ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta elektrotechnická
Katedra elektroniky a informačních technologií

BAKALÁŘSKÁ PRÁCE

Software pro analýzu dat pixelových detektorů částic

Autor práce: **Guillian Jacquot**
Vedoucí práce: **Ing. Ondřej Urban**

2022

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta elektrotechnická

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Guillian JACQUOT**
Osobní číslo:	**E19B0092P**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronika a telekomunikace**
Téma práce:	**Software pro analýzu dat pixelových detektorů částic**
Zadávající katedra:	**Katedra elektroniky a informačních technologií**

Zásady pro vypracování

1. Seznamte se s pixelovými detektory ionizujícího záření Timepix.
2. Vytvořte software pro základní statistickou analýzu dat získaných pomocí těchto detektorů.
3. Ověřte funkčnost vytvořeného software analýzou reálných naměřených dat.

Rozsah bakalářské práce: **30 – 40**
Rozsah grafických prací: **dle doporučení vedoucího**
Forma zpracování bakalářské práce: **elektronická**

Seznam doporučené literatury:

ANDĚL, Jiří. *Základy matematické statistiky*. Vyd. 3. Praha: Matfyzpress, 2011. ISBN 978-80-7378-162-0.

Vedoucí bakalářské práce: **Ing. Ondřej Urban**
Katedra elektroniky a informačních technologií

Datum zadání bakalářské práce: **8. října 2021**
Termín odevzdání bakalářské práce: **26. května 2022**

L.S.

Prof. Ing. Zdeněk Peroutka, Ph.D.
děkan

Doc. Ing. Jiří Hammerbauer, Ph.D.
vedoucí katedry

V Plzni dne 8. října 2021

Abstrakt

Bakalářská práce je zaměřena na realizaci softwaru pro podrobnější statistickou analýzu částečně zpracovaných dat z polovodičových detektorů částic Timepix. Software je postaven na platformě Microsoft .NET (dotnet) v integrovaném vývojovém prostředí (IDE) Microsoft Visual Studio v rámci knihovny Windows Presentation Foundation (WPF) zahrnuté v dotnetu a umožňuje vykreslovat několik typů histogramů ukazujících rozdělení různých vlastností částic nepřímým měřením při kolizi částic s citlivou oblastí detektoru Timepix.

Klíčová slova

Software, Timepix, Detektor, Ionizující záření, Analýza dat, Histogram, C#, Microsoft .NET, dotnet, WPF, XAML

Abstract

This bachelor thesis is focused on the implementation of software for more detailed statistical analysis of partially processed data from Timepix semiconductor particle detectors. The software is built on the Microsoft .NET platform (dotnet) in the Microsoft Visual Studio integrated development environment (IDE) within the Windows Presentation Foundation (WPF) library included in dotnet and allows plotting several types of histograms showing the distribution of various particle properties by indirect measurement when particles collide with the sensitive region of the Timepix detector.

Key Words

Software, Timepix, Detector, Ionizing radiation, Data analysis, Histogram, C#, Microsoft .NET, dotnet, WPF, XAML

Poděkování

Tímto bych rád poděkoval vedoucímu bakalářské práce Ing. Ondřejovi Urbanovi za nenahraditelnou a efektivní zpětnou vazbu, konstruktivní kritiku a cenné rady z praxe při vývoji softwaru.

Obsah

Seznam symbolů a zkratek

dotnet .....Microsoft .NET Framework

IDE .....Integrated Development Environment

WPF.....Windows Presentation Foundation

XAML .....Extensible Application Markup Language

MVVM .....Model View View-Model

THL .....Threshold

ToT .....Time over Threshold

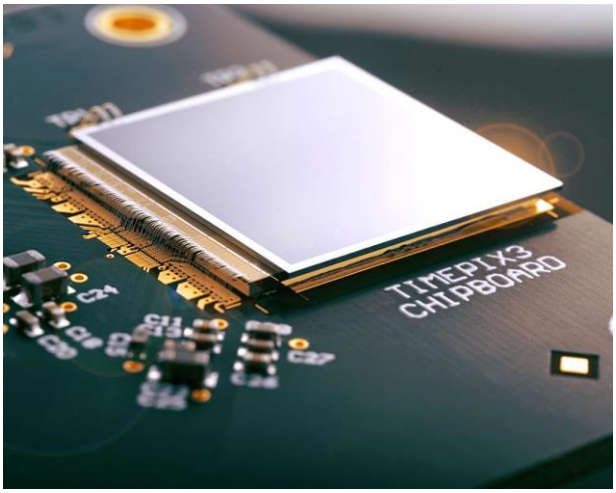
Úvod

Bakalářská práce se zabývá realizací softwaru pro podrobnější statistickou analýzu částečně zpracovaných dat z polovodičových detektorů částic Timepix. Software je napsán v programovacím jazyce C# na platformě Microsoft .NET 6.0 (dotnet)[1] a umožňuje jednoduchou vizuální analýzu vlastností ionizujícího záření dopadajícího na detektor (např. odvození druhů záření, úhly dopadu, distribuce energie apod.). Tato analýza například umožňuje přímo zkoumat parametry částic zanechávající obrazce na detektoru při kolizi a efektivně částice klasifikovat nebo z grafů odhadnout intervaly očekávaných hodnot a kalibrovat jiné nástroje či software na tento interval.

První kapitola je zaměřena na teorii detektorů Timepix, jejich nastavitelné hodnoty a vysvětlení hodnot zaznamenávaných detektorem. Druhá kapitola detailně popisuje vývoj softwaru pro statistickou analýzu dat z detektoru a implementaci algoritmů potřebných k analýze. Třetí kapitola porovnává výsledky z měření za pomoci histogramů vytvořených softwarem.

1 Hardware

V následujících kapitolách je popsán použitý hardware, sloužící k akvizici dat, které jsou následně dále softwarem zpracovávány.



Obr. 1: Detektor Timepix3, převzato z: [10]

1.1 Princip detektoru Timepix

Polovodičové detektory Timepix součástí rodiny čipů Medipix vyvinuté Evropským centrem jaderného výzkumu (CERN) fungují na principu povrchového p-n přechodu (citlivá oblast detektoru), který je schopen detekovat dopady částic předávající kinetickou energii plochám detektoru.

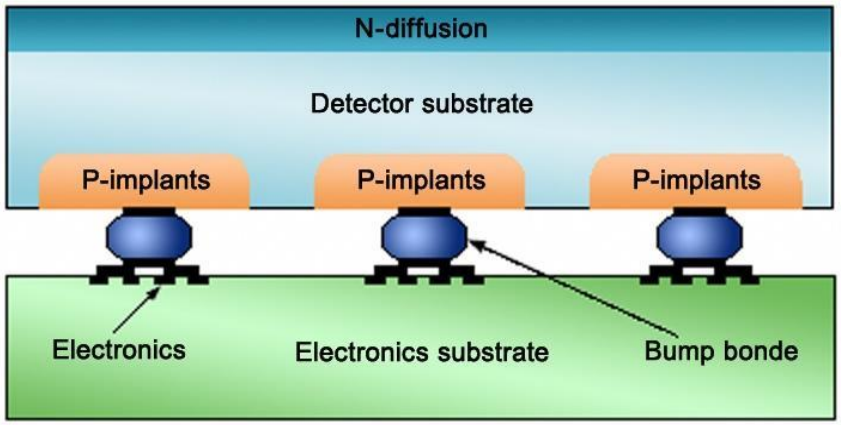
Plocha detektoru je diodou v závěrném směru. Na plochu přivádíme napětí (hodnota Bias) způsobující vyprázdnění p-n přechodu. Napětí nesmí překročit maximální elektrickou pevnost, což by způsobilo průraz a zničení čipu. Maximální napětí je závislé na materiálu citlivé oblasti (Si, GaAs, CdTe) a její tloušťce. Při nižším napětí nebo větší tloušťce citlivé oblasti se plochou náboj více rozlévá a tvoří mnohem větší „bloby“ (viz Obr. 22).

V ustáleném stavu plochou prochází minimální proud a spotřeba je v řádu stovek mW. Částice kolidující s citlivou oblastí detektoru v místě dopadu vytvoří volný náboj, který se určitou dobu odčerpává a pro každou čtecí buňku obrazového bodu (pixel) na detektoru se počítá, kolik hodinových pulzů uplyne, než náboj v určitém místě klesne pod nastavitelnou komparační úroveň závislé na hodnotě Threshold (viz Obr. 3).[3]

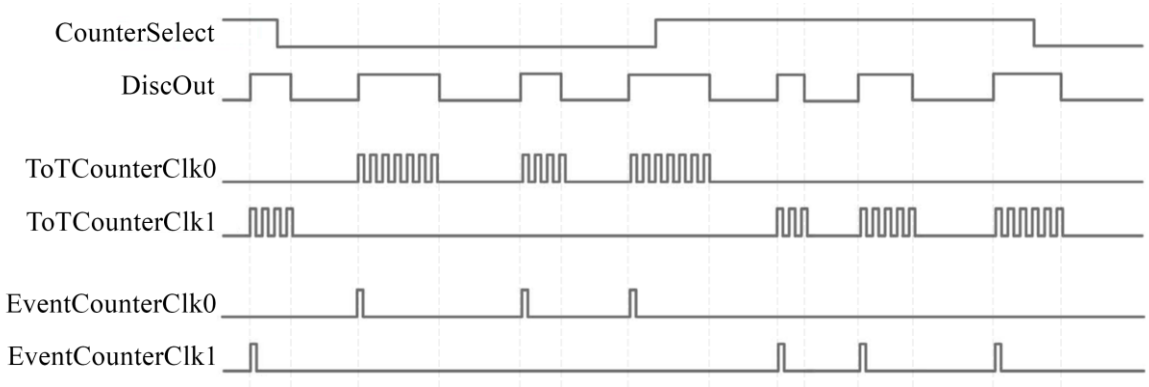
Po zpracování pro každou událost detekce dostáváme informace o souřadnicích X/Y a hodnotu Time Over Threshold (ToT), což je předem zmíněný počet hodinových pulzů. Z hodnoty ToT můžeme odvodit množství energie předané každému pixelu.

Hodnota ToT mimo jiné závisí na frekvenci hodinových pulzů a nastavené komparační úrovni, takže je potřeba jednotky spočítat z aktuálního nastavení, čehož vyvinutý software zatím není schopen. Proto bylo učiněno rozhodnutí ToT v softwaru zobrazovat jako bezrozměrnou jednotku pro zabránění vzniku chybné interpretace hodnot.

Pod citlivou oblastí jsou ve čtvercové mřížce umístěny mikroskopické kuličky, které každý pixel spojují s vyčítacím čipem (Bump Bonding, viz Obr. 2).[11]



Obr. 2: Průřez detektorem, převzato z [9]



Obr. 3: Mód průběžného čtení ToT (DiscOut = stav překročení Thresholdu), převzato a upraveno z [3]

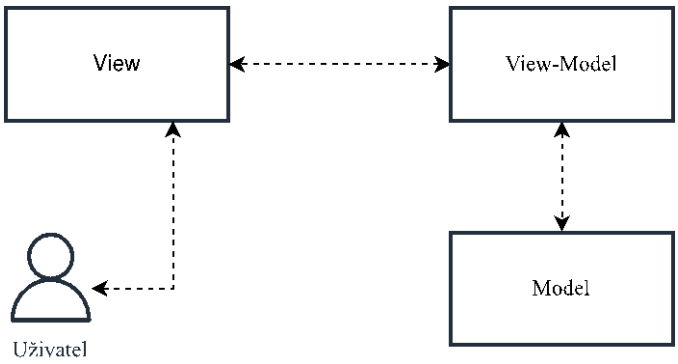
2 Software

Při vývoji na platformě dotnet[1] byla vybrána open-source architektura Windows Presentation Foundation (WPF)[2] pro její jednoduchou a rychlou implementaci grafického rozhraní ve značkovacím jazyce Extensible Application Markup Language (XAML)[4] vyvíjeného společností Microsoft založeném na principech Hypertext Markup Language (HTML).

Ačkoliv je dotnet multiplatformní (aplikace fungují v operačních systémech Windows, Linux, macOS, Android, iOS), WPF zatím multiplatformní není a funguje pouze v operačním systému Windows. Další limitací WPF je značná složitost vykreslování bodových grafů bez užití knihoven. Pro zjednodušení práce s grafy byla zvolena open-source knihovna ScottPlott WPF[5] s licencí MIT umožňující software zdarma používat, kopírovat, modifikovat a prodávat.

2.1 Architektura softwaru, MVVM

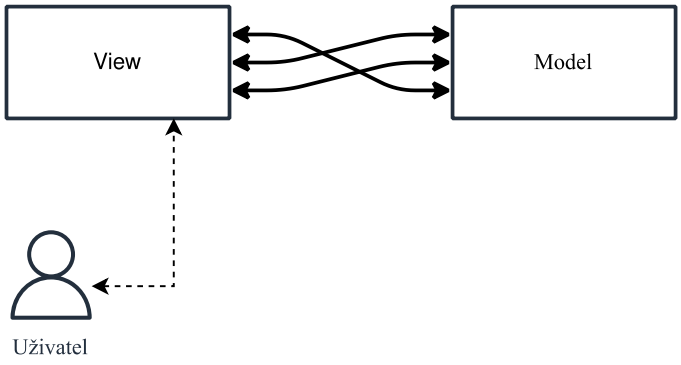
Cílem bylo při vývoji softwaru použít návrhový vzor s názvem Model View View-Model (MVVM)[6] doporučený a značně používaný společností Microsoft, kde je úkolem čistě oddělit obchodní logiku (business logic - algoritmy, matematické výpočty, práce s daty, práce s pamětí) od prezentační logiky (presentation logic - grafické rozhraní, interakce s uživatelem). V praxi se také používají názvy back-end pro obchodní logiku a front-end pro prezentační logiku. Veškerá obchodní logika je umístěna v části Model, prezentační logika v části View a část View-Model má na starost spojení těchto přísně enkapsulovaných objektů.



Obr. 4: Blokové schéma vzoru MVVM

Separováním obchodní a prezentační logiky vzorem MVVM zajistíme čistý vzhled kódu umožňující rychlejší pochopení programátorem, který s kódem nebyl seznámen. Dále separací zajistíme usnadnění údržby aplikace, testování aplikace a takzvané volné spojení (v překladu Loose Coupling), což znamená jednoduchost odtržení libovolného objektu ze skupiny Model nebo View této aplikace a použití objektu v jiné aplikaci.

Při přebrání objektů z jiné aplikace by teoreticky mělo stačit pracovat pouze na částech View-Model, které jsou laicky řečeno lepidlem spojujícím vše dohromady. V opačném případě pevného spojení (v překladu Tight Coupling) při odtržení objektu vznikne řada nejasností jak objekt správně implementovat a programátor je nucen pozorně a zdlouhavě studovat jak byl objekt původně se zbytkem původní aplikace spojen, jelikož není jasné kde obchodní nebo prezentační logika začíná a končí. Objekt také může být při pevném spojení kontaminován logikou z jiných objektů, která do něj vůbec nepatří a je vyžadována pouze v implementaci původní aplikace.



Obr. 5: Neideální případ pevného spojení objektů (Tight Coupling)

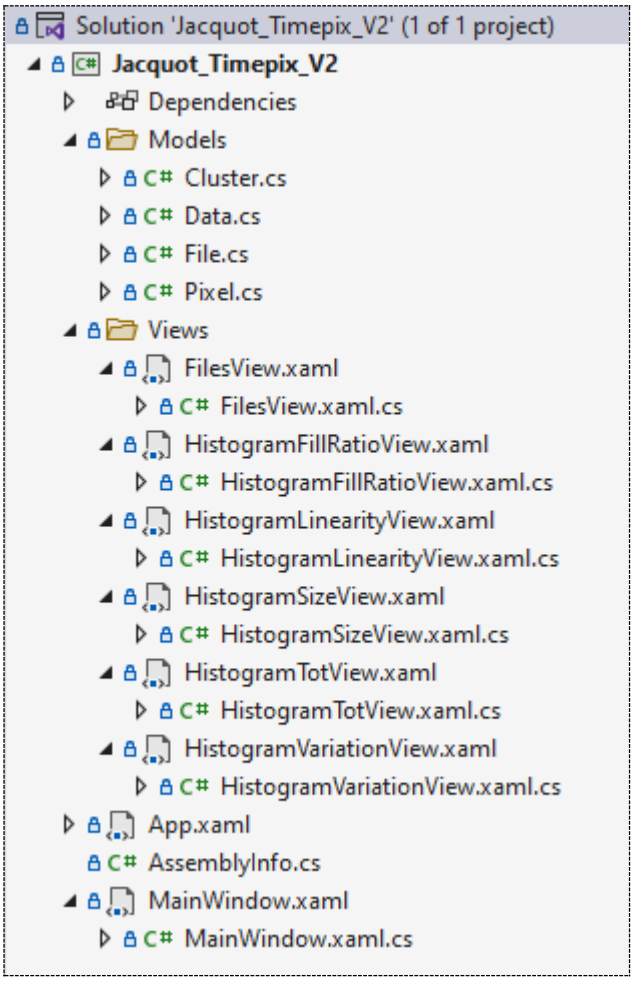
Proti vzoru MVVM se často objevuje kritika, že vzor malé aplikace zbytečně zesložiťuje a dobré praktiky pro vývoj softwaru má cenu využívat pouze u velkých projektů.[7][8]

Z počátku byl vzor použit. Později bylo učiněno rozhodnutí vzor opustit vzhledem k relativně malé velikosti projektu. V aplikaci je ale stále zachované rozdělení na složky Models zcela zaměřené na back-end a složky Views převážně zaměřené na front-end.

2.2 Vyvinutá aplikace

Následující kapitoly jsou zaměřeny na podrobný popis fungování aplikace. Předpokládají se základní znalosti v integrovaném vývojovém prostředí (IDE) Microsoft Visual Studio a programovacího jazyku C#.

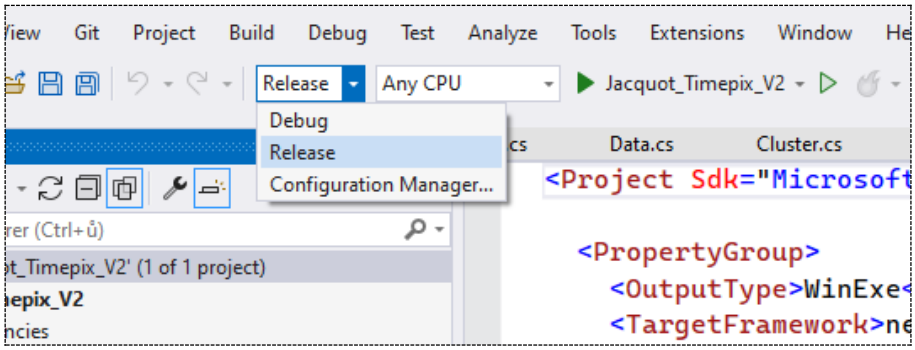
2.2.1 Struktura projektu uvnitř IDE



Obr. 6: Struktura projektu

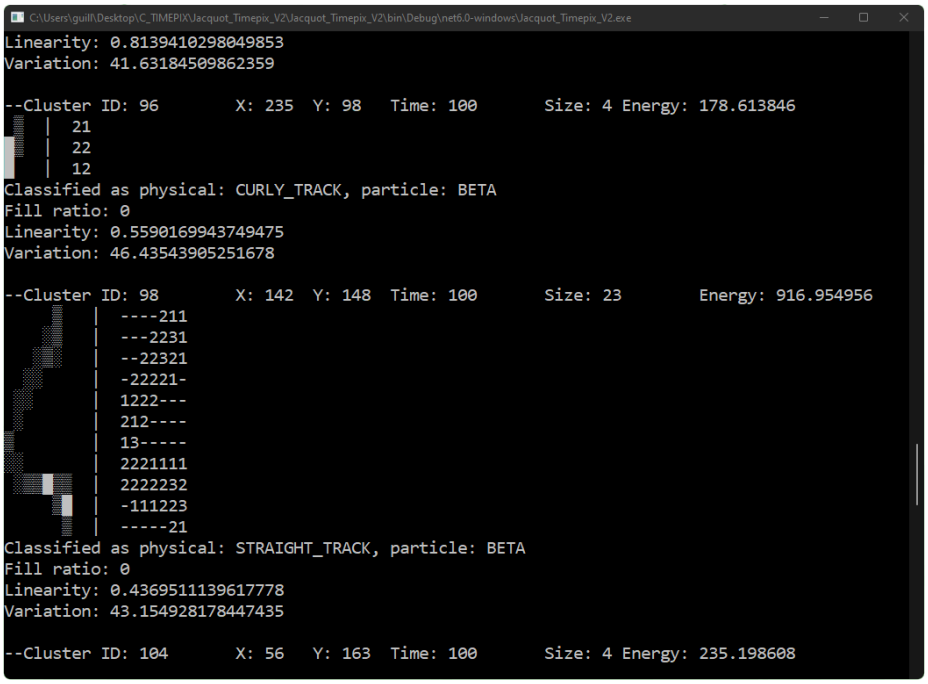
Jak již bylo zmíněno na konci kapitoly: *2.1 Architektura softwaru, MVVM*, aplikace je rozčleněna na část Models obsahující převážně obchodní logiku, která mimo jiné pracuje s importováním dat ze souboru měření nebo s algoritmy využitých ve výpočtech vlastností zobrazovaných v histogramech a na část Views obsahující prezenční logiku (design

uživatelského rozhraní, tlačítka apod.). Celý projekt je dostupný ke stažení v přílohách práce.



Obr. 7: Volby konfigurace (horní lišta Visual Studia)

Projekt lze zkompilovat v konfiguraci režimu ladění „Debug“ nebo v režimu „Release“. V režimu ladění se aplikace spustí společně s konzolí (viz Obr. 8), kterou autor využíval při vývoji aplikace a hledání zajímavých clusterů použitelných pro znázornění jejich vlastností (viz Obr. 17). Konzole se u každého clusteru zastaví a pokračovat lze stisknutím libovolné klávesy. Po importování souborů se konzole automaticky zavírá. **Při normálním užívání aplikace je nutné projekt před kompilací přepnout do režimu „Release“.**



Obr. 8: Konzole při konfiguraci „Debug“

2.2.2 Princip aplikace

Základním požadavkem pro aplikaci je schopnost importovat data z měření pro další analýzu. Katedra elektroniky a informačních technologií (KEI) již software pro částečné zpracování a klasifikaci typů částic má, proto aplikace nebude přebírat surová data z detektoru Timepix, ale tyto částečně zpracovaná data.

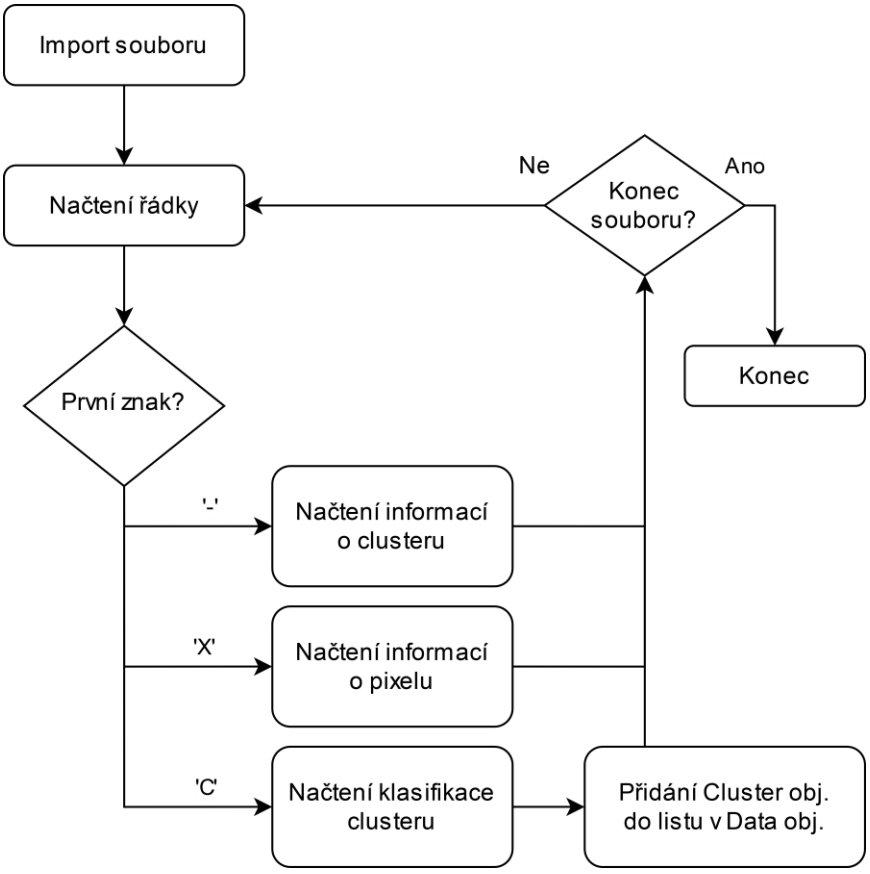
Zatímco surová data z detektoru obsahují pouze souřadnice X/Y a ToT, částečně zpracovaná data přicházejí v tomto formátu:

```
--Cluster ID: 1    X: 58    Y: 99    Time: 100        Size: 3   Energy: 6238.454102
X: 57    Y: 99    ToT: 2008
X: 58    Y: 99    ToT: 4814
X: 59    Y: 99    ToT: 5737
Classified as physical: CURLY_TRACK, particle: BETA

--Cluster ID: 2    X: 254   Y: 119   Time: 100        Size: 3   Energy: 8205.732422
X: 253   Y: 119   ToT: 1300
X: 254   Y: 119   ToT: 5508
X: 255   Y: 119   ToT: 6392
Classified as physical: CURLY_TRACK, particle: BETA
```

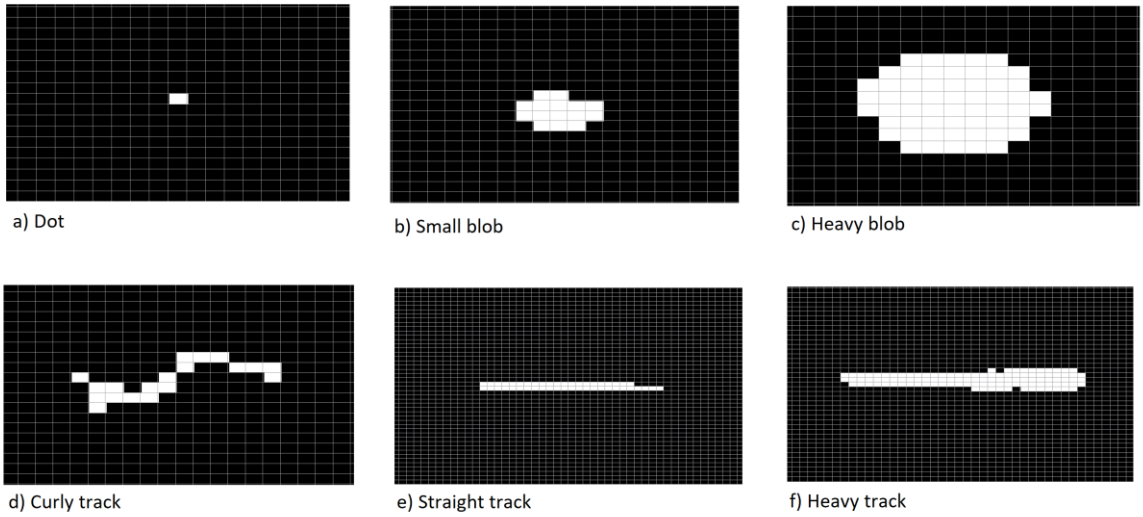
Individuální události z každého pixelu jsou seřazeny do clusterů, kde souřadnice X/Y clusteru jsou průměrem souřadnic pixelů a velikost (Size) je počtem pixelů. V budoucnu by bylo vhodné skript, který data částečně zpracovává, implementovat do této aplikace pro odstranění potřeby data z měření importovat/exportovat skrz dvě různé aplikace.

Pro každý import clusterů je vytvořen Data objekt, který se přidá do listu. Funkci parsující (přeměna dat z jednoho druhu na jiný) clustery z částečně zpracovaných dat do nového Data objektu, lze nalézt v souboru \Models\Data.cs a vypadá přibližně takto:



Obr. 9: Vývojový diagram importování částečně zpracovaných dat

Informace se načítají po řádcích a pro každý řádek z původního souboru se vytvoří pole slov. Podle prvního znaku (char) prvního slova se rozhodne, zdali navazující větev bude parsovat informace o clusteru, jednoho z pixelů nebo klasifikaci, kterou je každý cluster ukončen. Po záznamu klasifikace je cluster kompletní a vytvoří se nový objekt Cluster, který se přidá do listu clusterů obsaženého v každém Data objektu. Smyčka pokračuje, dokud nedorazí na konec importovaného souboru. Ve vývojovém diagramu Obr. 9 chybí sjednocení desetinné čárky/tečky a průběžné ověřování správnosti dat.

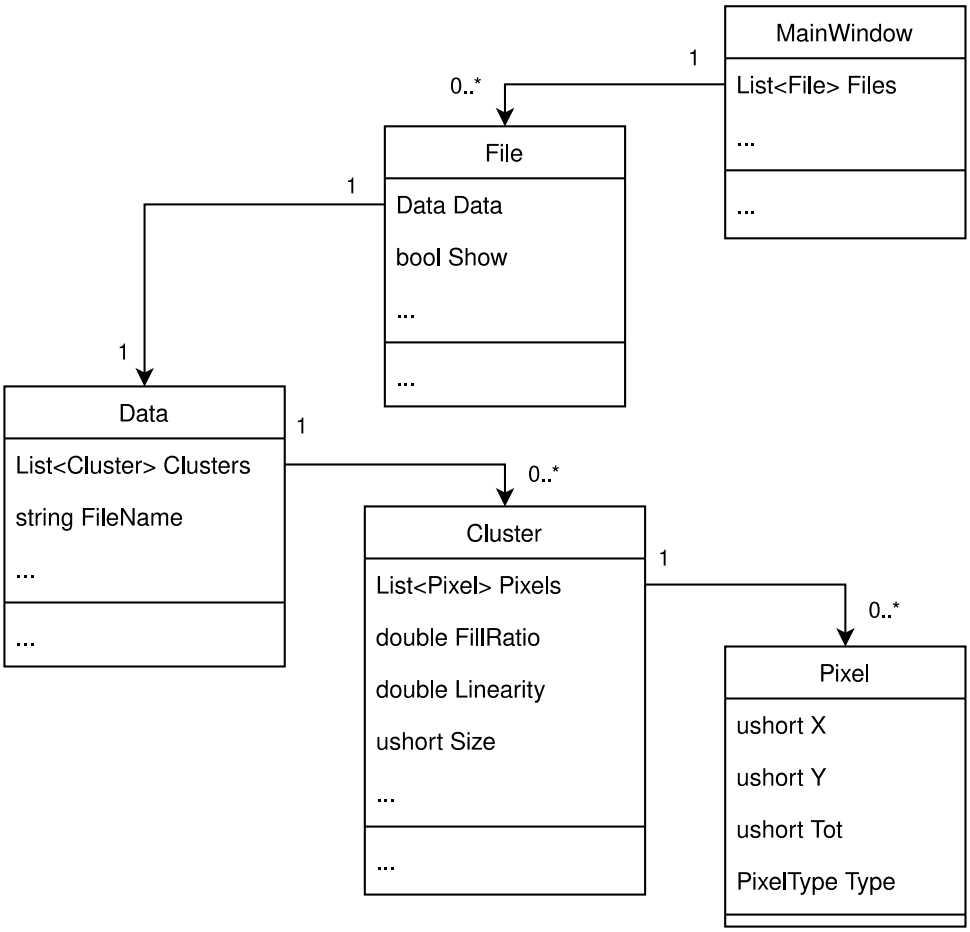


Obr. 10: Možné klasifikace clusterů, převzato z [12]

Všechny Clustery mají list[1] Pixel objektů (třída Pixel je k nalezení uvnitř souboru \Models\Pixel.cs). Každý Pixel má 4 vlastnosti: X, Y, Tot a PixelType. Až na výčtový typ (enum) PixelType, který rozlišuje okrajové a vnitřní pixely (použito při výpočtu FillRatio), jsou ostatní 3 vlastnosti datovým typem ushort (unsigned short, 16-bitová proměnná od 0 do 65535). Pro Timepix1 je rozlišení citlivé oblasti 256 x 256 pixelů, takže by teoreticky 8-bitová (0 až 255) proměnná na jediný detektor stačila, ale rozlišení budoucích detektorů může narůst a data také mohou přicházet z více propojených senzorů, na což by již 8 bitů nestačilo.

Okolo každého Data objektu je obálka ve formě třídy File, která už obsahuje funkce nezbytné pro manipulaci v různých částech aplikace - převážně funkce pro vykreslování histogramů. Prezenční logika aplikace nepracuje přímo s Data objekty, pracuje s File objekty.

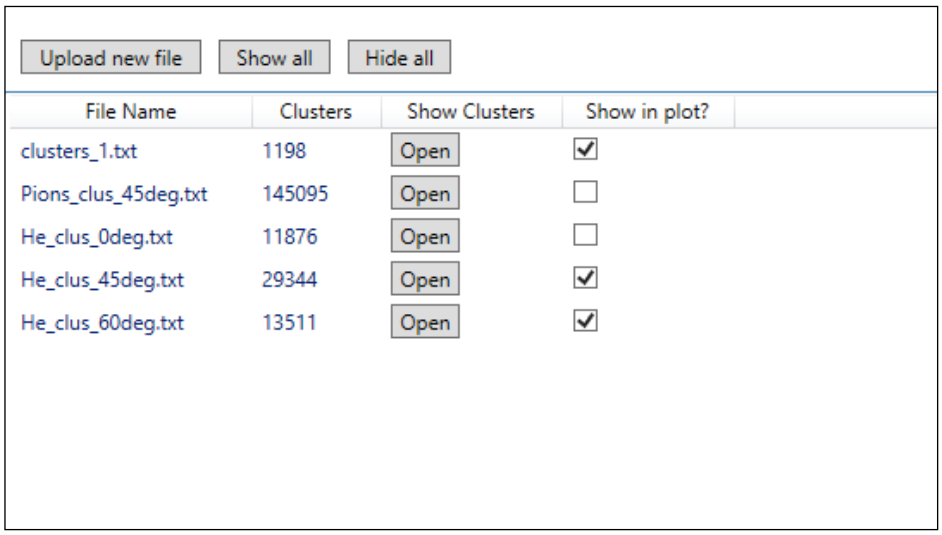
[1] I v případě zpracování jediného pixelu jako jednoho clusteru (většinou gamma/rentgenové fotony nebo šum) je tento Pixel uvnitř listu o délce jedna. Z objektu Data se k pixelům vždy musí přes clustery.



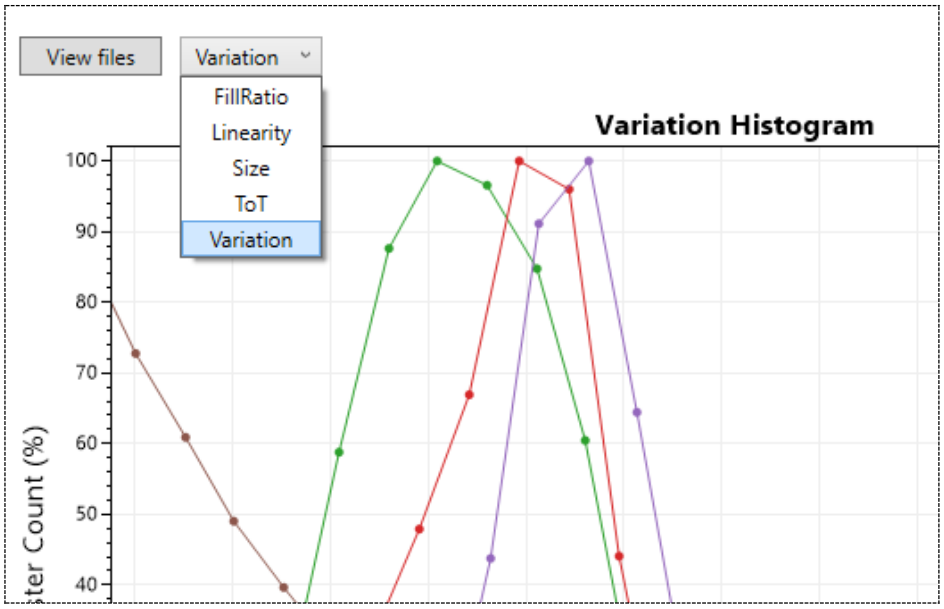
Obr. 11: Struktura načtených dat v aplikaci[2]

Nová data lze do aplikace nahrávat otevřením dialogu přes tlačítko v levém horním rohu hlavního okna „View files“ (viz Obr. 13) a poté tlačítkem „Upload new file“ také v levém horním rohu nově otevřeného dialogu (viz Obr. 12). Tento dialog dále umožňuje data z každého nahraného souboru skrýt nebo ukázat (Show) na histogramech. Dalším ovládacím prvkem hlavního okna je rozbalovací seznam (combo box), kterým lze vybírat z 5 různých typů histogramů: FillRatio, Linearity, Size, ToT, a Variance. Kromě histogramu ToT, kde se ignoruje seskupení po clusterech, se u všech ostatních vykreslují počty (Count) jedné z hodnot / vlastností clusteru. Histogramy jsou podrobněji popsány v následujících kapitolách.

[2] Proměnná „PixelType“ je výčtový typ (enum) pro rozlišení vnitřních/vnějších pixelů (viz kapitola: 2.2.3)



Obr. 12: View files dialog okno



Obr. 13: Ovládací prvky hlavního okna aplikace[3]

Histogramy jsou v aplikaci normalizovány tak, aby nejčetnější hodnota měla v grafu na svislé ose úroveň 100%, všechny ostatní hodnoty jsou pak k tomuto maximu vztažené. Bez normalizace by počty hodnot byly závislé na době měření a množství zaznamenaných dat, což by mohlo znemožnit porovnávání tvarů křivek několika různě dlouhých měření v jednom grafu.

[3] Graf lze v aplikaci uložit pravým kliknutím na okno grafu a vybráním možnosti: Save Image.

2.2.3 Analýza poměru vnitřních/vnějších pixelů v Clusteru (FillRatio)

Různé druhy částic tvoří velké rozdíly ve tvaru clusteru (viz Obr. 10). Pro hodnotu FillRatio platí jednoduchý poměr:

$$\text{FillRatio} = \frac{p_V}{p_V + p_K}$$

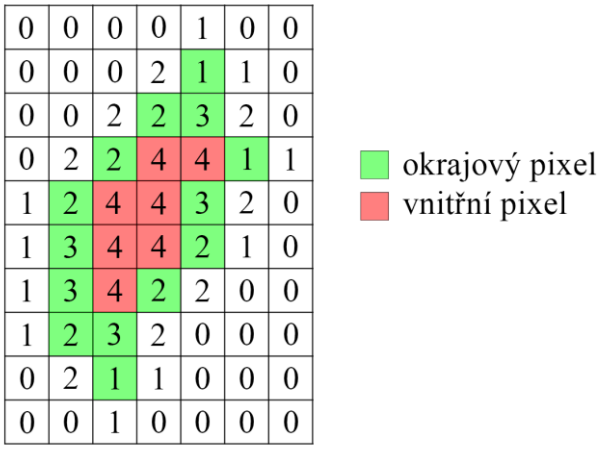
kde p_V je počtem vnitřních pixelů a p_K počtem vnějších pixelů (ve jmenovateli je tedy počet všech pixelů). FillRatio se může pohybovat pouze v intervalu od 0 do 1 a v aplikaci je hodnota uložena jako double (64-bitová proměnná s přesností na 15 desetinných míst). Proměnná double ale může nabývat hodnot v intervalu přibližně od $-1{,}7977 \cdot 10^{308}$ do $1{,}7977 \cdot 10^{308}$, takže by v případě, že se v budoucnu objeví potíže s pamětí, dávalo větší smysl interval od 0 do 1 pro hodnotu FillRatio interně ukládat v efektivnějším formátu.

Pixel je v aplikaci označen jako vnitřní, pokud má 4 sousední pixely, se kterými sdílí stranu. V deklaraci třídy Pixel uvnitř souboru \Models\Pixel.cs lze tato pravidla jednoduše podle potřeby upravit:

```
public class Pixel
{
    public static readonly int insidePixelNeighbours = 4;
    public static readonly bool checkCornerNeighbours = false;
// ...
```

Pravidla rozhodující o vnitřních pixelech není třeba za chodu aplikace měnit, proto jsou v kódu stanovena pevně, ale teoreticky by nemělo být příliš obtížné tuto část přepsat tak, aby ji bylo možné měnit v uživatelském rozhraní.

Funkce pro třídění pixelů na vnitřní/krajní a výpočet hodnoty FillRatio si nejdříve dočasně vytvoří pole 8-bitových hodnot, do kterého si bude ukládat počet sousedů pro každý pixel (viz Obr. 14). Program cykluje každou buňkou pole a při každém cyklu se podle pravidel podívá buď na 4 sousední buňky sdílející stranu, nebo všech 8 sousedů okolo buňky zahrnující buňky sdílející roh a pro každý nalezený sousední pixel inkrementuje 8-bitovou hodnotu v buňce, kde zrovna cyklus je. Po spočítání všech sousedů program znovu projede buňky pole obsahující pixely a podle pravidel rozhoduje, zdali je každý pixel vnitřní nebo krajní.



Obr. 14: Pole sousedních pixelů; 4 sousedi = vnitřní pixel

FillRatio je dalším nástrojem jak rozeznat různé druhy částic při měření, jelikož má každý druh relativně typický tvar clusteru. Největší hodnoty FillRatio se objevují u záření alfa (viz Obr. 17). Velký vliv má mimo jiné úhel dopadu částic.

2.2.4 Analýza linearity Clusterů

Klasifikaci částic ulehčuje pozorování tvaru Clusteru. Linearitu každého Clusteru aplikace počítá tímto vzorcem:

$$Linearity = \frac{l_{max}}{p}$$

kde l_{max} je maximální vzdálenost mezi dvěma pixely v clusteru. Implementace tohoto algoritmu v kódu vypadá takto:

```
double GetLinearity()
{
    List<Pixel> edgePixels = Pixels.FindAll(p => p.Type == PixelType.Edge);

    if (edgePixels.Count <= 1)
    {
        return 0;
    }

    List<Pixel> done = new();
    int lengthSquaredMax = 0;

    // checks every pair of 2 pixels exactly once
    foreach (Pixel pix1 in edgePixels)
    {
        done.Add(pix1);
        foreach (Pixel pix2 in edgePixels.Except(done))
        {
            int dx = pix2.X - pix1.X;
            int dy = pix2.Y - pix1.Y;

            // instead of square rooting every loop,
            // compare squared values and square root only once at the end
            int lengthSquared = dx * dx + dy * dy;
            lengthSquaredMax = lengthSquared > lengthSquaredMax ?
                lengthSquared : lengthSquaredMax;
        }
    }

    // square root
    double lengthMax = Math.Pow(lengthSquaredMax, 0.5);

    return lengthMax / Pixels.Count;
}
```

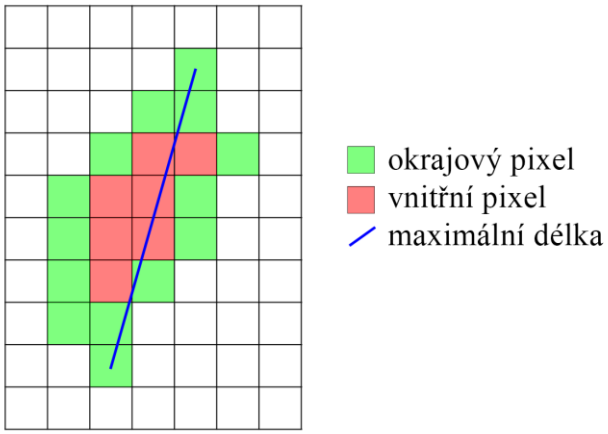
Jelikož maximální vzdálenost nemůže být mezi dvěma vnitřními pixely (okolo každého vnitřního pixelu jsou vždy okrajové pixely, viz Obr. 15), funkce porovnává vzdálenosti pouze v listu „edgePixels. Pokud je počet vnějších pixelů menší nebo roven jedné (cluster obsahuje pouze jeden pixel nebo někde došlo k chybě), funkce okamžitě vrací hodnotu linearity nula. Následující smyčka ověřuje vzdálenosti mezi každou dvojicí pixelů právě jednou[4] vzorcem:

$$l = \sqrt{(p_{2x} - p_{1x})^2 + (p_{2y} - p_{1y})^2}$$

kde l je vzdálenost mezi dvěma pixely, p_{nx} je souřadnice x pixelu n a p_{ny} je souřadnice y pixelu n. V kódu je smyčka urychlena vynecháním odmocniny při každém cyklu

[4] Pro zamezení spočtení vzdálenosti z pixelu A do pixelu B a později z pixelu B do pixelu A (stejná vzdálenost) se ukládají ověřené pixely do listu „done“ a vnitřní smyčka cykluje rozdílem listu všech okrajových pixelů a listu ověřených pixelů „done“.

(porovnávají se kvadráty délky) a odmocní se až finální maximální kvadrát délky, z čehož dostaneme maximální délku.



Obr. 15: Maximální délka clusteru

Jakmile známe maximální délku clusteru, pro získání linearity stačí tuto délku vydělit počtem pixelů v clusteru (Size). Teoreticky by se hodnota linearity měla pohybovat na intervalu nula až délka úhlopříčky detektoru v pixelech děleno dvěma.[5]

2.2.5 Analýza velikostí Clusterů

Velikost daného clusteru (Size) se jednoduše rovná počtu pixelů v tomto clusteru, takže vykreslování histogramu velikostí nevyžaduje žádné výpočty. Zkoumání velikosti clusteru je nejjistějším způsobem, jak například poznat záření alfa (viz Obr. 17).

2.2.6 Analýza ToT jednotlivých Pixelů (Time over Threshold)

Původ hodnoty ToT je vysvětlen v kapitole: *1.1 Princip detektoru Timepix* a znázorněn na Obr. 3.

Při stejných nastavených parametrech detektoru znamenají vyšší ToT více energie pro daný pixel. Některé částice (např. záření alfa) mohou mít větší energii, ale po kolizi se tato energie více rozprostře po povrchu (viz Obr. 17).

[5] Pro ideální kruh je linearita $2r / \pi r^2 \ldots 2 / \pi r$, kde s rostoucím poloměrem r se linearita blíží nule. Maximální linearita vychází při největší možné délce (úhlopříčka detektoru v pixelech) mezi dvěma pixely, což je ale chybný stav, jelikož 2 pixely takto daleko od sebe by nebyly v jednom clusteru.

Na základě této analýzy je možné stanovit rozdělení hodnot ToT pro dané měření a v případě jeho opakování umožní upravit nastavení tak, aby bylo dosaženo optimálních parametrů detektoru (viz Obr. 21).

2.2.7 Analýza variačního koeficientu clusterů

Variační koeficient je skvělým nástrojem pro pozorování „homogenity“ clusterů. Pro částice s vysokou penetrací jsou hodnoty ToT v celém clusteru velice podobné, zatímco částice, které mnohem víc interagují s citlivou částí detektoru, tvoří gradienty ToT často od nějaké maximální hodnoty až po hodnoty blízké nule pro pixely vybuzené těsně nad komparační úroveň (viz. Obr. 25).

Variační koeficient vychází z tohoto vzorce:

$$v_x = \frac{s_x}{\bar{x}}$$

kde v_x je variační koeficient daného clusteru, s_x je směrodatná odchylka hodnot ToT clusteru a $\bar{x}$ je aritmetický průměr hodnot ToT clusteru. Nízký variační koeficient znamená, že ToT všech pixelů v Clusteru jsou velmi blízké (viz Obr. 17).

Směrodatná odchylka se spočítá tímto vzorcem:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

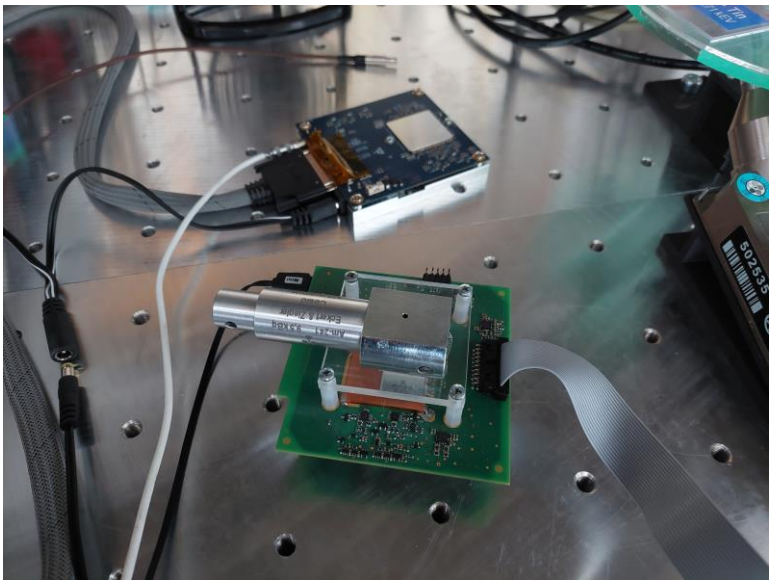
Implementace této rovnice v kódu vypadá takto:

```
double GetDeviation()
{
    if (Pixels.Count > 1)
    {
        double sum =
             Pixels.Select(p => (p.Tot - AverageTot) * (p.Tot - AverageTot)).Sum();
        return Math.Pow(sum / (Pixels.Count - 1), 0.5);
    }
    else
    {
        return 0;
    }
}
```

Pro získání variačního koeficientu poté stačí spočtenou směrodatnou odchylku vydělit aritmetickým průměrem:

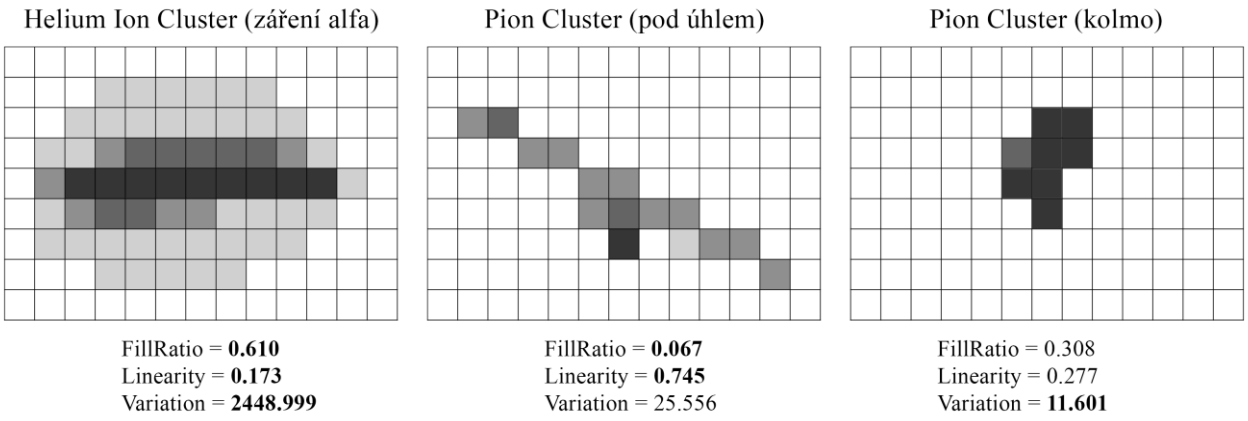
3 Měření

Uspořádání při měření záření alfa za užití zdroje ionizujícího záření Americium-241 s aktivitou 10 kBq (kilobecquerel) vypadalo takto:

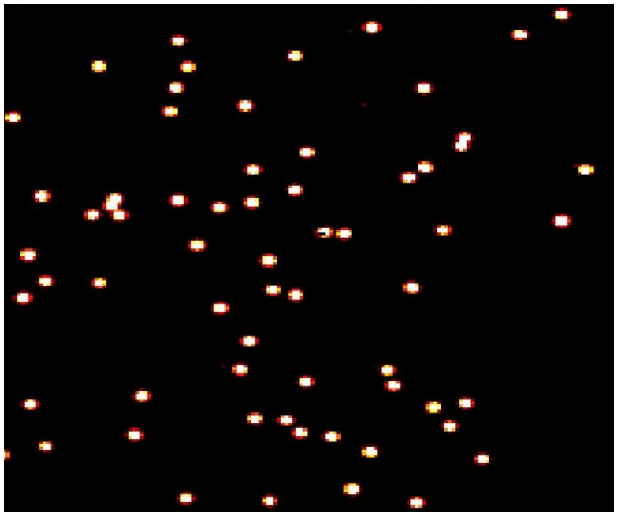


Obr. 16: Zdroj záření alfa Am-241 na detektoru Timepix (zelená deska)

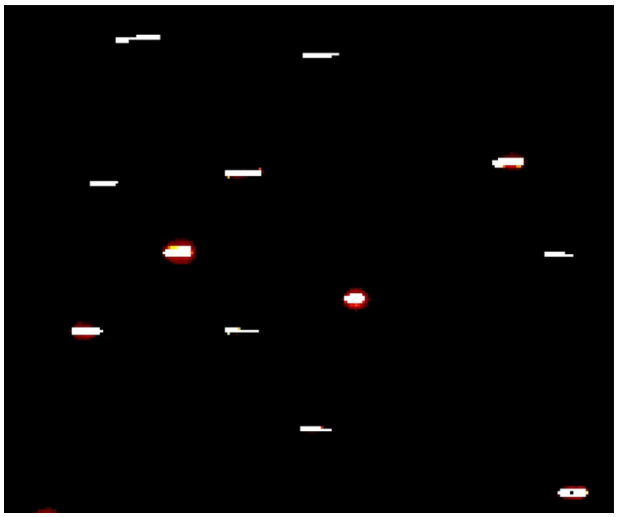
Výsledné snímky lze vidět na Obr. 18.



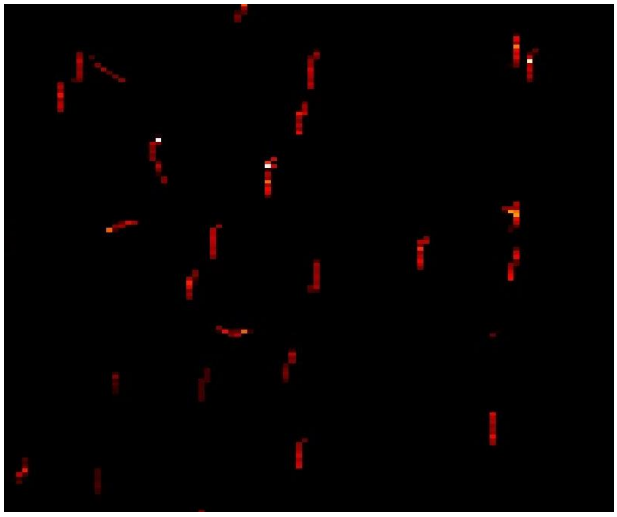
Obr. 17: Porovnání vlastností clusterů (tmavší barva = vyšší ToT)



Obr. 18: Měření záření alfa ze zdroje Am-241



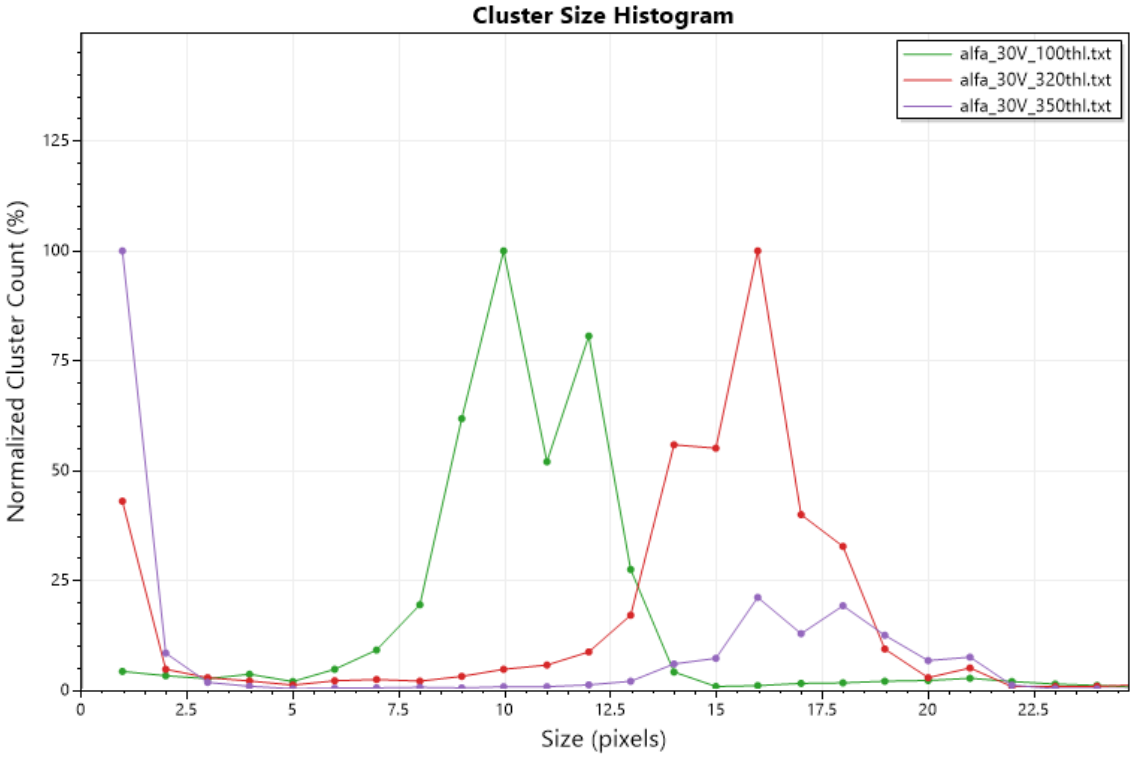
Obr. 19: Měření heliových iontů z cyklotronu UJV v Řeži, úhel dopadu 45°



Obr. 20: Měření pionů ze synchrotronu SPS v CERN, úhel dopadu 45°

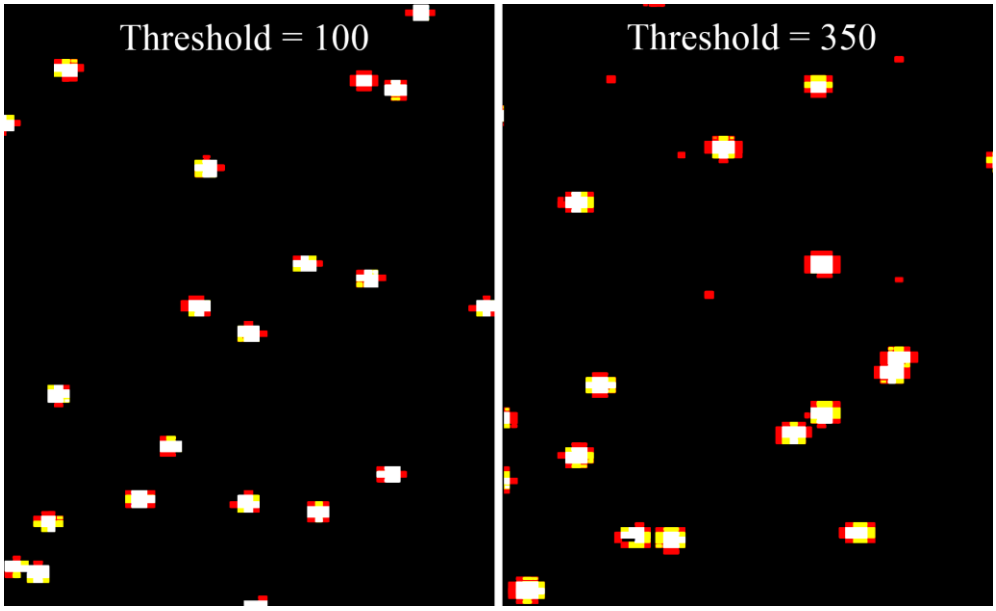
3.1 Vliv změny komparační úrovně (Threshold) na clustery

Zvyšováním Thresholdu (THL) detektoru se zvyšuje jeho citlivost a **snižuje** se komparační úroveň potřebného náboje, nad kterým je pixel označen jako vybuzený. Čas strávený nad touto úrovní je pak hodnotou ToT (Time Over Threshold) pro daný pixel.



Obr. 21: Histogram velikostí pro různé komparační úrovně Threshold (100, 320 a 350)

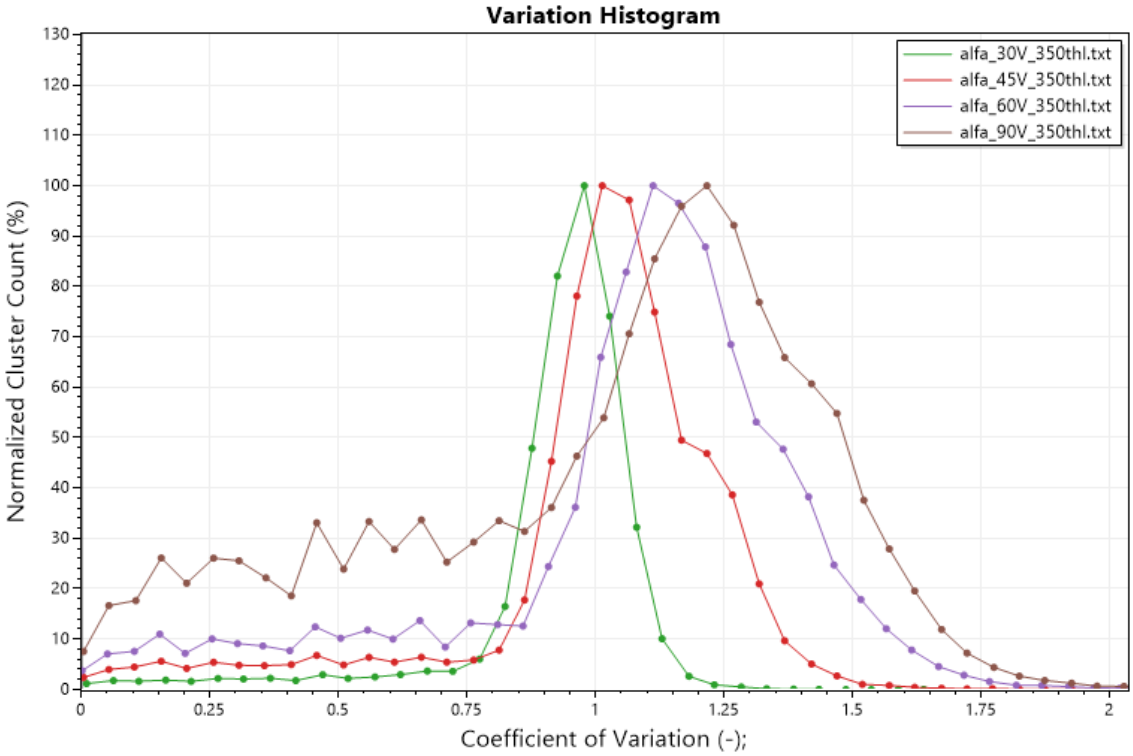
V grafu lze vidět, že zvyšováním Thresholdu (zelená=100, červená=320, fialová=350) se v pořízených snímcích objevuje více šumu (viz Obr. 22), proto se na vodorovné ose zvyšuje relativní četnost velikostí clusteru 1 (nejčastěji jednopixelové gamy, v grafu první hodnota zleva). Pro THL=350 (fialová) se už ve snímcích objevuje více jednopixelových clusterů, než měřených clusterů alfa záření. Zároveň také ale lze pozorovat, že se alfa clustery s rostoucím THL zvětšují, protože se do snímků dostávají okrajové pixely, které by pro nižší THL neměli dostatečný náboj pro překročení komparační úrovně.



Obr. 22: Stejný zdroj záření alfa při různém nastavení THL

3.2 Vliv změny úrovně předpětí (Bias) na clustery

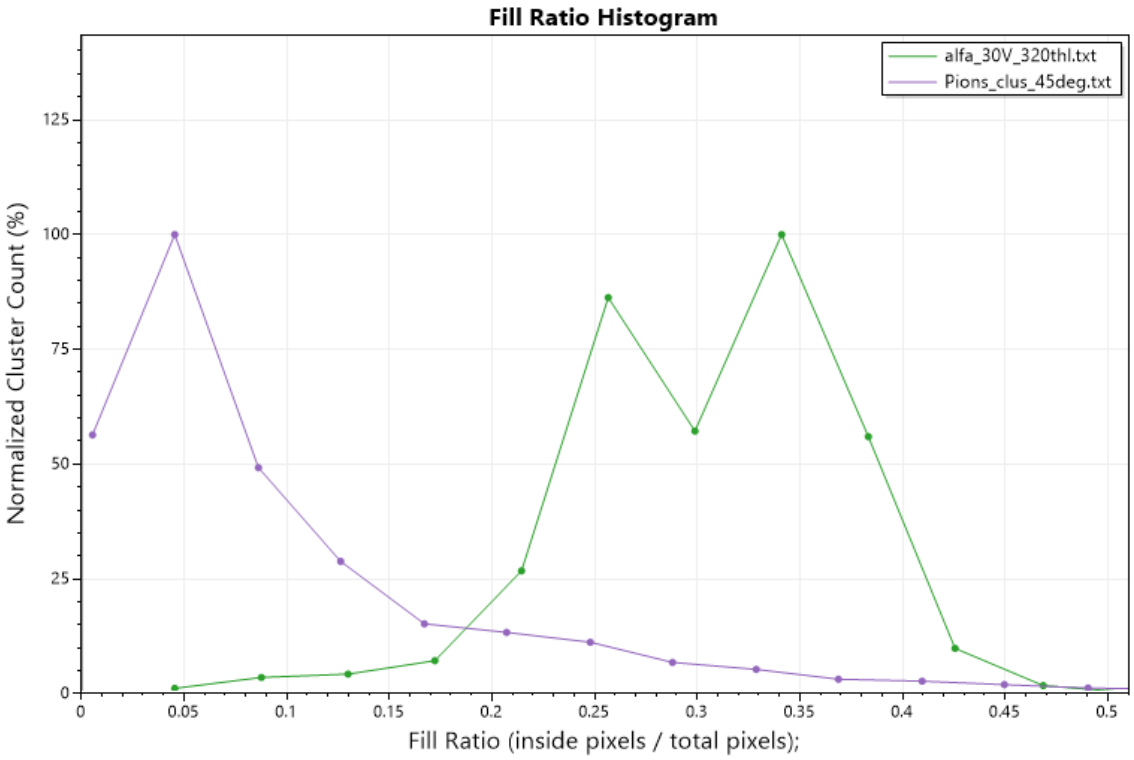
Zvyšováním předpětí se náboj z ploch zasažených ionizujícím zářením rychleji vybíjí a nestihne se tolik rozptýlit po povrchu. To způsobí vyšší detail clusterů namísto rozptýlených skvrn, které všechny vypadají podobně.



Obr. 23: Histogram variace clusterů pro různé úrovně předpětí Bias (30, 45, 60 a 90 Voltů)

V grafu lze vidět zvyšování variace clusterů s rostoucím předpětím.

3.3 Porovnání různých druhů ionizujícího záření

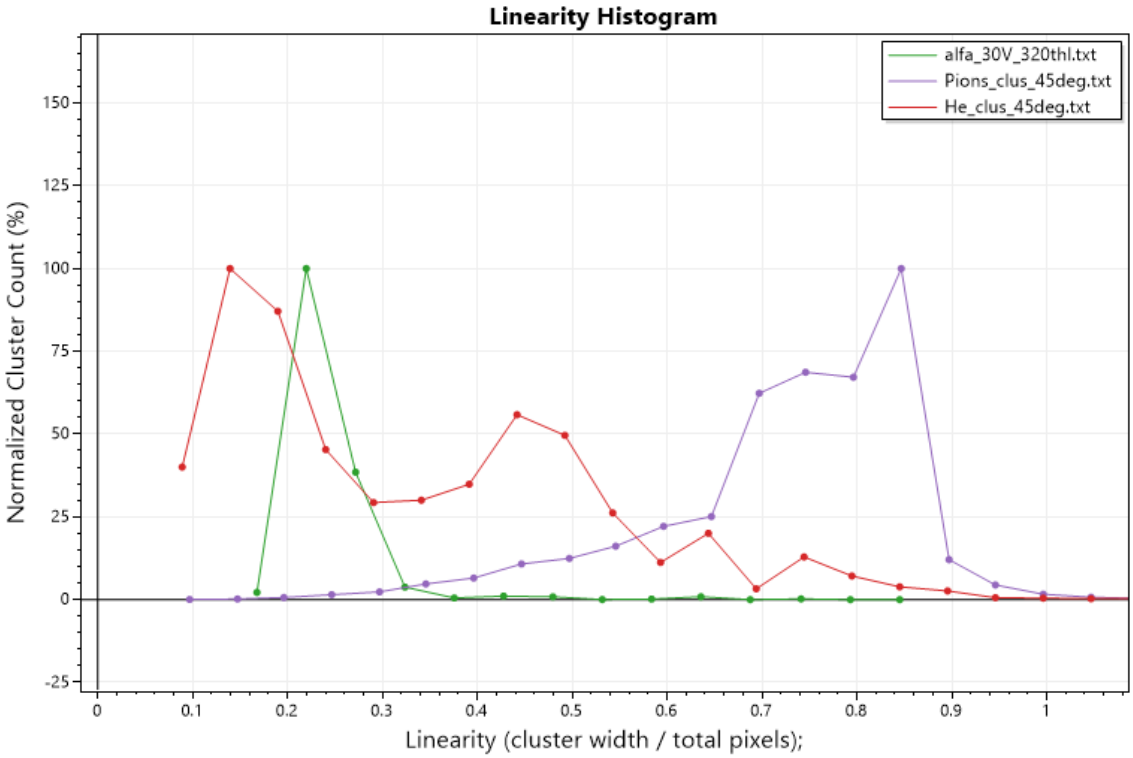


Obr. 24: Histogram FillRatio clusterů alfa (zelená) a pionů (fialová), vzorkování 0,04

V grafu lze pozorovat očekávaný výsledek vyšších hodnot FillRatio pro kulaté clustery záření alfa (viz Obr. 17, Obr. 18), zatímco piony místo kruhů tvoří rovné čáry při nenulových úhlech dopadu (viz Obr. 17, Obr. 20).

Obr. 25: Histogram variace clusterů alfa (zelená), iontů helia (červená) a pionů (fialová)

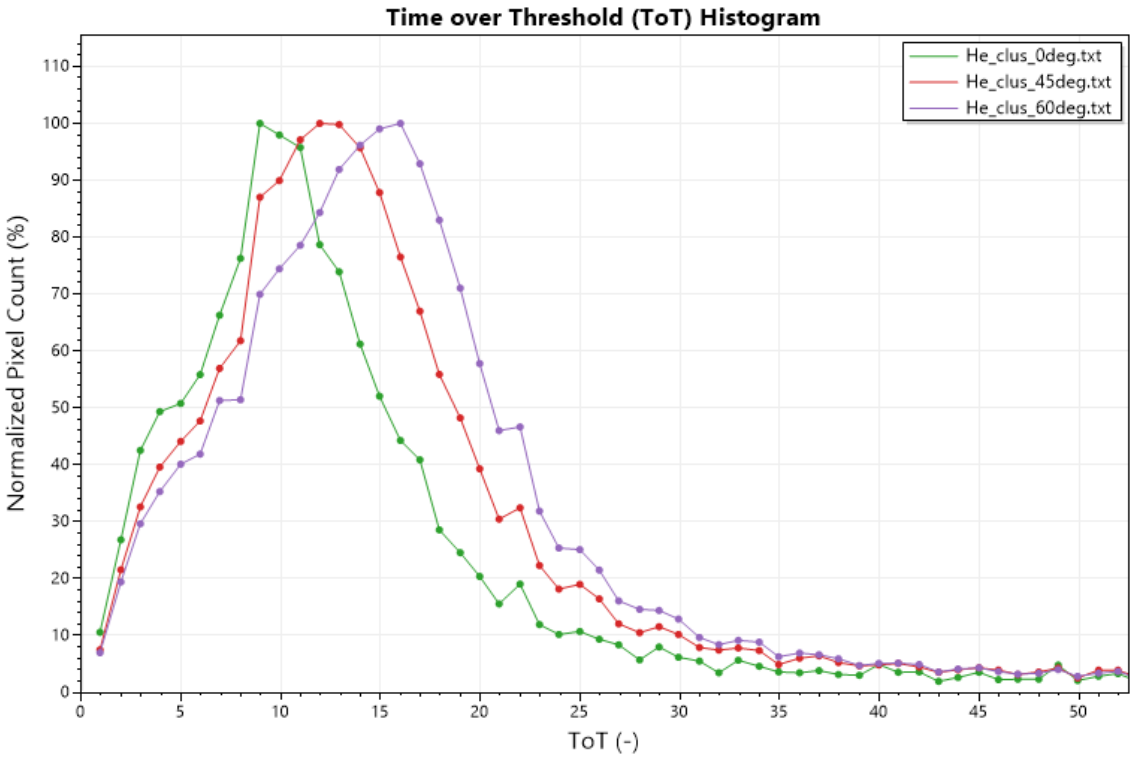
V grafu lze pozorovat, že variace pionů je mnohem nižší, protože piony po celé své trajektorii deponují téměř konstantní energií a s citlivou oblastí detektoru interagují málo, takže každému pixelu předávají podobná kvanta energie (ToT pixelů v clusteru jsou pak skoro stejné). Variace záření alfa je vyšší kvůli rozprostření ToT po větší ploše, ale distribuce je v grafu mnohem užší, protože si jsou všechny clustery velmi podobné. Pro ionty helia je distribuce široká - pravděpodobně kvůli výskytu více různých druhů částic (viz Obr. 19).



Obr. 26: Histogram linearity clusterů alfa (zelená), iontů helia (červená) a pionů (fialová)

Distribuce linearit iontů helia je v grafu opět mnohem širší ze stejných důvodů, jako v předchozím grafu. Rovné čáry zanechané trajektoriemi pionů citlivou oblastí detektoru očekávaně mají řádově vyšší linearitu v porovnání s kulatými bloby záření alfa.

3.4 Porovnání různých úhlů dopadu ionizujícího záření



Obr. 27: Histogram ToT iontů helia při různých úhlech dopadu (zelená=0°, červená=45°, fialová=90°)

Při zvyšování úhlu dopadu částice citlivé oblasti detektoru předávají více energie a na grafu lze pozorovat zvyšování střední hodnoty ToT s rostoucím úhlem dopadu. Tvar distribuce zůstává stejný, jelikož vychází z vlastností záření, které je zde ve všech případech stejné.

Závěr

Cílem práce bylo seznámení se s polovodičovými detektory ionizujícího záření Timepix a navržení softwaru umožňují základní statistické analýzy naměřených dat, které poté byly ověřeny skutečnými experimenty. Pro realizaci softwaru byla zvolena platforma Windows Presentation Foundation (WPF) od firmy Microsoft v programovacím jazyce C#. V softwaru byla částečně separována prezenční logika (front-end) od obchodní logiky (back-end) pro zjednodušení budoucího vývoje a lepší čitelnost kódu. Software bez problémů zvládá relativně rychle zpracovávat soubory dat z měření v řádech stovek megabitů.

Software nabízí snadno pochopitelné vizualizace, které jsou užitečné pro rychlou klasifikaci různých ionizujících částic a porovnání jejich vzájemných energetických úrovní, úhlů dopadu nebo velikostí. Může být také užitečným nástrojem pro nastavení optimálních parametrů detektoru na daný zdroj ionizujícího záření porovnáváním tvarů křivek v histogramech pro různá nastavení detektoru. Software v aktuálním stavu není schopen běžet souběžně s měřením v reálném čase, může pouze zpracovávat data až po skončení experimentu a importování souboru dat z měření.

V práci byly za užití softwaru porovnávány výsledky měření získaných z laboratoře ionizujícího záření přímo na FEL ZČU, cyklotronu UJV v Řeži a synchrotronu SPS v CERN. Všechny vlastnosti clusterů částic ionizujícího záření obsažené v softwaru a algoritmy využité pro jejich výpočet jsou popsány. Měnící se hodnoty těchto vlastností pro různá měření jsou vysvětleny a vizualizovány za pomoci histogramů v softwaru.

Možné budoucí zlepšení zahrnují například přebrání algoritmů a napsání programu funkčního ve více operačních systémech, asynchronní načítání souborů pro využití více než jednoho vlákna procesoru, přepsání importovací logiky pro možnost zobrazování histogramů v reálném čase a přidání možnosti výběru logaritmické osy do histogramů.

Literatura

[1] Microsoft Corporation. *Co je .NET? Úvod a přehled* [online]. Microsoft, 2022 [cit. 2022-05-10]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/core/introduction

[2] Microsoft Corporation. *Přehled windows WPF (WPF .NET)* [online]. Microsoft, 2022 [cit. 2022-05-10]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/desktop/wpf/windows/

[3] PINSKY, Lawrence. *An Update on the Timepix2* [online]. Physics Department University of Houston 2018 [cit. 2022-05-10]. Dostupné z: https://wrmiss.org/workshops/twentythird/Pinsky.pdf

[4] Microsoft Corporation. *Přehled XAML (WPF .NET)* [online]. Microsoft, 2022 [cit. 2022-05-10]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/desktop/wpf/xaml/

[5] HARDEN, Scott. *ScottPlot.WPF* [online]. NuGet 2022 [cit. 2022-05-10]. Dostupné z: https://www.nuget.org/packages/ScottPlot.WPF

[6] Microsoft Corporation. *Model-ViewModel* [online]. Microsoft, 2022 [cit. 2022-05-10]. Dostupné z: https://docs.microsoft.com/cs-cz/xamarin/xamarin-forms/enterprise-application-patterns/mvvm

[7] GOSSMAN, John. *Advantages and disadvantages of M-V-VM* [online]. Microsoft Team Blog 2006 [cit. 2022-05-10]. Dostupné z: https://docs.microsoft.com/en-gb/archive/blogs/johngossman/advantages-and-disadvantages-of-m-v-vm

[8] OSMANI, Addy. *Understanding MVVM – A guide for JavaScript Developers* [online]. Addy Osmani's Blog 2012 [cit. 2022-05-10]. Dostupné z: https://addyosmani.com/blog/understanding-mvvm-a-guide-for-javascript-developers/

[9] POSPÍŠIL, Stanislav. *TimePix hybrid pixels detectors for particle identification and dosimetry* [online]. Institute of Experimental and Applied Physics, Czech Technical University in Prague 2020 [cit. 2022-05-12]. Dostupné z: https://indico.cern.ch/event/954194/contributions/4009245/attachments/2156334/3637201/Stanislav%20Pospisil%20-%20SP-Dakar-Timepix-lecture-fin.pdf

[10] CERN. *Timepix3 Knowledge Transfer* [online]. CERN 2022 [cit. 2022-05-12]. Dostupné z: https://kt.cern/technologies/timepix3

[11] LIPP, John. *STFC Bump Bonding* [online]. UK Bump Bonding Workshop, Daresbury 2012 [cit. 2022-05-12]. Dostupné z:

https://indico.cern.ch/event/191355/contributions/346836/attachments/272967/381962/2012.05.15_DL_bump_bonding_workshop.pdf

[12] URBAN, Ondřej, VAVROCH, Ondřej, POLÁČEK, Libor, GEORGIEV, Vjačeslav, BURIAN, Petr, TURJANICA, Pavel, FIALA, Pavel, BROULÍM, Pavel, BERGMANN, Benedikt. *Hodoscope with Timepix detectors for PilsenCube2 cubesat.* ISSN 0168-9002. Dostupné z: https://www.semanticscholar.org/paper/Hodoscope-with-Timepix-detectors-for-PilsenCube2-Urban-Vavroch/7ca153ea0df0ad9c14564da599eb7bde8bec1400