μ-PIC と光電子増倍管を用いた
放射線の三次元測定

石神直大　　澤野達哉　　義川達人

2009 年 3 月 23 日

摘録

放射線を扱う多くの実験において、線源の位置特定や放射線の到来方向を知ることは重要な要素である。これらは放射線の三次元位置情報を得ることで容易に達成される。従来の TPC で得られる三次元位置情報は、検出面に対する高さ方向の位置情報について、相対的なものしか得られなかった。これは、放射線の軌跡の形を再現することはできるが、その軌跡がどの高さに存在するかは分からないということに相当する。

我々は、二次元位置情報を得られるガス検出器 μ-PIC と光電子増倍管を併用して、高さ方向に関しても相対的でない実際の位置情報を得ることに成功した。また、これを用いて、線源の三次元位置特定及び放射線の 3D-tracking にも成功した。

はじめに

本文に入る前に、本レポートの構成を簡単に説明する。

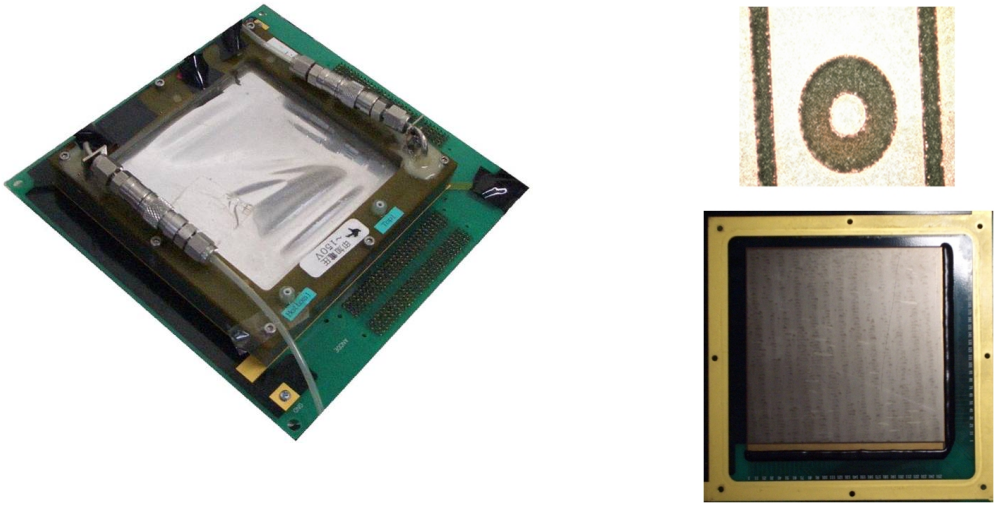
本レポートは課題研究の成果をまとめたものである。従って、摘録で書いた実験以前に行ったいくつかの予備実験に関しても、その詳細を書いてある。本レポートの大まかな流れは次のようになる。

- μ-PIC の概要
- いくつかの予備実験（ほぼ時系列順）
- 本実験
- まとめ

目次

1　μ-PIC

今回我々が使用した μ-PIC は京都大学宇宙線研究室で開発されたガス検出器である。高い位置分解能、放電に対して安定という特長を持つ。また、二次元位置情報を取得することができ、それにより二次元画像を構成することができる。その構造は図 1 のようになっている。



(a) 全体像　　(b) 検出面とピクセル

図 1　μ-PIC の写真. (a) の銀色の部分がドリフト面になる。ドリフト面の下には１０ cm 四方の検出面があり、(b) 下図の中央部分にあたる。検出面には (b) 上図のピクセルが 256×256 だけ敷き詰められており、各ピクセルは比例計数管を輪切りにしたような構造になっている。中央の円が Anode（陽極）で直径は 50μm である。

μ-PIC 内に放射線が入射すると放射線はガス分子を電離し、放射線の軌跡に沿って電子雲が生成される（放射線が γ 線や X 線の場合は光電吸収や Compton 効果によって生成された光電子の軌跡に沿う)。生成された電子雲は μ-PIC 内の電場を感じで検出面へドリフトし、検出面付近で雪崩増幅した後、信号として読み取られる（図 2)。

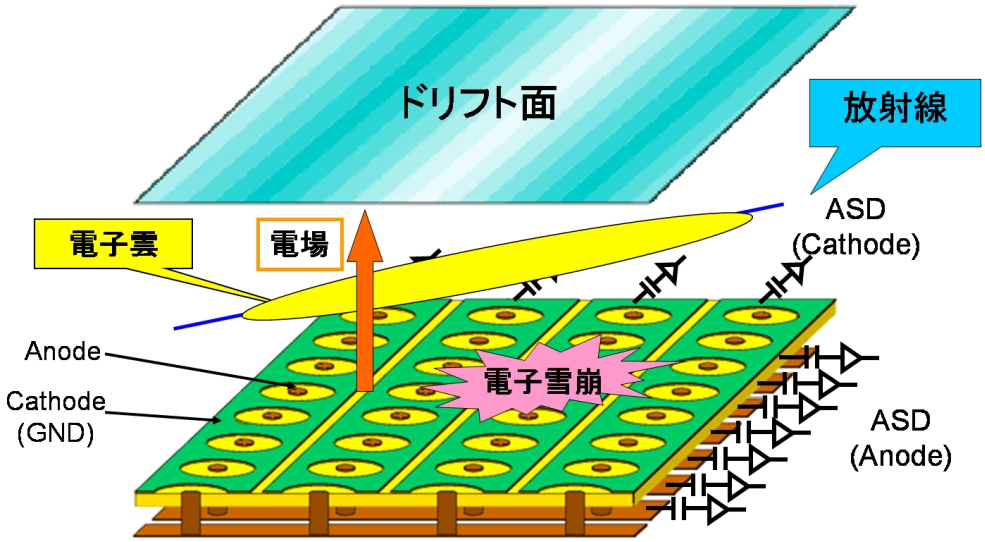


図 2　μ-PIC の動作原理. ドリフト面と検出面の間にはガスが封入されており、ここでは荷電粒子が入射した場合を想定している。

信号として得られるものは、各チャンネルを適当に sum したアナログ信号と、各チャンネルのデジタル信号で、Anode、Cathode という二方向の読み出しを行うことで、二次元位置情報が得られる。

2 ガス検出器としての μ-PIC

本実験へ進むための予備実験として、まずは単純にガス検出器としての μ-PIC の特性を調べた（以下、これを予備実験 I と呼ぶ）。具体的には、エネルギースペクトルを取得してエネルギー較正を行い、分解能や μ-PIC の増幅率を求めた。

なお、予備実験 I で用いた μ-PIC20 号機は放電（Anode と Cathode の導通）を起こしており、その影響を除去するために様々な労力を費やした。

2.1 実験環境

まず、予備実験 I を行った際の実験環境を示しておく。

μ-PIC は 20 号機を用いた。この μ-PIC はひどく放電を起こしており、当初は信号が見えるレベルまで Anode の電圧を上げることができなかった。そこで、放電を起こしている箇所、つまり、電流が漏れている箇所を調べて、その部分に高電圧がかからないようにした。これにより、放電による有効電圧の降下を抑えることができたが、一方で、有効な検出領域を狭めることにもなった。信号が見えるようになった頃には、検出領域は本来の 1/8 になってしまった。

封入するガスは Ar90%-$C_2H_6$10% を用い、ガスフロー型で実験を行った。ガスの流量は 165cc/min とした。

ガスパッケージ（ガス領域の検出面に対する高さ方向の長さ）は 5mm で、GEM 間に電圧差を設けず、GEM をドリフト面に代用した。ドリフト面に-550V、検出面（Cathode）に 0 V、Anode に 550V の電圧をそれぞれかけた。

以上の環境で予備実験 I を行った。用いた線源は γ 線を放射する ^{55}Fe と ^{109}Cd である。

2.2 実験原理

予備実験 I の目的を達成するためには、μ-PIC からの信号（波形）を取得する必要がある。今回、波形の取得には FADC（Flash ADC）を用いた。今回用いた Flash ADC は 10ns を 1 クロックとして、8192 クロック分のデータを保持することができ、1 データは 1/256V 刻みで 0〜1V までを示す。

予備実験 I で波形を取得する際に構成した機構は図 3 のようになる。予備実験 I では、放電対策のために有効検出領域を狭めたこともあって、Anode の 161ch〜192ch の 32 ストリップからのみ信号を取得した（アナログ信号）。また、図中の PreAmp は ASD（Amplifier Shaper Discriminator）の Amplifier の機能のみを用いたことを示す。

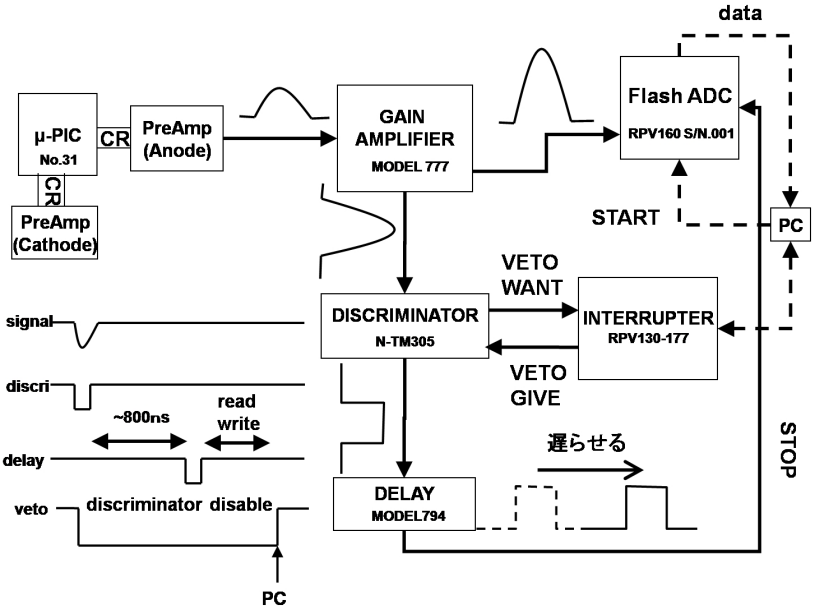


図 3　信号取得のための配線図

μ-PIC からの信号は CR 基盤、PreAmp を通って整形増幅された後、Gain Amplifier で再び増幅され、FADC で読み込まれる。こうして信号は取得されるわけだが、我々が欲しいデータは放射線による信号、つまり波形部分だけであり、その付近を効率よく取得する必要が出てくる。そのために、いくつかのモジュールを追加した。

まず、波形部分付近を取得するという目的から、Discriminator と Delay を追加した。Discriminator により、ある閾値よりも大きい信号、つまり放射線からの信号と思われる部分が来た際のみ、Delay にパルス信号が出力されることになる。そして、Delay で 800ns 程度遅らせた信号を FADC の stop 信号として用いれば、放射線からの信号と思われる部分の直後に FADC はデータの読み込みを止めることになる。ここから適当なだけさかのぼって（予備実験 I では 1μs）データを取得すれば、波形部分付近のデータを取得することができる。

stop 信号が入力されると、FADC はデータの取得及びファイルへの書き込みを行うわけだが、この際に新たな stop 信号は入力されると FADC は誤作動を起こしかねない。そこで更に Interrupter を追加した。Discriminator は Delay に加えて、Interrupter にもパルス信号を出力する。この信号を受けた Interrupter はパソコンへと信号を出し、パソコンは Interrupter を介して、Discriminator に VETO 信号を出力する。Discriminator は VETO 信号を受けると、不感状態となるため、FADC に stop 信号が何度も入ることが防がれることになる（図 3 左下）。

こうして誤作動を起こすことなく、波形部分付近の信号が FADC によって取得され、パソコンへ出力される。パソコンはデータを得ると Interrupter を介して今度は、VETO 解除の信号を Discriminator に出力し、これにより、Discriminator は不感状態から回復する。また、同時に FADC にも start 信号を出力する。ここで、μ-PIC からの信号を再び受け付ける状態に戻ることに

なる。

以上のようにして得られた波形部分の信号の一例が図 4 である。エネルギースペクトルを描くには、エネルギーに相当する情報が必要となるが、それは図 4 における波形の Area（面積）になる。常に 0 の部分から波形があったわけではないので、得られたデータの後ろ 200ns 間のノイズ部分を平均して、これをベースライン（基準値）とした。そして、ベースラインを超えた部分の波形の面積を計算して、これを Area とした（なお、予備実験 I 以降では、ベースライン以下の部分の面積を負の値にして加えたものを Area と呼んでいる）。

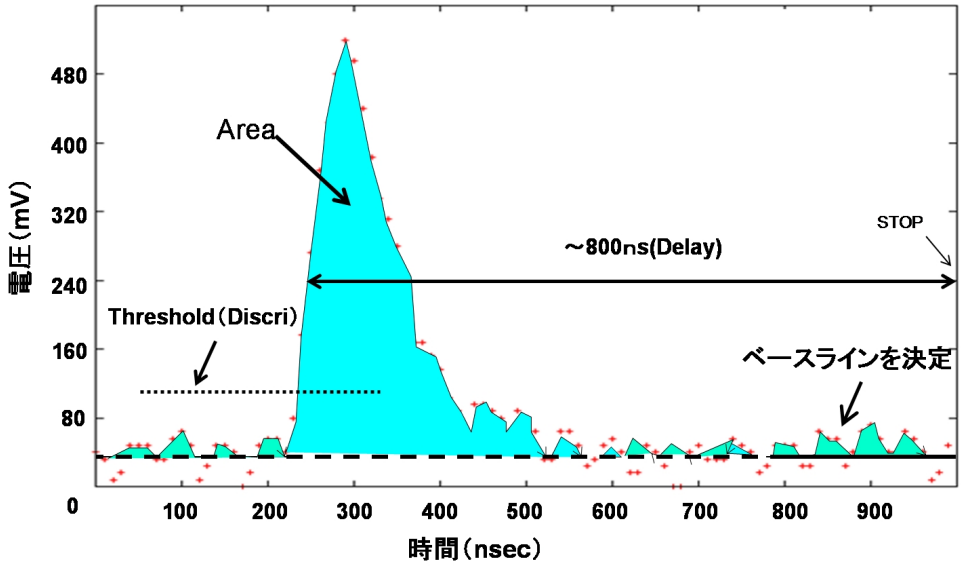


図 4　取得した信号の一例：縦軸は FADC から出力されるデータを電圧値に換算したもの。ベースラインを得られたデータのノイズ部分から決定し、エネルギースペクトルを描くのに必要な Area を求める。

2.3　結果

2.2 節で得られた Area をヒストグラムで描き、エネルギースペクトルとしたものが図 5 である。^{55}Fe、^{109}Cd のピークだけでなく、^{55}Fe のスペクトルには Ar のエスケープピーク、^{109}Cd のスペクトルには μ-PIC の基盤に用いられている銅の特性 X 線も見えている。また、各ピークをガウシアンでフィッティングし、エネルギー較正も行った。

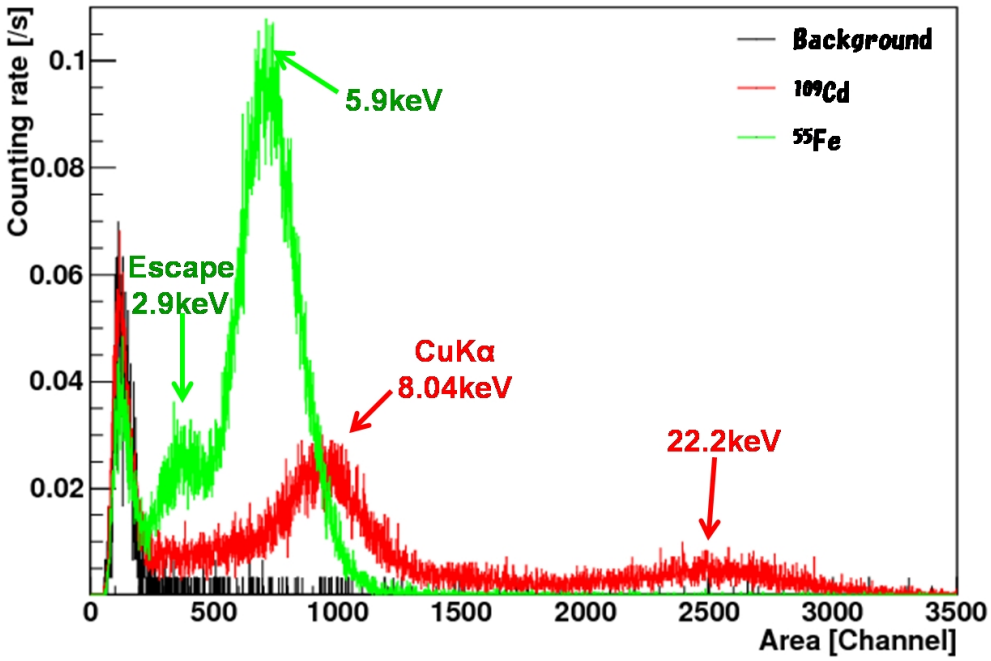


図 5　エネルギースペクトル：最もエネルギーが低い左端のピークは pedestal である。

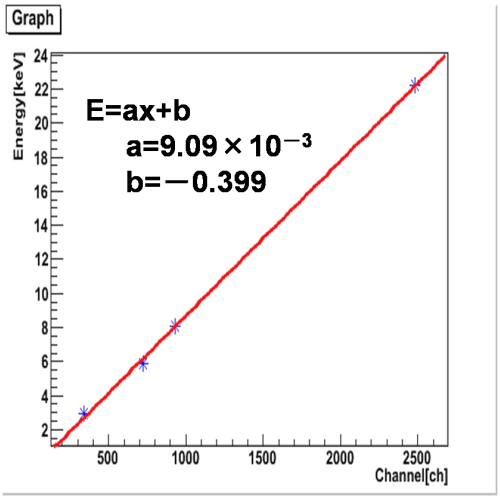


図 6　エネルギー較正

エネルギー較正がきちんと行えたので、各ピークのエネルギー分解能（FWHM）を求めたところ、

2.96keV…36.0 %

5.90keV…31.9 %

8.04keV…36.5 %

22.2keV…29.7 %

となった。エネルギースペクトルでピークを確認し、エネルギー較正を行ったところまでは順調であったが、エネルギー分解能は過去に報告されているものに比べ、悪い値となった。

エネルギー分解能の悪化については、以下のようなことが考えられる。

- 放電対策を講じたものの、完全に放電を排除することはできなかった。そのため、電流漏れによる電圧降下から増幅率が下がり、分解能が悪化した。
- 放電は各ピクセルで発生しているが、その程度（電流の漏れ具合、電圧の降下具合）はピクセルごとに異なる。そのため、増幅率がピクセル単位でばらつき、分解能が悪化した。
- 放電対策のために有効検出領域が狭まってしまい、比較的高いエネルギーの γ 線が、その領域内で全エネルギーを損失できなかった場合があったと考えられる。この結果、エネルギーの高い部分でエネルギー分解能が特に悪化した。

ただ、放電が起きていたことを考慮すればこのエネルギー分解能はリーズナブルな値であり、μ-PI Cはガス検出器として正常に動作していることが確認できた。

μ-PIC がガス検出器として正常に動作していることを確認できたので、次いで、ガス検出器の特性を示す増幅率を求めた。μ-PIC の増幅率とは検出面付近で起こす雪崩増幅によるガス増幅率を指す。

^{55}Fe の場合で、増幅率の求め方を示す。まず ^{55}Fe の放射線が μ-PIC に入射し、そのエネルギー 5.9keV 全てがアルゴンガスの電離に使われたとする。アルゴンガスの W 値は 26eV であるから、一次電子は 5900/26〜220 個生成する。この一次電子が μ-PIC により増幅され、更に PreAmp と GainAmp により増幅され、FADC で読み込まれた後、パソコンにデータとして出力される。パソコンでは電荷量に相当する情報として Area が計算できる。Area の平均が 700ch（1ch は 1 クロックにおける 1/256 刻みの 1 メモリ）程度、PreAmp の増幅率 700 倍（既知）、GainAmp の増幅率 16 倍（設定）を用いれば、増幅率は以下のように計算できる。

パソコンで得られる Area の 1ch は (1/256)V × 10ns に相当するから

$$\begin{aligned}\delta Q &= \delta t \times \frac{V}{R} \\ &= 10 \times 10^{-9}\text{sec} \times \frac{(1/256)\text{V}}{50\Omega} \\ &= 7.8 \times 10^{-13}\text{C/ch}\end{aligned}$$

従って、求める増幅率は

$$7.8 \times 10^{-13}\text{C/ch} \times 700\text{ch} = 0.55\text{nC}$$
$$\therefore 0.55\text{nC} \div 16 \div 700 \div (220 \times 1.6 \times 10^{-19}\text{C}) \approx 1400$$

よって、μ-PIC20 号機の増幅率は 1400 倍程度と求められた（Ar90%-$C_2H_6$10%、ドリフト-550V、検出面 0V、Anode550V）。これは過去の報告と比べると、比較的低いものだが、放電の影響を考慮すれば、リーズナブルなものであると言える。

2.4 まとめ

以上から、得られた結果は過去のものと比べてあまり良いものではなかったが、μ-PIC 本体の放電という事情を考慮すれば、リーズナブルなものであることが分かった。これにより、μ-PIC をガス検出器として扱えること、また、本章の方法でエネルギースペクトルや波形を取得することができることが確認された。

3　2D-imaging ver. 1

前章の予備実験 I により、μ-PIC のガス検出器としての特性を確認できたので、次に μ-PIC の特長でもある二次元画像の描写を行った（以下、予備実験 II と呼ぶ)。なお、予備実験 I の際に狭まった有効検出領域は、色々試みたものの、回復させるに至らなかった。

3.1　実験環境

まず、予備実験 II を行った際の実験環境を示しておく。

μ-PIC は予備実験 I で用いた μ-PIC20 号機を引き続き用いた。放電対策のために有効検出領域が 1/8 に狭まっている。

ガス、ガスパッケージ、各部分の電圧については、予備実験 I と同じ。線源は ^{55}Fe を用いた。

3.2　実験原理

μ-PIC は Anode、Cathode 二方向の信号により、二次元位置情報を出力することができるが、それを元に二次元画像を構成するには、Encoder と Memory Board を用いる必要がある。予備実験 II では予備実験 I と違い、適当に sum したアナログ信号ではなく、各チャンネルのデジタル信号を用いる。このデジタル信号は ASD の Discriminator の機能を用いて生成される。

μ-PIC からの Anode、Cathode それぞれのデジタル信号は Encoder に入力されるが、そのうち Anode と Cathode からの信号が同期したものだけ、Encoder は Memory Board を介し、パソコンへ出力する。なおこの際、Encoder は Encoder 内部で回っているクロック（100MHz）を加えた、X（Anode)、Y（Cathode)、T（clock）の 3 つの情報をパソコンに出力する (図 7)。

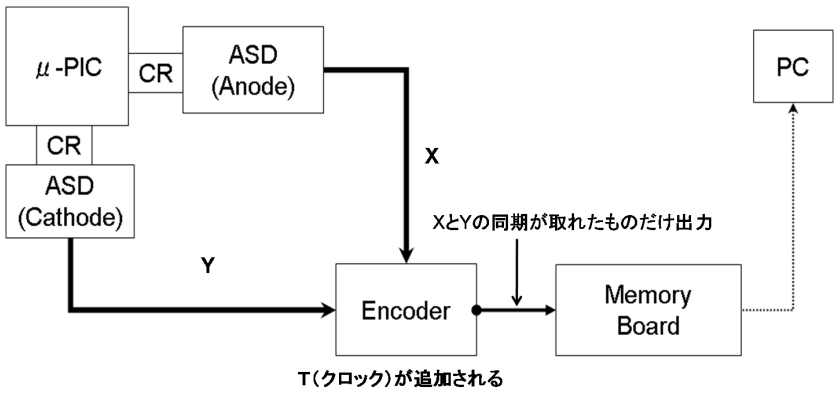


図 7　2D-imaging 取得のための配線図

予備実験 II では、μ-PIC のドリフト面上部に鍵を置き（図 8)、更にその上部 10cm の位置に線

源を置いた。こうすると放射線は鍵の部分でのみ遮られ、鍵の形が “ 白抜き ” になった二次元画像が得られるはずである。

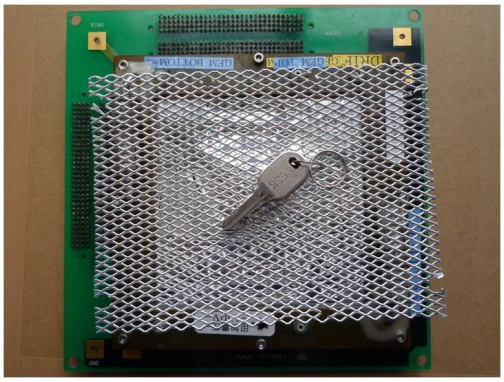


図 8　鍵の設置：この上部から放射線を照射した。

3.3　結果

鍵の 2D-imaging について、予備実験 II で得られた結果が図 3-3 である。図の詳細な説明は次章に譲るが、白い部分は放射線を検出しなかった部分である。鍵の大体の形に沿って放射線が遮られたことは窺えるが、μ-PIC の高い位置分解能を活かした詳細な二次元画像が得られたとは言い難い。なお、図 3-3 は 2000 個のデータから構成される二次元画像であるが、データ数を増やしても大きな変化は見られなかった。

本章冒頭で書いたような有効検出領域の回復や、ノイズの除去を行ったものの、図 9 以上のきれいな二次元画像を得ることはできなかった。

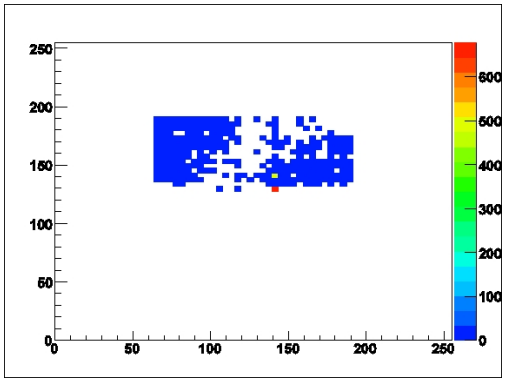


図 9　鍵の 2D-imaging の結果：横軸が Cathode、縦軸が Anode に対応している。Anode における放電対策に加え、ノイズののっていた両端の Cathode も検出しないようにした。白い部分が検出できなくなっている領域を表している。

4　2D-imaging ver.2

前章までの μ-PIC(20 号機) では放電の影響によって詳細な二次元画像を得ることができず、また検出面の有効面積が狭いという問題も解決することができなかった。

そこで新しい μ-PIC(31 号機) の設置された μTPC(Time Projection Chamber) を用いた、設備を導入した。ここではまず前章で出来なかった 2D-imaging を取得することを目的とする。

4.1　μTPC

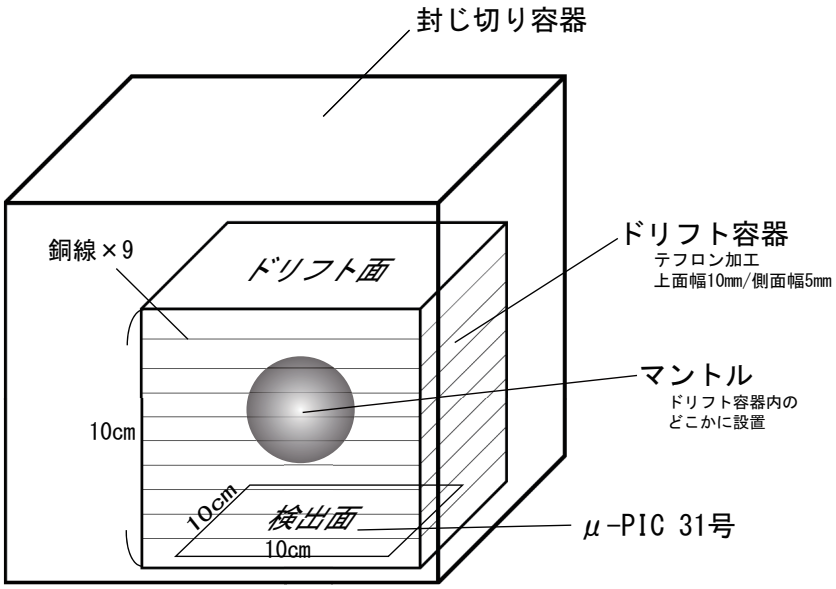


図 10　μTPC の概念図

これ以降は 2D-imaging だけではなく 3D-tracking も行うことを考えているので、ガス package 部分は深さ方向に厚いことが要求される。我々の使用した μTPC の Package の深さ方向の厚さは 10cm である。

なお、前章までの設備ではガスフロー型であったが、ここではガスを容器内に封じ切っている。使用ガスは前章と同じく、Ar90%-$C_2H_6$10% の混合ガスである。どこにあるかは知らされていないが、装置内のどこかにマントルが設置されており、このマントルからの放射線を計測する。

TPC の仕様

- 側面、上板　SUS304 3mm　　（内面電界研磨）
- 底面　アルミニウム 16.7mm(検出面部 5mm)
- 容器外形　280 × 280 × 170mm
- 使用 μPIC31 号機　S/N 080728-2
- 光トリガー用シンチレータ R2238-01 × 2　SC-A(MG0146)　SC-B (MG0146)
- ドリフト電場　10M オーム× 10　/　10cm
- ドリフト電場　0.5mm Φ　銅線
- ドリフト側面　10mm テフロン
- ドリフト上面　5mm テフロン

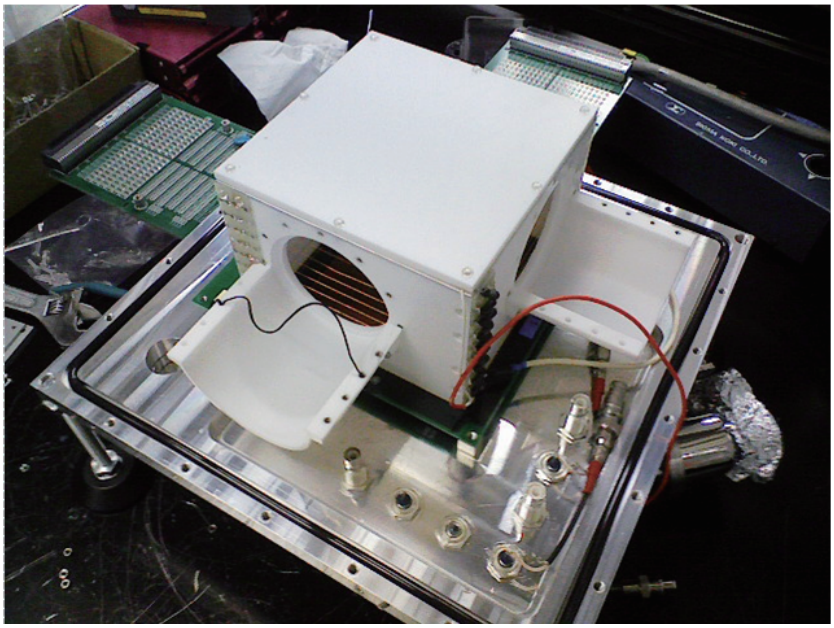


図 11　μTPC

使用ガス

Ar90%-$C_2H_6$10%

4.2 マントル

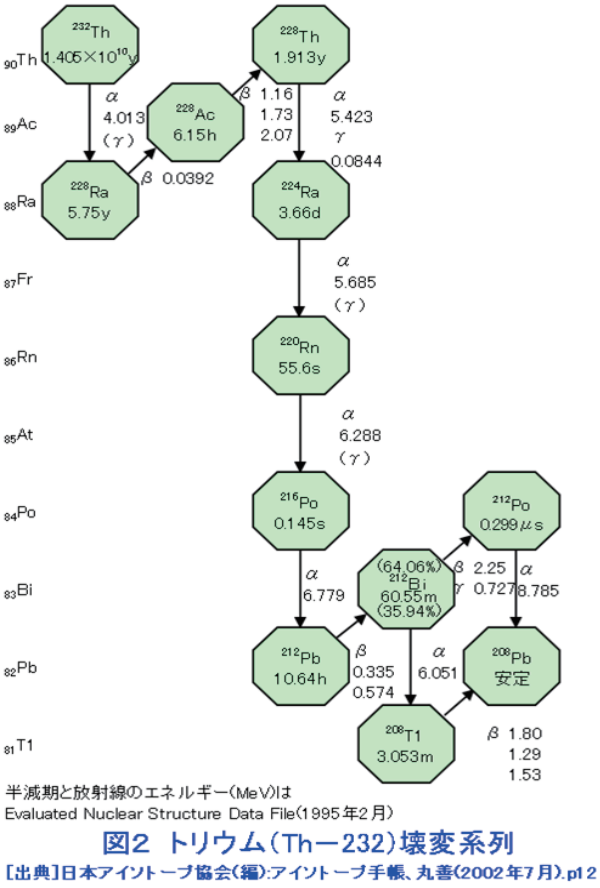


図 12　トリウム壊変系列

図 13　マントル

これ以降の実験では放射線源としてマントルを用いた。マントル中には微量の ^{232}Th が含まれており、上図で示したようにα線などを出しながら壊変していく。

まずはマントルからの放射線を使って 2D-imaging を行う。これにより装置が正しく動作していることを確認するとともに、マントルがドリフト容器内のどの位置に設置されているかを同定する。

4.3 実験方法とセッティング

装置の配線と実験方法は前章と同様である。HV は Drift 側に-3215V、Anode に 425V かけた。Encoder の Threshold は、Anode が-50.8mV、Cathode が 55.9mV である。

この設定の下、Memory Board から PC に出力されるデジタル信号を 10000 発分取得し、その位置情報を色グラフにして 2D-imaging を作成した。

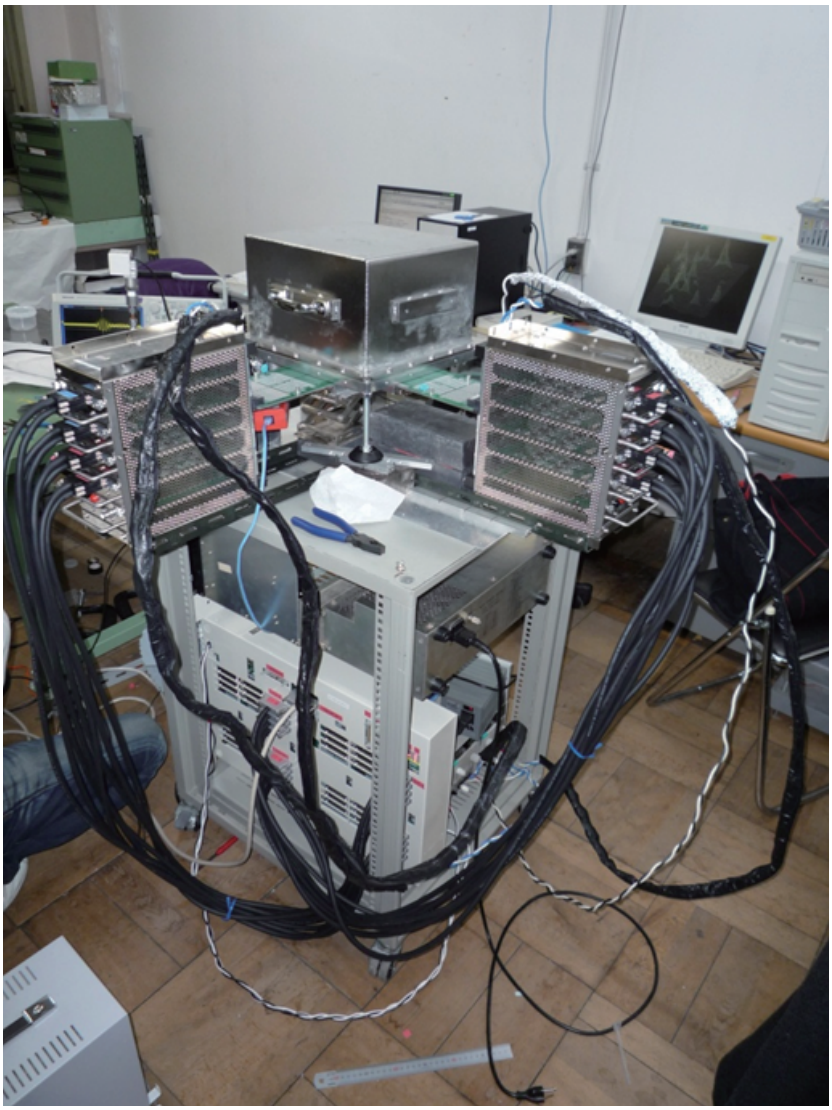


図 14　セッティング

4.4　2D-imaging の結果

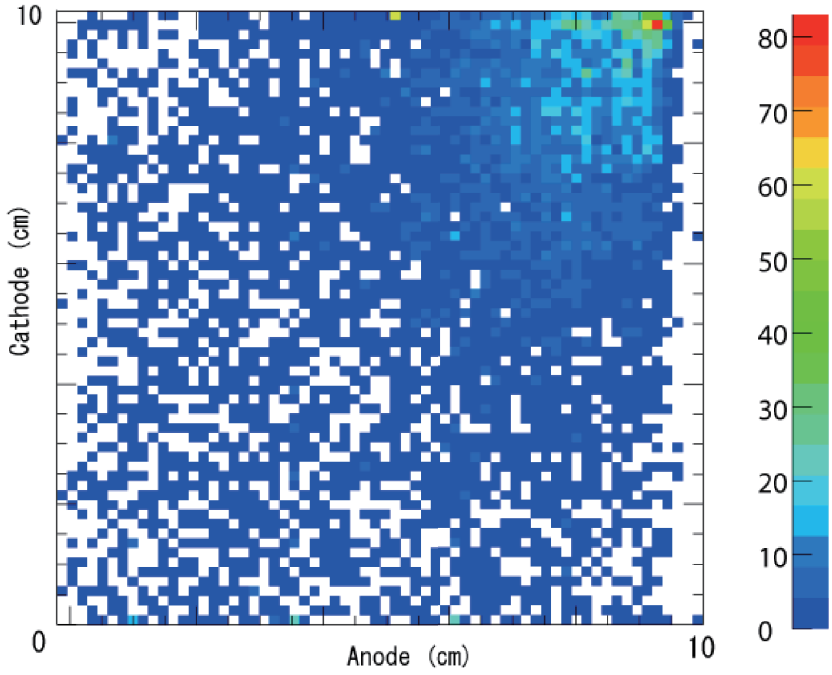


図 15　2D-imaging

計測の結果、上図のような 2D-imaging が得られた。各ビットごとの位置データの数を色で示しており、右側のカラーバーは色と個数の対応を示している。上のグラフのビット数は 64 × 64 個であり、μ-PIC での 16 ピクセル分を 1 ビットにまとめている。

位置データの最頻値は、グラフの左から 9.2cm、下から 9.8cm のところにある。位置データは放射線の飛跡位置に対応しており、線源付近に位置データが集まるのは明らかである。この結果により、マントルはドリフト容器の上から見て隅に設置されていると結論付けられる。

以上より実験設備は正しく動作していることが確認されたので、次に放射線の 3D-tracking に移行する。

5　3D-tracking

前章で放射線の 2D-imaging に成功したので、次は二次元の位置情報に深さ情報を加えて、放射線の三次元での飛跡を得ることに挑戦した。

5.1　**実験原理**

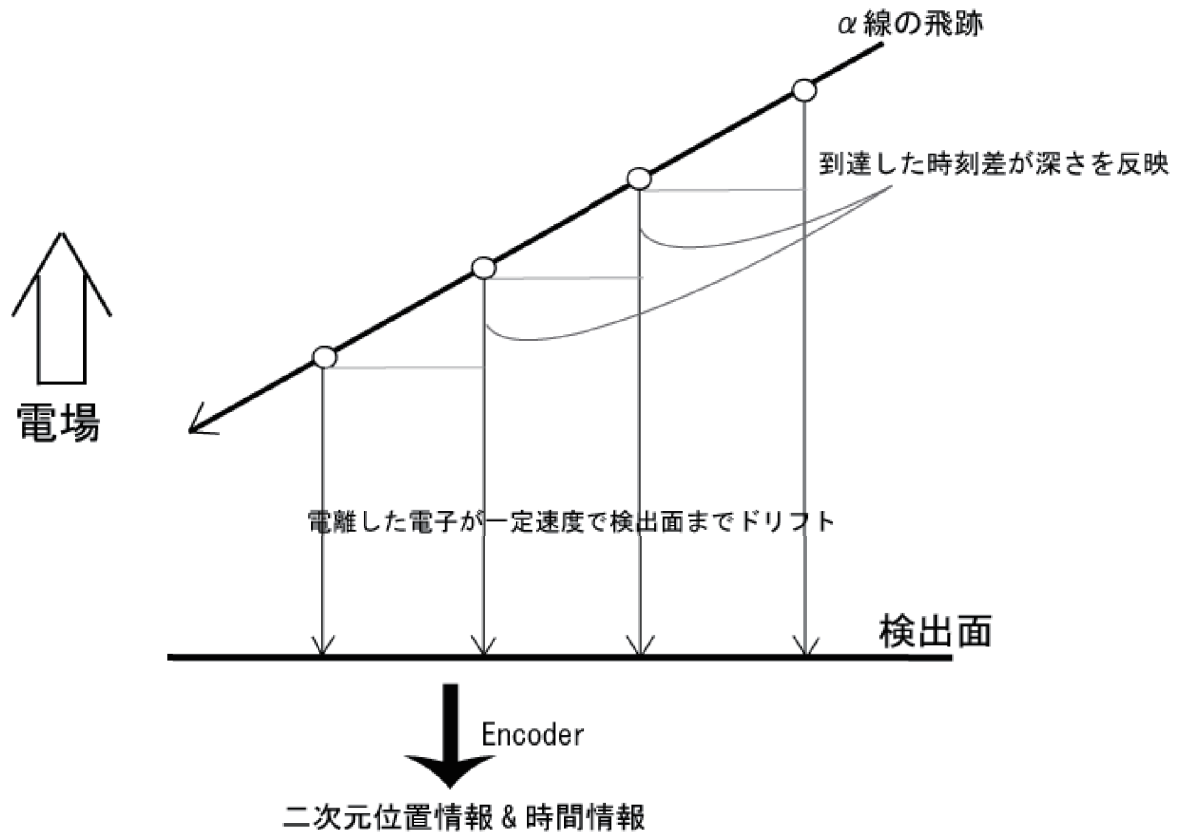


図 16　3D-tracking の原理図

2D-imaging の時は Encoder から出力される情報のうちの位置情報のみを使用していたが、Encoder からは位置情報に加えて時間情報も出力されている。検出面に到達した時間が深さを反映している。

Encoder は内部に時計の機能を持っており、100MHz で ASD からの入力信号を処理している。すなわち 1clock=10ns を単位として、位置情報を出力した際の clock を位置情報と同時に出力している。この clock を電離電子が検出面に到達した時刻とみなし、深さを議論する。

まずα線が入射した際に次々とガスを電離し電子を出すが、放射線の飛ぶ速さは電子のドリフト速度に比べて十分速く、電子は同時に作られるとみなしてよい。次に電子が電場によって検出面へ

とドリフトするが、電場の強さは一様なのでそれぞれの電子の動く速さは同じである。従って、時間情報にドリフト速度を掛け合わせることで深さを逆算することができるのである。

ただし Encoder の時計は放射線の入射するタイミングとは関係なく回しているだけであるので、放射線の入射した時間は分からない。従って意味があるのはそれぞれの位置情報間の時間差であり、分かるのは飛跡の三次元的な形までであって飛跡の絶対的な深さまでは分からない。

5.2 ドリフト速度

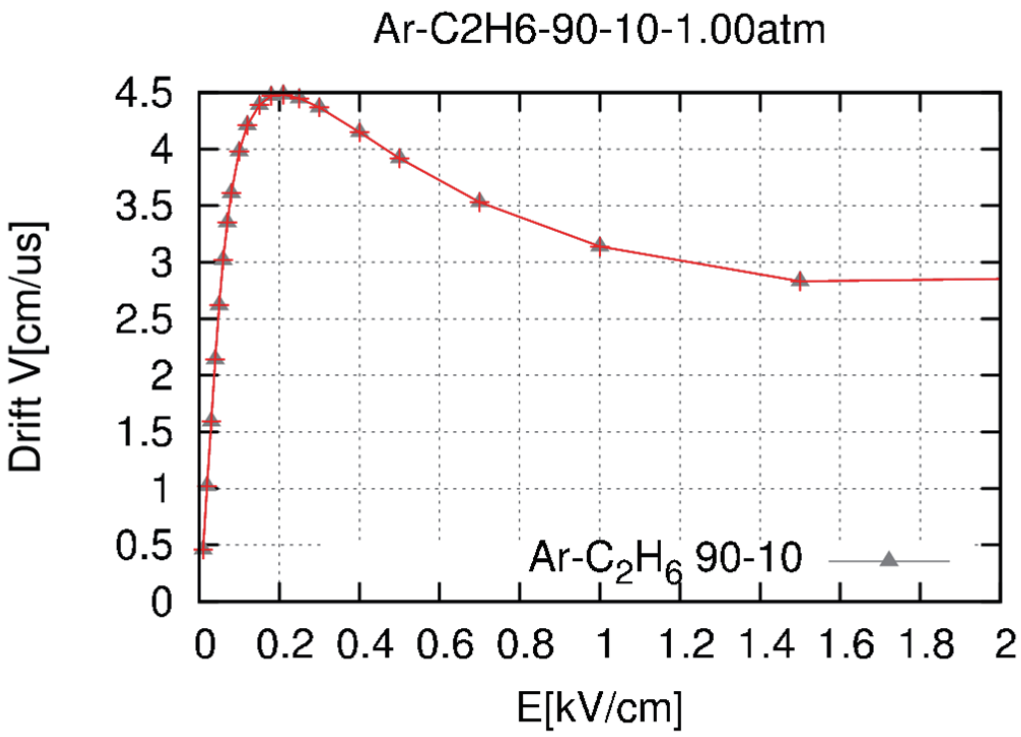


図 17　ドリフト速度のシミュレーション

深さを求めるためには電子のドリフト速度を知る必要がある。上図は 1 気圧の Ar90%-$C_2H_6$10% 中でのドリフト速度のシミュレーションである。我々の実験では電場を 0.32kV/cm として、上図からドリフト速度を 4.2cm/μs と見積もった。以降ドリフト速度には 4.2cm/μs という値を使っていく。

5.3 実験方法

実験装置の配線は 2D-imaging 取得の時と同じである。

Encoder から出力される個々の位置情報 +clock 情報はそのままではバラバラになっているため、一つの放射線が走ったイベントでのデータ群をそこから選び出す必要がある。そのために自動で同一イベントと考えられるデータを集めて一つのデータ列として扱えるようなソースプログラムを用いた。同一イベントを判別する条件として、データ間の間隔が 10clock 以内になるように設定した。つまり隣の点から隣の点までの間隔が 100ns 以内のものを同一イベントとみなす。カウントレートは約 3Hz 程度であり、放射線ごとの時間間隔は clock に対して十分長い。よって clock が近いという条件で判定するのが妥当である。実際の計測では、同一イベントでの点から点の clock 間隔は 5clock 以内には収まっていた。　さらに 1 イベントのデータ点の数が 4 点以上という条件をつけ 500 イベント分のデータを取得し、いくつか選んで三次元グラフにプロットした。

clock にドリフト速度を掛け合わせることで Z 方向の深さが出せるが、絶対的な深さは分からないため、適当な高さを基準として飛跡の出発点の高さが揃って見えるように較正を行った。

5.4　3D-tracking の結果

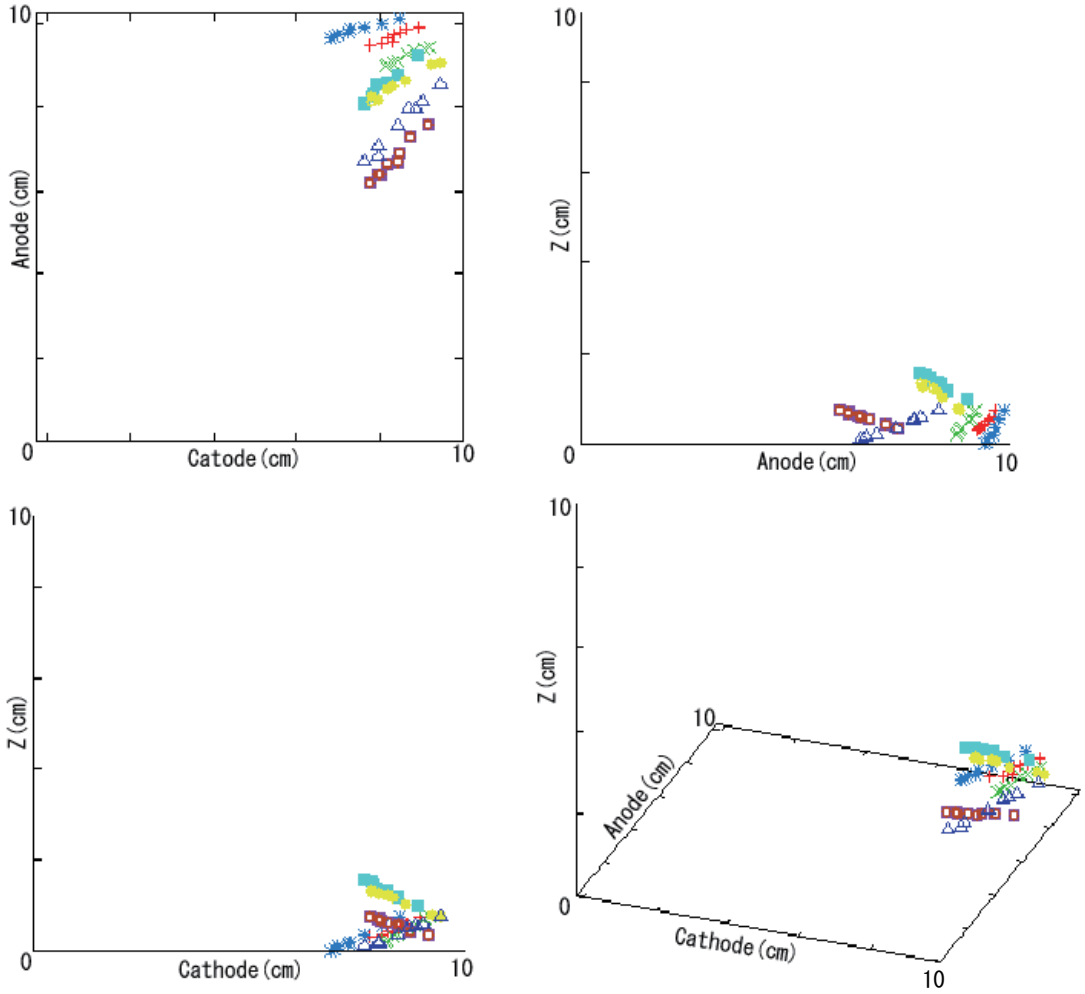


図 18　3D-tracking

500 イベントのデータ中から 7 イベントを選んで飛跡の 3D-tracking を作成した。ここでは高さの基準を 1.6cm として飛跡の端の高さを揃えている。上から見た図はまさしく線源から放射線が放射状に出ている様子を示しており、線源が存在すると思われる位置も 2D-imaging の結果と一致している。

6　α線の飛距離とエネルギー

これまでの実験によって、μ-PIC を使用して放射線のエネルギー及び飛距離を求めることができることを確認した。

次のステップでは、今まで別々にやっていた波形取得と位置データ取得を同時に行うことによって、放射線の落とすエネルギーと飛距離の関係を議論することを目的とする。

6.1　実験原理

マントルから放射されたα線は、ガス中を飛行する過程でエネルギーを落としていく。荷電粒子のエネルギー損失は次の Bethe-Bloch の式で表される。

$$-\frac{dE}{dx} = 2\pi N_{\mathrm{a}} r_{\mathrm{e}}^2 m_{\mathrm{e}} c^2 \frac{Z}{A}\frac{z^2}{\beta^2}\left[\ln\left(\frac{2m_{\mathrm{e}}\gamma^2 v^2 W_{\max}}{I^2}\right)\right] \tag{1}$$

N_{a} はアボガドロ定数、r_{e} は古典電子半径、Z は吸収物質の原子番号、A は吸収物質の原子量、z は入射粒子の電荷、m_{e} は電子の質量、v^2 は入射粒子の質量、$W_{\max}$ は 1 回の衝突で入射粒子が与えうる最大エネルギー、I は平均励起ポテンシャルである。

右図は 1 気圧の空気中における ^{214}Po からのα線の Bragg 曲線の例である。荷電粒子の飛程中でのエネルギー損失は、右図のような Bragg 曲線と呼ばれる曲線に従うことが知られている。

始めはほぼ平坦にエネルギーを落としていき、α線が止まる直前にもっとも大きくエネルギーを落とす。飛程中で最もエネルギー損失の大きいところは Bragg Peak と呼ばれる。

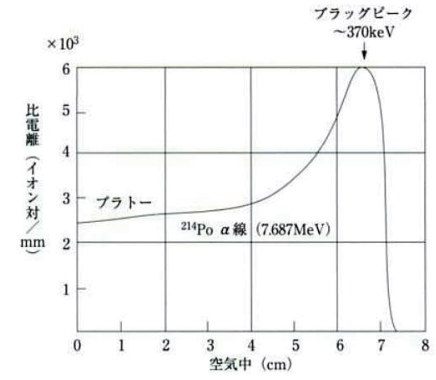


図 19　空気中でのα線による Bragg 曲線（出典『放射線基礎計測学』2008 年）

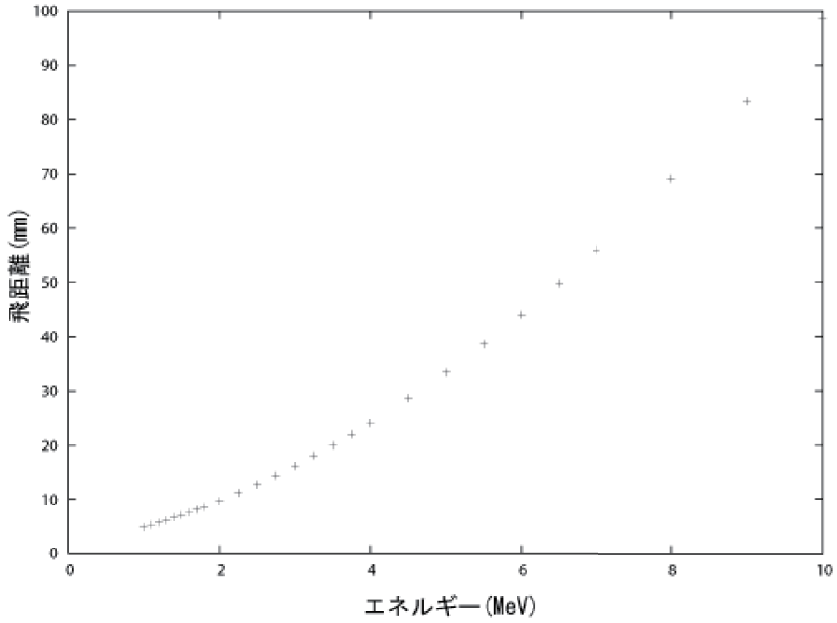


図 20　Ar-C_2H_6 ガス中での飛距離とエネルギー

上図はシミュレーションソフト、SRIM2008 による
α線の飛距離のシミュレーションである。トリウム壊変系列より、マントルから放射されるα線のエネルギーは 4MeV から 9MeV 程度である。ガスパッケージは一辺 10cm の立方体であり、マントルは上から見てパッケージの隅に設置されていると考えられるので、検出されるα線はほぼ全てのエネルギーをガスパッケージ内で落とし切ると考えられる。

位置データと clock から飛跡を取得するのと同時に、全ての ASD から波形のアナログ信号を FlashADC によって読み出し、その信号の Area の合計を取るという操作を行う。Area の合計からスペクトルが得られるので、第 2 章の手法でエネルギー較正を行い、Area をエネルギー値に直すことができる。これによって放射線の飛跡とその放射線が落とした全エネルギーを同時に取得することができる。このようにして測定した飛距離とエネルギーの分布は、シミュレーション結果に対応することが期待される。

6.2 実験方法と TPC モード

デジタルの位置情報とアナログ波形を同時に取るために、Encoder の機能である TPC モードを用い、実験装置もそれに合わせて設置した。

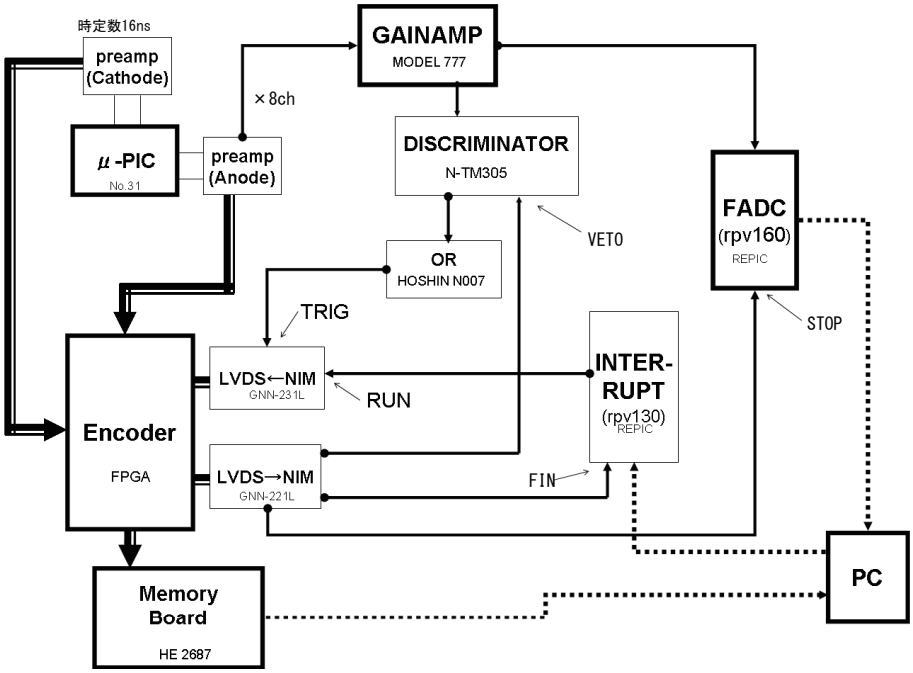


図 21　TPC モードの概念図

位置と Clock データは、Anode、Cathode の ASD から Encoder に入り Memory Boad を通して PC に入る。波形のアナログ信号は Anode 側から NIM で 8ch 分出力されて Gainamp で増幅された後、FADC を通って PC へ入るようになっている。アナログ信号のそれぞれのそれぞれの ch は、検出面を 8 個に分割した Anode 側のスリットにそれぞれ対応する。

これらのデジタル信号と波形信号の同期を取る役目は、主に Encoder の機能が担っている。次のページでもう少し詳しく説明する。

まず PC から INTERRUPT に RUN 信号を送るよう命令を出す。Encoder は RUN 信号を受けて Discri の VETO を解除し、同時に位置データを取り始める。VETO 解除後、Discri は Gainamp から入ったアナログ信号を矩形波にして OR に出力する。このとき、8ch のうちどれかでも、最も早く来た信号が TRIGER となって OR から出力される。OR から Encoder に信号が来た時点で、Discri に VETO 信号を出し、Discri から信号が来るのを停止させる。TRIG がかかってから 8μs 遅れて Encoder から FADC に STOP 信号を出し、FADC はアナログ信号の読み込みを止める。そして TRIG がかかってから 25μs 遅れて、Encoder から INTERRUPT に信号が入り、PC は MemoryBoad と FADC からのデータの読み込みを開始する。

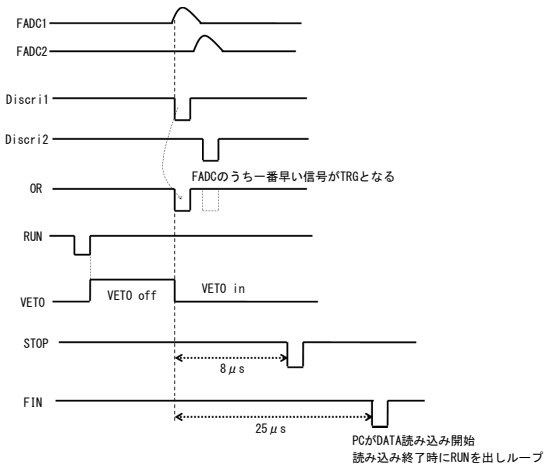


図 22　TPC モードの概念図 2

読み込みが完了次第、再び INTERRUPT から RUN 信号が出され、始めに戻って信号取得を開始する。

このようにして、放射線 1 イベントごとに、位置データ及び Clock と波形データを取得することができる。

測定に用いたソースプログラムは付録に記載する。

装置を一晩かけて回し、32768 個分のデータを取得した。

6.3 出力例

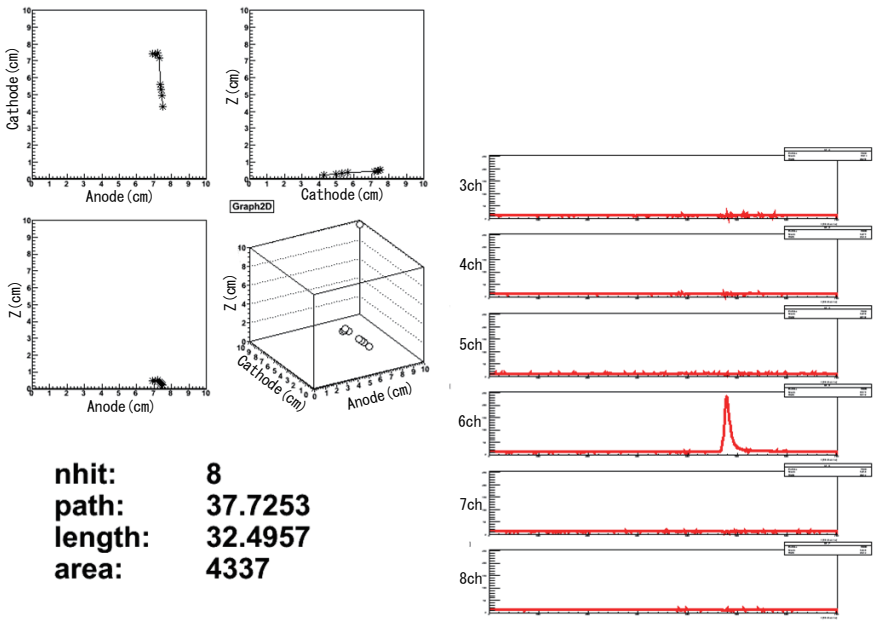


図 23　出力例

取得した位置データと波形データから、放射線 1 イベントについて上図のような図が描ける。

左側が飛跡のデータ、右側が波形データである。nhit は同一イベントでの位置データの数、path は点と点の距離の合計、length は最初と最後の点の距離、area は波形データの積分値の合計をそれぞれ表す。

このデータでは飛跡は Anode 側のスリットに沿って縦に伸びているが、そのことは、そのスリットに対応する 6ch だけに波形データが立っていることとよく対応している。

波形データからは 1,2ch のデータを抜いてあるが、これは次章の本実験へ向けて考慮したためである。右図に 1,2ch を入れた場合（赤）と 1,2ch を抜いた場合（黒）の Area のヒストグラムの比較を示す。1,2ch は線源から最も遠いスリットであって、測定にはほぼ影響は無いと考えられる。図もそれを示している。

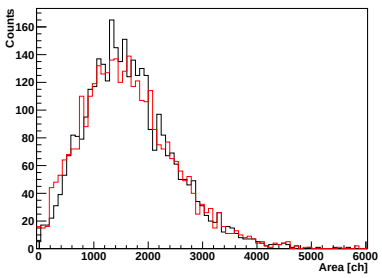


図 24　Area のヒストグラムの比較

6.4　結果と考察

計測によって得られたデータから、path と Area についての散布図及び、Area、path に関するヒストグラムを描いた。なお散布図及びヒストグラムに使用したデータは、イベント中の位置データの数が 4 点以上という条件で選び出した。

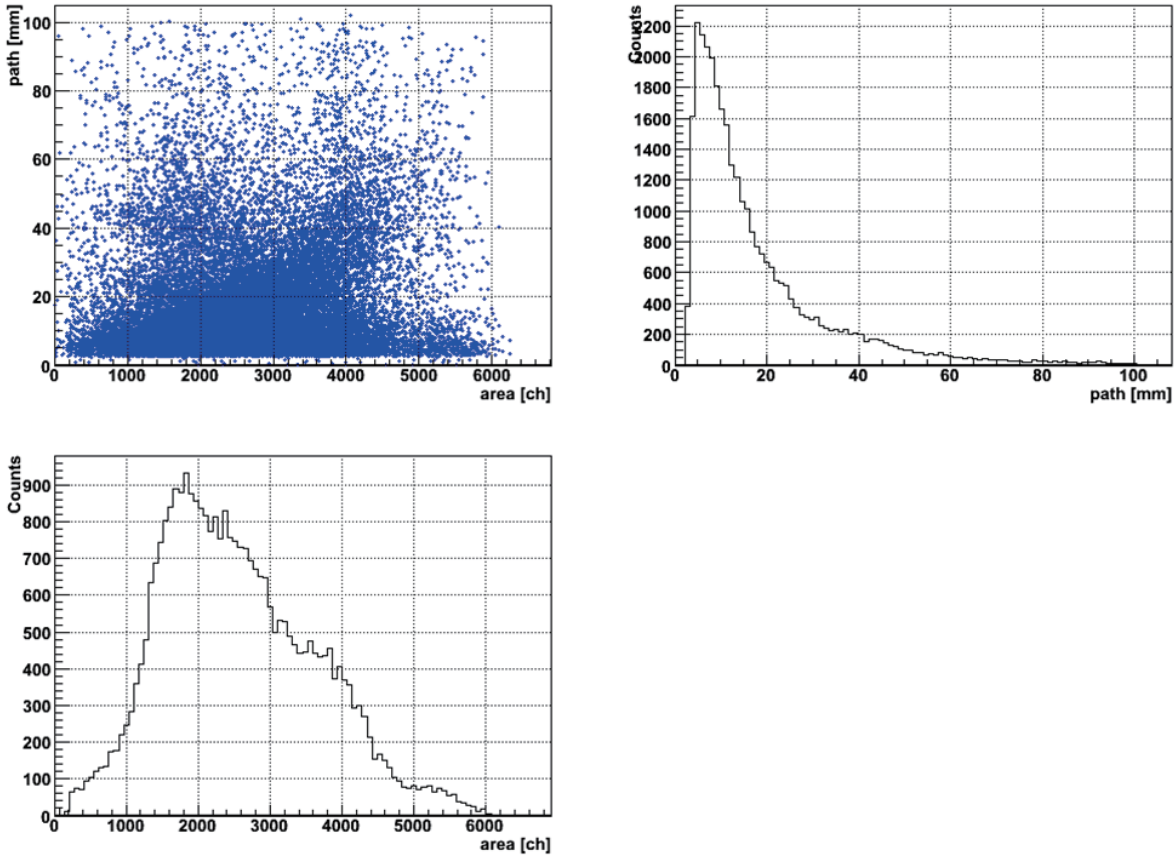


図 25　散布図

散布図には、上方へ伸びている成分、斜めに伸びている成分、真横に伸びている成分と、何かしらの構造が見られるが、マントルからのα線の飛程を表しているのかどうかはこのままでは分からない。

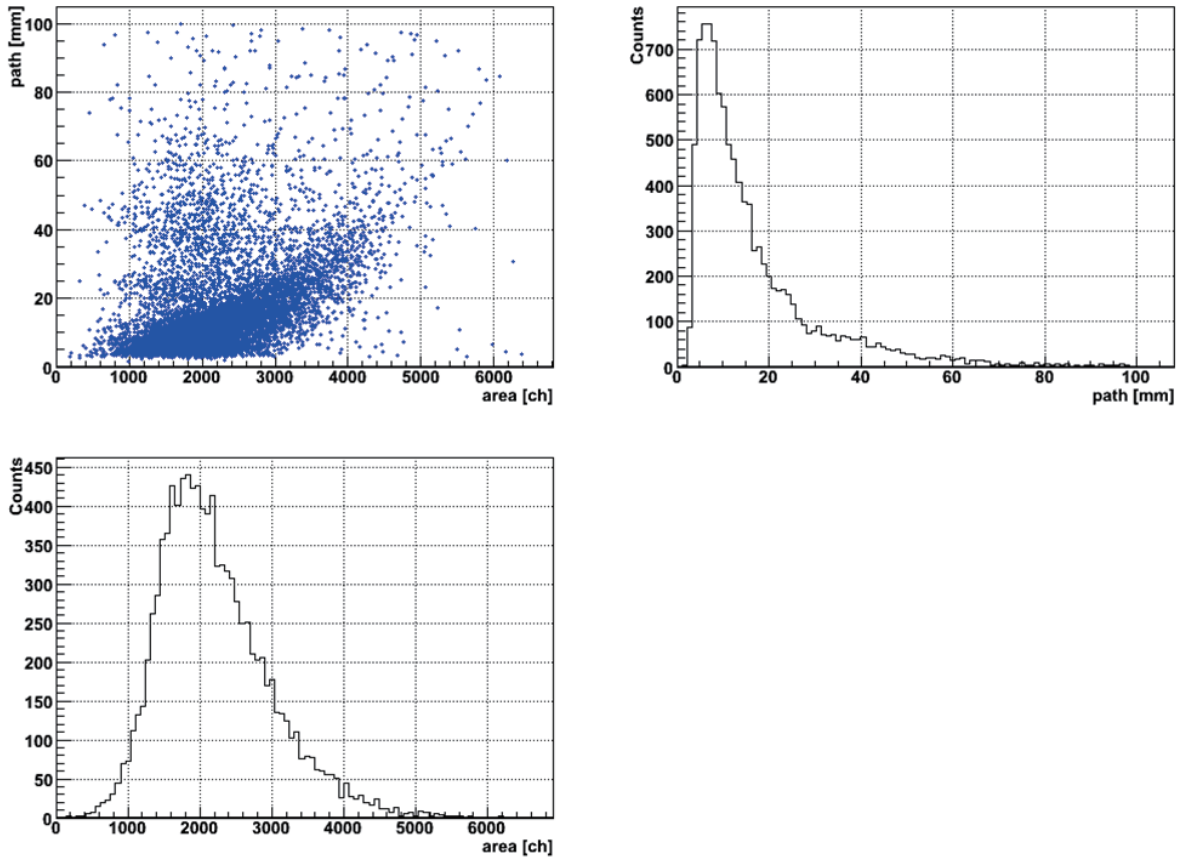


図 26　散布図 x-ycut

次に、線源付近のイベントに限って散布図を描くと上図のようになる。データを線源付近で選び出すというのは、2Dimaging の時に特定した、検出面の隅から 1.28cm 四方以内にデータ点が最低一つ入っているイベントを抽出することに相当する。これによってマントルからの放射線とは関係ないと思われるイベントを除外することができる。

このとき、散布図上では path の小さい領域で真横に伸びている成分が消えている。このことから、真横にべったりと伸びている成分はマントルからの放射線とは関係ないイベントであると考えられる。

しかし、まだ 2000ch 付近で上方に集まっている成分が存在する。これらの成分は飛距離が長く計測されている割りにエネルギーをあまり落としていないイベントである。

2000ch 付近で縦に伸びている成分のイベントを詳しく調べたところ、α線の飛跡にしては不自然にギザギザしているものが多数見られた。我々はこれを Encoder 周りのノイズなどの影響によるものと考えて、これら飛跡のギザギザしたイベントを排除した。排除する際には、path と length の差が path の 10% 以上という条件を用いた。

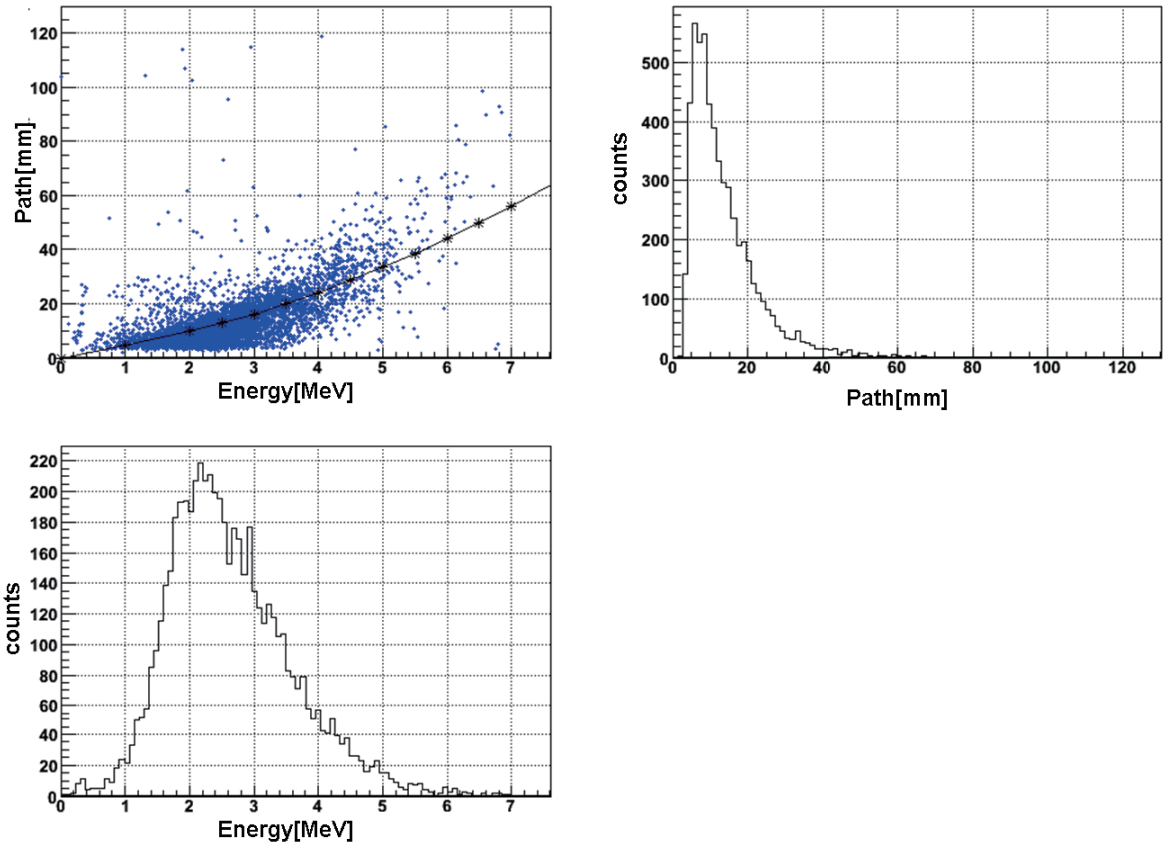


図 27　散布図 pathcut

こうして線源付近にあり、かつ飛跡が真っ直ぐという条件にあうイベントを集めたのが上図である。path と Area の散布図にはシミュレーション結果に対応するような相関がみられるように思われる。

散布図中に、シミュレーションの曲線を相関に当てはまるよう描き入れた。それを元にシミュレーションからエネルギー較正を行い、散布図の横軸の ch を MeV に直してある。

シミュレーションの曲線を挿入したがこれはあくまで目で見て当てはめたものであり、なんら解析的な手法を使っていない。本来ならば Spectrogram からエネルギー較正を行うべきであるが、我々の計測ではうまくトリウム系列に合致するようなスペクトルが得られなかった。トリウム系列のスペクトルについては、2003 年度 P6 課題研究、希ガスシンチレータ班のレポートを参照のこと。

α線の飛距離とエネルギーに相関が存在することを確認することは出来たが、本格的な解析を行えるまでには至らなかった。今後の課題としては、ドリフト容器内から線源を除いたバックグラウンドを計測する、あるいはイベント一つ一つのデータを詳細に見ていき、散布図上の座標と、飛跡の形や走った方向などのデータの傾向との関係を調べる、などの方法が挙げられる。

7　本実験

さて、これまでの実験で用いてきた μTPC では、放射線の入射高さまでは計測できなかった。この問題を解決するには、放射線が入射して生成された電子雲が陽極に到着する時刻を知るだけでは不十分である。そこで、我々は放射線が入射したときにガス分子が放出するシンチレーション光を光電子増倍管で捕捉することで、放射線が入射して電子雲が生成された時刻情報の取得も試みる。この情報と、μTPC で得た放射線飛跡の不完全な 3 次元位置情報とを組み合わせることで、放射線飛跡に関する完全な 3 次元情報を得ることが出来る。

7.1　原理

7.1.1　シンチレーション光の発生機構

励起状態にある物質が基底状態に遷移する過程で光子が放出される。これをシンチレーション光という。励起状態のエネルギー準位を E_{ex}、基底状態のエネルギー準位を E_{gr} とすると、放出されるシンチレーション光の波長 λ は

$$\lambda = \frac{hc}{E_{\mathrm{ex}} - E_{\mathrm{gr}}} \tag{2}$$

となる。ここで h はプランク定数、c は光速である。また、気体分子の場合、遷移にかかる時間は通常 2〜3 ns あるいはそれより短い。純粋な Ar ガスの場合、シンチレーション光の平均発光波長は 1 atm の条件下で 250 nm である。

7.1.2　シンチレーション光発生のタイミング

本実験ではシンチレータとして μTPC に封入してあるガス Ar90%-$C_2H_6$10% を採用する。これらのガス分子が励起される、すなわちシンチレーション光が発生するのは本実験では主に次の 2 つの場合である。一つは、放射線である荷電粒子がガス内を通過するとき、もう一つは、入射放射線によってつくられた電子が陽極付近の強い電場で加速しながら電子なだれを起こすときである。後者は入射放射線によって作られた電子が陽極に到達し始めてから全て到達し終えるまで、断続的に光り続ける。

7.2 実験装置

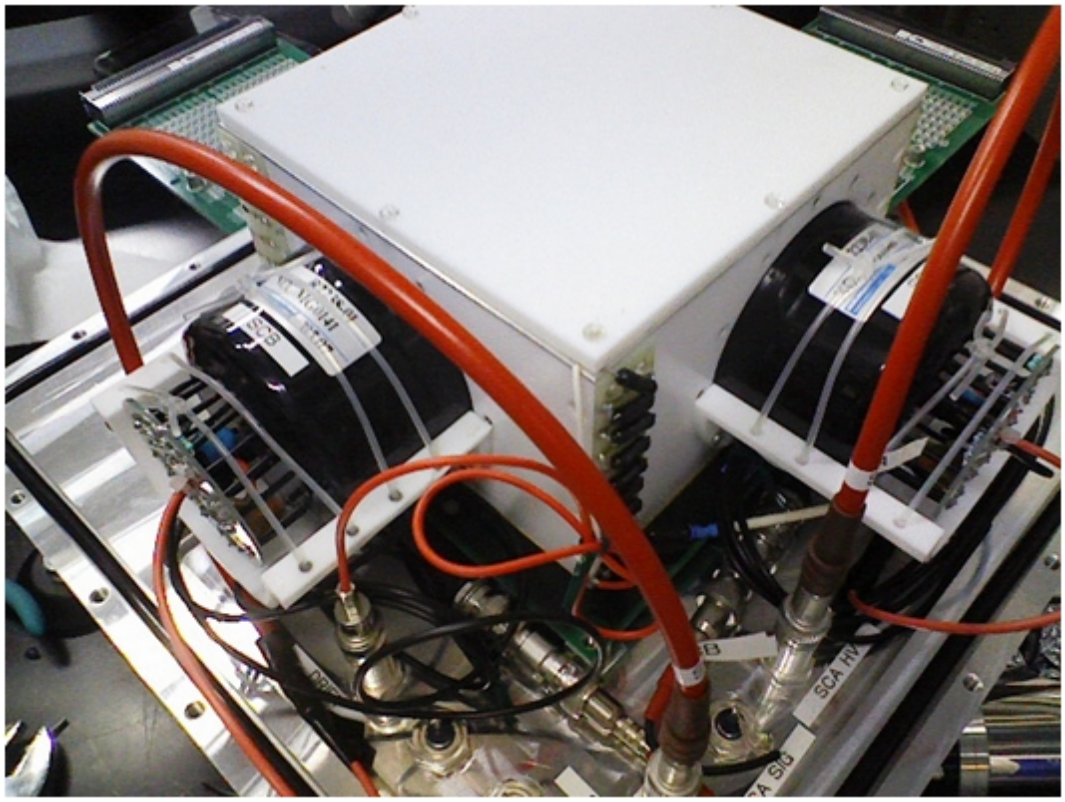


図 28　光電子増倍管の取り付け図

実験装置は前節で用いたものをそのまま今回も使用する。本実験ではそれらに加えて、新たに光電子増倍管（浜松ホトニクス製　R2238-01）をドリフトケージの側面 2 か所に計 2 個設置する（図 28）。陰極-陽極間にはそれぞれ −1250 V の電圧を印加する。また、ドリフトケージ内側にはテフロンがコーティングされており、ケージ内で発生した光を反射して効率よく光電子増倍管に捕捉させる。

光電子増倍管の出力信号は計算機で解析できるように、それぞれ増幅器で 40 倍に増幅してから Flash ADC に入力する。増幅器に通す理由は、光電子増倍管からのシンチレーション光と見られるピーク波形はそのままではそのピーク高さが 数 mV であり、これは Flash ADC の分解能と同程度の大きさしかないことが後述の予備実験において判明したためである。

データ取得のアルゴリズムは前節のものを引き続き使用する。Flash ADC のストップ信号入力には、前節と同様に μ-PIC の Anode 信号 6 つがそれぞれ Discriminator で NIM 信号に変換されたものの論理和信号を採用する。これにより、本実験においても放射線 1 イベントに対して波形データと飛跡データが同時に取得できる。

7.3 予備実験

7.3.1 予備実験の目的

本実験の実験方法に見通しをつけるため、光電子増倍管からの信号をオシロスコープで観測する。

7.3.2 予備実験の方法

実験装置は本実験と同様のものを使用し、 μ-PIC の Anode アナログ信号のうち、線源であるマントルがある位置に最も近い出力のものを増幅器で 2 倍したもの 1 つ、光電子増倍管からの信号 2 つの計 3 つの信号をそれぞれオシロスコープに入力する。3 つの信号波形の特徴を確認する。

7.3.3 予備実験の結果

トリガソースを μ-PIC のアナログ信号にしてみると、 μ-PIC のピーク波形と同期して光電子増倍管の信号には深さ 5 mV 程度、時定数が 30 ns 程度の多数のピーク波形が見られた。その一例を図 29 に示す。

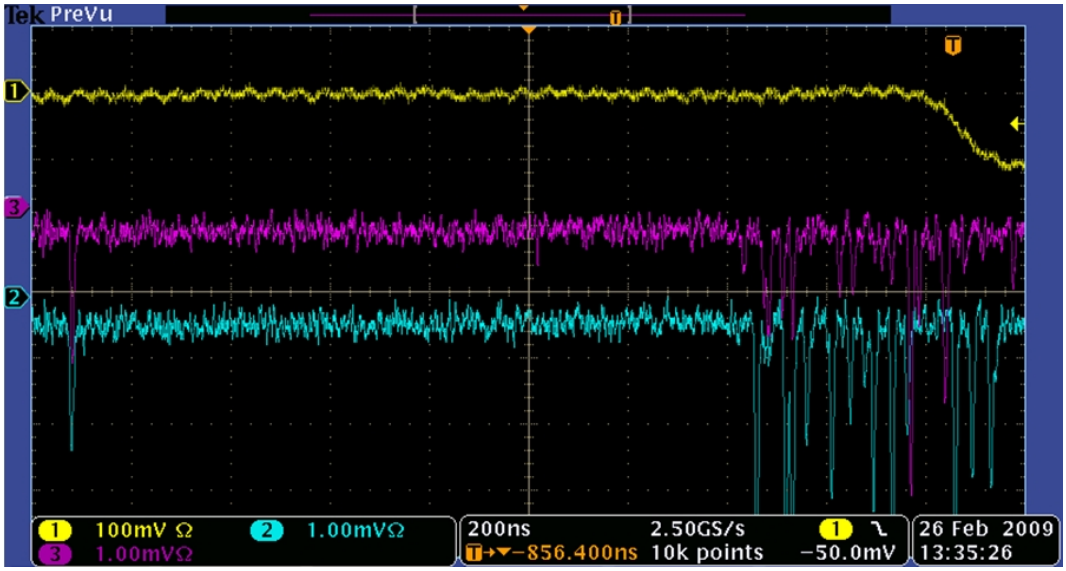


図 29　オシロスコープで見た信号の一例

そのピーク波形群は μ-PIC のピーク波形の時定数と同程度の 数百 ns にわたって分布していた。次に、光電子増倍管の信号に焦点を絞って、多数のピーク波形が現れる時刻から時間を遡っていくと、2.5 μs 遡るまでに孤立したピーク波形がイベントによって見られたり見られなかったりした。この孤立したピーク波形の特徴を以下に列挙する。

- いつも孤立したピークが見られるわけではない。
- ピークが現れた場合、光電子増倍管の 2 つの信号のうちの片方のみで見られることがほとんどである。
- まれに、光電子増倍管の 2 つの信号で同時に孤立したピークが現れる。
- 時定数はピーク波形群のものと同程度のおよそ 30 ns である。
- ピークの深さは 1 ~ 4 mV でばらつきがあるが、ピーク波形群の深さよりも低い傾向にある。
- ピークが現れる時刻に偏りがあり、ピーク波形群が現れるおよそ 2 μs だけ遡った時刻に現れることが多い。

また、トリガソースを光電子増倍管の信号にしてみると、 μ-PIC のピーク波形が前後 4 μs には現れないタイミングで孤立したピーク波形が見られた。孤立したピーク波形の深さは 数 mV のものから 20 mV 程度のものまで様々であった。

7.3.4 予備実験の考察

μ-PIC のピーク波形と同期して現れる光電子増倍管の多数のピーク波形について、 μ-PIC のピーク波形が現れるときには常に現れることと、その出現の同期性と発現期間から、これは入射放射線によって生成された電子雲が陽極付近で電子なだれを起こしたときに発生したシンチレーション光によるものであると考えられる。

次に、ピーク波形群から時間的に遡ったところで現れる孤立したピーク波形であるが、ここでわれわれが期待するのは、これが入射放射線がガス分子を電離した際に発生したシンチレーション光によるものであるということである。その決定可能性について、ピーク波形の時定数とピーク波形の現れるタイミングの 2 つの観点から議論を進める。

ピーク波形の時定数について考える。観測された孤立したピーク波形の時定数が 30 ns 程度であるのに対して、本実験で使用した光電子増倍管の時間特性は上昇時間 2.6 ns、走行時間 48 ns と光電子増倍管の時定数とオーダーのレベルで等しい。従って捕捉した光の発光時間は長くとも 数 ns 程度である。

一方、放射線が入射してシンチレーション光が発生した場合を考えてみる。入射放射線として 4 MeV のアルファ線を仮定しよう。マントルに含まれる ^{232}Th は 4.01 MeV のアルファ線を放出する。この場合、アルファ線の速度は光速の 10 %である。この速度でドリフトケージ内部を 10 cm 通過するのにかかる時間は 3 ns 程度である。前節の実験で判明したようにマントルから放出される放射線の飛程距離のうちで典型的なものは 1 cm 程度であったから、アルファ線がガス分子を電離していく過程はおおざっぱに 1 ns 未満としてよい。ここでもう少し厳密に議論しようと思えば、アルファ線がガス分子を電離してエネルギーを失っていく過程で減速していくことも当然考慮に入れなければならない。またガス分子が遷移の際にかかる時間は原理のところで述べたとおり高々 数 ns である。これらを総括すると、放射線が入射してシンチレーション光が発生するとき、その発光時間は高々 数 ns であると見積もられる。

従って、観測された孤立ピーク波形の時定数については、放射線入射によるシンチレーション光のものと矛盾しない。

ピーク波形が現れるタイミングの偏りについて考える。予備実験の結果でも述べたとおり、孤立したピーク波形は電子なだれ増幅によるシンチレーション光のピーク波形群が現れるおよそ 2 μs だけ早いタイミングでよく現れた。これが、放射線入射によるシンチレーション光によるものであると仮定すると、電子雲がドリフトした時間が 2 μs のものが多いということを示す。また、電子雲の発生した高さは陽極面から 2 [μs] $\times$ 4.2 [cm/s] $\sim$ 8 cm と見積もられ、これはドリフトケージの天井よりも低い位置にあり、電子雲の発生場所としては問題ない。さらに都合よく考えると、線源であるマントルが高さ 8 cm 付近にあるために電子雲がその高さでより多く生成したと考えることもできる。

しかしながら、楽観的な考察ばかりもできない。トリガソースを光電子増倍管の信号にしたときに見られた孤立ピーク波形について、これらは我々の想定しないプロセスによる信号で、ノイズである。その原因として第一に考えられるのは、密封容器の光漏れであるが、他にも熱電子によるノイズなど光電子増倍管の内的な要因も考えられる。このノイズ信号がどのタイミングで現れるかについて、我々は知りえないので、なだれ増幅によるピーク波形群の出現より前に現れる孤立したピーク波形について、それが入射放射線がガス分子を電離した際に発生したシンチレーション光によるものであるか、たまたまノイズ信号がそのタイミングで現れたのかどうかについて現段階で決定することはできない。

7.3.5 予備実験のまとめと本実験への移行

光電子増倍管の信号からは、 μ-PIC のピーク波形に同期して電子なだれ由来のシンチレーション光によるピーク波形群がみられる。そこから時間を遡ったところに孤立したピーク波形が見られる場合があるが、それが放射線入射由来のシンチレーション光によるものかどうかは判別できない。

そこで本実験では、電子なだれ由来のシンチレーション光によるピーク波形群より時間を最大 2.5 μs より遡って初めて現れるピーク波形を放射線入射由来のシンチレーション光によるものであるとひとまず仮定する。2.5 μs という時間はドリフト速度 4.2 cm/s とかけ合わせて 10cm、つまりドリフトケージの高さいっぱいまでは電子雲が発生する場所として適切だからである。

7.4 実験方法

光電子増倍管の信号でみられる電子なだれ由来のシンチレーション光のピーク波形群は、Flash ADC から読み込んだデータの特定の場所に現れる（図 30)。今回の実験の場合、Flash ADC 読み込みのストップがかかったところから 8.2 μs 遡ったあたりからピーク群の最初の立ち上がりが始まる。そこで、Flash ADC 読み込みのストップがかかったところから 8.3 μs ～ 12.8 μs 以前のデータに対して平均値を求め、これを信号のベースラインとする。次に、Flash ADC 読み込みのストップがかかったところから 8.3 μs 遡った位置から時間的に進む方向へ、ベースラインの電圧値よりはじめて 3 [ch] $\times$ 1000/256 [mV/ch] $\simeq$ 11.7 mV 高くなるものを探し、その時刻を電子なだれ増幅由来のシンチレーション光が発生し始めた時刻 t_2 とする。さらに t_2 から最大 2.5 μs だけ遡りながら初めてベースライン電圧値より 11.7 mV 高くなるものを探し、あればその時刻を t_1 とする。11.7 mV という値は、シンチレーション光によるピーク波形が目測では見られた 10 イベントの信号に対して試験的に設定して上手くピーク位置を取得できたことから取り決めた。これを光電子増倍管 A,B の 2 つ信号に対して評価し、$t_2^{(\mathrm{A})}$、$t_2^{(\mathrm{B})}$ と、存在するなら $t_1^{(\mathrm{A})}$、$t_1^{(\mathrm{B})}$ を求める。

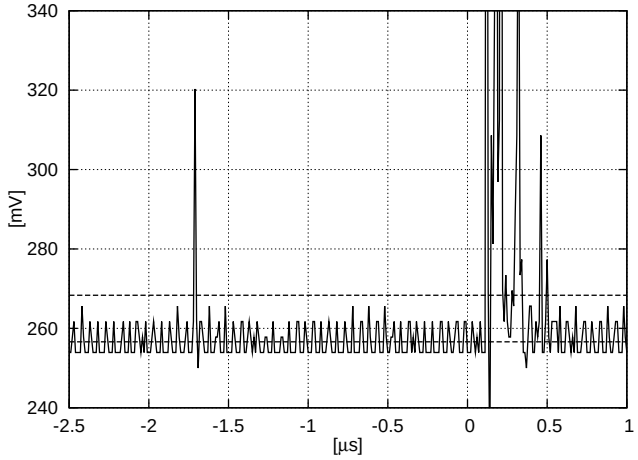


図 30　Flash ADC で読み込んだ PMT の信号：2 本ある破線のうち、下方のものはベースライン電圧値、上方のものはベースラインより 11.7 mV 高い電圧値。また、時刻の原点は Flash ADC 読み込みのストップがかかったところから 8.3 μs 遡った位置。

$t_1^{(\mathrm{A})}$ または $t_1^{(\mathrm{B})}$ が存在するイベントに対して t_1、t_2 の時間差 $T_{\mathrm{lag}} := t_2 - t_1$ を求める。もし $t_1^{(\mathrm{A})}$、$t_1^{(\mathrm{B})}$ が両方存在するならば、$T_{\mathrm{lag}}^{(\mathrm{A})}$、$T_{\mathrm{lag}}^{(\mathrm{B})}$ のうち小さいものを T_{lag} とする。多数のイベントに対してこの T_{lag} の分布を調べる。

次に、先で得られた $T_{\rm lag}$ と μTPC で取得したイベントごとの 3 次元位置データとの統合の方法について述べる。μTPC で得られる 3 次元放射線飛跡情報のうち、高さ情報に対応するデータ列のうちで最も高さが低い点の高さ情報を z_1 [cm] とする。較正した高さ情報 $z_k (k = 1, \cdots)$ を、較正前の高さ情報を $\tilde{z}_k$ とし、また電子のドリフト速度を $v_{\rm drift} = 4.2\,{\rm cm}/\mu{\rm s}$ として

$$z_k = \tilde{z}_k - \tilde{z}_1 + v_{\rm drift} T_{\rm lag} \tag{3}$$

とする。すなわち、$z_1 = v_{\rm drift} T_{\rm lag}$ は入射放射線によってガス分子が電離したときに生成された電子雲のうち、最も陽極面に近い部分の生成高さを与えると考える。また、他のデータの較正された高さ情報は z_1 との相対的な距離から求められる。ここでの議論で、$T_{\rm lag}$ が電子雲の最も陽極面に近い部分が陽極面に到達するまでの時間であるという仮定が要求されることに留意する。

このようにして、完全に較正された 3 次元飛跡情報を導出することが可能であるので、多数のイベントに対してこれを適用して、較正できた全イベントの飛跡データを重ねてプロットし、場所ごとの強度分布図すなわちイメージ図を作成してみる。

7.5 結果

32768 個のイベントについて飛跡情報と波形データを取得した。そのうち、$T_{\rm lag}$ を求めることができたのは、14502 個であった。$T_{\rm lag}$ の度数分布をヒストグラムにしたものを図 31 に示す。

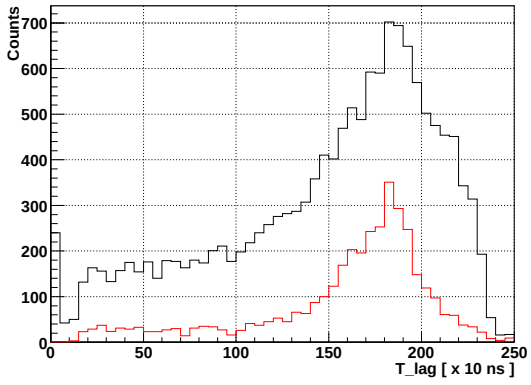


図 31　$T_{\rm lag}$ の度数分布：黒のヒストグラムは全イベントに対する度数分布、赤のヒストグラムはマントル付近にあるイベントの度数分布を表す。

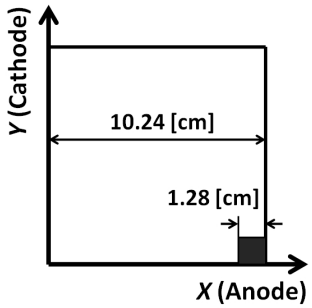


図 32　マントル付近にあるイベントの抽出方法：図の右下隅にマントルがある。灰色の区域内に少なくとも飛跡の位置データが一つあるものをマントル付近にあるイベントとする。

最頻値が 1.8 μs 辺りにあることが見て取れるが、マントル付近にあるイベントに限って度数分布を取り直すとそのピークはさらに際立つ。ここにいう「マントル付近にあるイベント」とは、放射線飛跡の X-Y 平面つまり陽極面に投影した位置データの中の少なくとも 1 点についてマントルのある検出面の隅の角から 1.28cm 角の正方形の区域の中に位置するという条件を満たすイベントである（図 32）。

さらに、較正された 3 次元位置情報を持つイベント群 14502 個を用いて 3D イメージングを構成したところ、図 33 のようになった。

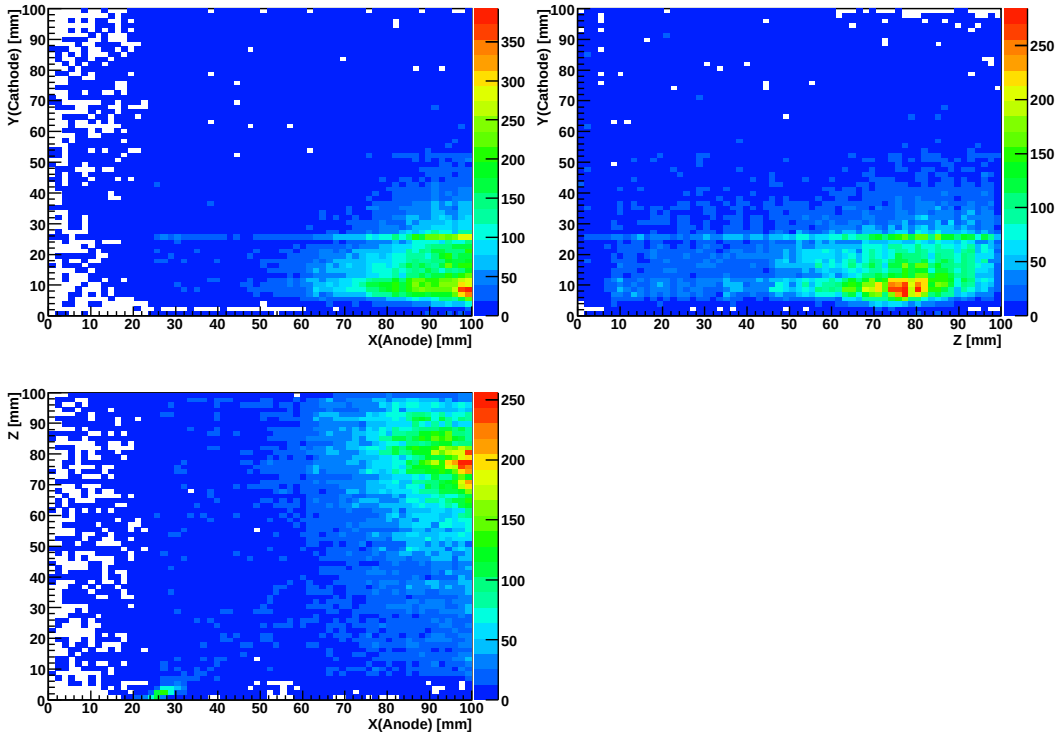


図 33　3D イメージ：左上図は X-Y 投影面、右上図は Z-Y 投影面、左下図は X-Z 投影面のイメージ図である。

X-Y 投影面のイメージについては、以前の実験で得た結果と同様の結果であるが、今回は新しく X-Z 投影面や Y-Z 投影面のイメージを得た。これによると、ある高さの点を中心に放射状に放射線が強く観測されている様子が描かれている。そのコアの位置は最頻値と強度の広がりの中心という二つの観点から見て (99 mm、9 mm、77 mm) の位置である。

ところで、我々は現段階において現実のマントルが、容器内のどの位置に仕込まれていてどのくらいのサイズであるかについて知らなかったのだが、ここでその実際についても明らかにしておく。設置されたマントルは図 34 の写真のように、検出面からおよそ 8 cm 程度の高さの位置にあり、また、マントルのサイズはおよそ 2 cm 程度である。

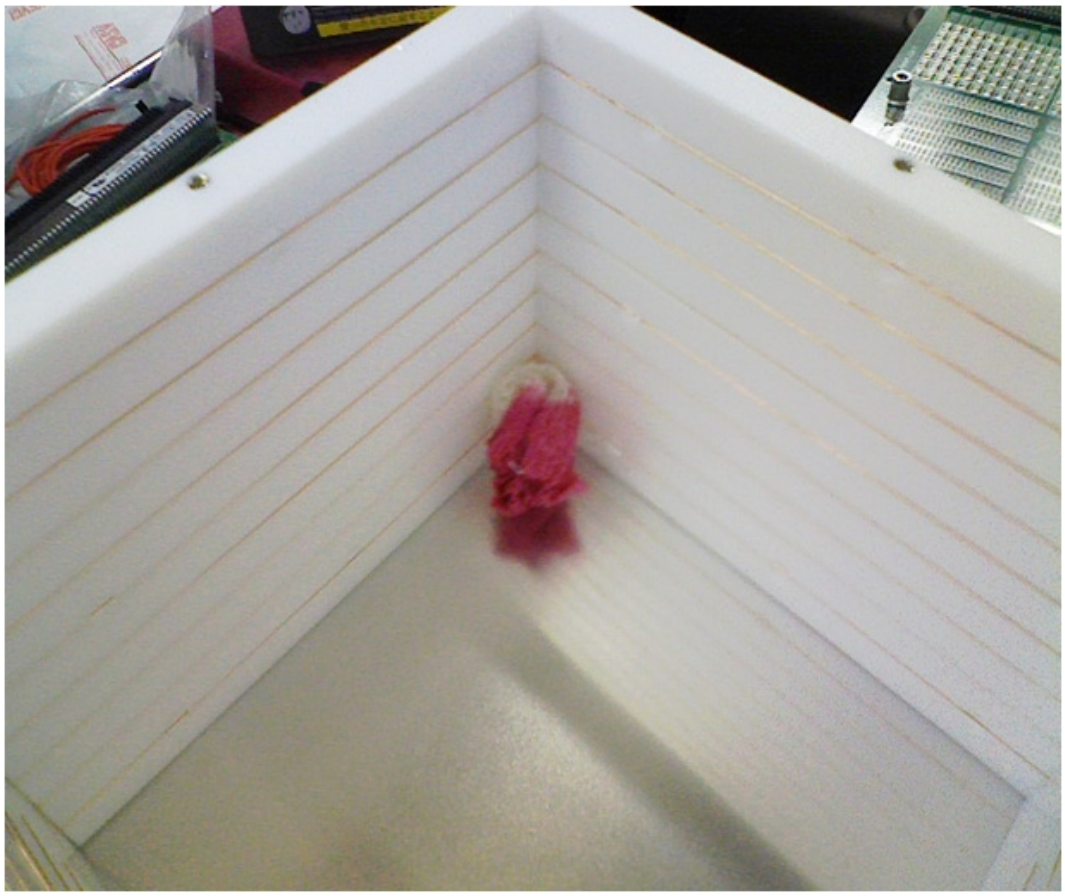


図 34　マントルの実際の位置１：この写真はドリフトケージを逆さまにした状態で撮影したもの。

7.6　考察

7.6.1　実験の妥当性

図 35 の写真とイメージ図 36 を比較すれば、イメージ図がマントルの位置を見事に再現できていることはまず第一に強調したい。

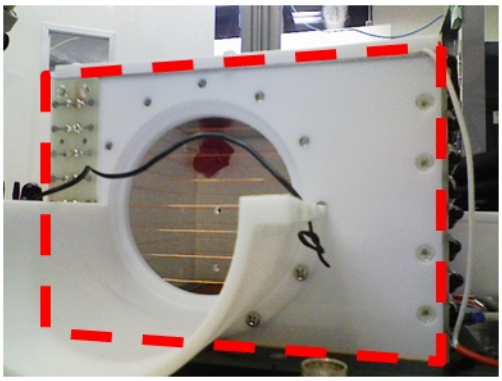


図 35　マントルの実際の位置 2

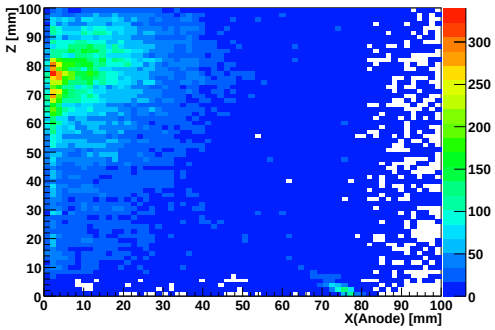


図 36　図 35 の赤枠に対応した投影面のイメージ図

また、Tlag の度数分布について、1.8 μs 辺りに集中することは、予備実験において予言されたものであった。さらに、そこから示唆される 1.8 [μs] $\times$ 4.2 [cm/s] $\simeq$ 7.6 cm 辺りに、電子雲の陽極面に近い部分が多く生成されたという結果はマントルの高さがおよそ 8 cm にあったことが直接的要因といえる。というのも、前節の結果で見たように、マントルの放射線飛跡の典型的な飛距離は 1 cm 程度であるため、生成される電子雲もせいぜいその程度の幅の不定性しかもたないためである。

以上の結果から、Tlag の取り方、つまり t_1 の選び方はイメージ図を適切に描くのに十分なほどには、入射放射線由来のシンチレーション光の発光時刻をとらえていたといえる。

7.6.2　線源の詳細な位置

再び Tlag の度数分布について見てみよう。ピークである 1.8 μs よりも短い時間についても連続的な成分がみられる。この正体について大きく 2 つの場合が考えられる。一つは、バックグラウンド放射線が適当な高さで走ったときに生じたシンチレーション光をとらえた場合である。もう一つは、光電子増倍管に生じるノイズ信号がたまたま入射放射線由来のシンチレーション光の信号よりも時間的に遅れたところで発生したために、t_1 が本来の意図とは異なるタイミングで評価された、すなわち入射放射線由来のシンチレーション光の信号を誤認した場合である。これらを判別するための一つ方法として、光電子増倍管のピーク波形の電荷量と入射放射線のエネルギーとの相関を確かめることにより、正しく入射放射線由来のシンチレーション光の信号をキャッチしたかどうかについての議論ができるかもしれない。

線源であるマントルの位置に対するもう少し厳密な位置の評価方法について考える。そのために、みたび Tlag の度数分布について考える。ここではイベントはマントル付近にあるものだけに注目することにする。「マントル付近にある」という条件は先述の通りである。そもそも t_1 が入射放射線によるシンチレーション光の発光時刻をとらえていても、Tlag が与える情報は、発生した電子雲の陽極面に近い部分が陽極面に到達するまでの時間であって、電子雲の線源に近い部分が陽極面に到達するまでの時間ではない。そこで、電子雲の線源に近い部分が陽極面に到達するまでの時間 Tsrc を次のようにして求めてみよう。まず、3D-imaging による考察から線源のおよその位置 $\boldsymbol{x}_0$ は $x_0 = 99\,\mathrm{mm}$、$y_0 = 9\,\mathrm{mm}$、$z_0 = 77\,\mathrm{mm}$ である。次に、放射線の飛跡データのうちで Z が最も小さい点の位置を $\boldsymbol{x}_\mathrm{f} = (x_\mathrm{f}, y_\mathrm{f}, z_\mathrm{f})$ とし、z_f に較正する前の高さ $\tilde{z}_\mathrm{f}$ は Memory Board から得た時刻情報を t_f として $\tilde{z}_\mathrm{f} = v_\mathrm{drift} t_\mathrm{f}$ であったとする。f は最初（first）に陽極面に到達するという意味である。また、放射線の飛跡データのうちで Z が最も大きい点の位置を $\boldsymbol{x}_\mathrm{l} = (x_\mathrm{l}, y_\mathrm{l}, z_\mathrm{l})$ とし、z_l に較正する前の高さ $\tilde{z}_\mathrm{l}$ は Memory Board から得た時刻情報を t_l として $\tilde{z}_\mathrm{l} = v_\mathrm{drift} t_\mathrm{l}$ であったとする。l は最後（last）に陽極面に到達するという意味である。ここで、$|\boldsymbol{x}_\mathrm{l} - \boldsymbol{x}_0| > |\boldsymbol{x}_\mathrm{f} - \boldsymbol{x}_0|$ ならば、放射線は $\boldsymbol{x}_\mathrm{f}$ から $\boldsymbol{x}_\mathrm{l}$ へ走っていると判断し、$|\boldsymbol{x}_\mathrm{l} - \boldsymbol{x}_0| < |\boldsymbol{x}_\mathrm{f} - \boldsymbol{x}_0|$ ならば、放射線は $\boldsymbol{x}_\mathrm{l}$ から $\boldsymbol{x}_\mathrm{f}$ へ走っていると判断する。このようにして線源から出る放射線の飛跡方向を評価し、

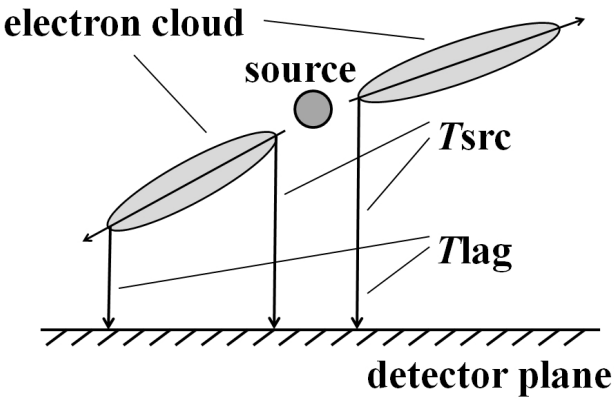


図 37　T_src と T_lag の概念図：T_src は電子雲の線源に近い部分が陽極面に到達するまでの時間。

$$T_\mathrm{src} = \begin{cases} T_\mathrm{lag} & (|\boldsymbol{x}_\mathrm{l} - \boldsymbol{x}_0| > |\boldsymbol{x}_\mathrm{f} - \boldsymbol{x}_0|) \\ T_\mathrm{lag} + t_\mathrm{l} - t_\mathrm{f} & (|\boldsymbol{x}_\mathrm{l} - \boldsymbol{x}_0| < |\boldsymbol{x}_\mathrm{f} - \boldsymbol{x}_0|) \end{cases} \tag{4}$$

と T_src を評価する。式 (4) の意味するところは次のとおりである。すなわち、放射線が陽極面から遠ざかる方向へ向かって走っているときには、電子雲の陽極面に最も近い部分は線源に最も近い部分でもある。逆に、放射線が陽極面へ近づく方向へ向かって走っているときには、電子雲の陽極面から最も離れた部分が線源に最も近い部分となる（図 37）。

表 1　$T_{\rm src}$ と $T_{\rm lag}$ に対する Fitting：ガウシアン $a\exp(-(x-b)^2/2c^2)$ において a = Constant、b = Mean、c = Sigma。フィッテングに際してその範囲はどちらも 1.5 μs ∼ 2.2 μs で行った。

	$T_{\rm src}$	$T_{\rm lag}$
Constant	315.6 ± 8.9	281.4 ± 8.1
Mean	182.7 ± 0.4	178.9 ± 0.5
Sigma	17.4 ± 0.5	19.3 ± 0.7

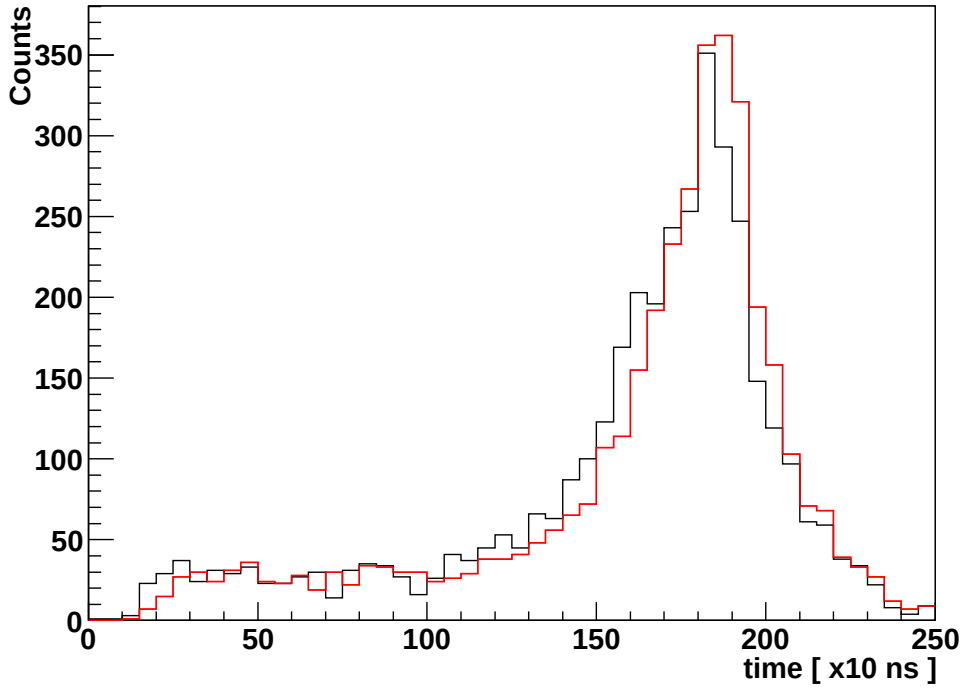


図 38　T_{src} の度数分布：黒のヒストグラムは $T_{\rm lag}$ に対する度数分布、赤のヒストグラムは $T_{\rm src}$ の度数分布を表す。

このようにして得た $T_{\rm src}$ について度数分布のヒストグラムを描いたのが図 38 である。$T_{\rm lag}$ のピークに比べて、$T_{\rm src}$ のピークは幾分左右対称形に近くなっているように見受けられる。$T_{\rm src}$ はその決め方から $T_{\rm lag}$ よりも線源の位置を表す指標としてよいものと考えられる。これをもう少し定量的に議論できるようにするために、$T_{\rm src}$ と $T_{\rm lag}$ のピークをそれぞれガウシアンでフィットしたところ、表 1 のようになった。　この表を見るに $T_{\rm src}$ のピークに対する分散のほうが 9.8 %小さくなっている。これは式 (4) による補正が効いている証拠であり、やはり $T_{\rm src}$ のほうが線源の位置を表す指標としてより適切である。

さて、T_{src} のピークの平均値と分散から線源のコアの高さ位置は定量的に (7.69 ± 0.72) cm と求められる。この標準誤差値の意味は、サイズに広がりをもった線源の放射線強度に関する重心位置の不定性を表す量である。従って、標準誤差が線源のサイズを表す直接の指標とはならない。例えば今回の場合、線源であるマントルのサイズが 2 cm 程度と標準誤差よりも大きい。

様々なサイズの線源に対して同様に実験を行ったときに、標準誤差と線源のサイズについてどのような関係があるかについては興味のあるところである。

7.6.3 高さ較正した 3D-tracking

放射線全イベントに対するイメージは結果で示した通り、マントルの位置を与えるものとなった。第 5 章の 3D-tracking のように、本実験についてもいくつかイベントを抽出してプロットしたのが図 39 である。図を見ると、ほぼ水平方向に走るトラックが高さ 7 ～ 9 cm と線源のサイズと同じ 2 cm ほどの広がりをもって分布している。もっと小さいサイズの線源で同様に実験したときにその広がりが小さくなることを確かめれば、この広がりは線源のサイズに対応していることを示せるだろう。

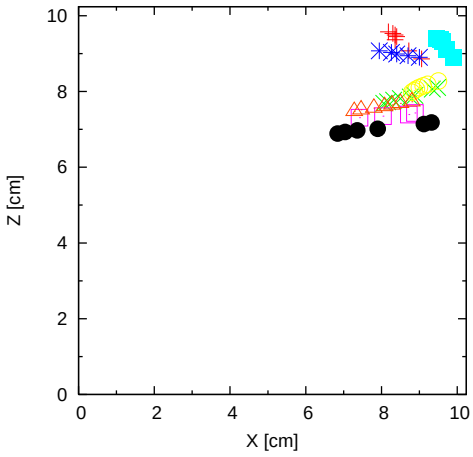


図 39　X-Z 投影面に対する放射線飛跡の様子。

7.6.4 シンチレーション光の検出効率

T_{lag}、あるいは t_1 は常に得られるものではなかった。これは、放射線入射時に発生するはずのシンチレーション光が必ずしも光電子増倍管で検出できないことを示す。放射線が通過した場所と、発生したシンチレーション光の検出効率との関係をみるため、1 イベントの飛跡の重心位置と、そこでの検出効率をカラーマップにしたものが図 41、図 42、図 43、図 44 である。

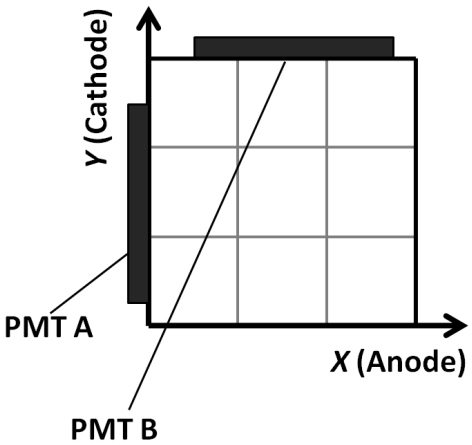


図 40　光電子増倍管の光電面の位置。

光電子増倍管の取り付け位置は図 40 のように 2 か所であり、図に従ってそれぞれ PMTA、PMTB とする。PMTA の検出効率は PMTA の光電面近くを放射線が走ったとき、PMTB の検出効率は PMTB の光電面近くを放射線が走ったときにそれぞれ高く、およそ 50 %である。また、各領域での A または B で検出された割合は 40 %〜60 %であった。この検出効率を上げるために有効と思われる方法を以下に列挙する。

- 光電子増倍管を取り付ける数を側面 2 か所でなく 4 か所に取り付ける。
- 光電子増倍管の感度波長と封入ガスのシンチレーション光の平均波長がなるべく一致するように、ガスまたは光電子増倍管を選ぶ。

1 つめについて、光電子増倍管の光電面に近い領域では遠い領域よりも検出効率が高いことから有効であるといえる。2 つめについて、今回の実験で用いた光電子増倍管は感度領域が 300 nm ∼ 650 nm であるのに対して、大気圧下での純粋 Ar のシンチレーション光の平均発光波長は 250 nm である。今回の実験で用いたガスはエタンが 10 %混合されたものであるため、発光波長は 250 nm からずれると予想されるが、それでもベストなコンディションとはいえないだろう。ここを改善し光電子増倍管の量子効率を稼ぐことで、シンチレーション光の検出効率上昇が期待される。

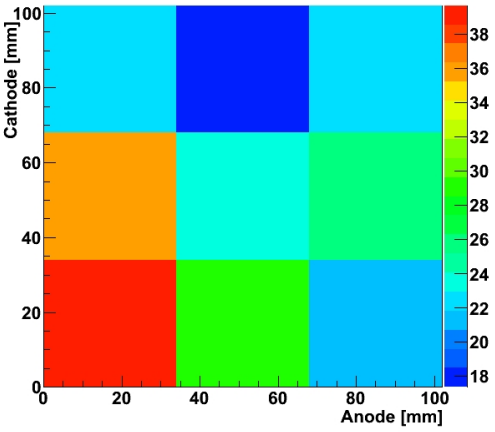


図 41　PMTA の検出効率

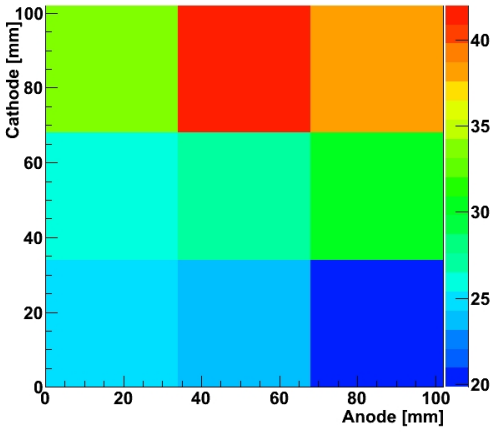


図 42　PMTB の検出効率

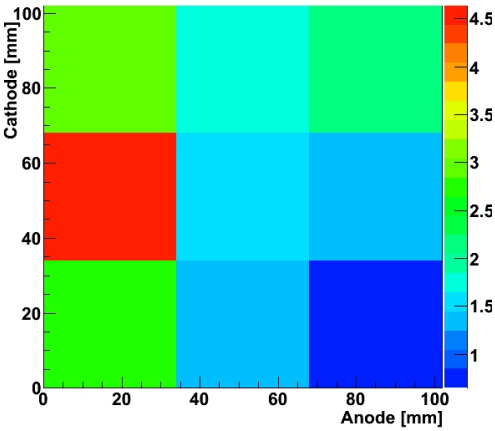


図 43　PMTA かつ PMTB の検出効率

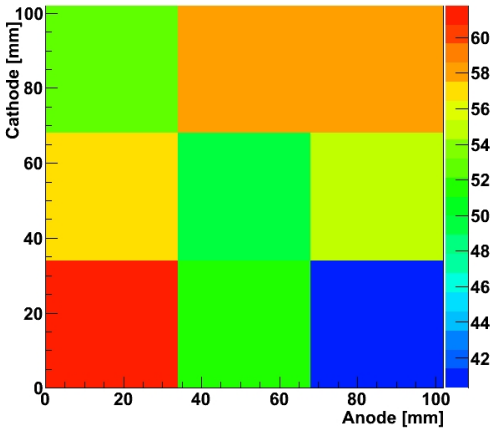


図 44　PMTA または PMTB の検出効率

8　まとめと展望

光電子増倍管と μ-PIC を併用することで、当初の目的である放射線の完全な三次元位置特定に成功した。ただ、第 6 章で書いた問題については、目的となる信号を効率よくとるという観点からも改善が必要である。また、特に第 7 章の本実験においては、今後この三次元測定を有効に使用していくためにも、比較実験などから一層の定量的評価が必要であろう。

9　参考文献

G.F.Knoll 木村逸郎/阪井英次訳 『放射線計測ハンドブック第 3 版』
日刊工業新聞社 2001 年
日本アイソトープ協会編 『アイソトープ手帳第 10 版』 丸善株式会社
2002 年
福士政広/齋藤秀敏/三枝健二/入船寅二/中谷儀一郎 『放射線基礎計測学』
医療科学社 2008 年

10 付録〜プログラムソースなど

10.1 第 2 章で用いた、波形を得るプログラムのメイン関数

```
//readFADC.cxx
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <ctime>
#include <string.h>
#include <ctype.h>
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <sys/param.h>
#include <vmedrv.h>
#include "rpv160sa.h"
#include "rpv130.h"
#include "MIudaq_val.h"  // define value
#define BASE 0x400000
#define VETO_ON 0xff //

using namespace std;

//variables for rpv130
char *rpv;
int rpvdev;
size_t rpvmap;
  //variables for rpv160
```

```
int         dev;
size_t      map;
char        *addr;

void intrpted(int rpvdev){
dis_intr_rpv130(rpvdev);

//VETO on
level_rpv130( (unsigned int)rpv, VETO_ON );
cerr << "triggered, ... ";
}


int main(int argc,char *argv[]){

time_t start_time, end_time, jikoku;
int mode=0;
int num = 100;
unsigned char ch[8][8192];

string fname = "";
ofstream fout;

int counter=0;
int evNum=10;
//unsigned int j;
unsigned short *ptr;

int ARGC_MODIFIER;
if(argc>=2)
fname = argv[1];
string::size_type tmp = fname.rfind(".dat\0");
if( tmp == string::npos )
{
//第 1 引数が?.dat でなかった.
fname = "tmp.raw.dat";
ARGC_MODIFIER = 0;
```

```
}
else
{
//第 1 引数が?.dat であった.
ARGC_MODIFIER = 1;
}


if(argc >= 2 + ARGC_MODIFIER){
if(atoi(argv[1 + ARGC_MODIFIER]) > 8192)
{
cerr << argv[1 + ARGC_MODIFIER] << "is resized to 8192.";
num = 8192;
}
else
{
num = atoi(argv[1 + ARGC_MODIFIER]);
}
}
if(argc >= 3 + ARGC_MODIFIER){
evNum = atoi(argv[2 + ARGC_MODIFIER]);
}
if(argc >= 4 + ARGC_MODIFIER){
mode = atoi(argv[3 + ARGC_MODIFIER]);
}


//rpv160
addr = (char *)init_fadc( &dev, BASE, &map );
set_fadc((unsigned int)addr);
clear_fadc((unsigned int)addr);
////rpv130
rpv = (char *)init_rpv130(&rpvdev, RPV130_BASE_ADD_DEFAULT, &rpvmap);
pulse_rpv130( (unsigned int)rpv, 0x00);
level_rpv130( (unsigned int)rpv, 0x22);
ptr = (unsigned short *)((unsigned int)rpv + 0xc);
*ptr = 0x03; //0000 0011
```

```
ptr = (unsigned short *)((unsigned int)rpv + 0xe);
*ptr = 0x03; //0000 0011

//set interrupt
intr_rpv130( rpvdev, intrpted );

//file output
fout.open(fname.c_str(),ios_base::app);
if(!fout){
cerr << "Cannot open file, \"" << fname << "\".\n";
return 1;
}

time(&start_time);

while(1){
//start_fadc((unsigned int)addr);
//cerr << "start!" << endl;
//usleep(100000);
//stop_fadc((unsigned int)addr);

//zero fadc memory
//for(j=0x0000; j <= 0xfffe; j=j+2){
// ptr = (unsigned short *)((unsigned int)addr + j);
// *ptr = 0xffff;
//}

//start fadc.
clear_fadc((unsigned int)addr);
start_fadc((unsigned int)addr);

usleep(4);

ena_intr_rpv130( rpvdev, (unsigned int)rpv );

//VETO off ==> permit stop-signal.
level_rpv130( (unsigned int)rpv, 0x00 );
```

```
//Wait for fadc stopping.
int ctr = data_num_fadc((unsigned int) addr);
//cerr  << "addres counter : " << ctr << endl;

//Write data to file.
if(ctr != -1){
cerr << counter << " success.\n";
for( int i = 0; i < 8; i ++ ){
read_fadc((unsigned int) addr, ctr, i, &ch[i][0], num);
}

for( int i = 0; i < num; i ++ ){
fout << i;
for( int j = 0; j < 8; j ++ ){
fout << "\t" << (int)ch[j][i];
  }
fout << "\n";
}
fout << "\n";
counter++;
}
else{
//cerr << counter << " failure. retry...\n";
//VETO on ==> forbid stop-signal.
level_rpv130( (unsigned int)rpv, VETO_ON );

}

time(&end_time);

//Break or continue.
if(mode == 0){
if(!(counter < evNum)) break;
}
else{
if( (end_time - start_time) >= evNum) break;
```

```
}

}
fout.close();

//Write log.
time(&jikoku);
fout.clear();
fout.open("info.log",ios_base::app);
fout << "date:" << ctime(&jikoku);
fout << "counts:" << counter << endl;
fout << "time:" << (end_time - start_time) << endl;
fout << "frequency:" << (double)counter/(end_time - start_time) << endl;
fout << endl;
fout.close();

//End devices.
dis_intr_rpv130(rpvdev);
level_rpv130( (unsigned int)rpv, 0x00 );
end_fadc( dev, map, (unsigned int) addr);
reset_intr_rpv130( rpvdev );
end_rpv130( rpvdev, rpvmap, (unsigned int)rpv );

cerr << "counts      : " << counter << endl;
cerr << "frequency   : " << (double)counter/(end_time - start_time) << endl;
cerr << "output file : " << fname << endl;
  return 0;
}
```

10.2 2D-imaging や 3D-tracking に必要な位置情報を得るプログラムのメイン関数

```
//main.cxx
#include <stdio.h>
#include <stdlib.h>
```

```
#include <fcntl.h>
#include <unistd.h>
#include <fstream.h>
#include <iomanip.h>
#include <string.h>
#include <ctype.h>
#include <errno.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <sys/param.h>
#include <vmedrv.h>

#include "memory.h"
#include "main.h"

using namespace std;

int take_data(){

  int flag = 0;

cerr << "take_data() called." << endl;
  // HE2687
 //reset_memory( (unsigned int)addr_he2687 );
  read_memory2( (unsigned int)addr_he2687, num_event );

  return flag;

};


void write_header( int depth ){
  time_t    now_t;
  struct tm *t;
```

```
  char      message[HEADERSIZE];

  now_t = time( NULL );
  t     = localtime( &now_t );

  sprintf( message, "%d/%d/%d/%d:%d:%d %d",
           t->tm_year + 1900,
           t->tm_mon  + 1,
           t->tm_mday,
           t->tm_hour,
           t->tm_min,
           t->tm_sec,
           depth
         );

  cerr << "Writing the header in the out file.\n"
       << "Header: " << message << endl;

  outf.write( message, HEADERSIZE );

}

long Now( void ){
  time_t timer, now;

  now = time( &timer );
  return (long)now;
}


int main( int argc, char **argv ){

  long   start_time, end_time, elapsed_time;
  double rate;
string dummy;

  // set default values
```

```
  num_event     = 10000;

  if(argc>1) num_event=atoi(argv[1]);
  cerr << "num of event" << num_event << endl;
  // HE2687
  addr_he2687=(char *)init_memory( &dev_he2687, BASE_HE2687, &map_he2687 );
  reset_memory( (unsigned)addr_he2687 );
  set_memory( (unsigned int)addr_he2687, num_event );
  abort_memory( (unsigned int)addr_he2687 );

  // DAQ start

  start_time = Now();
  cerr << "DAQ start...";

  start_memory( (unsigned int)addr_he2687 );
  take_data();

  end_time = Now();
  elapsed_time = end_time - start_time;
  rate = num_event / elapsed_time;

  cerr << "...done." << endl;
  cerr << "time: " << elapsed_time << " sec" << endl;
  cerr << "rate: " << rate         << " Hz"  << endl;

  return 0;
}
```

10.3 VME モジュールのインターフェイス関数群

10.3.1 rpv130.cxx

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#include <errno.h>
#include <unistd.h>
#include <fcntl.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <sys/param.h>
#include <vmedrv.h>

#include "rpv130.h"

static struct vmedrv_interrupt_property_t  pr;

int latch_1_rpv130(unsigned int addr){
  unsigned short  *ptr;
  int             data;

  ptr = (unsigned short *)(addr + 0x0);
  data = (int) *ptr;
  return data;
};

int latch_2_rpv130(unsigned int addr){
  unsigned short  *ptr;
  int             data;

  ptr = (unsigned short *)(addr + 0x2);
  data = (int) *ptr;
  return data;
};

int latch_ff_rpv130(unsigned int addr){
  unsigned short  *ptr;
  int             data;
```

```
  ptr = (unsigned short *)(addr + 0x4);
  data = (int) *ptr;
  return data;
};

int read_rpv130(unsigned int addr){
  unsigned short  *ptr;
  int             data;

  ptr = (unsigned short *)(addr + 0x6);
  data = (int) *ptr;
  return data;
};

void pulse_rpv130(unsigned int addr, int data){
  unsigned short  *ptr;
  ptr = (unsigned short *)(addr + 0x8);
  *ptr = (unsigned short) data;
};

void level_rpv130(unsigned int addr, int data){
  unsigned short  *ptr;

  ptr = (unsigned short *)(addr + 0xa);
  *ptr = (unsigned short) data;
};

void intr_rpv130(int dev, void (*sigh)(int)){
  pr.irq = LEVEL;
  pr.vector = VECTOR;
  pr.signal_id = SIGNAL_ID;

  signal(SIGNAL_ID, sigh);
  ioctl(dev, VMEDRV_IOC_REGISTER_INTERRUPT, &pr);
};

void ena_intr_rpv130(int dev, unsigned int addr){
```

```
  unsigned short  *ptr;

  ptr = (unsigned short *)(addr + 0xc);
  *ptr = 0x1b; //0001 1011
  ptr = (unsigned short *)(addr + 0xe);
  *ptr = 0x1b;
  ioctl(dev, VMEDRV_IOC_ENABLE_INTERRUPT);
};

void dis_intr_rpv130(int dev){
  ioctl(dev, VMEDRV_IOC_DISABLE_INTERRUPT);
};

void reset_intr_rpv130(int dev){
  ioctl(dev, VMEDRV_IOC_UNREGISTER_INTERRUPT, &pr);
  signal(SIGNAL_ID, SIG_DFL);
};

unsigned int init_rpv130(int *dev, long base, size_t *mapsize){
  int       offset, pagesize;
  char      *addr;

  printf("  RPV-130 base address   : 0x%lx\n", base);

  if((*dev = open("/dev/vmedrv16d16", O_RDWR)) == -1){
    perror(" open : /dev/vmedrv16d16");
    exit(1);
  }

  //  printf("aho");
  pagesize = getpagesize();
  *mapsize = 0x1000;
  offset = base % pagesize;
  base = base - offset;
  addr = (char *)mmap(0, *mapsize, PROT_READ|PROT_WRITE,
      MAP_SHARED, *dev, base);
  if(addr == MAP_FAILED){
```

```
    perror(" mmap : RPV130 failed.");
    exit(1);
  }
  addr += offset;

  printf("        pagesize         : 0x%x\n", pagesize);
  printf("        mapsize          : 0x%x\n", *mapsize);
  printf("  RPV130  addr on linux  : 0x%x\n", (unsigned int) addr);

  return (unsigned int)addr;
};

int end_rpv130(int dev, size_t mapsize, unsigned int addr){
    munmap((char *) addr, mapsize);
  close(dev);
  printf(" RPV130 : unmapped.\n");
  return 0;
};
```

10.3.2 rpv160sa.cxx

```
#include "rpv160sa.h"

int data_num_fadc(unsigned int addr){
  volatile unsigned short  *ptr;
  volatile int    rpv160_done_flag;
  int             ctr=0;
  int             limit_time=0;
  while(1){
    //printf(".");

    ptr = (unsigned short *)(addr + 0x30004);
    rpv160_done_flag = *ptr & 0x20;
    if(rpv160_done_flag){
      break;
     }
 //else if(rpv160_busy_flag == 0x40){
```

```
    else{
      usleep(100);
      limit_time++;
      if(limit_time>=100){
        ctr = -1;
        break;
       }

       //ptr = (unsigned short *)(addr + 0x30000);
       //fprintf(stderr, "address counter: %x \n", *ptr);
      //ctr = -1;
      //break;
     }
    //
    //
// break;
     // printf("limit break, ptr : %x \n", *ptr);
     // stop_fadc((unsigned int)addr);
    //usleep(1);
  }
  if(ctr != -1){
    ptr = (unsigned short *)(addr + 0x30000);
    ctr = *ptr;
  }
  //else{
 //   ctr = -1;
  //}
  return ctr;
};

void read_fadc(unsigned int addr, int ctr, int ch,
 unsigned char *data, int num){
  volatile unsigned short  *ptr;
  int i;
  num = num/2;
  for(i=0; i<num; i++){
    ptr = (unsigned short *)(addr +
```

```
    (i*2 + (ctr + 8191 - num)*2)%0x2000 + ch*0x2000);
    *data = (unsigned char)(*ptr >> 8);
    data++;
    *data = (unsigned char)(*ptr & 0x00ff);
    data++;
  }
  //if((num%2)==1){
  //  ptr = (unsigned short *)(addr + i*2 + ch*0x2000);
  //  *data = (unsigned char)(*ptr >> 8);
  //}
};

void start_fadc(unsigned int addr){
  volatile unsigned short  *ptr;

  ptr = (unsigned short *)(addr + 0x30008);
  *ptr = 0x0001;
};

void stop_fadc(unsigned int addr){
  volatile unsigned short  *ptr;

  ptr = (unsigned short *)(addr + 0x30008);
  *ptr = 0x0002;
};

void clear_fadc(unsigned int addr){
  volatile unsigned short  *ptr;

  ptr = (unsigned short *)(addr + 0x30008);
  *ptr = 0x0004;
};
```

```
void set_fadc(unsigned int addr){
  volatile unsigned short  *ptr;

  ptr = (unsigned short *)(addr + 0x20000);
  *ptr = 0x0003;
  ptr = (unsigned short *)(addr + 0x20002);
  *ptr = 0x0012;
  ptr = (unsigned short *)(addr + 0x20004);
  *ptr = 0x5678;
  ptr = (unsigned short *)(addr + 0x20006);
  *ptr = 0x0003;

  for(int i=0; i<8; i++){
    ptr = (unsigned short *)(addr + 0x21002 + i*0x200);
    *ptr = 0x0080;
  }
};

unsigned int init_fadc(int *dev, long base, size_t *mapsize){
  int         offset, pagesize;
  char        *addr;

  printf("    fadc base address    : 0x%lx\n", base);

  if((*dev = open("/dev/vmedrv32d16", O_RDWR)) == -1){
    perror(" open : /dev/vmedrv32d16 ");
    exit(1);
  }

  pagesize = getpagesize();
  *mapsize = 0x40000;
  offset = base % pagesize;
  base = base - offset;
  addr = (char *)mmap(0, *mapsize, PROT_READ|PROT_WRITE,
                      MAP_SHARED, *dev, base);
  if(addr == MAP_FAILED){
```

```
    perror(" mmap : fadc failed. ");
    exit(1);
  }
  addr += offset;

  printf("        pagesize         : 0x%x\n", pagesize);
  printf("        mapsize          : 0x%x\n", *mapsize);
  printf("  fadc address on linux  : 0x%x\n", (unsigned int) addr);

  return (unsigned int)addr;
};

int end_fadc(int dev, size_t mapsize, unsigned int addr){
  munmap((char *) addr, mapsize);
  close(dev);
  printf(" memory : unmapped.\n");
  return 0;

};
```

10.3.3 memory.cxx

```
#include <fstream.h>
#include <stdio.h>
#include <iomanip.h>
#include "memory.h"


int read_memory( unsigned int addr, unsigned int *data, int num ){
  int         *ptr;
  int         flag;
  unsigned    tmp;
  int         anode, cathode;

  for( int i = 0; i < 100000; i ++ ){
    ptr = (int *)( addr + 0x4 );
    flag = *ptr;
```

```
     if( flag == 0x0 ) break;
  }

  if( flag == 0x0 ){
    ptr = (int *)( addr + 0x0 );
    for( int i = 0; i < num; i ++ ){
      tmp = 0xffffffff - *ptr;
      *(data + i*sizeof(unsigned int)/HE2687DATASIZE) = tmp;

      cathode = ( tmp >>  9 ) & 0x1ff;
      anode   = ( tmp >> 18 ) & 0x1ff;
      cout << anode << " " << cathode << endl;
    }
  }

  return flag;
};

int read_memory2( unsigned int addr, int num ){
  int         *ptr;
  int         flag;
  unsigned    tmp;
  int         a_center, a_width;
  int         c_center, c_width;
  int clock;
  do{
  //  cerr << flag << endl;
    ptr = (int *)( addr + 0x4 );
    flag = *ptr;
  }while( flag > 0 );
 /*
  for( int i = 0; i < 100000; i ++ ){
    ptr = (int *)( addr + 0x4 );
    flag = *ptr;
     if( flag == 0x0 ) break;
  }
 */
```

```
 // flag = 0;

  if( flag == 0x0 ){

    for( int i = 0; i < num; i ++ ){
      ptr = (int *)( addr + 0x0 );
      //      tmp = 0xffffffff - (unsigned)*ptr;
      tmp = (unsigned)*ptr;

      //      c_width  = ( tmp       ) & 0x1f;
      c_center = ( tmp ) & 0x7ff;
      //      a_width  = ( tmp >> 15 ) & 0x1f;
      a_center = ( tmp >> 11 ) & 0x7ff;
      clock =  ( tmp >> 22 ) & 0x3ff;
      cout << a_center << "\t"
  //   << a_width  << \t
           << c_center << "\t"
   << clock << endl;
// << c_width  << endl;
    }
  }

  return flag;
};

/*
int read_memory( unsigned int addr, char *data, int num ){
  int         *ptr;
  int         flag;

  for( int i = 0; i < 100000; i ++ ){
    ptr = (int *)( addr + 0x4 );
    flag = *ptr;
     if( flag == 0x0 ) break;
  }

  if( flag == 0x0 ){
```

```
    ptr = (int *)( addr + 0x0 );
    for( int i = 0; i < num; i ++ ){
      *(data + (i*4) + 0) = ( 0xffffffff - *ptr ) & 0xff;
      *(data + (i*4) + 1) = ( ( 0xffffffff - *ptr ) >>  8 ) & 0xff;
      *(data + (i*4) + 2) = ( ( 0xffffffff - *ptr ) >> 16 ) & 0xff;
      *(data + (i*4) + 3) = ( ( 0xffffffff - *ptr ) >> 24 ) & 0xff;
    }
  }

  return flag;
};
*/

int start_memory(unsigned int addr){
  int         *ptr;
  int         flag;

  do{
    ptr = (int *)( addr + 0x4 );
    flag = *ptr;
  }while( flag );

  usleep( 50 );
  ptr = (int *)( addr + 0x0 );
  *ptr = 0x1;
  usleep( 100 );

  return flag;
};

void set_memory(unsigned int addr, int num){
  int         *ptr;

  ptr = (int *)(addr + 0x8);
  *ptr = num;
};
```

```
void abort_memory(unsigned int addr){
  int         *ptr;

  ptr = (int *)(addr + 0x0);
  *ptr = 0x0;
};

void reset_memory(unsigned int addr){
  int         *ptr;

  ptr = (int *)(addr + 0x0);
  *ptr = 0x2;
};

unsigned int init_memory(int *dev, long base, size_t *mapsize){
  int         offset, pagesize;
  char        *addr;

  cerr << "\t memory base address:     " << base << endl;

  if( (*dev = open("/dev/vmedrv32d32", O_RDWR)) == -1 ){
    perror( " open : /dev/vmedrv32d32 " );
    exit(1);
  }

  pagesize = getpagesize();
  *mapsize = 0x1000;
  offset = base % pagesize;
  base = base - offset;
  addr = (char *)mmap( 0, *mapsize, PROT_READ|PROT_WRITE,
                       MAP_SHARED, *dev, base );
  if( addr == MAP_FAILED ){
    perror( " mmap : memory board failed. " );
    exit(1);
  }
  addr += offset;
```

```
  cerr << "\t pagesize:                " << pagesize << endl;
  cerr << "\t mapsize:                 " << *mapsize << endl;
  cerr << "\t memory address on linux: " << (unsigned int)addr << endl;

  return (unsigned int)addr;
};

int end_memory(int dev, size_t mapsize, unsigned int addr){
  munmap((char *) addr, mapsize);
  close(dev);
  cerr << "Memory : unmapped." << endl;
  return 0;
};
```

10.4 第 5 章以降で用いた、波形と位置情報を同時に得るプログラムのメイン関数

```
//main_1.cxx
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
#include <unistd.h>
#include "vme_class.cpp"

using namespace std;

class rpv160 fadc;
class rpv130 rgstr;
class he2687 memory;

volatile int interrupt_flag=0;

void intrpted(int);

//#Main function
```

```
int main(int argc, char *argv[]){
//#Declare consts and variables.
const long base_fadc   = 0x400000;
const long base_rgstr  =   0x2000;
const long base_memory = 0x300000;

string file_fadc = "fadc.dat";
string file_he2687 = "he2687.dat";
ofstream fadc_out;
ofstream he2687_out;

const int depth_fadc = 1024+256;
const int depth_fadc_for_write = 1024;
unsigned char data_fadc[8][depth_fadc];

const int depth_he2687 = 1024;
const int he2687_data_size = 4;
unsigned int *data_he2687 = new unsigned [depth_he2687 * he2687_data_size];
unsigned int tmp;
int nani, min, max, ac_flag;

int num_event_cur = 0;
int num_event = 0;

int error_counter = 0;
bool success_flag;
/*
//#Read arguments.
if(argc >= 2)
file_fadc = argv[1];
if(argc >= 3)
file_he2687 = argv[2];
*/
//#Open devices and files.
fadc.open(base_fadc);
if(!fadc){
cerr << "Canoot open device, RPV160.\n";
```

```
return 0;
}
rgstr.open(base_rgstr);
if(!rgstr){
cerr << "Canoot open device, RPV130.\n";
return 0;
}
memory.open(base_memory);
if(!memory){
cerr << "Canoot open device, HE2687.\n";
return 0;
}
fadc_out.open(file_fadc.c_str(), ios_base::app);
if(!fadc_out){
cerr << "Cannot open file, \"" << file_fadc << "\".\n";
return 0;
}
he2687_out.open(file_he2687.c_str(), (ios_base::app)|(ios_base::binary));
if(!he2687_out){
cerr << "Cannot open file, \"" << file_he2687 << "\".\n";
return 0;
}

//#Initialize devices.
memory.reset();
memory.set(depth_he2687);
memory.abort();

fadc.set();
fadc.clear();
fadc.write(0);

rgstr.pulse(0);
rgstr.level(0);
rgstr.set_intr(intrpted);

/*
```

```
cerr << "Press Enter key, and you continue.";
string ss;
getline(cin, ss);
cin.clear();
*/

//#DAQ System
while(1){
usleep(1);
fadc.start();
while(1){
if(memory.busy() == false)
break;
}
usleep(50);
memory.start();
usleep(100);

rgstr.ena_intr();
rgstr.pulse(0x03); //#Send RUN signal into Encoder.

while(!interrupt_flag){
//#Waining for interrupt signal.
}
memory.reset();

success_flag = false;
for(int i=0 ; i<100 ; i++){
if( (memory.busy() == false)&&(fadc.done() == true) ){
success_flag = true;
break;
}
usleep(100);
}

if(success_flag == true){
for(int i=0; i<depth_he2687; i++)
```

```
data_he2687[i] = memory.read();
//memory.read(data_he2687, depth_he2687);
for(int ch=0 ; ch<8 ; ch++)
fadc.read(ch, data_fadc[ch], depth_fadc);
//#Write data to files.

for(int i=0; i<depth_he2687; i++){
tmp = data_he2687[i];
nani=      tmp         & 0x7ff;
min =     (tmp >> 11)  & 0x3ff;
max =     (tmp >> 21)  & 0x3ff;
ac_flag = (tmp >> 31)  & 0x1;
}
he2687_out.write((char *)data_he2687, depth_he2687*he2687_data_size);
for(int i=0 ; i<depth_fadc_for_write ; i++){
fadc_out << i;
for(int ch=0 ; ch<8 ; ch++){
fadc_out << "\t" << (int)data_fadc[ch][i];
}
fadc_out << endl;
}
fadc_out << endl;
//#Increment success event number.
num_event_cur++;
}
else{
memory.reset();
usleep(100);
while(1){
if(memory.busy() == false)
break;
}
error_counter++;
}
(success_flag == true) ? (cerr << "success!\n") : (cerr << "falure.\n");

if(num_event_cur >= num_event)
```

```
break;

fadc.stop();
fadc.clear();

memory.reset();
memory.set(depth_he2687);
memory.abort();

}//#Back to loop.

cerr << "Break!\n";
cerr << "error times : " << error_counter << endl;

//#Close devices and files.
rgstr.dis_intr();
rgstr.unset_intr();
fadc_out.close();
he2687_out.close();
fadc.close();
rgstr.close();
memory.close();

return 0;
}

void intrpted(int){
rgstr.dis_intr();
interrupt_flag = 1;
//cerr << "itrpted.\n" ;
}
```

10.5 VME モジュールのインターフェイス関数群 (クラス版)

```
//vme_class.cpp
#include <unistd.h>
```

```
#include <fcntl.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <sys/param.h>
#include <vmedrv.h>

using namespace std;

namespace vme{
//error information
const int open_error = 1;
const int map_error = 2;

//HE2687 information
const int HE2687DATASIZE = 4;

//for rpv130 interrupt
static struct vmedrv_interrupt_property_t  pr;
const int LEVEL = 3;
const int VECTOR = 0xfff0;
const int SIGNAL_ID = SIGINT;
}

//base class
class vme_module{
protected:
unsigned int addr;
int dev;
size_t mapsize;
char *dev_file;
int error_code;
public:
vme_module();
int open(long base);
```

```
int close();
int fail();
unsigned int get_addr();
bool operator!();
};

//RPV-160
class rpv160 : public vme_module{
private:
unsigned short ctr;
public:
rpv160();
void set();
void clear(); //software clear.
void start(); //software start.
void stop(); //software stop.
int get_ready(int limit_time);
unsigned short addrctr(); //return Address counter.
bool done(); //return "done" flag, in CRS1.
bool busy(); //return "busy" flag, in CRS1.
int read(const int ch, unsigned char *data, const int data_num);
int write(const char data);
};

//RPV-130
class rpv130 : public vme_module{
private:

public:
rpv130();
int latch_1();
int latch_2();
int latch_ff();
int read();
void pulse(const int data);
void level(const int data);
void set_intr(void (*sigh)(int));
```

```
void unset_intr();
void ena_intr();
void dis_intr();
unsigned short csr1();
unsigned short csr2();
};

//HE2687
class he2687 : public vme_module{
private:

public:
he2687();
void reset();
void abort();
void set(const int num); // set event number
void start();
void read(unsigned int *data, const int num);
unsigned int read();
bool busy();
};

//###############################################

vme_module::vme_module(){
vme_module::error_code = -1;
}

int vme_module::open(long base){
int offset, pagesize;
char *addr;
if((vme_module::dev = ::open(dev_file, O_RDWR)) == -1){
vme_module::error_code = vme::open_error;
return 1;//error
}
pagesize = getpagesize();
offset = base % pagesize;
```

```
base = base - offset;
addr = (char *)mmap(0, vme_module::mapsize,
 PROT_READ|PROT_WRITE, MAP_SHARED, vme_module::dev, base);
if(addr == MAP_FAILED){
error_code = vme::map_error;
return 1;//error
}
addr += offset;
vme_module::addr = (unsigned int)addr;
vme_module::error_code = 0;
return 0;
}

int vme_module::close(){
munmap((char *)vme_module::addr, vme_module::mapsize);
::close(vme_module::dev);
return 0;
}

int vme_module::fail(){
return vme_module::error_code;
}

unsigned int vme_module::get_addr(){
return vme_module::addr;
}

bool vme_module::operator!(){
bool ret;
(vme_module::error_code != 0) ? (ret = true) : (ret = false);
return ret;
}

//RPV-160
rpv160::rpv160(){
rpv160::dev_file = "/dev/vmedrv32d16";
rpv160::mapsize = 0x40000;
```

```
}

void rpv160::set(){
unsigned short *ptr;
ptr = (unsigned short *)(rpv160::addr + 0x20000);
*ptr = 0x0003;
ptr = (unsigned short *)(rpv160::addr + 0x20002);
*ptr = 0x0012;
ptr = (unsigned short *)(rpv160::addr + 0x20004);
*ptr = 0x5678;
ptr = (unsigned short *)(rpv160::addr + 0x20006);
*ptr = 0x0003;

for(int i=0; i<8; i++){
ptr = (unsigned short *)(rpv160::addr + 0x21002 + i*0x200);
*ptr = 0x0080;
}
}

void rpv160::clear(){
unsigned short *ptr;
ptr = (unsigned short *)(rpv160::addr + 0x30008);
*ptr = 0x0004;
}

void rpv160::start(){
unsigned short *ptr;
ptr = (unsigned short *)(rpv160::addr + 0x30008);
*ptr = 0x0001;
}

void rpv160::stop(){
unsigned short *ptr;
ptr = (unsigned short *)(rpv160::addr + 0x30008);
*ptr = 0x0002;
}
```

```
int rpv160::get_ready(int limit_time = 0){
volatile unsigned short *ptr;
int flag;
while(1){
usleep(10);
ptr = (unsigned short *)(rpv160::addr + 0x30004);
flag = *ptr & (0x1 << 5);
if(flag){
ptr = (unsigned short *)(rpv160::addr + 0x30000);
rpv160::ctr = *ptr;
break;
}
limit_time--;
if(limit_time <= 0){
rpv160::ctr = (unsigned short)(-1);
}
}
return rpv160::ctr;
}

unsigned short rpv160::addrctr(){
unsigned short *ptr;
ptr = (unsigned short *)(rpv160::addr + 0x30000);
return *ptr;
}

bool rpv160::done(){
bool ret = false;
unsigned short *ptr;
ptr = (unsigned short *)(rpv160::addr + 0x30004);
if( (*ptr & (1<<5)) )
ret = true;
return ret;
}

bool rpv160::busy(){
bool ret = false;
```

```
unsigned short *ptr;
ptr = (unsigned short *)(rpv160::addr + 0x30004);
if( (*ptr & (1<<6)) )
ret = true;
return ret;
}

int rpv160::read(const int ch, unsigned char *data, const int data_num){
if ( (ch>7) || (ch<0) )
return -1;
int half_data_num = data_num/2;
volatile unsigned short *ptr;
for(int i=0 ; i<half_data_num ; i++){
ptr = (unsigned short *)( rpv160::addr + 0x2000 * ch +
 ( (2*((int)addrctr() - 1 + i - half_data_num )) & 0x1fff) );
*data = (unsigned char)(*ptr >> 8);
  data++;
  *data = (unsigned char)(*ptr & 0x00ff);
  data++;
  }
return 0;
}

int rpv160::write(const char data){
unsigned short *ptr;
for(int i=0; i<(8192/2); i++){
ptr = (unsigned short *)( rpv160::addr + i*2 );
*ptr = ((unsigned short)data << 8) || ((unsigned short)data);
}
return 0;
}

//RPV-130
rpv130::rpv130(){
rpv130::dev_file = "/dev/vmedrv16d16";
rpv130::mapsize = 0x1000;
}
```

```
int rpv130::latch_1(){
unsigned short  *ptr;
ptr = (unsigned short *)(rpv130::addr + 0x00);
return (int) *ptr;
}

int rpv130::latch_2(){
unsigned short  *ptr;
ptr = (unsigned short *)(rpv130::addr + 0x02);
return (int) *ptr;
}

int rpv130::latch_ff(){
unsigned short  *ptr;
ptr = (unsigned short *)(rpv130::addr + 0x04);
return (int) *ptr;
}

int rpv130::read(){
unsigned short  *ptr;
ptr = (unsigned short *)(rpv130::addr + 0x06);
return (int) *ptr;
}

void rpv130::pulse(const int data){
unsigned short  *ptr;
ptr = (unsigned short *)(rpv130::addr + 0x08);
*ptr = (unsigned short) data;
}

void rpv130::level(const int data){
unsigned short  *ptr;
ptr = (unsigned short *)(rpv130::addr + 0x0a);
*ptr = (unsigned short) data;
}
```

```
void rpv130::set_intr(void (*sigh)(int)){
vme::pr.irq = vme::LEVEL;
vme::pr.vector = vme::VECTOR;
vme::pr.signal_id = vme::SIGNAL_ID;

signal(vme::SIGNAL_ID, sigh);
ioctl(rpv130::dev, VMEDRV_IOC_REGISTER_INTERRUPT, &vme::pr);
}

void rpv130::unset_intr(){
  ioctl(rpv130::dev, VMEDRV_IOC_UNREGISTER_INTERRUPT, &vme::pr);
  signal(vme::SIGNAL_ID, SIG_DFL);
}

void rpv130::ena_intr(){
unsigned short  *ptr;
ptr = (unsigned short *)(rpv130::addr + 0xc);
*ptr = 0x1b;
ptr = (unsigned short *)(rpv130::addr + 0xe);
*ptr = 0x1b;
ioctl(rpv130::dev, VMEDRV_IOC_ENABLE_INTERRUPT);
}

void rpv130::dis_intr(){
  ioctl(rpv130::dev, VMEDRV_IOC_DISABLE_INTERRUPT);
}

unsigned short rpv130::csr1(){
unsigned short  *ptr;
ptr = (unsigned short *)(rpv130::addr + 0xc);
return *ptr;
}

unsigned short rpv130::csr2(){
unsigned short  *ptr;
ptr = (unsigned short *)(rpv130::addr + 0xe);
return *ptr;
```

```
}

//HE2687
he2687::he2687(){
he2687::dev_file = "/dev/vmedrv32d32";
he2687::mapsize = 0x1000;
}

void he2687::reset(){
int *ptr;
ptr = (int *)(he2687::addr + 0x0);
*ptr = 0x2;
}

void he2687::start(){
int *ptr;
ptr = (int *)(he2687::addr + 0x0);
*ptr = 0x1;
}

void he2687::abort(){
int *ptr;
ptr = (int *)(he2687::addr + 0x0);
*ptr = 0x0;
}

void he2687::set(const int num){
int *ptr;
ptr = (int *)(he2687::addr + 0x8);
*ptr = num;
}

bool he2687::busy(){
bool ret;
int *ptr;
ptr = (int *)(he2687::addr + 0x4);
(*ptr != 0) ? (ret = true) : (ret = false);
```

```
return ret;
}

void he2687::read(unsigned int *data, const int num){
int *ptr;
unsigned int tmp;
for( int i = 0; i < num; i ++ ){
ptr = (int *)( he2687::addr + 0x0 );
tmp = (unsigned int)*ptr;
*(data + i*sizeof(unsigned int)/vme::HE2687DATASIZE) = tmp;
//data[i] = tmp;
}
}

unsigned int he2687::read(){
int *ptr;
unsigned int tmp;
ptr = (int *)( he2687::addr + 0x0 );
tmp = (unsigned int)*ptr;
return tmp;
}
```

10.6 デコーダ

```
//analyzer.cxx
/*data file format*/
//#index x y clock xmin xmax ymin ymax ?th_hit

#include <iostream>
#include <fstream>
#include <string>

using namespace std;

const int depth_he2687 = 1024;
```

```
class data{
public:
int id[depth_he2687];
int clock[depth_he2687];
double x[depth_he2687];
double y[depth_he2687];
int xmin[depth_he2687];
int xmax[depth_he2687];
int ymin[depth_he2687];
int ymax[depth_he2687];
int nhit;
} data;

int main(int argc, char *argv[]){
string file_fadc = "fadc.dat";
string file_he2687 = "he2687.dat";
ifstream fadc_in;
ifstream he2687_in;

int index=1;
bool status_flag;

const int he2687_data_size = 4;
unsigned int *data_he2687 = new unsigned [depth_he2687 * he2687_data_size];
int data_num_he2687;
unsigned int tmp;

he2687_in.open(file_he2687.c_str(), ios_base::binary);
if(!he2687_in){
cerr << "Cannot open file, \"" << file_he2687 << "\".\n";
return 0;
}

cout << "#index x y clock xmin xmax ymin ymax ?th_hit\n";
while(!he2687_in.eof()){
cerr << "loop in\n";
data.nhit = 0;
```

```
status_flag = true;
he2687_in.read((char *)data_he2687, depth_he2687 * he2687_data_size);
for(int i=0 ; i<(depth_he2687/2); i++){
tmp = data_he2687[2*i];
if(tmp == 0xffffffff){
status_flag = false;
break;
}
data.id[i]    =  tmp         & 0x7ff;
data.xmin[i]  = (tmp >> 11)  & 0x3ff;
data.xmax[i]  = (tmp >> 21)  & 0x3ff;
tmp = data_he2687[2*i+1];
if(tmp == 0xffffffff){
status_flag = false;
break;
}
data.clock[i] =  tmp         & 0x7ff;
data.ymin[i]  = (tmp >> 11)  & 0x3ff;
data.ymax[i]  = (tmp >> 21)  & 0x3ff;
data.x[i] = (data.xmin[i]+data.xmax[i])/2.0;
data.y[i] = (data.ymin[i]+data.ymax[i])/2.0;
if(i!=0){
if(data.id[i] != data.id[i-1]){
status_flag = false;
break;
}
}
cout << index << "\t" << data.x[i] << "\t" << data.y[i] << "\t";
cout << data.clock[i] << "\t";
cout << data.xmin[i] << "\t" << data.xmax[i] << "\t";
cout << data.ymin[i] << "\t" << data.ymax[i] << "\t";
cout << (i+1) << endl;
data.nhit++;
}
if(status_flag == false)
break;
index++;
```

```
}
he2687_in.close();
return 0;
}
```

10.7 解析用プログラム

```
//parameter_getter4.cxx
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <math.h>

const double xy_factor = 0.4; //# 0.4 [mm/ch]
const double z_factor = 42.0 * 0.01; //# V[mm/us] * 0.010[us] *1[/ch]

int get_AVE(const char fname[] , const int iscope_i , const int iscope_f ,
 double *data_ave );
int get_AREA( const char fin_name[] , int *area_p,
 int *area_pm, double *Threshold );
int get_PATH( const char fin_name[] , int &nhit,
 double &path, double &length );

using namespace std;

int main(const int argc,char *argv[]){

//--Declare Consts.
const int noise_start_index=0;
const int noise_end_index=450;
//--Declare Variables.
double bl[8];
int area_p[8];
int area_pm[8];
int nhit;
```

```
double path, length;


ostringstream strout;
string fin_name;
string fout_name;
fstream fout;
string file_number = argv[1];

strout.clear();
strout.str("");
strout << "fadc_" << file_number << ".dat";
fin_name = strout.str(); //# "fadc_??.dat"
if(get_AVE(fin_name.c_str(), noise_start_index, noise_end_index, bl) == -1)
return 0;
get_AREA(fin_name.c_str(), area_p, area_pm, bl);

strout.clear();
strout.str("");
strout << "tpc_" << file_number << ".dat";
fin_name = strout.str(); //# "tpc_??.dat"
if(get_PATH( fin_name.c_str() , nhit, path, length ) == -1)
return 0;

strout.clear();
strout.str("");
strout << "para4_" << file_number << ".dat";
fout_name = strout.str(); //# "para_??.dat"

fout.open(fout_name.c_str(), (ios::out));
if(!fout){
cerr << "Error, cannot open file \"" << fout_name << "\".\n";
return 0;
}
fout << nhit << "\t" << path << "\t" << length;
for(int i=0; i<8; i++)
fout << "\t" << area_p[i];
```

```
for(int i=0; i<8; i++)
fout << "\t" << area_pm[i];
for(int i=0; i<8; i++)
fout << "\t" << bl[i];
fout << endl;
fout.close();
cerr << "ok.\n";

return 0;
}



int get_AVE(const char fname[] , const int iscope_i , const int iscope_f ,
 double *data_ave ){
int index;
int data[8];
int sum[8];
for(int i=0; i<8; i++)
sum[i] = 0;
int ave_factor=0;
string ss;
ifstream fin;
istringstream strin;

fin.open(fname);
if(!fin)
return -1;

while(getline(fin, ss)){
strin.clear();
strin.str(ss);
if( !(strin >> index) )
{
continue;
}
if( ( iscope_i <= index ) && ( index <= iscope_f ) )
```

```
{
for(int i=0 ; i<8 ; i++){
strin >> data[i];
sum[i] += data[i];
}
ave_factor++;
}
}
fin.close();
for(int i=0 ; i<8 ; i++){
data_ave[i] = ((double)sum[i]) / ave_factor;
}
return 0;
}



int get_AREA( const char fin_name[] , int *area_p, int *area_pm,
 double *Threshold ){
int index;
int data[8][1024];
double d_area_p[8], d_area_pm[8];
double geta = 10;
double geta_bl[8];
ifstream fin;
istringstream strin;
string ss;
int nhit = 0;
int nhit_max = 0;
int tmp_start, tmp_end, start[8], end[8];

for(int i=0; i<8; i++){
area_p[i] = 0;
area_pm[i] = 0;
d_area_p[i] = 0;
d_area_pm[i] = 0;
geta_bl[i] = Threshold[i] + geta;
```

```
}
//#read data
fin.clear();
fin.open(fin_name);
if(!fin)
return -1;
while(getline(fin,ss)){
strin.clear();
strin.str(ss);
if(!(strin >> index))
continue;

for ( int i=0 ; i<8 ; i++ ){
strin >> data[i][index];
}
}
fin.close();

//#
for (int i=0; i<8; i++){
nhit = nhit_max = 0;
for(int t=450; t<700; t++){
if(data[i][t] > geta_bl[i]){
if(nhit == 0)
tmp_start = t;
nhit++;
tmp_end = t;
if(nhit > nhit_max){
start[i] = tmp_start;
end[i] = tmp_end;
}
}
else{
nhit = 0;
}
}
//#
```

```
for(int t=end[i]; t<700; t++){
if(data[i][t] < Threshold[i]){
end[i] = t;
break;
}
}
for(int t=start[i]; t>=450; t--){
if(data[i][t] < Threshold[i]){
start[i] = t;
break;
}
}
//#
for(int t=start[i]; t<=end[i]; t++){
if (data[i][t] > Threshold[i])
d_area_p[i] += data[i][t] - Threshold[i];
d_area_pm[i] += data[i][t] - Threshold[i];
}
}


for ( int i=0 ; i<8 ; i++ ){
area_p[i] = (int)d_area_p[i];
area_pm[i] = (int)d_area_pm[i];
}
return 0;
}


int get_PATH( const char fin_name[] , int &nhit, double &path,
 double &length ){
const int nhit_max = 100;
int index;
double xyz[nhit_max][3]; //# xyz[ nhit ][ xyz ]
ifstream fin;
istringstream strin;
string ss;
```

```
bool from_src;
int ret=0;

nhit = 0;
path=0.0;
length=0.0;

fin.clear();
fin.open(fin_name);
if(!fin)
return -1;
from_src = false;
while(getline(fin,ss)){
strin.clear();
strin.str(ss);
if(ss[0] == '#')
continue;
if(!(strin >> index))
continue;
strin >> xyz[nhit][0] >> xyz[nhit][1] >> xyz[nhit][2];
nhit++;
}
fin.close();
for(int i=1; i<nhit; i++){
if( (xyz[i][2] - xyz[i-1][2]) > 10 ) //CUT DATA ON CONDITION ABOUT Z(CLOCK)
ret = -1;
}
for(int i=0; i<nhit; i++){
if( (xyz[i][0] > 224) && (xyz[i][1] < 32) )
from_src = true;
xyz[i][0] = xyz[i][0] * xy_factor;
xyz[i][1] = xyz[i][1] * xy_factor;
xyz[i][2] = xyz[i][2] * z_factor;
}
if(nhit >= 2){
for(int i=1; i<nhit; i++)
path += sqrt( pow((xyz[i][0]-xyz[i-1][0]), 2.0) +
```

```
pow((xyz[i][1]-xyz[i-1][1]), 2.0) + pow((xyz[i][2]-xyz[i-1][2]), 2.0) );
length = sqrt( pow((xyz[nhit-1][0]-xyz[0][0]), 2.0) +
pow((xyz[nhit-1][1]-xyz[0][1]), 2.0) +
 pow((xyz[nhit-1][2]-xyz[0][2]), 2.0) );
}
//if(from_src == false)
// ret = -1;  //CUT DATA ON CONDITION ABOUT XY
return ret;
}
```

10.8 メイン関数から解析までを行うシェルスクリプト

```
#/bin/sh
##main.sh

exe_main="nice /home/p6/src/main/main_1"
exe_analyzer="nice /home/p6/src/main/analyzer"
exe_para="nice /home/p6/src/main/parameter_getter4"
dev_null="/dev/null"

input_file_fadc="fadc.dat"
input_file_tpc="he2687.dat"
output_file_log="log.dat"

hit_low=2
hit_low=‘expr ${hit_low} \* 2 + 1‘

counter=1
loop_time=1

if [ $# -ge 1 ]; then
loop_time=$1
fi

start_date=‘date ’+%Y %m %d %T’‘
start_t=‘date ’+%s’‘
```

```
while [ $counter -le $loop_time ]; do
rm ${input_file_fadc} 2> ${dev_null}
rm ${input_file_tpc}  2> ${dev_null}
${exe_main} 2> ${dev_null}
output_file_fadc="fadc_${counter}.dat"
mv ${input_file_fadc} ${output_file_fadc}
output_file_tpc="tpc_${counter}.dat"
${exe_analyzer} > ${output_file_tpc} 2> ${dev_null}
rm ${input_file_tpc}
${exe_para} ${counter} 2> ${dev_null}
output_file_para="para_${counter}.dat"
hit=‘wc -l ${output_file_tpc} | cut -c1‘
if [ ! -f ${output_file_para} -o ${hit} -lt ${hit_low} ]; then
rm ${output_file_fadc} 2> ${dev_null}
rm ${output_file_tpc} 2> ${dev_null}
rm ${output_file_para} 2> ${dev_null}
else
echo "No.${counter} data derived."
counter=‘expr ${counter} + 1‘
fi
done
stop_t=‘date ’+%s’‘
span_t=‘expr ${stop_t} - ${start_t}‘
frequency=‘echo "scale=4; ${loop_time} / ${span_t}" | bc‘

cat <<- EOF > ${output_file_log}
date:${start_date}
event_count:${loop_time}
measurement_time:${span_t}
frequency:${frequency}
lower_limit_hit:${hit_low}
EOF

echo "completed!"
```

10.9 T_{lag} を求めるプログラム

```
//scinti_time.cxx
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <math.h>

int get_AVE(const char fname[] ,
 const int iscope_i , const int iscope_f , double *data_ave );
int get_clocklag( const char fin_name[] ,
 int *start, int *end, double *Threshold );


using namespace std;

int main(const int argc,char *argv[]){
if(argc < 2)
return -1;

//--Declare Consts.
const int noise_start_index=0;
const int noise_end_index=450;
//--Declare Variables.
double bl[2];
int start[2], end[2];
int status = 0;
int lag = -1;

ostringstream strout;
string fin_name;
string fout_name;
fstream fout;
string file_number = argv[1];
```

```
strout.clear();
strout.str("");
strout << "fadc_" << file_number << ".dat";
fin_name = strout.str(); //# "fadc_??.dat"
if(get_AVE(fin_name.c_str(), noise_start_index, noise_end_index, bl) == -1)
return 0;
get_clocklag(fin_name.c_str(), start, end, bl);

if(start[0] != -1)
status += 1;
if(start[1] != -1)
status += 2;

if(status == 1)
lag = end[0] - start[0];
if(status == 2)
lag = end[1] - start[1];
if(status == 3){
if( (end[0] - start[0] - end[1] + start[1] ) == 0){
lag = end[0] - start[0];
}
else{
if((end[0] - start[0])<(end[1] - start[1])){
lag = end[0] - start[0];
status = 1;
}
else{
lag = end[1] - start[1];
status = 2;
}
}
}

strout.clear();
strout.str("");
strout << "scinti_" << file_number << ".dat";
fout_name = strout.str(); //# "scinti_??.dat"
```

```
fout.open(fout_name.c_str(), (ios::out));
if(!fout){
cerr << "Error, cannot open file \"" << fout_name << "\".\n";
return 0;
}


fout << status << "\t" << lag;
for(int i=0; i<2; i++)
fout << "\t" << start[i];
fout << endl;
fout.close();
cerr << "ok.\n";

return 0;
}



int get_AVE(const char fname[] , const int iscope_i , const int iscope_f ,
 double *data_ave ){
int index;
int data[2];
int sum[2];
for(int i=0; i<2; i++)
sum[i] = 0;
int ave_factor=0;
string ss;
ifstream fin;
istringstream strin;

fin.open(fname);
if(!fin)
return -1;

while(getline(fin, ss)){
```

```
strin.clear();
strin.str(ss);
if( !(strin >> index) )
{
continue;
}
if( ( iscope_i <= index ) && ( index <= iscope_f ) )
{
for(int i=0 ; i<2 ; i++){
strin >> data[i];
sum[i] += data[i];
}
ave_factor++;
}
}
fin.close();
for(int i=0 ; i<2 ; i++){
data_ave[i] = ((double)sum[i]) / ave_factor;
}
return 0;
}



int get_clocklag( const char fin_name[] ,
 int *start, int *end, double *Threshold ){
int index;
int data[2][1024];
double geta = 3;
double geta_bl[2];
ifstream fin;
istringstream strin;
string ss;


for(int i=0; i<2; i++){
geta_bl[i] = Threshold[i] + geta;
```

```
start[i] = -1;
end[i] = -1;
}
//#read data
fin.clear();
fin.open(fin_name);
if(!fin)
return -1;
while(getline(fin,ss)){
strin.clear();
strin.str(ss);
if(!(strin >> index))
continue;

for ( int i=0 ; i<2 ; i++ ){
strin >> data[i][index];
}
}
fin.close();

//#
for (int i=0; i<2; i++){
for(int t=450; t<500; t++){
if(data[i][t] > geta_bl[i]){
end[i] = t;
break;
}
}
if(end[i] != -1){
for(int t=end[i]-1; t>(end[i]-250); t--){
if(data[i][t] > geta_bl[i]){
start[i] = t;
break;
}
}
}
}
```

```
return 0;
}
```

10.10 T_{src} を求めるプログラム

```
//Tsrc.cxx
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include <math.h>

using namespace std;

int main(){;

vector<double> data(10);
double d_tmp;
double Tsrc, Tlag, tfirst, tlast;
double p[3][3];
double df, dl;
enum p_1{
SRC,
FST,
LST
};
enum p_2{
X,
Y,
Z
};
bool flag;
int file_number;
string ss;
string infile;
```

```
string outfile;
ifstream fin, srcfin;
ofstream fout;
istringstream strin;
ostringstream strout;

p[SRC][X] = 99;
p[SRC][Y] =  9;
p[SRC][Z] = 77;

strout.clear();
strout.str("");
strout << "Tsrc.dat";
outfile = strout.str();
fout.clear();
fout.open(outfile.c_str());

srcfin.open("src.dat", ios::binary);
if(!srcfin){
cerr << "error: src.dat does not exist.\n";
return 0;
}
while(1){
srcfin.read( (char *)(&file_number), sizeof(int) );
if(!srcfin.good())
break;

strout.clear();
strout.str("");
strout << "scinti_" << file_number << ".dat";
infile = strout.str();
fin.clear();
fin.open(infile.c_str());
if(!fin)
continue;
flag=false;
while(getline(fin,ss)){
```

```
strin.clear();
strin.str(ss);
if(ss[0] == '#')
continue;
data.clear();
for(int i=0; i<2; i++){
if(!(strin >> d_tmp))
break;
data.push_back(d_tmp);
}
if(data.size() < 2)
continue;
if(data[0] != 0.0){ //#data[0]: status
flag = true;
Tlag = data[1];
}
}
fin.close();

if(flag == false)
continue;

strout.clear();
strout.str("");
strout << "track_" << file_number << ".dat";
infile = strout.str();
fin.clear();
fin.open(infile.c_str());
if(!fin)
continue;
flag=true;
while(getline(fin,ss)){
strin.clear();
strin.str(ss);
if(ss[0] == '#')
continue;
data.clear();
```

```
for(int i=0; i<3; i++){
if(!(strin >> d_tmp))
break;
data.push_back(d_tmp);
}
if(data.size() < 3)
continue;
if(flag == true){
p[FST][X] = data[0];
p[FST][Y] = data[1];
p[FST][Z] = data[2];
flag = false;
}
else{
p[LST][X] = data[0];
p[LST][Y] = data[1];
p[LST][Z] = data[2];
}
}
fin.close();

d_tmp = 0.0;
for(int i=X; i<=Z; i++){
d_tmp += pow( p[FST][i] - p[SRC][i] , 2.0 );
}
df = d_tmp;
d_tmp = 0.0;
for(int i=X; i<=Z; i++){
d_tmp += pow( p[LST][i] - p[SRC][i] , 2.0 );
}
dl = d_tmp;

if(df < dl){
Tsrc = Tlag;
}
else{//#if(df <= dl)
strout.clear();
```

```
strout.str("");
strout << "tpc_" << file_number << ".dat";
infile = strout.str();
fin.clear();
fin.open(infile.c_str());
if(!fin)
continue;
flag=true;
while(getline(fin,ss)){
strin.clear();
strin.str(ss);
if(ss[0] == '#')
continue;
data.clear();
for(int i=0; i<4; i++){
if(!(strin >> d_tmp))
break;
data.push_back(d_tmp);
}
if(data.size() < 4)
continue;
if(flag == true){
flag = false;
tfirst = data[3];
}
else{
tlast = data[3];
}
}
fin.close();
Tsrc = Tlag + tlast - tfirst;
}//#endif(df <= dl)
fout << file_number << "\t" << Tsrc << endl;
}
fout.close();
return 0;
}
```

謝辞

今回の課題研究においては、宇宙線研究室の皆様に数多くの助言をいただきました。特に、助教の身内賢太朗先生には実験の進め方からトラブルの対処法、発表資料の作り方など、はじめから最後まで適切なタイミングで適切な助言をいただき、なんとか結果を出すにいたりました。また、TA として上野一樹さんには、実験装置の詳しい説明や時折の冷やかしなどにより、より楽しく有意義な実験環境作りにご助力いただきました。以上の皆様のおかげで無事、課題研究をやり終えることができました。はなはだ簡単ではありますが、ここに感謝の意を記したいと思います。本当にありがとうございました。

最後に、細やかな気配りで場を和やかにしてくれた河手さん、度々みかんを分け与えてくれた蔵本君、実験に対するストイックな姿勢で我々を励ましてくれた今野君、いつも前向きな態度で我々の不安を解きほぐしてくれた福間君、みんなのおかげでいい課題研究生活を送ることができました。ありがとう。