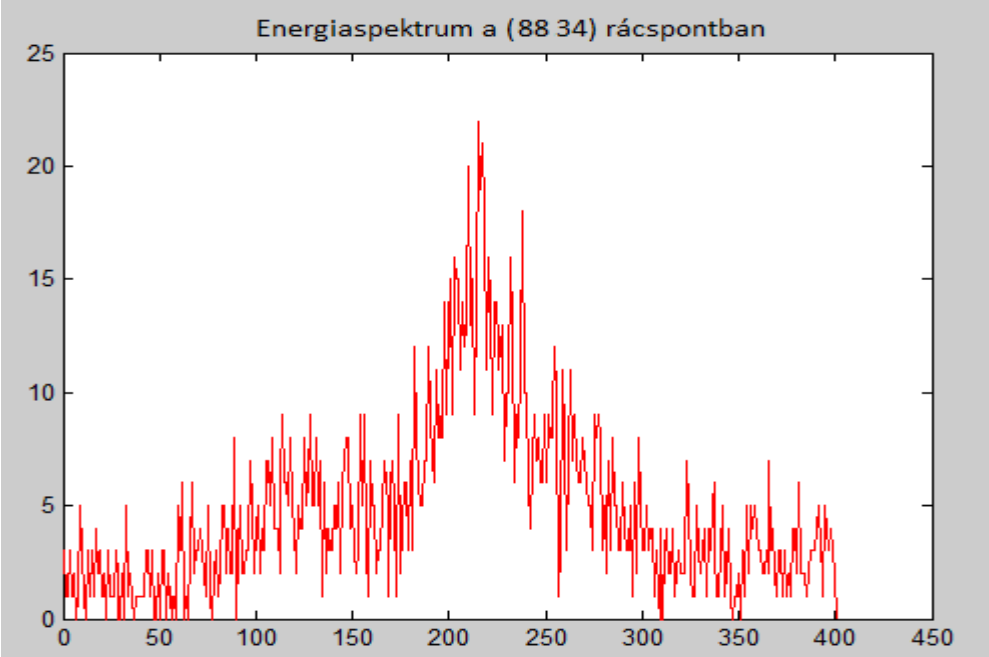
19. ábra: Energiaspektrum tipikus alakja a látómező középső tartományán

Ugyanakkor egyes pontokban előfordult az is, hogy mivel a beütésszám-különbségek kicsik, a zaj nagyobb volt a jelnél. Erre példát a 20. ábra spektrumán láthatunk:



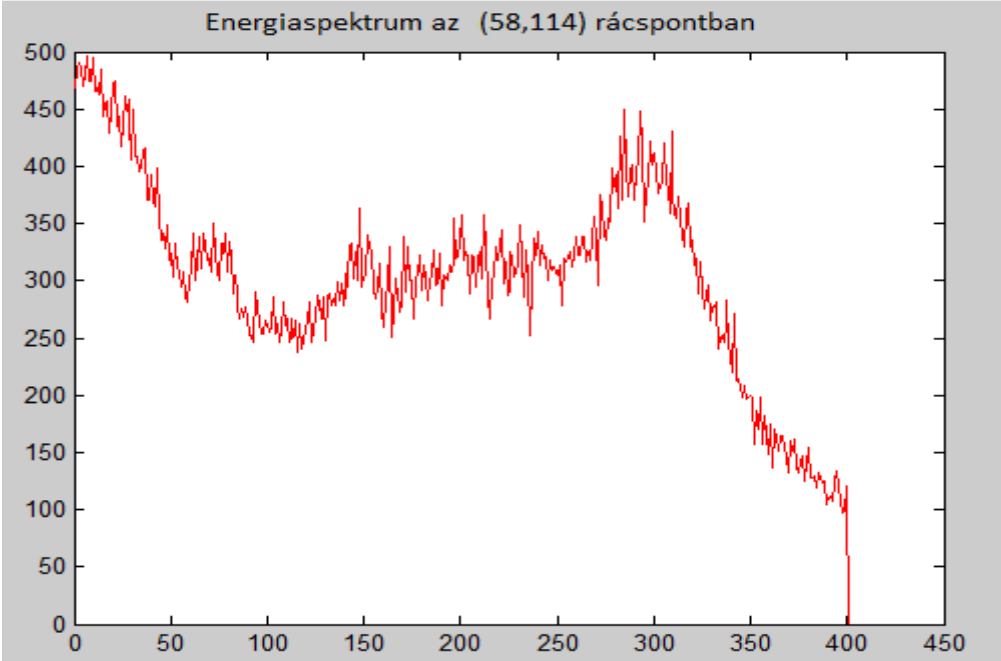
20. ábra: Egy tipikus zajos energiaspektrum alakja a látómező középső tartományán

A szélső pixelek vizsgálatánál előnyünkre szolgált, hogy a mérésekben külön vettük fel azok képeit, így ott egy-egy rácspontba több beütés jutott, vagyis a spektrumaink is szebbek. A 21. ábrán látható egy példa, ahol a fotocsúcs szépen kirajzolódik:



21. ábra: Energiaspektrum tipikus alakja a látómező szélső tartományán

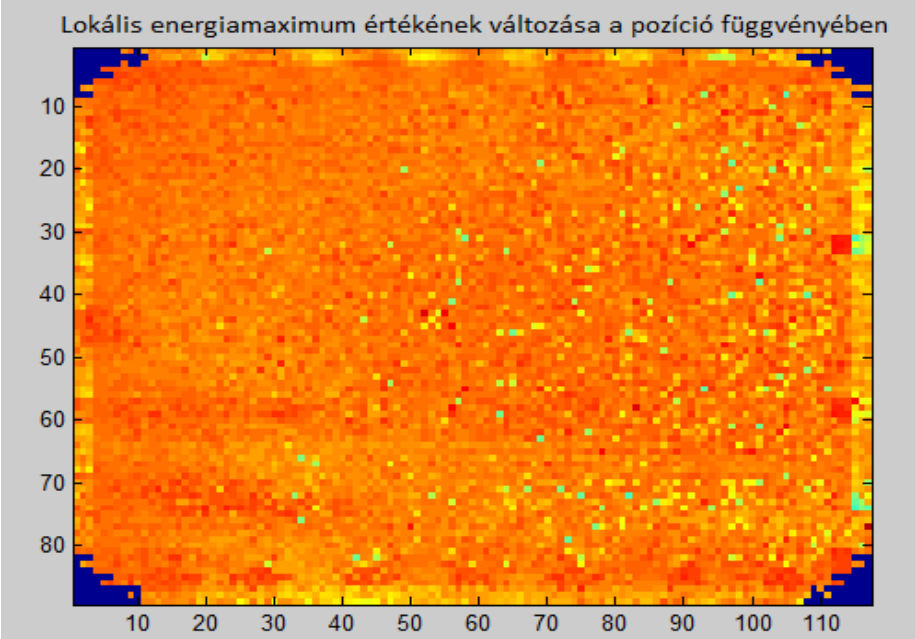
Ugyanakkor, sok esetben például a kép jobb szélén látható „kék” tartományokban a spektrum alakja az alábbihoz hasonló (22. ábra):



22. ábra: Egy tipikus zajos energiaspektrum alakja a látómező szélső tartományán

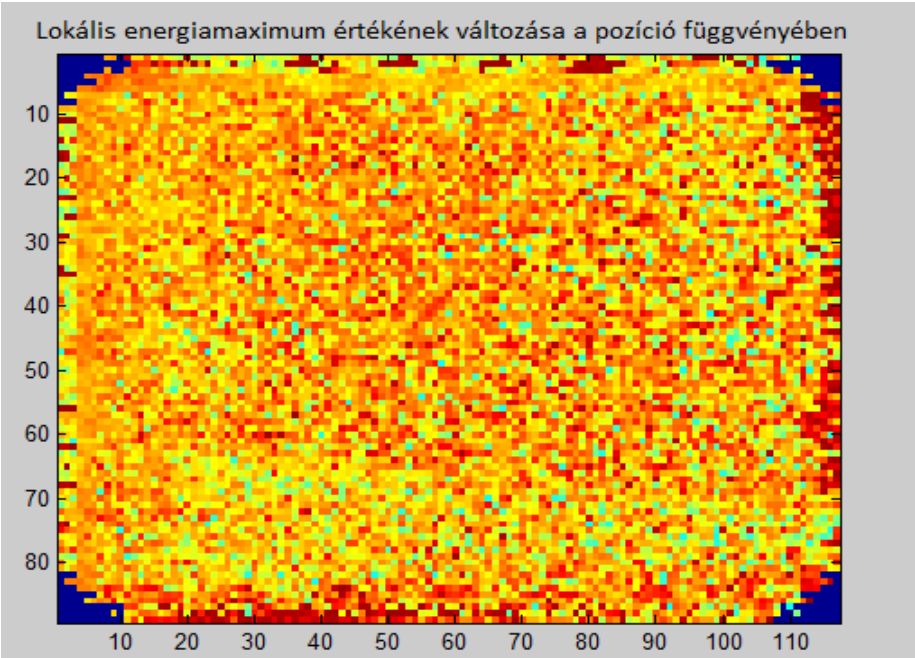
Ilyen esetben, bár a fotocsúcs szemre jól elkülönül, az abszolút maximumot mégis a spektrum legalján találjuk, mivel a zaj mértéke meghaladja a hasznos jelét.

Ezeket a problémákat részben kiküszöbölhetjük, ha magasabbra tesszük a vágási küszöböt, és például a spektrumoknak csak a 10-25 ezer közti tartományán vizsgálódunk, így a lokális energiamaximumok eloszlása a 23. ábrán látható módon módosul:



23. ábra: A lokális fotocsúcs energia értékek az automatikus maximumkereséssel és megemelt zajvágási küszöbbel

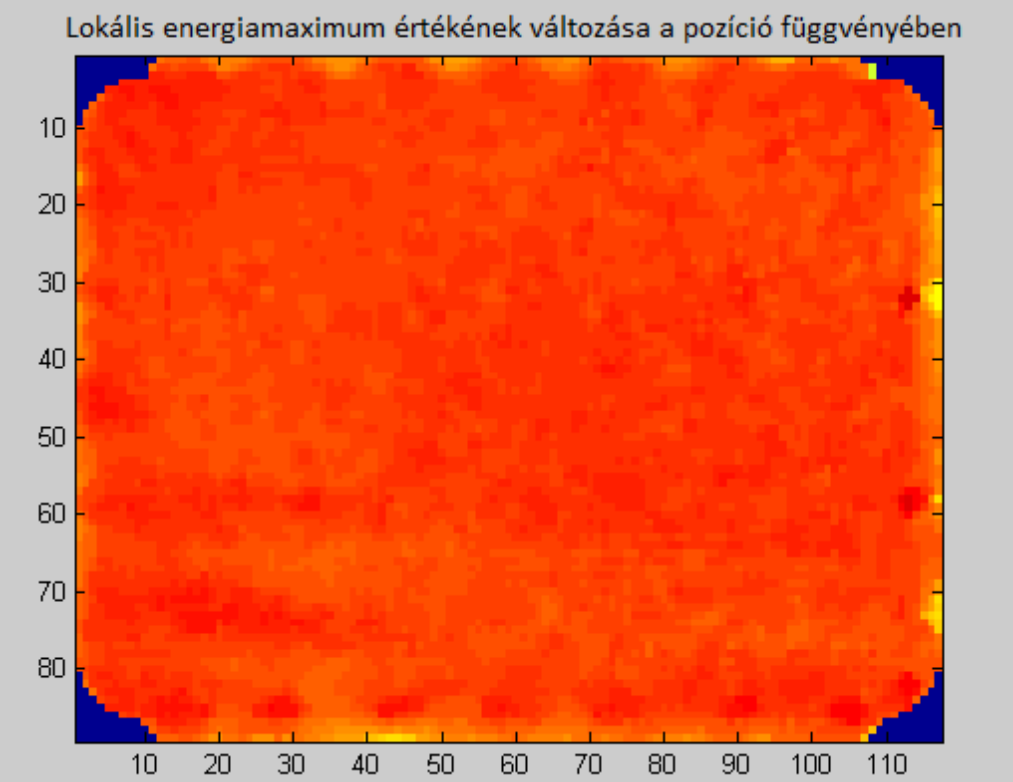
Látható, hogy a kép szélén lévő foltok így javarészt eltűntek, ugyanakkor a kép közepén lévő egyes kiszúró pontok problémáját ez sem oldja meg. (Az így kapott mátrixunk átlagos értéke és szórása: 17551 ± 2484.) Amivel próbálkozhatunk, az például a spektrum újra-binnelése, 50 helyett 150-es lépésközt használva, ez ugyanis kisebb szórású spektrumot eredményezne a kis beütésszámú középső tartományon:



24. ábra: A lokális fotocsúcs energia értékek az automatikus maximumkereséssel és újrabinneléssel

Az így eredményül kapott kép ugyanakkor a rosszabb energiafelbontás miatt sokkal nagyobb szórású, mint az eredeti, így semmiképp nem tanácsos ezt a javítási módszert alkalmazni. ($17964 \pm 3574{,}4$ a mátrixunk átlaga és szórása ebben az esetben).

Összességében a legsimább képet a második esetben kaptuk, itt a legkisebb a szórás az egyes rácspontokban mérhető fotocsúcs-energia közt. Feltételezhető ugyanakkor, hogy a lokális energiamaximum értéke folytonosan változik a térben, így a maradék kiszórópontok eltüntetése érdekében medián-szűrővel megsimítottam a képet. Ez látható a 25. ábrán:



25. ábra: A végleges lokális fotocsúcs energia értékek medián-szűréssel

IV.2.4. LRF és VAR függvények elkészítése

Miután megállapítottam, hogy az egyes furatokhoz milyen teljesenergia-csúcs érték tartozik, illetve, hogy az egyes mérések során az NLCC módszer hova képezi le az egyes furatokat, a Nucline-nal kiértékelt nyers adatok kielemezhetővé váltak. Középső helyzetben a középső furat koordinátái (512,512), ehhez képest a valóságban 5 mm-es eltolás a képen 8 pixeles elmozdulásokat eredményez, így minden rácspont koordinátája könnyen megkapható.

A feladat tehát lényegében itt is annyi, hogy soronként beolvassuk a .txt adat fájlokat és megvizsgáljuk, hogy az NLCC koordináták melyik furathoz tartoznak, majd, ha a beütésszámok összege beleesik a megfelelő lokális fotocsúcs érték körüli 20%-os tartományba, akkor a koordináták mögött tárolt 60 értéket elmentjük az adott furathoz rendelve.

Ezekből az adatokból számos jellemző tulajdonságot kiszámíthatunk. Az egyes PMT-khez tartozó jelnagyságok átlagát képezve a megfelelő furathoz tartozó LRF értékeket, az értékek négyzeteinek átlagát véve, és abból az LRF négyzetét kivonva pedig a megfelelő VAR értékeket kaphatjuk meg. Ha a 60 értéket minden esetben normáljuk a jelösszeggel, akkor ugyanilyen módon normált LRF és VAR értékeket kaphatunk. Emellett két másik jellemző adatot érdemes eltárolni, azt, hogy az egyes furatokban hány beütést számláltunk (COUNT), illetve, hogy a furatokban mi a jellemző jelösszeg (SumS).

Ezt a feladatot végzi el az lrfkeszito.m fájl, amit mind a tizennyolc mérési adatsorra le kell futtatni, a visszaadott értéke pedig két LRF és két VAR mátrix, jelösszegre kiértékelés közben normálva és normálás nélkül. Mind a négy mátrix 60 oszlopos, a 60 PMT-nek megfelelően és annyi soruk van, ahány furat látható a képeken. Emellett a .m fájlból ugyanilyen formában megkapjuk a beütésszám és jelnagyság adatokat is a COUNT és a SumS vektorban.

Ahhoz hogy a fentiekben leírt LRF és VAR mátrixból valódi LRF és VAR függvényeket hozhassunk létre még két .m fájlt írtam. Az lrf.m fájl feladata az, hogy beolvassa mind a kilenc mérési helyzethez tartozó LRF és VAR mátrixot, és azok megfelelő értékeit már egy-egy 89x117x60-as tömbben tárolja el, úgy, hogy a tömb harmadik dimenziója mentén helyezkednek el az egyes PMT-khez tartozó LRF illetve VAR értékek. Ugyanígy megkaphatjuk 89x117-es mátrix alakban a COUNT és SumS adatokat is.

A feladat így már csaknem kész, az utolsó lépést, az interpolációt az interp_smoothingspline.m fájl végzi, ami a MATLAB beépített kétdimenziós simított spline interpolációra alkalmas függvényét, a csaps-ot használja, arra, hogy a megfelelő tartományon (mivel a függvény csak négyzetes tartományon működik x és y irányban is 24-1000 között) kitöltse az 1024x1024x60-as LRF és VAR tömbök összes elemét. Ahhoz, hogy használható adatokat kapjunk a simítási paramétert 0.3-re állítottam be. Az interpoláció végeztével a .m fájl elmenti az adatokat tömbként .mat formátumban.

IV.2.5. Furatonkénti beütésszám és jelösszeg eloszlások vizsgálata

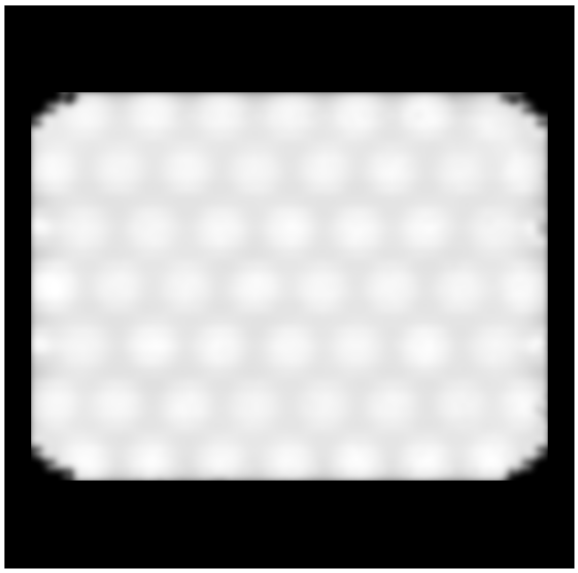
Az lrfkeszito.m függvényből a nyers LRF és VAR adatok mellett a furatok beütésszáma is kinyerhető, ezt tartalmazza a COUNT vektor, illetve vizsgálható a jelösszeg nagyságának térbeli változása is a SumS vektor ismeretében. Az lrf.m függvény felépítésével teljesen analóg módon létrehoztam egy count.m függvényt is, ami a kilenc mérési elrendezés vektoraiból 89x117-es mátrixokat hoz létre, amik kiterjeszthetőek interpolációval 1024x1024-es felbontásra is.

Az elkészült COUNT mátrix azonos formátumú az LRF és VAR adatokkal, így egyszerűen az előzőekben bemutatott interp_smoothingspline.m fájl segítségével simítható és interpolálható. Az így nyert beütésszám-adatokat ábrázolhatjuk is, ezt mutatja a 26. ábra:

26. ábra: Beütésszámok eloszlása a gammakamera látómezejében

Látható, hogy a beütésszám-eloszlás meglehetősen homogén a látómező középső illetve szélső részein, ugyanakkor tükrözi a szcintillációs kristály mögött elhelyezkedő PMT-rács szimmetriáit, hiszen közvetlenül az egyes PMT-k fölött a nagyobb érzékenység miatt valamivel hatékonyabban detektálunk.

Hasonlóan ábrázolhatjuk az interpolált SumS adatokat is (27. ábra):



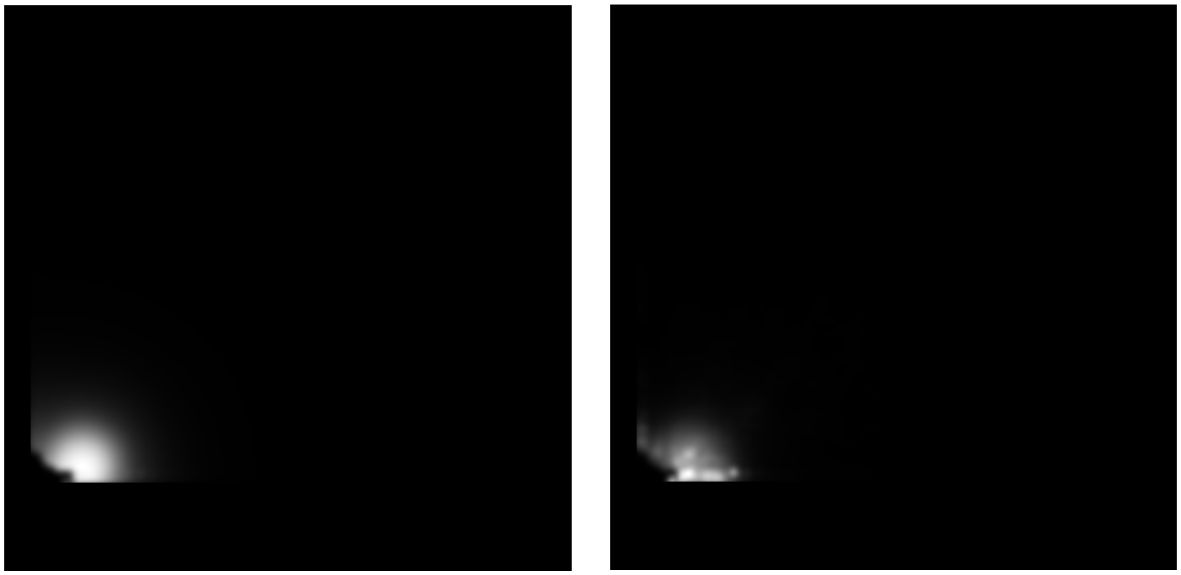
27. ábra: Jelösszeg eloszlása a gammakamera látómezejében

A jelösszeg nagysága a beütésszámok eloszlásához hasonló jelleget mutat, vagyis közel egyenletes, de közvetlenül a PMT-k felett valamivel nagyobb értéket vesz fel, mivel ezeken a területeken lehet a legtöbb szcintillációs fotont detektálni.

IV.2.6. LRF és VAR függvények kiértékelése

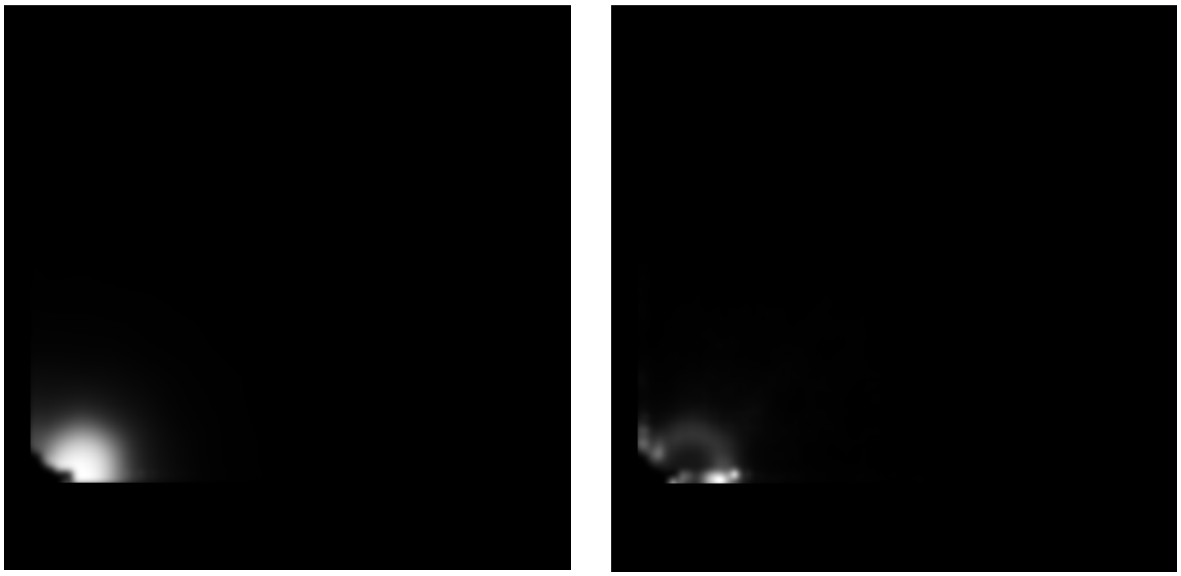
A későbbi mérések kiértékeléséhez mindenképpen normált LRF és VAR adatokra van szükségünk, hiszen a Bhattacharyya-metrika normált vektorok összehasonlítására alkalmas. Így a készítés fázisában az aktuális jelösszegekkel még nem normált függvényeket utólag mindenképpen normálnunk kell, immár a jelösszeg átlagával. Ez után a két módon létrehozott LRF és VAR adatok összehasonlíthatóak.

A sarokban található 1. PMT függvényei:



LRF (PMT1) *VAR (PMT1)*

28. ábra: Az 1. PMT LRF és VAR függvénye utólagos normálással

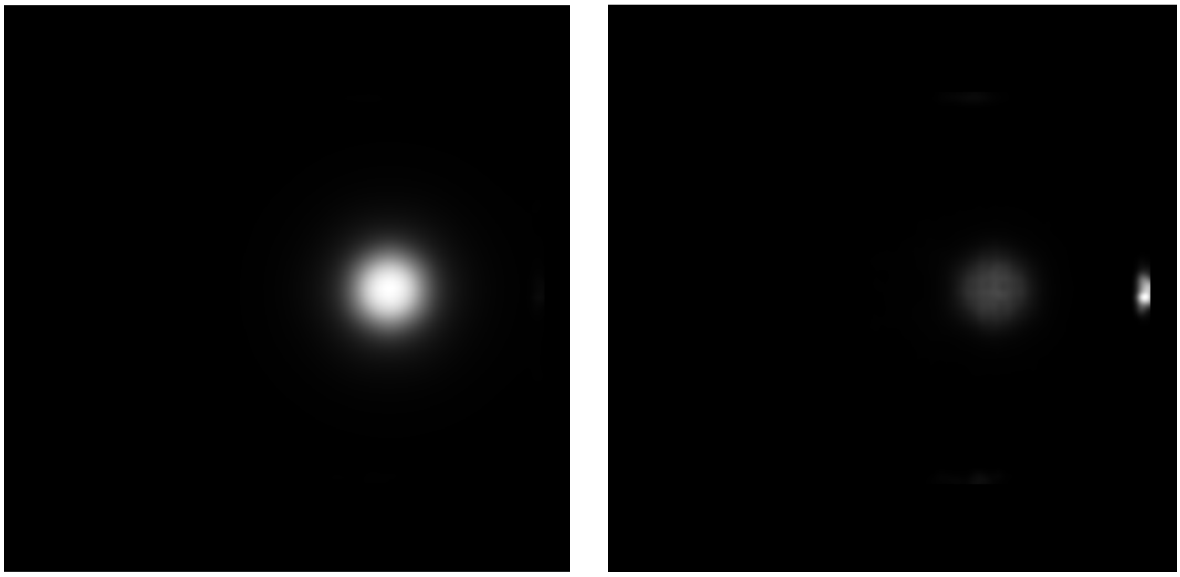


LRF (PMT1) *VAR (PMT1)*

29. ábra: Az 1. PMT LRF és VAR függvénye kiértékelés közben normálva

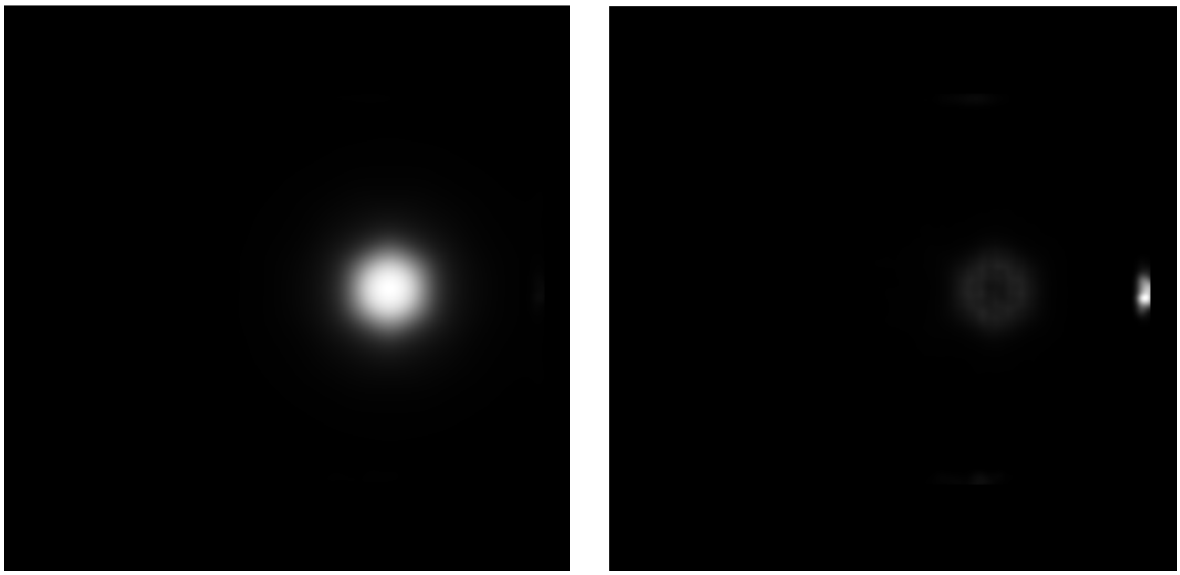
Egymás mellett ábrázolva a két módszerrel készített függvényeket, látható, hogy bár az LRF-ek alakjában nincsen nagy különbség, de a VAR függvények alakja jelentősen függ a

normálás módjától. Ha a normálást már a kiértékelés közben elvégezzük, akkor a PMT középpontja felett a VAR függvény behorpad. Ezt a jelenséget sokkal jobban megfigyelhetjük egy közép tájt elhelyezkedő PMT függvényeit ábrázolva. A 30. PMT LRF és VAR függvényei (30. ábra és 31. ábra):



LRF (PMT30) *VAR (PMT30)*

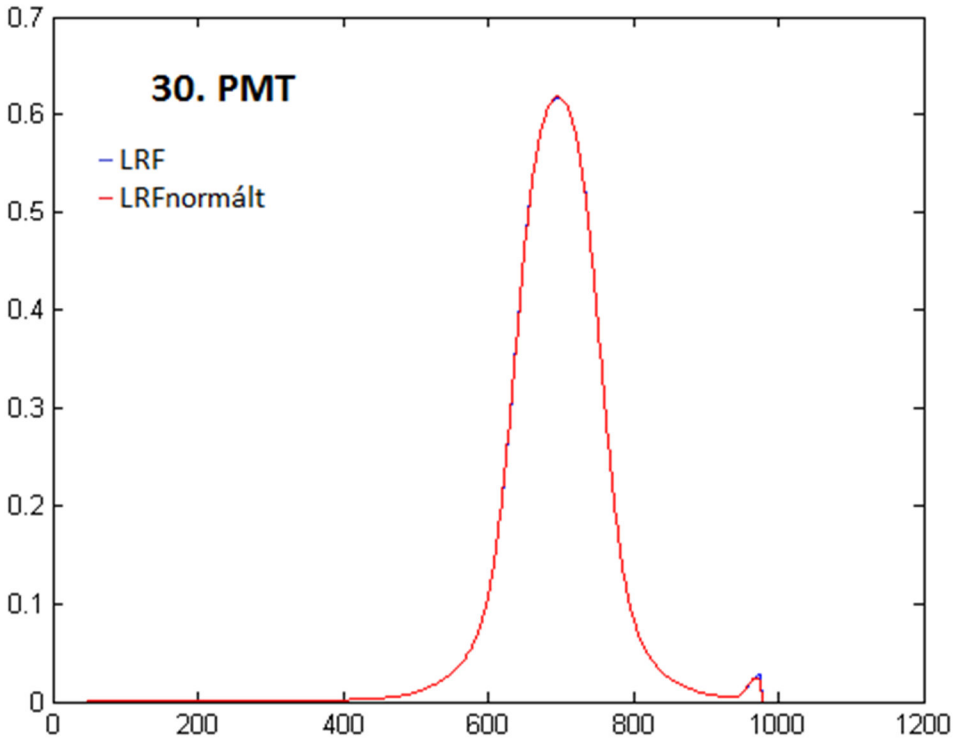
30. ábra: A 30. PMT LRF és VAR függvénye utólagos normálással



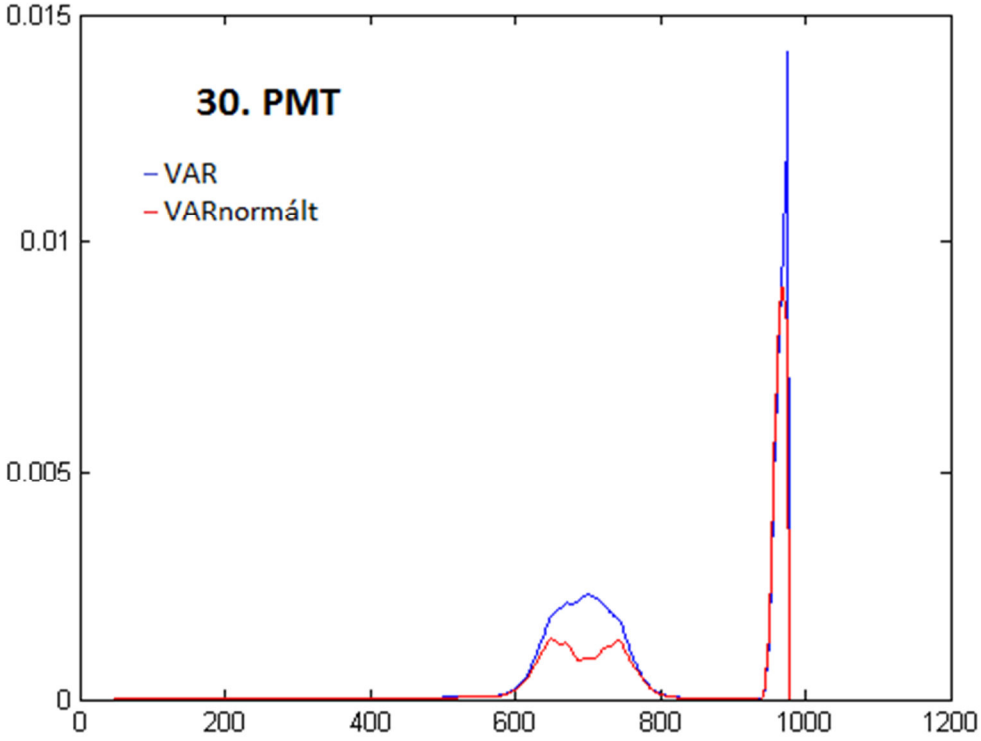
LRF (PMT30) *VAR (PMT30)*

31. ábra: A 30. PMT LRF és VAR függvénye kiértékelés közben normálva

Azt, hogy a két módszerrel készült LRF függvények mennyire hasonlítanak (32. ábra), illetve a VAR függvények miben különböznek egymástól (33. ábra), igazán jól egy középső metszetet ábrázolva figyelhetjük meg:



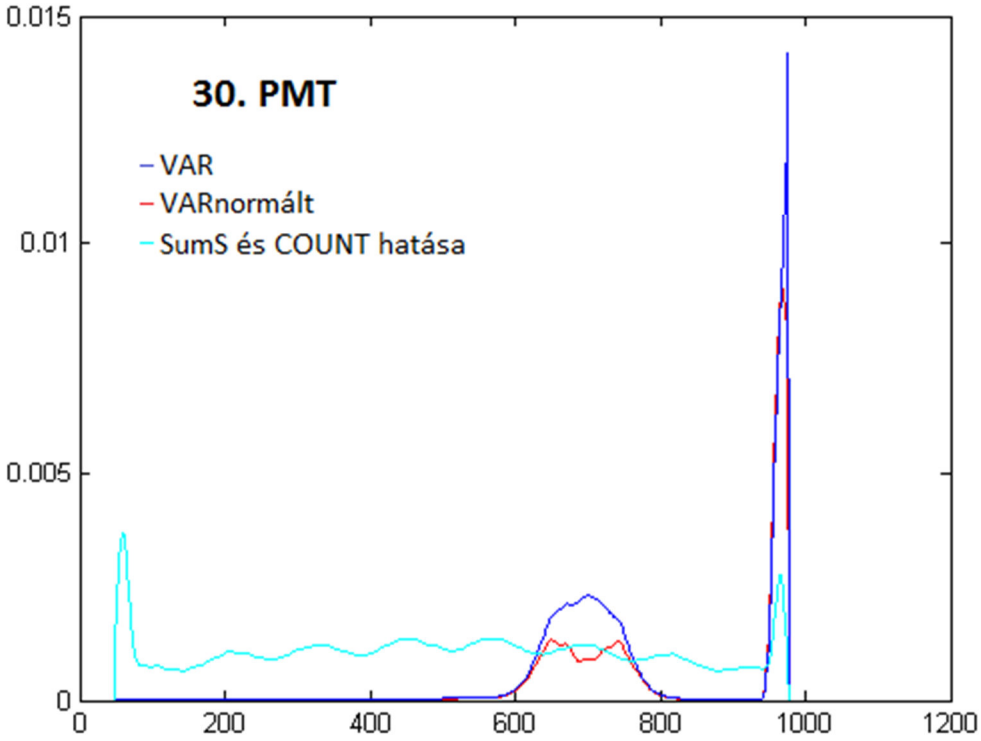
32. ábra: LRF függvények középső metszete



33. ábra: VAR függvények középső metszete

A 33. ábrán jól látható, hogy a VAR függvények között valódi jellegbeli különbség van. Ezt a jelenséget úgy okozza a kiértékelés közbeni normálás, hogy a normálás során figyelmen kívül hagyjuk mind a beütésszámok, mind a jelösszeg kis fluktuációit, amik ugyanakkor

egymás hatását erősítik, ráadásul a VAR függvények számításába ezek négyzetesen számítanak bele. A 34. ábra a két VAR metszet mellett ábrázoltam a COUNT és SumS képek megfelelő metszetének négyzetes szorzatát is.



34. ábra: VAR függvények, és a beütésszám, valamint a jelösszeg hatását érzékeltető függvény

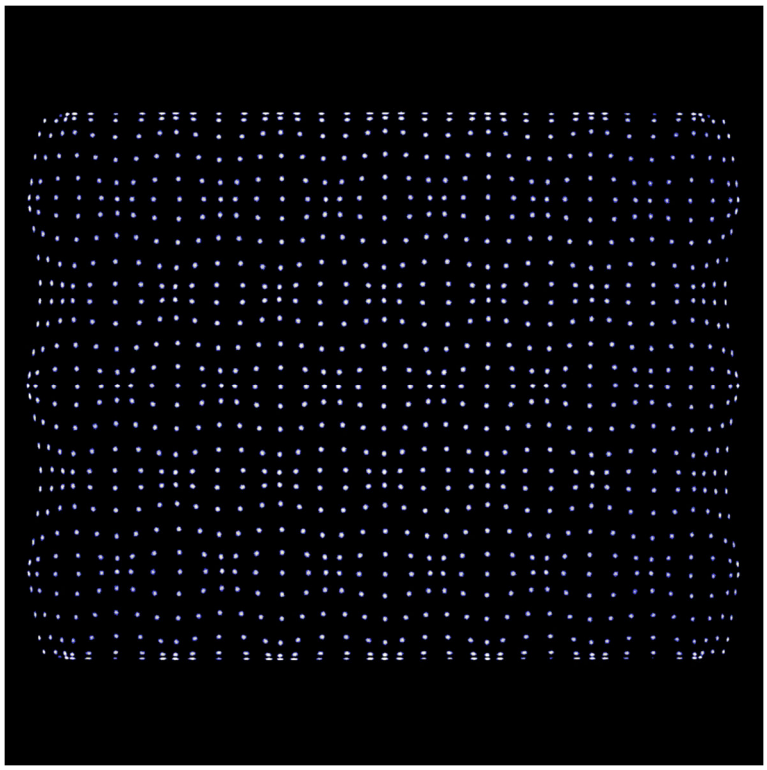
Látható, hogy a beütésszám és a jelösszeg függvény épp ott nő meg, ahol az előre normált VAR függvény behorpad, vagyis ezeknek a fluktuációknak a figyelmen kívül hagyása valóban okozhatja az előre illetve utólag normált VAR függvények közötti látványos különbséget.

V. Mérési eredmények kiértékelése a mért LRF és VAR értékek felhasználásával

Az elkészített LRF és VAR táblázatok segítségével már lehetőség volt arra, hogy a mérési adatokat a Mediso Kft. Nucline statisztikus módszerek felhasználására alkalmas szoftverével feldolgozzam, és azokból DICOM képeket készítsek. A különböző képszámítási algoritmusok segítségével készített képek összehasonlításával megvizsgálhatjuk, hogy melyik módszer a legalkalmasabb a gamma kamerák képalkotó eljárásának.

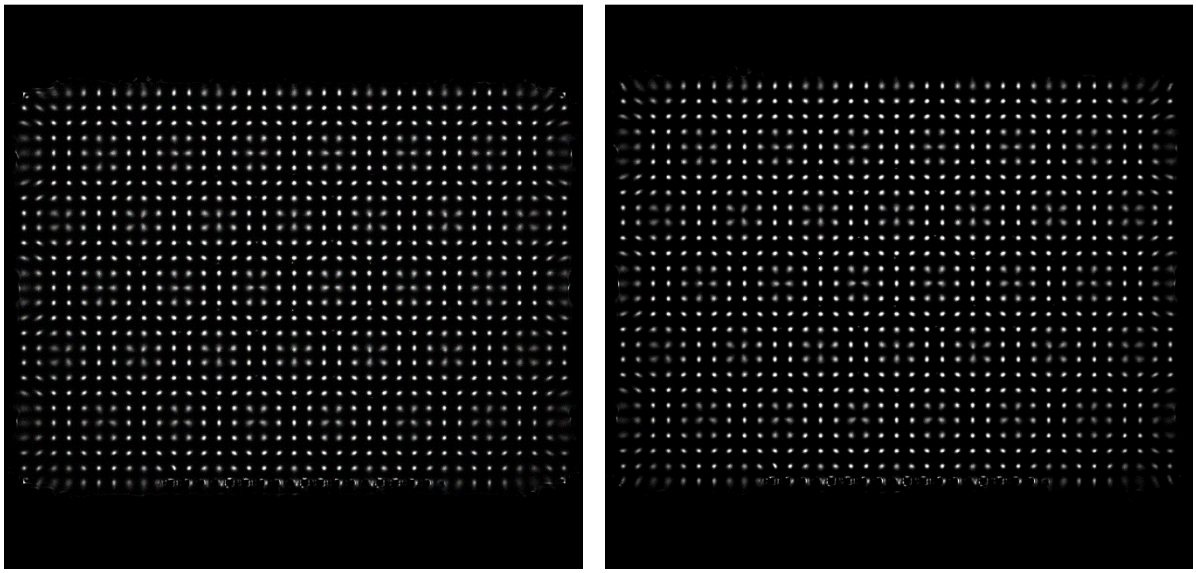
Ahhoz, hogy az előző fejezetben leírt kétféleképpen elkészített LRF és VAR adatokat felhasználhassam a Nucline szoftverben, szükséges volt megfelelően átírni a statisztikus képszámítást végző függvény beolvasó modulját, valamint, a képszámítást végző kódrészletbe beleírtam egy-egy metódust a CSE, az LS és a Bhattacharyya-metrika használatához is. Így ugyanazokon a mérési adatokon kipróbálhattam mindhárom statisztikus képszámítási módszert, és a kétféleképp kiszámított LRF és VAR adatokat is. Az elkészített képszámítást végző függvény mindhárom esetben egyszerű gradiens módszerrel keresi meg az adott metrika szerint minimális távolságú pontot.

A 35. ábrán összehasonlítási alapként látható az eredeti NLCC kép a középső helyzetben felvett mérési adatokból:



35. ábra: Eredeti NLCC kép

Ehhez képest két statisztikus képszámítási módszerrel, az LS és a Bhattacharyya-metrikával nyert képek jelentős javulást mutatnak linearitás és homogenitás tekintetében is. Az L2 normán alapuló LS módszer eredményei (36. ábra):

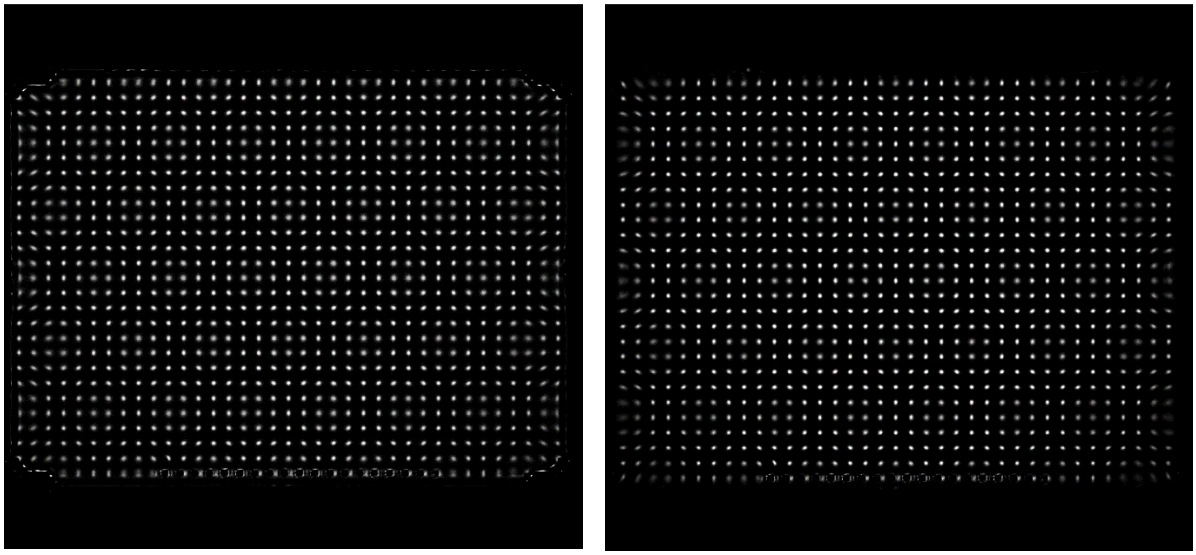


Kiértékelés közben normált LRF *Utólag normált LRF*

36. ábra: LS metrika hatása a képre

Jól látható, hogy az LS metrika is jelentősen javít a képalkotás lineritásán, különösen az utólag normált LRF adatok használatával, itt a leképezés már csaknem tökéletes, kivéve a legszélső oszlopok furatainak képét, illetve jól látható az is, hogy az egyes furatok intenzitása, és alakja sem teljesen homogén, a háttérben húzódó PMT rács alakját magán hordozza.

Ezzel szemben a Bhattacharyya-metrika alkalmazásával kapható képek (37. ábra):



Kiértékelés közben normált LRF *Utólag normált LRF*

37. ábra: Bhattacharyya-metrika hatása a képre

A két ábrán látszik, hogy ezzel a metrikával és az utólagos normálással már a legszélső két oszlop leképezése is tökéletesen működött, és bár a furatok intenzitása és mérete itt sem teljesen egyenletes, de a szórás már jelentősen kisebb.

Hogy a kiértékelést számszerűsíthessem, a MATLAB regionprops függvényét használtam. Ez a függvény bináris képek összefüggő tartományait keresi meg, és azok különböző tulajdonságait méri meg, mint a terület, illetve a rá illeszkedő ellipszis nagy és kistengelye. Ezekből statisztikát készítettem, ezek eredményeit foglalja össze a 2. táblázat:

	LS		LSnorm			Bhattacharyya		Bh.norm	
	Átlag	Szórás	Átlag	Szórás		Átlag	Szórás	Átlag	Szórás
Terület	97,26	23,25	96,52	25,50	Terület	72,55	13,85	83,55	19,86
Nagytengely	12,86	2,66	12,92	2,14	Nagytengely	10,98	1,27	11,99	1,78
Kistengely	10,01	1,72	9,81	1,63	Kistengely	8,62	1,18	9,12	1,38
Arány	1,43	2,07	1,36	0,39	Arány	1,30	0,27	1,34	0,32

2. táblázat: Furatok képeiből készült statisztika, az adatokat pixelben mérjük

Az átlag és szórás adatokból világosan látszik, hogy a négy kép közül a két Bhattacharyya-metrikával készült felvétel minden alaki tulajdonságban jobban teljesít az LS módszernél. A furatok képei kisebbek, vagyis pontszerűbbek, így jobb felbontást eredményeznek, és a tengelyek aránya is valamivel közelebb van az ideális kört jellemző 1-es értékhez. Emellett az adatok szórása is kisebb, vagyis az új metrikával készített felvételek valóban homogénebbek az LS módszerrel készült képeknél.

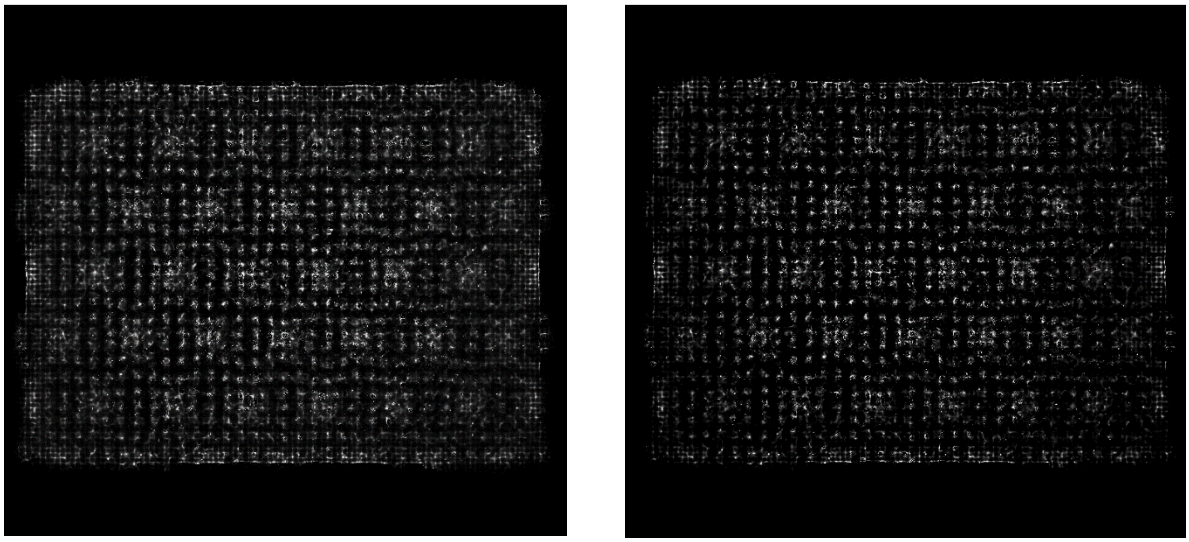
A regionprops függvény segítségével az illeszthető ellipszisek középpontjait is megkaphatjuk, így megvizsgálható az is, hogy a furatok képei milyen messze kerülnek a valós helyüktől. Ezt a tulajdonságot már csak az utólag normált LRF-ekkel készített képek esetében vizsgáltam meg, mivel ezek helyleképezése szemmel láthatóan pontosabb volt, különösen a látómező szélén.

	LS		Bhattacharyya	
	Átlag	Szórás	Átlag	Szórás
Eltérés	1,66	0,96	1,61	0,78

3. táblázat: Furatok képének és tényleges helyének távolsága pixelben

A 3. táblázat adataiból látszik, hogy a Bhattacharyya-metrikával készült kép nem csupán ránézésre tűnik pontosabbnak, hiszen átlagra pontosabban helyezi el a furatok képét, mint az LS módszer, és a szórása is kisebb.

A fenti két eljárással szemben a CSE módszer váratlanul rosszul teljesített. A CSE az LS módszerhez hasonlóan L2-es normán alapul, azonban figyelembe veszi az adatok szórását is, ezzel elméletben pontosabb és gyorsabban konvergáló megoldást eredményezve. A CSE módszerrel készített képeket mutatja a 38. ábra:



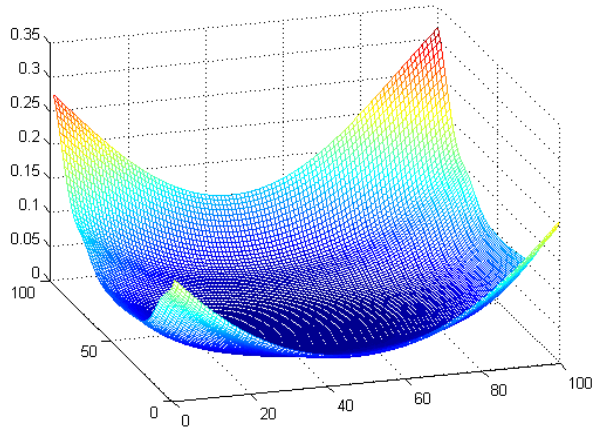
Kiértékelés közben normált LRF *Utólag normált LRF*

38. ábra: CSE metrika hatása a képekre

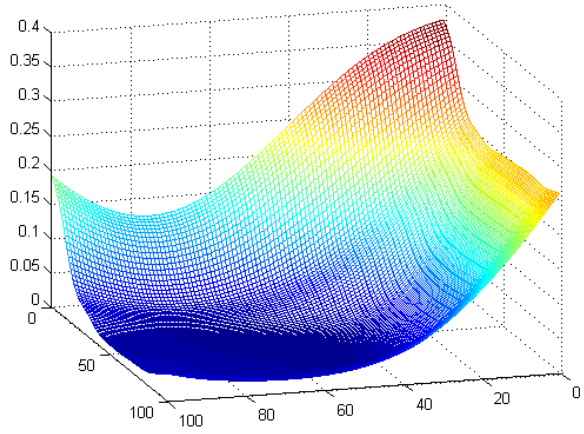
A 38. ábra képein ugyan látható a pontrács, de a furatok hely-leképezése nem javult jelentősen az NLCC módszerhez képest, viszont az egyes furatok képeit ez az eljárás nagyon zajossá, elkentté teszi, gyakorlatilag a teljes látómezőben láthatunk beütéseket.

Ennek a váratlan alulteljesítésnek meglehetősen egyszerű oka van. A CSE módszerben a másik két statisztikus eljárással szemben felhasználtuk a lemért VAR táblázatok adatait is. A szórás ismerete elméletben még javítana, gyorsítana is az implementált gradiens-módszeren, ugyanakkor kis vizsgálódás után jól látható, hogy a minimalizálandó ε függvényünk ugyan általánosságban meredekebben lejt az abszolút minimum-hely felé, viszont sokkal több lokális minimummal rendelkezik annak közelében, mint az LS illetve a gyökvonás simító hatásának köszönhetően még robosztusabb Bhattacharrya-metrikával készített függvények. Ily módon a közönséges gradiens módszer CSE metrika használata esetén jó eséllyel egy lokális minimumban fog megrekedni, az abszolút minimumhelyet pedig leghatékonyabban Bhattacharyya-módszerrel kaphatjuk meg.

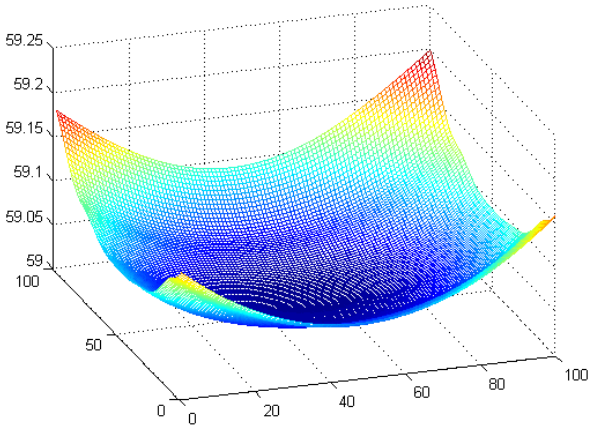
Az állítás alátámasztására a 39. ábrán bemutatom, hogy hogyan néz ki az ε felület a három statisztikus módszer esetében, ha a mért értékek egy, közvetlenül valamelyik PMT feletti szcintillációból származnak, vagy ha attól kissé távolabbról érkező fotonok detektálásával nyerjük őket.



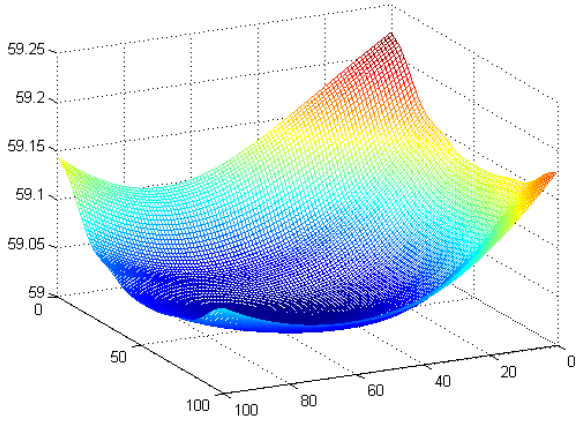
ε_{LS} *a PMT felett*



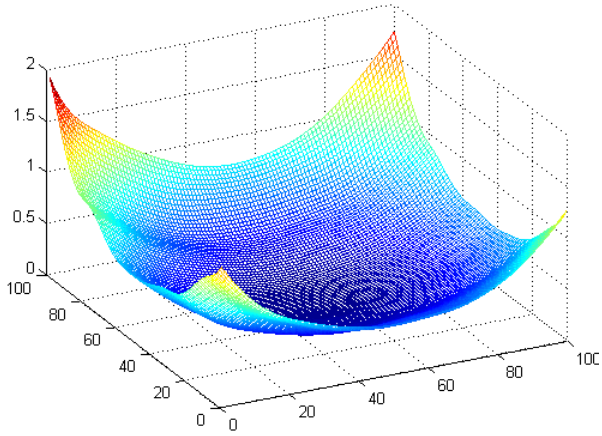
ε_{LS} *a PMT-től távolabb*



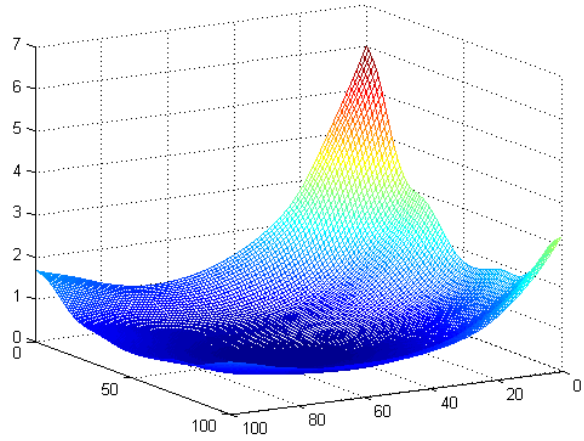
$\varepsilon_{Bhattacharyya}$ *a PMT felett*



$\varepsilon_{Bhattacharyya}$ *a PMT-től távolabb*



ε_{CSE} *a PMT felett*



ε_{CSE} *a PMT-től távolabb*

39. ábra: ε felületek tipikus alakja a három statisztikus módszer esetében

Összességében tehát a jelenleg alkalmazott minimumkeresési eljárás mellett a három statisztikus módszer közül pontos képalkotásra messzemenőkig a legalkalmasabb a Bhattacharyya-távolságon alapuló módszer. Bár a fentiekből nyilvánvaló, hogy a CSE módszer hatékonyságán rengeteget javítana, hogyha valamilyen a lokális minimumokra kevésbé

érzékeny minimalizálási eljárást, például valamilyen szimulált hűtési algoritmust használnánk, egy ilyen módszer feltehetően jelentősen nagyobb futásidővel járna, így mindenképp praktikusabb lehet a Bhattacharrya-módszer mellett dönteni, ami azt is lehetővé teszi, hogy az egyes kameráknak csupán az LRF adatait kelljen kimérni, hiszen a szórástáblázatokat ez a módszer fel sem használja.

VI. Konklúzió

Diplomamunkámban tehát két független módszerrel, szimulációkkal és mérésekkel is igazoltam, hogy az általam a statisztikus képszámítás magjának javasolt távolságmérték, a Bhattacharyya-metrika valóban jobb intrinsic felbontást eredményez a szakirodalomban széles körben kutatott L2 metrikán alapuló eljárásoknál, valamint beláttam, hogy a statisztikus módszerekkel a hasznos látómező mérete valóban növelhető a súlypontszámítási módszerekéhez képest.

Vizsgálataim alapján a Bhattacharyya-módszer alkalmas lehet tényleges ipari felhasználásra is, mivel amellett, hogy igen jó felbontású és linearitású képeket eredményez, ez az eljárás jól alkalmazható gyors minimumkeresési algoritmusokkal együtt, így képes a mérések során rendelkezésre álló hatalmas mennyiségű adatot kellően rövid időn belül feldolgozni.

Mindezek alapján tehát úgy gondolom, hogy a későbbiekben érdemes lehet a gamma kamerák LRF adatainak kimérésére egy jól automatizálható standard eljárást kifejleszteni, valamint a Bhattacharyya-módszert tovább vizsgálni, illetve optimalizálni ipari felhasználásra.

VII. Függelék

VII.1. Forráskódok a szimmulációkhoz

VII.1.1. lrferedeti.m

```
%A file beolvassa a PetDetSim által kiadott adatokat, és összefésüli őket.
%Az elmentett adatokat feldolgozza, az átlagból LRF a szórásból VAR
%adatokat álltít elő. Ezután létrehozza az adatokból a teljes LRF és VAR
%táblázatokat (1. 2. 5.). Ezekből tükrözésekkel előállíthatók a többi PMT adatai is.

%Adatbeolvasás és összefésülés:
LRF=zeros(2760,9,100);

for i=1:100
    LRF(:,:,i)=load(['NaI9.5_glass8_LRF_',num2str(i),'.dat']);
end
save ('NaI9.5_glass8_LRF_1_100.mat','LRF')
%load('NaI9.5_glass8_LRF_1_100.mat');
LRFeredeti=LRF;

LRFdata=zeros(276,9,1000);
for k=1:276
    for i=1:10
        LRFdata(k,:,(i-1)*100+1:i*100)=LRFeredeti((k-1)*10+i, :,:);
    end
end

save('LRFdata.mat','LRFdata')
%Normálás:
%LRFdata=LRFdata./repmat(sum(LRFdata,2),1,9);
%LRFdata=LRFdata./repmat(LRFdata(:,5,:),1,9);


%LRF adatok az átlagból, VAR adatok a szórásból
Lrf=sum(LRFdata*0.001, 3);
Negy=sum((LRFdata.^2)*0.001,3);
Var=(Negy-(Lrf.^2)).^(0.5);

%Adatok elrendezése a látómezőnek megfelelő alakban
LRF=zeros(45,45,9);
VAR=zeros(45,45,9);
k=0;
for i=1:23
    for j=i:23
        k=k+1;
        LRF(i,j,:)=Lrf(k,:);
        VAR(i,j,:)=Var(k,:);
    end
end

figure(1)

for i=1:9

    subplot(3,3,i); image(LRF(:,:,i)*0.1);
    axis square
end

figure(2)

for i=1:9

    subplot(3,3,i); image(VAR(:,:,i));
    axis square
end

%A középső PMT LRF és VARadatai:
LRF5=[LRF(:,1:23,5),fliplr(LRF(:,1:22,5))];
LRF5=[LRF5(1:23,:);flipud(LRF5(1:22,:))];
for i=1:45
LRF5(i,i)=0;
```

```
end
LRF5=LRF5+rot90(LRF5,3);
LRF5(23,23)=LRF(23,23,5);
figure(3)
image(LRF5*0.10)
axis square

%save('LRF5.mat','LRF5')

VAR5=[VAR(:,1:23,5),fliplr(VAR(:,1:22,5))];
VAR5=[VAR5(1:23,:);flipud(VAR5(1:22,:))];
for i=1:45
VAR5(i,i)=0;
end
VAR5=VAR5+rot90(VAR5,3);
VAR5(23,23)=VAR(23,23,5);
figure(4)
image(VAR5)
axis square

%save('VAR5.mat','VAR5')


%A 2. PMT LRF és VARadatai:
LRF2=[LRF(:,1:23,2),fliplr(LRF(:,1:22,2))];
LRF8=[LRF(:,1:23,8),fliplr(LRF(:,1:22,8))];
LRF2=[LRF2(1:23,:);flipud(LRF8(1:22,:))];
LRF4=LRF(:,:,4)';
LRF4=[LRF4(:,1:23),fliplr(LRF4(:,1:22))];
for i=1:23
    LRF4(i,i)=0;
    LRF4(i,46-i)=0;
end

LRF6=LRF(:,:,6)';
LRF6=[LRF6(:,1:23),fliplr(LRF6(:,1:22))];
for i=1:23
    LRF6(i,i)=0;
    LRF6(i,46-i)=0;
end
LRF6(23,:)=zeros(1,45);
LRF6=flipud(LRF6);
LRF2=LRF2+LRF4+LRF6;
figure (5)
image(LRF2*0.100)

axis square

%save('LRF2.mat','LRF2')

VAR2=[VAR(:,1:23,2),fliplr(VAR(:,1:22,2))];
VAR8=[VAR(:,1:23,8),fliplr(VAR(:,1:22,8))];
VAR2=[VAR2(1:23,:);flipud(VAR8(1:22,:))];
VAR4=VAR(:,:,4)';
VAR4=[VAR4(:,1:23),fliplr(VAR4(:,1:22))];
for i=1:23
    VAR4(i,i)=0;
    VAR4(i,46-i)=0;
end

VAR6=VAR(:,:,6)';
VAR6=[VAR6(:,1:23),fliplr(VAR6(:,1:22))];
for i=1:23
    VAR6(i,i)=0;
    VAR6(i,46-i)=0;
end
VAR6(23,:)=zeros(1,45);
VAR6=flipud(VAR6);
VAR2=VAR2+VAR4+VAR6;
figure (6)
image(VAR2)

axis square

%save('VAR2.mat','VAR2')
```

```
%Az 1. PMT LRF és VAR adatai:
LRF1_1=[LRF(:,1:23,1),fliplr(LRF(:,1:22,3))];
for i=1:23
    LRF1_1(i,46-i)=0;
end
LRF1_2=[LRF(:,1:23,7),fliplr(LRF(:,1:22,9))];
LRF1=LRF1_1+rot90(LRF1_2,3);
for i=1:45
    LRF1(i,i)=0;
end
LRF1=LRF1'+LRF1_1+rot90(LRF1_2,3);

figure (7)
image(LRF1*0.100)

axis square

%save('LRF1.mat','LRF1')

VAR1_1=[VAR(:,1:23,1),fliplr(VAR(:,1:22,3))];
for i=1:23
    VAR1_1(i,46-i)=0;
end
VAR1_2=[VAR(:,1:23,7),fliplr(VAR(:,1:22,9))];
VAR1=VAR1_1+rot90(VAR1_2,3);
for i=1:45
    VAR1(i,i)=0;
end
VAR1=VAR1'+VAR1_1+rot90(VAR1_2,3);

figure (8)
image(VAR1)

axis square

%save('VAR1.mat','VAR1')

%Egyéb metszetek ábrázolásai:
figure(9)
plot(1:45,LRF5(:,23),'b',1:45,VAR5(:,23),'r')

figure(10)
plot(VAR5(:,23),'r*')

figure(11)
plot(1:45,LRF2(:,23),'b',1:45,VAR2(:,23),'r')

figure(12)
plot(VAR2(:,23),'r*')

for i=1:45
    lrf5(i)=LRF5(i,i);
    var5(i)=VAR5(i,i);
end
figure(13)
plot(1:45,LRF5(:,23),'c',1:45,VAR5(:,23),'g',1:45,lrf5,'b',1:45,var5,'r')
figure(14)
plot(var5,'r*')
figure(15)
plot(1:45,LRF5(:,23),'b',1:45,LRF5(:,23).^0.5,'r',1:45,VAR5(:,23),'c')
figure(16)
plot(1:45,lrf5,'b',1:45,lrf5.^0.5,'r',1:45,var5,'c')

%Az 1. 2. 5. PMT jeleiből a többi LRF és VAR táblázat is megkapható

LRF=zeros(45,45,9);
VAR=zeros(45,45,9);

LRF(:,:,1)=LRF1;
VAR(:,:,1)=VAR1;
LRF(:,:,2)=LRF2;
VAR(:,:,2)=VAR2;
LRF(:,:,3)=fliplr(LRF1);
VAR(:,:,3)=fliplr(VAR1);
LRF(:,:,4)=LRF2';
VAR(:,:,4)=VAR2';
LRF(:,:,5)=LRF5;
```

```
VAR(:,:,5)=VAR5;
LRF(:,:,6)=fliplr(LRF(:,:,4));
VAR(:,:,6)=fliplr(VAR(:,:,4));
LRF(:,:,7)=LRF(:,:,3)';
VAR(:,:,7)=VAR(:,:,3)';
LRF(:,:,8)=flipud(LRF2);
VAR(:,:,8)=flipud(VAR2);
LRF(:,:,9)=flipud(LRF(:,:,3));
VAR(:,:,9)=flipud(VAR(:,:,3));

save('LRFteljestomb.mat','LRF')
save('VARteljestomb.mat','VAR')
```

VII.1.2. szimulációk.m

```
%A file beolvassa a teljes LRF és VAR tömböke, majd ezekből 46
%pontforráshoz tartozó várhatóérték-vektort állítunk elő. a pontforrások
%közül 23 a látómező átlóján, 23 a látómező oldalfelező-merőlegese mentén.
%a várhatóértékek körül Poisson-eloszlással 10000 eseményt szimulálunk, a
%"mérési" adatokból pedig 4 képet készítünk, Anger-elvvel, LS, CSE és
%Bhattacharyya-metrikával.

load('LRFteljestomb.mat')
load('VARteljestomb.mat')

LRF=LRF./repmat(sum(LRF,3),[1,1,9]);
VAR=VAR./repmat(sum(LRF,3).^2,[1,1,9]);
IVAR=1/VAR;
IVAR(isnan(IVAR))=max(max(max(IVAR)));

%Várhatóérték-vektor létrehozása
v=1:23;
LAMBDA=zeros(46,9);
for i=1:23
    LAMBDA(2*i-1,:)=squeeze(LRF(v(i),v(i),:));
    LAMBDA(2*i,:)=squeeze(LRF(v(i),23,:));
end

LAMBDA=1064*LAMBDA;

Lambda=repmat(LAMBDA,[10000,1]);

%Mérési adatok tömbje
S=poissrnd(Lambda);
S=S./repmat(sum(S,2),[1,9]);

%A négy módszerrel készült 46 kép
KEPanger=zeros(45,45,46);
KEPls=zeros(45,45,46);
KEPcse=zeros(45,45,46);
KEPbh=zeros(45,45,46);

PMTX=[6,23,40,6,23,40,6,23,40];
PMTY=[6,6,6,23,23,23,40,40,40];

eps=zeros(45);

for i=1:46:46000
    for j=1:46
        %Anger
        Xa=(sum(S(i+j-1,:),2).^-1).*(S(i+j-1,:)*PMTX');
        Ya=(sum(S(i+j-1,:),2).^-1).*(S(i+j-1,:)*PMTY');
        if (round(Xa)>0 && round(Xa)<46)
            if (round(Ya)>0 && round(Ya)<46)
                KEPanger(round(Xa),round(Ya),j)=KEPanger(round(Xa),round(Ya),j)+1;
            end
        end

        Stomb=permute((repmat(S(i+j-1,:),[45,1,45])),[1,3,2]);
        %Bhattacharyya
        eps=1-sum((Stomb.*LRF).^0.5,3);
        [Xb,Yb]=find(eps==min(min(eps)));
        if length(Xb)==1
```

```
            if (round(Xb)>0 && round(Xb)<46)
                if (round(Yb)>0 && round(Yb)<46)
                    KEPbh(round(Xb),round(Yb),j)=KEPbh(round(Xb),round(Yb),j)+1;
                end
            end
        end

        %LS
        eps=sum((Stomb-LRF).^2,3);
        [Xls,Yls]=find(eps==min(min(eps)));
        if length(Xls)==1
            if (round(Xls)>0 && round(Xls)<46)
                if (round(Yls)>0 && round(Yls)<46)
                    KEPls(round(Xls),round(Yls),j)=KEPls(round(Xls),round(Yls),j)+1;
                end
            end
        end

        %CSE
        eps=sum(IVAR.*(Stomb-LRF).^2,3);
        [Xcse,Ycse]=find(eps==min(min(eps)));
        if length(Xcse)==1
            if (round(Xcse)>0 && round(Xcse)<46)
                if (round(Ycse)>0 && round(Ycse)<46)
                    KEPcse(round(Xcse),round(Ycse),j)=KEPcse(round(Xcse),round(Ycse),j)+1;
                end
            end
        end

    end
end

%Képek elmentése
save('Kepa.mat','KEPanger')
save('Kepb.mat','KEPbh')
save('Kepls.mat','KEPls')
save('Kepcse.mat','KEPcse')
```

VII.1.3. kiertekelo.m

```
%A fájl beolvassa a szimulációk végeredményét, és a képekből kiszámítja a
%pontforrás becsült pozícióját(súlypont), és félértékszélességét (NEMA
%alapján parabolaillesztéssel)

load('Kepa.mat')
load('Kepb.mat')
load('Kepls.mat')
load('Kepcse.mat')

v=1:23;

%anger
ERRanger=zeros(46,1);
Positionanger=zeros(46,2);
FWHManger=zeros(46,2);

for i=1:23
    %Anger pozíció-becslés
    Sumkep1=sum(sum(KEPanger(:,:,2*i-1)));
    Sumkep2=sum(sum(KEPanger(:,:,2*i)));
    for j=1:45
        for k=1:45
            ERRanger(2*i-1)=ERRanger(2*i-1)+KEPanger(k,j,2*i-1)*((j-v(i))^2+(k-v(i))^2)^0.5;
            ERRanger(2*i)=ERRanger(2*i)+KEPanger(k,j,2*i-1)*((j-v(i))^2+(k-23)^2)^0.5;

            Positionanger(2*i-1,1)=Positionanger(2*i-1,1)+KEPanger(k,j,2*i-1)*j/Sumkep1;
            Positionanger(2*i,1)=Positionanger(2*i,1)+KEPanger(k,j,2*i)*j/Sumkep2;

            Positionanger(2*i-1,2)=Positionanger(2*i-1,2)+KEPanger(k,j,2*i-1)*k/Sumkep1;
            Positionanger(2*i,2)=Positionanger(2*i,2)+KEPanger(k,j,2*i)*k/Sumkep2;
        end
    end

    %Anger FWHM számítás
```

```
    syms('x')

    %fwhm x1
    if round(Positionanger(2*i-1,1))>2
        p=polyfit((round(Positionanger(2*i-1,1))-2:round(Positionanger(2*i-1,1))+2),
KEPanger(round(Positionanger(2*i-1,2)),round(Positionanger(2*i-1,1))-
2:round(Positionanger(2*i-1,1))+2,2*i-1), 2);
    end

    if round(Positionanger(2*i-1,1))==2
        p=polyfit((round(Positionanger(2*i-1,1))-1:round(Positionanger(2*i-1,1))+3),
KEPanger(round(Positionanger(2*i-1,2)),round(Positionanger(2*i-1,1))-
2:round(Positionanger(2*i-1,1))+2,2*i-1), 2);
    end

    if round(Positionanger(2*i-1,1))==1
        p=polyfit((round(Positionanger(2*i-1,1)):round(Positionanger(2*i-1,1))+4),
KEPanger(round(Positionanger(2*i-1,2)),round(Positionanger(2*i-1,1))-
2:round(Positionanger(2*i-1,1))+2,2*i-1), 2);
    end

    y=solve(p(1)*x^2+p(2)*x+p(3)==(4*p(1)*p(3)-p(2)^2)/(8*p(1)),x);
    y=double(y);
    if length(y)==2
    FWHManger(2*i-1,1)=abs(y(1)-y(2));
    end

    %fwhm x2
    if round(Positionanger(2*i,1))>2
        p=polyfit((round(Positionanger(2*i,1))-2:round(Positionanger(2*i,1))+2),
KEPanger(round(Positionanger(2*i,2)),round(Positionanger(2*i,1))-
2:round(Positionanger(2*i,1))+2,2*i), 2);
    end

    if round(Positionanger(2*i,1))==2
        p=polyfit((round(Positionanger(2*i,1))-1:round(Positionanger(2*i,1))+3),
KEPanger(round(Positionanger(2*i,2)),round(Positionanger(2*i,1))-
1:round(Positionanger(2*i,1))+3,2*i), 2);
    end

    if round(Positionanger(2*i,1))==1
        p=polyfit((round(Positionanger(2*i,1)):round(Positionanger(2*i,1))+4),
KEPanger(round(Positionanger(2*i,2)),round(Positionanger(2*i,1)):round(Positionanger(2*i,1))+4
,2*i), 2);
    end

    y=solve(p(1)*x^2+p(2)*x+p(3)==(4*p(1)*p(3)-p(2)^2)/(8*p(1)),x);
    y=double(y);
    if length(y)==2
    FWHManger(2*i,1)=abs(y(1)-y(2));
    end

    %fwhm y1
    if round(Positionanger(2*i-1,2))>2
        p=polyfit((round(Positionanger(2*i-1,2))-2:round(Positionanger(2*i-1,2))+2)',
KEPanger(round(Positionanger(2*i-1,2))-2:round(Positionanger(2*i-
1,2))+2,round(Positionanger(2*i-1,1)),2*i-1), 2);
    end

    if round(Positionanger(2*i-1,2))==2
        p=polyfit((round(Positionanger(2*i-1,2))-1:round(Positionanger(2*i-1,2))+3)',
KEPanger(round(Positionanger(2*i-1,2))-1:round(Positionanger(2*i-
1,2))+3,round(Positionanger(2*i-1,1)),2*i-1), 2);
    end

    if round(Positionanger(2*i-1,2))==1
        p=polyfit((round(Positionanger(2*i-1,2)):round(Positionanger(2*i-1,2))+4)',
KEPanger(round(Positionanger(2*i-1,2))-1:round(Positionanger(2*i-
1,2))+3,round(Positionanger(2*i-1,1)),2*i-1), 2);
    end

    y=solve(p(1)*x^2+p(2)*x+p(3)==(4*p(1)*p(3)-p(2)^2)/(8*p(1)),x);
    y=double(y);
    if length(y)==2
    FWHManger(2*i-1,2)=abs(y(1)-y(2));
    end
```

```
    %fwhm y2
    if round(Positionanger(2*i,2))>2
        p=polyfit((round(Positionanger(2*i,2))-2:round(Positionanger(2*i,2))+2)',
KEPanger(round(Positionanger(2*i,2))-
2:round(Positionanger(2*i,2))+2,round(Positionanger(2*i,1)),2*i), 2);
    end

    if round(Positionanger(2*i,2))==2
        p=polyfit((round(Positionanger(2*i,2))-1:round(Positionanger(2*i,2))+3)',
KEPanger(round(Positionanger(2*i,2))-
1:round(Positionanger(2*i,2))+3,round(Positionanger(2*i,1)),2*i), 2);
    end

    if round(Positionanger(2*i,2))==1
        p=polyfit((round(Positionanger(2*i,2)):round(Positionanger(2*i,2))+4)',
KEPanger(round(Positionanger(2*i,2))-
1:round(Positionanger(2*i,2))+3,round(Positionanger(2*i,1)),2*i), 2);
    end

    y=solve(p(1)*x^2+p(2)*x+p(3)==(4*p(1)*p(3)-p(2)^2)/(8*p(1)),x);
    y=double(y);
    if length(y)==2
    FWHManger(2*i,2)=abs(y(1)-y(2));
    end
end


%Bhattacharyya-mérték
ERRbh=zeros(46,1);
Positionbh=zeros(46,2);
FWHMbh=zeros(46,2);

for i=1:23
    %Bhattacharyya-pozíció számítás
    Sumkep1=sum(sum(KEPbh(:,:,2*i-1)));
    Sumkep2=sum(sum(KEPbh(:,:,2*i)));
    for j=1:45
        for k=1:45
            ERRbh(2*i-1)=ERRbh(2*i-1)+KEPbh(k,j,2*i-1)*((j-v(i))^2+(k-v(i))^2)^0.5;
            ERRbh(2*i)=ERRbh(2*i)+KEPbh(k,j,2*i-1)*((j-v(i))^2+(k-23)^2)^0.5;

            Positionbh(2*i-1,1)=Positionbh(2*i-1,1)+KEPbh(k,j,2*i-1)*j/Sumkep1;
            Positionbh(2*i,1)=Positionbh(2*i,1)+KEPbh(k,j,2*i)*j/Sumkep2;

            Positionbh(2*i-1,2)=Positionbh(2*i-1,2)+KEPbh(k,j,2*i-1)*k/Sumkep1;
            Positionbh(2*i,2)=Positionbh(2*i,2)+KEPbh(k,j,2*i)*k/Sumkep2;
        end
    end

    %Bhattacharyya-félértékszélesség
    syms('x')

    %fwhm x1
    if round(Positionbh(2*i-1,1))>2
        p=polyfit((round(Positionbh(2*i-1,1))-2:round(Positionbh(2*i-1,1))+2),
KEPbh(round(Positionbh(2*i-1,2)),round(Positionbh(2*i-1,1))-2:round(Positionbh(2*i-
1,1))+2,2*i-1), 2);
    end

    if round(Positionbh(2*i-1,1))==2
        p=polyfit((round(Positionbh(2*i-1,1))-1:round(Positionbh(2*i-1,1))+3),
KEPbh(round(Positionbh(2*i-1,2)),round(Positionbh(2*i-1,1))-2:round(Positionbh(2*i-
1,1))+2,2*i-1), 2);
    end

    if round(Positionbh(2*i-1,1))==1
        p=polyfit((round(Positionbh(2*i-1,1)):round(Positionbh(2*i-1,1))+4),
KEPbh(round(Positionbh(2*i-1,2)),round(Positionbh(2*i-1,1))-2:round(Positionbh(2*i-
1,1))+2,2*i-1), 2);
    end

    y=solve(p(1)*x^2+p(2)*x+p(3)==(4*p(1)*p(3)-p(2)^2)/(8*p(1)),x);
    y=double(y);
    if length(y)==2
    FWHMbh(2*i-1,1)=abs(y(1)-y(2));
    end
```

```
    %fwhm x2
    if round(Positionbh(2*i,1))>2
        p=polyfit((round(Positionbh(2*i,1))-2:round(Positionbh(2*i,1))+2),
KEPbh(round(Positionbh(2*i,2)),round(Positionbh(2*i,1))-2:round(Positionbh(2*i,1))+2,2*i), 2);
    end

    if round(Positionbh(2*i,1))==2
        p=polyfit((round(Positionbh(2*i,1))-1:round(Positionbh(2*i,1))+3),
KEPbh(round(Positionbh(2*i,2)),round(Positionbh(2*i,1))-1:round(Positionbh(2*i,1))+3,2*i), 2);
    end

    if round(Positionbh(2*i,1))==1
        p=polyfit((round(Positionbh(2*i,1)):round(Positionbh(2*i,1))+4),
KEPbh(round(Positionbh(2*i,2)),round(Positionbh(2*i,1)):round(Positionbh(2*i,1))+4,2*i), 2);
    end

    y=solve(p(1)*x^2+p(2)*x+p(3)==(4*p(1)*p(3)-p(2)^2)/(8*p(1)),x);
    y=double(y);
    if length(y)==2
    FWHMbh(2*i,1)=abs(y(1)-y(2));
    end

    %fwhm y1
    if round(Positionbh(2*i-1,2))>2
        p=polyfit((round(Positionbh(2*i-1,2))-2:round(Positionbh(2*i-1,2))+2)',
KEPbh(round(Positionbh(2*i-1,2))-2:round(Positionbh(2*i-1,2))+2,round(Positionbh(2*i-
1,1)),2*i-1), 2);
    end

    if round(Positionbh(2*i-1,2))==2
        p=polyfit((round(Positionbh(2*i-1,2))-1:round(Positionbh(2*i-1,2))+3)',
KEPbh(round(Positionbh(2*i-1,2))-1:round(Positionbh(2*i-1,2))+3,round(Positionbh(2*i-
1,1)),2*i-1), 2);
    end

    if round(Positionbh(2*i-1,2))==1
        p=polyfit((round(Positionbh(2*i-1,2)):round(Positionbh(2*i-1,2))+4)',
KEPbh(round(Positionbh(2*i-1,2))-1:round(Positionbh(2*i-1,2))+3,round(Positionbh(2*i-
1,1)),2*i-1), 2);
    end

    y=solve(p(1)*x^2+p(2)*x+p(3)==(4*p(1)*p(3)-p(2)^2)/(8*p(1)),x);
    y=double(y);
    if length(y)==2
    FWHMbh(2*i-1,2)=abs(y(1)-y(2));
    end

    %fwhm y2
    if round(Positionbh(2*i,2))>2
        p=polyfit((round(Positionbh(2*i,2))-2:round(Positionbh(2*i,2))+2)',
KEPbh(round(Positionbh(2*i,2))-2:round(Positionbh(2*i,2))+2,round(Positionbh(2*i,1)),2*i), 2);
    end

    if round(Positionbh(2*i,2))==2
        p=polyfit((round(Positionbh(2*i,2))-1:round(Positionbh(2*i,2))+3)',
KEPbh(round(Positionbh(2*i,2))-1:round(Positionbh(2*i,2))+3,round(Positionbh(2*i,1)),2*i), 2);
    end

    if round(Positionbh(2*i,2))==1
        p=polyfit((round(Positionbh(2*i,2)):round(Positionbh(2*i,2))+4)',
KEPbh(round(Positionbh(2*i,2))-1:round(Positionbh(2*i,2))+3,round(Positionbh(2*i,1)),2*i), 2);
    end

    y=solve(p(1)*x^2+p(2)*x+p(3)==(4*p(1)*p(3)-p(2)^2)/(8*p(1)),x);
    y=double(y);
    if length(y)==2
    FWHMbh(2*i,2)=abs(y(1)-y(2));
    end
end

%LS-módszer
ERRls=zeros(46,1);
Positionls=zeros(46,2);
FWHMls=zeros(46,2);

for i=1:23
    %LS-pozíció számítás
```

```
    Sumkep1=sum(sum(KEPls(:,:,2*i-1)));
    Sumkep2=sum(sum(KEPls(:,:,2*i)));
    for j=1:45
        for k=1:45
            ERRls(2*i-1)=ERRls(2*i-1)+KEPls(k,j,2*i-1)*((j-v(i))^2+(k-v(i))^2)^0.5;
            ERRls(2*i)=ERRls(2*i)+KEPls(k,j,2*i-1)*((j-v(i))^2+(k-23)^2)^0.5;

            Positionls(2*i-1,1)=Positionls(2*i-1,1)+KEPls(k,j,2*i-1)*j/Sumkep1;
            Positionls(2*i,1)=Positionls(2*i,1)+KEPls(k,j,2*i)*j/Sumkep2;

            Positionls(2*i-1,2)=Positionls(2*i-1,2)+KEPls(k,j,2*i-1)*k/Sumkep1;
            Positionls(2*i,2)=Positionls(2*i,2)+KEPls(k,j,2*i)*k/Sumkep2;
        end
    end

    %LS-félértékszélesség
    syms('x')

    %fwhm x1
    if round(Positionls(2*i-1,1))>2
        p=polyfit((round(Positionls(2*i-1,1))-2:round(Positionls(2*i-1,1))+2),
KEPls(round(Positionls(2*i-1,2)),round(Positionls(2*i-1,1))-2:round(Positionls(2*i-
1,1))+2,2*i-1), 2);
    end

    if round(Positionls(2*i-1,1))==2
        p=polyfit((round(Positionls(2*i-1,1))-1:round(Positionls(2*i-1,1))+3),
KEPls(round(Positionls(2*i-1,2)),round(Positionls(2*i-1,1))-2:round(Positionls(2*i-
1,1))+2,2*i-1), 2);
    end

    if round(Positionls(2*i-1,1))==1
        p=polyfit((round(Positionls(2*i-1,1)):round(Positionls(2*i-1,1))+4),
KEPls(round(Positionls(2*i-1,2)),round(Positionls(2*i-1,1))-2:round(Positionls(2*i-
1,1))+2,2*i-1), 2);
    end

    y=solve(p(1)*x^2+p(2)*x+p(3)==(4*p(1)*p(3)-p(2)^2)/(8*p(1)),x);
    y=double(y);
    if length(y)==2
    FWHMls(2*i-1,1)=abs(y(1)-y(2));
    end

    %fwhm x2
    if round(Positionls(2*i,1))>2
        p=polyfit((round(Positionls(2*i,1))-2:round(Positionls(2*i,1))+2),
KEPls(round(Positionls(2*i,2)),round(Positionls(2*i,1))-2:round(Positionls(2*i,1))+2,2*i), 2);
    end

    if round(Positionls(2*i,1))==2
        p=polyfit((round(Positionls(2*i,1))-1:round(Positionls(2*i,1))+3),
KEPls(round(Positionls(2*i,2)),round(Positionls(2*i,1))-1:round(Positionls(2*i,1))+3,2*i), 2);
    end

    if round(Positionls(2*i,1))==1
        p=polyfit((round(Positionls(2*i,1)):round(Positionls(2*i,1))+4),
KEPls(round(Positionls(2*i,2)),round(Positionls(2*i,1)):round(Positionls(2*i,1))+4,2*i), 2);
    end

    y=solve(p(1)*x^2+p(2)*x+p(3)==(4*p(1)*p(3)-p(2)^2)/(8*p(1)),x);
    y=double(y);
    if length(y)==2
    FWHMls(2*i,1)=abs(y(1)-y(2));
    end

    %fwhm y1
    if round(Positionls(2*i-1,2))>2
        p=polyfit((round(Positionls(2*i-1,2))-2:round(Positionls(2*i-1,2))+2)',
KEPls(round(Positionls(2*i-1,2))-2:round(Positionls(2*i-1,2))+2,round(Positionls(2*i-
1,1)),2*i-1), 2);
    end

    if round(Positionls(2*i-1,2))==2
        p=polyfit((round(Positionls(2*i-1,2))-1:round(Positionls(2*i-1,2))+3)',
KEPls(round(Positionls(2*i-1,2))-1:round(Positionls(2*i-1,2))+3,round(Positionls(2*i-
1,1)),2*i-1), 2);
    end
```

```
    if round(Positionls(2*i-1,2))==1
        p=polyfit((round(Positionls(2*i-1,2)):round(Positionls(2*i-1,2))+4)',
KEPls(round(Positionls(2*i-1,2))-1:round(Positionls(2*i-1,2))+3,round(Positionls(2*i-
1,1)),2*i-1), 2);
    end

    y=solve(p(1)*x^2+p(2)*x+p(3)==(4*p(1)*p(3)-p(2)^2)/(8*p(1)),x);
    y=double(y);
    if length(y)==2
    FWHMls(2*i-1,2)=abs(y(1)-y(2));
    end

    %fwhm y2
    if round(Positionls(2*i,2))>2
        p=polyfit((round(Positionls(2*i,2))-2:round(Positionls(2*i,2))+2)',
KEPls(round(Positionls(2*i,2))-2:round(Positionls(2*i,2))+2,round(Positionls(2*i,1)),2*i), 2);
    end

    if round(Positionls(2*i,2))==2
        p=polyfit((round(Positionls(2*i,2))-1:round(Positionls(2*i,2))+3)',
KEPls(round(Positionls(2*i,2))-1:round(Positionls(2*i,2))+3,round(Positionls(2*i,1)),2*i), 2);
    end

    if round(Positionls(2*i,2))==1
        p=polyfit((round(Positionls(2*i,2)):round(Positionls(2*i,2))+4)',
KEPls(round(Positionls(2*i,2))-1:round(Positionls(2*i,2))+3,round(Positionls(2*i,1)),2*i), 2);
    end

    y=solve(p(1)*x^2+p(2)*x+p(3)==(4*p(1)*p(3)-p(2)^2)/(8*p(1)),x);
    y=double(y);
    if length(y)==2
    FWHMls(2*i,2)=abs(y(1)-y(2));
    end
end

%CSE-módszer
ERRcse=zeros(46,1);
Positioncse=zeros(46,2);
FWHMcse=zeros(46,2);

for i=1:23
    %CSE-pozíció
    Sumkep1=sum(sum(KEPcse(:,:,2*i-1)));
    Sumkep2=sum(sum(KEPcse(:,:,2*i)));
    for j=1:45
        for k=1:45
            ERRcse(2*i-1)=ERRcse(2*i-1)+KEPcse(k,j,2*i-1)*((j-v(i))^2+(k-v(i))^2)^0.5;
            ERRcse(2*i)=ERRcse(2*i)+KEPcse(k,j,2*i-1)*((j-v(i))^2+(k-23)^2)^0.5;

            Positioncse(2*i-1,1)=Positioncse(2*i-1,1)+KEPcse(k,j,2*i-1)*j/Sumkep1;
            Positioncse(2*i,1)=Positioncse(2*i,1)+KEPcse(k,j,2*i)*j/Sumkep2;

            Positioncse(2*i-1,2)=Positioncse(2*i-1,2)+KEPcse(k,j,2*i-1)*k/Sumkep1;
            Positioncse(2*i,2)=Positioncse(2*i,2)+KEPcse(k,j,2*i)*k/Sumkep2;
        end
    end

    %CSE-félértékszélesség
    syms('x')

    %fwhm x1
    if round(Positioncse(2*i-1,1))>2
        p=polyfit((round(Positioncse(2*i-1,1))-2:round(Positioncse(2*i-1,1))+2),
KEPcse(round(Positioncse(2*i-1,2)),round(Positioncse(2*i-1,1))-2:round(Positioncse(2*i-
1,1))+2,2*i-1), 2);
    end

    if round(Positioncse(2*i-1,1))==2
        p=polyfit((round(Positioncse(2*i-1,1))-1:round(Positioncse(2*i-1,1))+3),
KEPcse(round(Positioncse(2*i-1,2)),round(Positioncse(2*i-1,1))-2:round(Positioncse(2*i-
1,1))+2,2*i-1), 2);
    end

    if round(Positioncse(2*i-1,1))==1
```

```
        p=polyfit((round(Positioncse(2*i-1,1)):round(Positioncse(2*i-1,1))+4),
KEPcse(round(Positioncse(2*i-1,2)),round(Positioncse(2*i-1,1))-2:round(Positioncse(2*i-
1,1))+2,2*i-1), 2);
    end

    y=solve(p(1)*x^2+p(2)*x+p(3)==(4*p(1)*p(3)-p(2)^2)/(8*p(1)),x);
    y=double(y);
    if length(y)==2
    FWHMcse(2*i-1,1)=abs(y(1)-y(2));
    end

    %fwhm x2
    if round(Positioncse(2*i,1))>2
        p=polyfit((round(Positioncse(2*i,1))-2:round(Positioncse(2*i,1))+2),
KEPcse(round(Positioncse(2*i,2)),round(Positioncse(2*i,1))-2:round(Positioncse(2*i,1))+2,2*i),
2);
    end

    if round(Positioncse(2*i,1))==2
        p=polyfit((round(Positioncse(2*i,1))-1:round(Positioncse(2*i,1))+3),
KEPcse(round(Positioncse(2*i,2)),round(Positioncse(2*i,1))-1:round(Positioncse(2*i,1))+3,2*i),
2);
    end

    if round(Positioncse(2*i,1))==1
        p=polyfit((round(Positioncse(2*i,1)):round(Positioncse(2*i,1))+4),
KEPcse(round(Positioncse(2*i,2)),round(Positioncse(2*i,1)):round(Positioncse(2*i,1))+4,2*i),
2);
    end

    y=solve(p(1)*x^2+p(2)*x+p(3)==(4*p(1)*p(3)-p(2)^2)/(8*p(1)),x);
    y=double(y);
    if length(y)==2
    FWHMcse(2*i,1)=abs(y(1)-y(2));
    end

    %fwhm y1
    if round(Positioncse(2*i-1,2))>2
        p=polyfit((round(Positioncse(2*i-1,2))-2:round(Positioncse(2*i-1,2))+2)',
KEPcse(round(Positioncse(2*i-1,2))-2:round(Positioncse(2*i-1,2))+2,round(Positioncse(2*i-
1,1)),2*i-1), 2);
    end

    if round(Positioncse(2*i-1,2))==2
        p=polyfit((round(Positioncse(2*i-1,2))-1:round(Positioncse(2*i-1,2))+3)',
KEPcse(round(Positioncse(2*i-1,2))-1:round(Positioncse(2*i-1,2))+3,round(Positioncse(2*i-
1,1)),2*i-1), 2);
    end

    if round(Positioncse(2*i-1,2))==1
        p=polyfit((round(Positioncse(2*i-1,2)):round(Positioncse(2*i-1,2))+4)',
KEPcse(round(Positioncse(2*i-1,2))-1:round(Positioncse(2*i-1,2))+3,round(Positioncse(2*i-
1,1)),2*i-1), 2);
    end

    y=solve(p(1)*x^2+p(2)*x+p(3)==(4*p(1)*p(3)-p(2)^2)/(8*p(1)),x);
    y=double(y);
    if length(y)==2
    FWHMcse(2*i-1,2)=abs(y(1)-y(2));
    end

    %fwhm y2
    if round(Positioncse(2*i,2))>2
        p=polyfit((round(Positioncse(2*i,2))-2:round(Positioncse(2*i,2))+2)',
KEPcse(round(Positioncse(2*i,2))-2:round(Positioncse(2*i,2))+2,round(Positioncse(2*i,1)),2*i),
2);
    end

    if round(Positioncse(2*i,2))==2
        p=polyfit((round(Positioncse(2*i,2))-1:round(Positioncse(2*i,2))+3)',
KEPcse(round(Positioncse(2*i,2))-1:round(Positioncse(2*i,2))+3,round(Positioncse(2*i,1)),2*i),
2);
    end

    if round(Positioncse(2*i,2))==1
```

```
        p=polyfit((round(Positioncse(2*i,2)):round(Positioncse(2*i,2))+4)',
KEPcse(round(Positioncse(2*i,2))-1:round(Positioncse(2*i,2))+3,round(Positioncse(2*i,1)),2*i),
2);
    end

    y=solve(p(1)*x^2+p(2)*x+p(3)==(4*p(1)*p(3)-p(2)^2)/(8*p(1)),x);
    y=double(y);
    if length(y)==2
    FWHMcse(2*i,2)=abs(y(1)-y(2));
    end
end
```

VII.2. Forráskódok a mért LRF és VAR adatok kiértékeléséhez

VII.2.1. imagemaker.m

```
%A file beolvassa a mérési adatokat tartalmazó .dat file-okat és létrehozza
%a beolvasott NLCC koordinátákból a képet

close all
clear all
image=zeros(1024);

fid = fopen('adatok/adatok33s.txt');
szamlalo=0;
while ~feof(fid)

   S=fscanf(fid, '%f',[62 100000]);
   Xa=S(1,:);
   Ya=S(2,:);
   Coord=[Xa;Ya]';
   coord = mat2cell(Coord, size(Coord, 1), ones(1, size(Coord, 2)));
   image(sub2ind(size(image), coord{:}))=image(sub2ind(size(image), coord{:}))+1;
   %image(Xa,Ya)=image(Xa,Ya)+1;
   szamlalo=szamlalo+1;
end
save('image33s.mat','image')
fclose(fid)
clear all
```

VII.2.2. racslinearis.m

```
%A file betölti a gamma kamera linearitás-tábláit, és az extrema2.m
%függvényből kapott maximumhely-koordinátákat ezekkel transzformálja. A
%végeredményt a viscircles parancs segítségével meg is jeleníti.


%a linearitás táblák betöltése
Fid=fopen('LIN_X1.TBL.59_101','r');
LinTblX=fread(Fid,[1024,1024],'int16');
fclose(Fid);
Fid=fopen('LIN_Y1.TBL.59_101','r');
LinTblY=fread(Fid,[1024,1024],'int16');
fclose(Fid);

%linearitás tábla alkalmazása (a készülék és a képek x-ytengelye invertált)
for i=1:size(centers1,1)
y(i)=centers1(i,2)+LinTblX(round(centers1(i,2)),round(centers1(i,1)))/64;
x(i)=centers1(i,1)+LinTblY(round(centers1(i,2)),round(centers1(i,1)))/64;
end

%ábrázolás
centers_lin=[x;y]';
radii1=3*ones(length(x),1);
imshow(zeros(1024))
viscircles(centers_lin,radii1)
```

VII.2.3. racsrendezes.m

```
% A file a racslinearis.m végeredményeként kapott transzformált
% maximumhelyeket rendezi rács alakba, majd az így kapott permutációt
% végrehajtja az eredeti, transzformálatlan koordinátákon is, így a
% végeredmény az eredeti maximumhely-koordináták megfelelő mátrix alakú
% elrendezése

%ideális méretek tárolása
DB=1036;
dbx=37;
dby=28;

%rendezés az y koordináta szerint
[v1,v2] = sort(centers_lin(:,1));
racs_=centers_lin(v2,:);

%ideális méretű tömb feltöltése a valós adatokkal: hiányzó rácspontokon a
%koordináták (0,0)
racs=zeros(DB,2);
racs(2:end-2,:)=racs_(68:end,:);
%racs(1:29,:)=racs_(56:84,:);
%racs(30:end,:)=racs_(86:end,:);
racs(1,:)=[0,0];
%racs(3:end,:)=racs_(71:end,:);
racs(end-1:end,:)=zeros(2);

racs=reshape(racs, [dbx,2*dby]);
Racs=zeros(size(racs));
%rendezés x koordináta szerint
for i=1:dby
[w1(:,i),w2(:,i)] = sort(racs(:,dby+i));
r=[racs(:,i),racs(:,dby+i)];
r=r(w2(:,i),:);
Racs(:,(2*i)-1:2*i)=r;
end


% transzformálatlan adatok rendezése azonos permutációkkal:
RACS_=centers1(v2,:);

RACS=zeros(DB,2);
RACS(2:end-2,:)=RACS_(68:end,:);
%RACS(1:29,:)=RACS_(56:84,:);
%RACS(30:end,:)=RACS_(86:end,:);

RACS(1,:)=[0,0];
%RACS(3:end,:)=RACS_(71:end,:);
RACS(end-1:end,:)=zeros(2);

RACS=reshape(RACS, [dbx,2*dby]);
RACS_xy=zeros(size(RACS));

for i=1:dby
    r=[RACS(:,i),RACS(:,dby+i)];
    r=r(w2(:,i),:);
    RACS_xy(:,(2*i)-1:2*i)=r;
end

RACS_xy(1:end-1,1:2)=RACS_xy(2:end,1:2);
%RACS_xy(1,1:2)=[0,0];
RACS_xy(end,1:2)=[0,0];

RACS_xy(2:end-1,end-1:end)=RACS_xy(3:end,end-1:end);
RACS_xy(end,end-1:end)=[0,0];

%adatok mentése
save('RACS33.mat','RACS_xy')
RACS_x=RACS_xy(:,1:2:end-1);
RACS_x=RACS_x';
RACS_y=RACS_xy(:,2:2:end);
RACS_y=RACS_y';

save('RACSx33.mat','RACS_x')
save('RACSy33.mat','RACS_y')
```

VII.2.4. lok_energiamax.m

```
%A file beolvassa a megfelelő méréshez tartozó rácspontok koordinátáit,
%majd soronként beolvassa a mérési adatokat tartalmazó .dat fileokat, és
%rácspontonként előállít egy spektrumot, amiből kiszámítható a lokális
%energiamaximum helye

load('racs11_x.mat');
load('racs11_y.mat');

korcenter(:,:,1)=RACS11_x;
korcenter(:,:,2)=RACS11_y;

Size=size(korcenter);

v=5000:50:25000;   %a spektrum felbontása beütésszámban: 50
l=length(v);

%memóriaallokálás a spektrum számára
spect=zeros(Size(1)*Size(2),l);

fid = fopen('adatok/adatok11s.txt');  %mérési adatok megnyitása

while ~feof(fid)


    S=fscanf(fid, '%f %f %u',[62 10000]);           %mérési adatok beolvasása
    x=S(1,:);                                      %NLCC x koordináta
    y=S(2,:);                                      %NLCC x koordináta
    sumS=sum(S(3:62,:),1);                          %összbeütés


        %vizsgáljuk meg, hogy az x,y pont melyik rácsponthoz esik közel
        M=(repmat(korcenter(:,:,1),[1,1,length(x)])-permute(repmat(x,
[Size(1),1,Size(2)]),[1,3,2])).^2+(repmat(korcenter(:,:,2),[1,1,length(x)])-permute(repmat(y,
[Size(1),1,Size(2)]),[1,3,2])).^2;

            M(2:(end-1),2:(end-1),:)=10000000000*ones(Size(1)-2,Size(2)-2,length(x));   %ha a
közepe nem kell

            %N=ones(Size(1),Size(2),length(x));                                  %ha a széle nem
kell
            %N(2:(end-1),2:(end-1),:)=zeros(Size(1)-2,Size(2)-2,length(x));
            %M=M+10000000000*N;

            mins=squeeze(min(min(M)));

     for i=1:length(x)
        linearInd=find(M(:,:,i)==mins(i)); %keressük meg amelyik középpontjához a legközelebb
van
            if length(linearInd)==1
               for j=1:l-1
                    if v(j)<sumS(i) && sumS(i)<v(j+1)
                    %ha a mért összbeütésszám a megfelelő beütésszámtartományban
                    %van, a spektum adott pontjához adunk egyet
                        spect(linearInd,j)=spect(linearInd,j)+1;
                    end
               end
            end
     end


end
```

VII.2.5. lokmax.m

```
%A file beolvassa a lok_energiamax.m file által előállított spektrumokat,
%és azok alapján megkeresi a maximumhelyeket

load('spectall.mat')

lokmax=zeros(89,117);

v=10000:250:25000;

for i=1:89
```

```
    for k=1:117
    %s=squeeze(spectall(i,k,:));
    sz=squeeze(spectall(i,k,101:end));
    for i=1:60
        s(i)=sz(i)+sz(i+1)+sz(i+2)+sz(i+3)+sz(i+4);
    end
    s(61)=sz(301);

    j=find(s==max(s));
    %ha a maximumhely egyértelmű:
    if length(j)==1
        lokmax(i,k)=(v(j)+v(j+1))/2;
    %ha több maximumhely is van a spektrumban:
    elseif max(s)>0
        for m=1:length(j)
            lokmax(i,k)=lokmax(i,k)+v(j(m))/length(j);
        end
    end
    end
end
```

VII.2.6. lrf_11.m

```
%Az lrf_ii.m beolvassa az adott méréshez tartozó rács x és y koordinátáit, soronként
%beolvassa a mérési adatokat, és azokat a megfelelő furathoz rendeli.
%Szűrésként a lokális energiamaximum körüli 20%-os ablakot használunk. A
%beolvasás után elmentjük az értékek, azokból várhatóértéket (LRF) és
%szórást(VAR) számolunk

clear all
close all
tic

%rácsok betöltése
load('racs11_x.mat');
load('racs11_y.mat');

korcenter(:,:,2)=RACS11_x;
korcenter(:,:,1)=RACS11_y;

Size=size(korcenter);

%lokális energiamaximum adatok betöltése
load ('lokmax11.mat')
lokmax=lokmax11;

fid = fopen('adatok/adatok11.txt');  %mérési adatok megnyitása

%értékes adatok tömbje: PMT-nként és rácshelyenként eltároljaa beütésszámok összegét,
négyzetösszegét és a felvillanások számát
LRFadatok=zeros(Size(1)*Size(2), 60,3);
LRFadatoknorm=zeros(Size(1)*Size(2), 60, 3);

toc
tic
i=0;
while ~feof(fid)


    S=fscanf(fid, '%f %f %u',[62 1]);          %mérési adatok beolvasása
    x=S(1);                                    %NLCC x koordináta
    y=S(2);                                    %NLCC y koordináta
    sumS=sum(S(3:62));                         %összbeütés
    Snorm(3:62)=S(3:62)*sumS^(-1);             %ha egyenletes beütésszámot akarünk látni

          %keressük meg amelyik középpontjához a legközelebb van
          M=((korcenter(:,:,1)-x).^2)+((korcenter(:,:,2)-y).^2);
          %M(2:(end-1),2:(end-1))=10000000000*ones(Size(1)-2,Size(2)-2);   %ha a közepe nem
kell

          N=ones(Size(1),Size(2));                                  %ha a széle nem kell
          N(2:(end-1),2:(end-1))=zeros(Size(1)-2,Size(2)-2);
          M=M+10000000000*N;
```

```
          [r,c]=find(M==min(min(M)));

            if length(r)==1          %ha van egyértelmű rácspont, akkor eltároljuk az adatokat
            linearInd = sub2ind(size(korcenter(:,:,1)), r, c);

            %zajszűrés energiaablakolással
            if 0.8*lokmax(linearInd)<sumS && sumS<1.2*lokmax(linearInd)

            LRFadatok(linearInd,:,1)=LRFadatok(linearInd,:,1)+(S(3:62))';    %átlag
            LRFadatok(linearInd,:,2)=LRFadatok(linearInd,:,2)+(S(3:62).^2)'; %négyzetes átlag
            LRFadatok(linearInd,:,3)=LRFadatok(linearInd,:,3)+1;          %felvillanások száma


LRFadatoknorm(linearInd,:,1)=LRFadatoknorm(linearInd,:,1)+(Snorm(3:62));    %átlag
LRFadatoknorm(linearInd,:,2)=LRFadatoknorm(linearInd,:,2)+(Snorm(3:62).^2); %négyzetes átlag
LRFadatoknorm(linearInd,:,3)=LRFadatoknorm(linearInd,:,3)+sumS;             %összenergia átlag

            end

            end

end
toc

%memóriaallokálás a mentendő adatok tömbjének
LRF=zeros(Size(1)*Size(2), 60);
VAR=zeros(Size(1)*Size(2), 60);
LRFnorm=zeros(Size(1)*Size(2), 60);
VARnorm=zeros(Size(1)*Size(2), 60);


for i=1:Size(1)*Size(2)
    LRF(i,:)=LRFadatok(i,:,1)./LRFadatok(i,:,3);    %átlag beütésszám adott rácshelyen
    VAR(i,:)=LRFadatok(i,:,2)./LRFadatok(i,:,3);    %négyzetes átlag az adott rácshelyen
    LRFnorm(i,:)=LRFadatoknorm(i,:,1)./LRFadatok(i,:,3);    %átlag beütésszám adott rácshelyen
    VARnorm(i,:)=LRFadatoknorm(i,:,2)./LRFadatok(i,:,3);    %négyzetes átlag az adott
rácshelyen
end
VAR=VAR-LRF.^2;                 %tényleges szórás
VARnorm=VARnorm-LRFnorm.^2;     %tényleges szórás
COUNT=LRFadatok(:,1,3);
SumS=LRFadatoknorm(:,1,3)./LRFadatok(:,1,3);


save('LRF11.mat','LRF')
save('VAR11.mat','VAR')
save('LRFnorm11.mat','LRFnorm')
save('VARnorm11.mat','VARnorm')
save('COUNT11.mat','COUNT')
save('SumS11.mat','SumS')

fclose(fid);
```

VII.2.7. interp_smoothingspline.m

```
%Az interp_smoothingspline.m fájl a 18 lrf_ii file összefésült végeredményét olvassa be, és az
azokban tárolt adatokat interpolálja a teljes 1024x1024-es rácsra 2d simított spline
illesztéssel.

clear all, close all

%memória allokálás a végső tömbök számára
LRF=zeros(1024,1024,60);
VAR=zeros(1024,1024,60);
LRFnorm=zeros(1024,1024,60);
VARnorm=zeros(1024,1024,60);
COUNT=zeros(1024,1024);
SumS=zeros(1024,1024);

% a furatok valós helyzeteinek koordinátái pixelekben(középső:(512,512)
% táv:8 pixel)
x=24:8:1000;
y=24:8:1000;
```

```
%a rács amire interpolálni szeretnénk:
xx=(x(1)):1:(x(end));
yy=(y(1)):1:(y(end));

%az lrf.m file által létrehozott tömbök betöltése
load('LRFMATLAB.mat');
load('VARMATLAB.mat');
load('LRFnormMATLAB.mat');
load('VARnormMATLAB.mat');
load('COUNTMATLAB.mat');
load('SumSMATLAB.mat');

%lrf.m adatainak betöltése
lrf=zeros(123,123,60);
var=zeros(123,123,60);
lrfnorm=zeros(123,123,60);
varnorm=zeros(123,123,60);
count=zeros(123,123);
sums=zeros(123,123);

lrf(18:end-17,4:end-3,:)=LRFall;
var(18:end-17,4:end-3,:)=VARall;
lrfnorm(18:end-17,4:end-3,:)=LRFnormall;
varnorm(18:end-17,4:end-3,:)=VARnormall;
count(18:end-17,4:end-3)=COUNTall;
sums(18:end-17,4:end-3)=SumSall;

%simító faktor
p={0.3,0.3};

for i=1:60

    %interpolált értékek létrehozása
    Lrf= csaps({x,y},lrf(:,:,i),p,{xx,yy});
    Var= csaps({x,y},var(:,:,i),p,{xx,yy});

    Lrfnorm= csaps({x,y},lrfnorm(:,:,i),p,{xx,yy});
    Varnorm= csaps({x,y},varnorm(:,:,i),p,{xx,yy});

    %interpolált értékek elhelyezése az 1024x1024-es képen
    LRF((y(1)):(y(end)),(x(1)):(x(end)),i)=Lrf;
    VAR((y(1)):(y(end)),(x(1)):(x(end)),i)=Var;

    LRFnorm((y(1)):(y(end)),(x(1)):(x(end)),i)=Lrfnorm;
    VARnorm((y(1)):(y(end)),(x(1)):(x(end)),i)=Varnorm;

end

    %interpolált értékek létrehozása
    Count= csaps({x,y},count,p,{xx,yy});
    Sums= csaps({x,y},sums,p,{xx,yy});

    %interpolált értékek elhelyezése az 1024x1024-es képen
    COUNT((y(1)):(y(end)),(x(1)):(x(end)))=Count;
    SumS((y(1)):(y(end)),(x(1)):(x(end)))=Sums;

%transzponálás

%adatok mentése

save('LRFcsapsMATLAB.mat','LRF')
save('VARcsapsMATLAB.mat','VAR')
save('LRFnormcsapsMATLAB.mat','LRFnorm')
save('VARnormcsapsMATLAB.mat','VARnorm')
save('COUNTcsapsMATLAB.mat','COUNT')
save('SumScsapsMATLAB.mat','SumS')
```

VII.3. Feldolgozott mérési adatok kiértékelése

VII.3.1. kiertekelo.m

```
% A file beolvassa a Nucline statisztikus képszámító moduljával készített
% képeket, majd a MATLAB regionprops függvényével statisztikát készítünk a
% képekről (rácspontok képének területe, középpontja, illeszkedő ellipszis
% kis és nagytengelye)

%képek beolvasása
B_norm=imread('bh_norm_nagyobb.png');
Bnorm=imread('bhnorm_nagyobb.png');
LS_norm=imread('LS_norm_nagyobb.png');
LSnorm=imread('LSnorm_nagyobb.png');

%képek átalakítása binárissá
BWB_norm = im2bw(B_norm, 0.1);
BWBnorm = im2bw(Bnorm, 0.1);
BWLS_norm = im2bw(LS_norm, 0.1);
BWLSnorm = im2bw(LSnorm, 0.1);

BWB_norm = bwareaopen(BWB_norm,20);
BWBnorm = bwareaopen(BWBnorm,20);
BWLS_norm = bwareaopen(BWLS_norm,20);
BWLSnorm = bwareaopen(BWLSnorm,20);

figure, imshow(BWB_norm,[])
figure, imshow(BWBnorm,[])
figure, imshow(BWLS_norm,[])
figure, imshow(BWLSnorm,[])

%teljes statisztika a képekről
STATSB_norm = regionprops(BWB_norm, 'all');
STATSBnorm = regionprops(BWBnorm, 'all');
STATSLS_norm = regionprops(BWLS_norm, 'all');
STATSLSnorm = regionprops(BWLSnorm, 'all');

%a releváns adatok kinyerése a teljes statisztikából
for i=1:length(STATSB_norm)
    B_normArea(i)=(STATSB_norm(i).Area);
    B_normMajor(i)=(STATSB_norm(i).MajorAxisLength);
    B_normMinor(i)=(STATSB_norm(i).MinorAxisLength);
    B_normMajor_Minor(i)=(STATSB_norm(i).MajorAxisLength)/(STATSB_norm(i).MinorAxisLength);
    B_normCentroid(i,:)=STATSB_norm(i).Centroid;
end

for i=1:length(STATSBnorm)
    BnormArea(i)=(STATSBnorm(i).Area);
    BnormMajor(i)=(STATSBnorm(i).MajorAxisLength);
    BnormMinor(i)=(STATSBnorm(i).MinorAxisLength);
    BnormMajor_Minor(i)=(STATSBnorm(i).MajorAxisLength)/(STATSBnorm(i).MinorAxisLength);
    BnormCentroid(i,:)=STATSBnorm(i).Centroid;
end

for i=1:length(STATSLS_norm)
    LS_normArea(i)=(STATSLS_norm(i).Area);
    LS_normMajor(i)=(STATSLS_norm(i).MajorAxisLength);
    LS_normMinor(i)=(STATSLS_norm(i).MinorAxisLength);
    LS_normMajor_Minor(i)=(STATSLS_norm(i).MajorAxisLength)/(STATSLS_norm(i).MinorAxisLength);
    LS_normCentroid(i,:)=STATSLS_norm(i).Centroid;
end

for i=1:length(STATSLSnorm)
    LSnormArea(i)=(STATSLSnorm(i).Area);
    LSnormMajor(i)=(STATSLSnorm(i).MajorAxisLength);
    LSnormMinor(i)=(STATSLSnorm(i).MinorAxisLength);
    LSnormMajor_Minor(i)=(STATSLSnorm(i).MajorAxisLength)/(STATSLSnorm(i).MinorAxisLength);
    LSnormCentroid(i,:)=STATSLSnorm(i).Centroid;
end

%átlagok és szórások számítása
M=zeros(8,4);

M(1,1)=mean(B_normArea);
```

```
M(1,2)=std(B_normArea);
M(1,3)=mean(BnormArea);
M(1,4)=std(BnormArea);

M(2,1)=mean(B_normMajor);
M(2,2)=std(B_normMajor);
M(2,3)=mean(BnormMajor);
M(2,4)=std(BnormMajor);

M(3,1)=mean(B_normMinor);
M(3,2)=std(B_normMinor);
M(3,3)=mean(BnormMinor);
M(3,4)=std(BnormMinor);

M(4,1)=mean(B_normMajor_Minor);
M(4,2)=std(B_normMajor_Minor);
M(4,3)=mean(BnormMajor_Minor);
M(4,4)=std(BnormMajor_Minor);


M(5,1)=mean(LS_normArea);
M(5,2)=std(LS_normArea);
M(5,3)=mean(LSnormArea);
M(5,4)=std(LSnormArea);

M(6,1)=mean(LS_normMajor);
M(6,2)=std(LS_normMajor);
M(6,3)=mean(LSnormMajor);
M(6,4)=std(LSnormMajor);

M(7,1)=mean(LS_normMinor);
M(7,2)=std(LS_normMinor);
M(7,3)=mean(LSnormMinor);
M(7,4)=std(LSnormMinor);

M(8,1)=mean(LS_normMajor_Minor);
M(8,2)=std(LS_normMajor_Minor);
M(8,3)=mean(LSnormMajor_Minor);
M(8,4)=std(LSnormMajor_Minor);

%foltok középpontjának távolsága a tényleges rácskoordinátáktól
load('XYkoord.mat');

for i=1:length(B_normMinor)
    B_Elteres(i)=min((B_normCentroid(i,1)-XY(:,1)).^2+(B_normCentroid(i,2)-XY(:,2)).^2).^0.5;
end

 mean(B_Elteres)
 std(B_Elteres)

for i=1:length(LS_normMinor)
    LS_Elteres(i)=min((LS_normCentroid(i,1)-XY(:,1)).^2+(LS_normCentroid(i,2)-
XY(:,2)).^2).^0.5;
end

%átlag és szórás
mean(LS_Elteres)
std(LS_Elteres)
```

VIII. Hivatkozások

1. Online Orvosi Fizika Tankönyv, Nukleáris Medicina fejezet: http://oftankonyv.reak.bme.hu/tiki-index.php?page=Nukle %C3%A1ris%20medicina %20fizikusoknak&structure=Tank%C3%B6nyv%20Fizikusoknak

2. Hal O. Anger: Scintillation Camera with Multichannel Collimators, Journal of Nuclear Medicine, No. 5., pp. 515-531., 1964

3. John Vesel és Micheal Petrillo: Improved Gamma Camera Performance Using Event Positioning Method Based On Distance Dependent Weighting, IEEE Nuclear Science Symposium Conference Record, 2005

4. J. Joung, R. S. Miyaoka, S. G. Kohlmyer, és T. K. Lewellen: Investigation of Bias-Free Positioning Estimators for the Scintillation Cameras, IEEE Transactions on Nuclear Science, Vol. 48., No. 3, pp. 715-719., 2001

5. Jinhun Joung, Robert S. Miyaoka, Steven Kohlmyer, Tom K. Lewellen: Implementation of ML based positioning algorithms for Scintillation Cameras, Transactions on Nuclear Science, Vol. 47., No. 3., pp. 1104-1111., 2000

6. N. A. Thacker, F. J. Aherne and P. I. Rockett: The Bhattacharyya Metric as an Absolute Similarity Measure for Frequency Coded Data, Kybernetika, Vol. 34., No. 4., pp. 363-368., 1997

7. C.O. Steinbach, A. Szlavecz, B. Benyo, T. Bukki, E. Lorincz,Validation of Detect2000-Based PetDetSim by Simulated and Measured Light Output of Scintillator Crystal Pins for PET Detectors, IEEE Transactions on Nuclear Science, Vol. 57., No. 5, pp. 20460-2467., 2010

8. NEMA NU 1-2012:

http://www.nema.org/standards/Pages/Performance-Measurements-of-Gamma-Cameras.aspx

9. Carlos Adrian Vargas Aguilera, Extrema2.m:

http://www.mathworks.com/matlabcentral/fileexchange/12275-extrema-m--extrema2-m