霧箱を用いた放射線の測定

大阪大学理学部物理学科　山中卓研究室
久郷　莉奈
村井　凛久

2022 年 3 月吉日

目次

概要

電子対生成を肉眼で観測するために霧箱を用いて測定を試みた。電子対生成は発生頻度が少ないことが予想されるので、α 線を用いて霧箱の動作条件の最適化を行った。最適化を行う際の指標として、飛跡の発生頻度を調べるために OpenCV を用いた画像解析アルゴリズムを開発した。これにより、飛跡の発生時刻を系統的に導くことができるようになった。これを利用し、霧箱に用いるドライアイスとエタノールの量を変化させて飛跡が最も多く観測される条件を調べた結果、ドライアイスは 200 mL、エタノールは 80 mL が最適であることが分かった。この最適条件で γ 線と鉛の反応の測定を行った結果、α 線だけでなく $\beta^{\pm}$ 線、光電効果・コンプトン散乱による電子線、ミューオンなどの宇宙線と思われる放射線も観測することができたが、対生成を判別することはできなかった。今後はより強力な線源や磁場を使い、測定時間を長くすることで対生成の観測につながるのではないかと考えられる。

第 1 章

序章

1.1　研究の背景

宇宙は物質で満ちているが、対生成という反応が起きると物質と反物質の組が生まれる。我々は γ 線が対生成を起こして電子と陽電子の組を放出する様子を観測したいと思い、本研究のテーマに設定した。

放射線測定にはシンチレーション検出器や半導体検出器などの実験機器が用いられることが多い。しかし、これらの装置は放射線が入ってきたことを電気的な信号に変換しているため、我々の目で確かめることができない。そこで我々は、対生成の様子を実際に目で見て測定してみたいと考え、放射線の飛跡を目で見ることのできる霧箱を用いることにした。

1.2　本研究の目的

本研究の目的は、対生成によって生じる電子・陽電子を霧箱を用いて観測することである。ただし、この目的を達成するにあたり 2 つの問題点があった。1 つ目は、簡易的な実験を行ったところ α 線の飛跡は容易に観測できるが、β 線の飛跡は全く見えなかったことである。これを解消するために α 線を用いて装置の最適な動作条件を求めることにした。2 つ目は、放射線の飛跡を目視で解析することが難しい点である。そのため本研究では α 線の飛跡を動画で撮影し、画像解析の手法を取ることにした。

第 2 章

放射線の性質

2.1　放射線の種類

放射線は大きく 2 種類に分けられる。1 つは α 線、β 線のように電荷を持つ荷電粒子線である。もう 1 つは γ 線、X 線、中性子線などのように電荷を持たない中性の粒子線である。これらの放射線はいずれも人間が直接感じることができないため、他の物質との相互作用を利用して観測している。次の節では、特に本研究で関連のある荷電粒子線と γ 線が、物質とどのような相互作用をするのか説明する。

2.2　放射線と物質の相互作用

2.2.1　荷電粒子との反応

荷電粒子が物質中を通過すると、物質内の原子中の電子をイオン化させることでエネルギーを落とす。このとき単位長さあたりに落とすエネルギー $-\frac{dE}{dx}$ は Bethe-Bloch の式によって以下のように表される [1]。

$$-\frac{dE}{dx} = 2\pi N_a r_e^2 m_e c^2 \rho \frac{Z}{A}\frac{z^2}{\beta^2}\left[\ln\left(\frac{2m_e\gamma^2 v^2 W_{max}}{I^2}\right) - 2\beta^2\right] \tag{2.1}$$

ただし、r_e：古典的電子半径、m_e：電子の質量、N_a：アボガドロ定数、I：平均励起ポテンシャル、Z：吸収物質の原子番号、A：吸収物質の質量数、ρ：吸収物質の密度、z：入射粒子の電荷、β：入射粒子の v/c、γ：$1/\sqrt{1-\beta^2}$、W$_{max}$：衝突で遷移する最大エネル

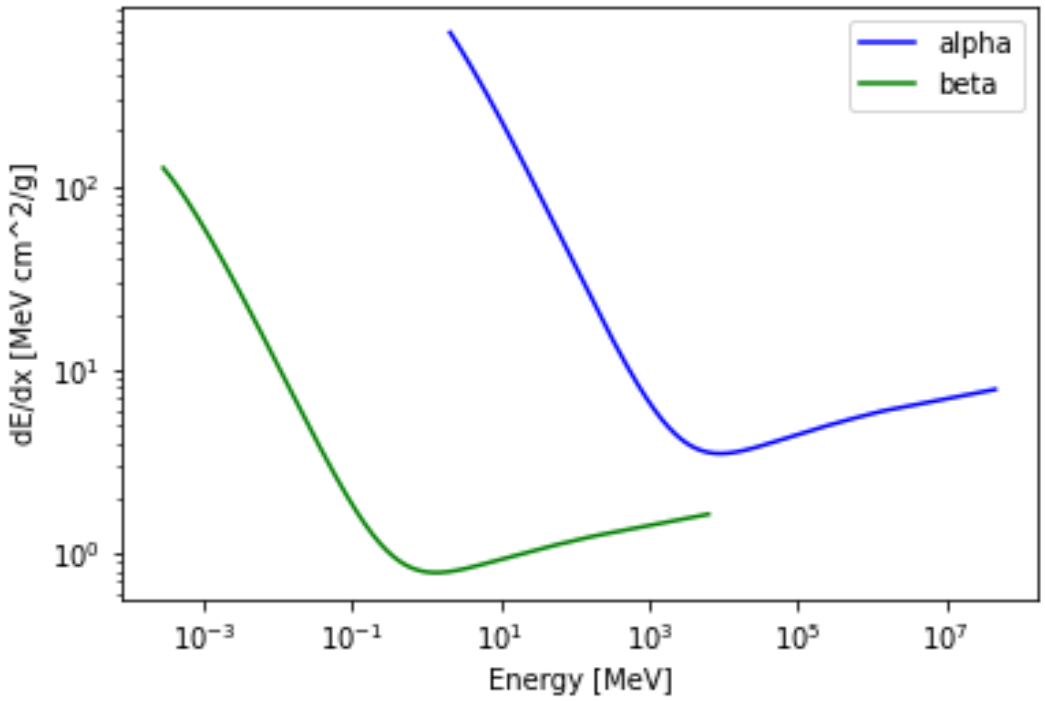

図 2.1　α 線と β 線の物質に対するエネルギー損失

ギー である。
この式を用いて α 線と β 線の物質によるエネルギー損失を図 2.1 に示す。

α 線は β 線と比べて質量や電荷が大きいためエネルギーを落としやすくなっている。β 線源として知られる ^{90}Sr から放出される β 線の最大エネルギーはおよそ 2.3 MeV である。したがって、例えば密度が 1 g/cm^3 の水の場合、厚さ数 cm で完全に静止させることができる。

2.2.2　γ 線との反応

γ 線 (光子) と物質との反応には「光電効果」「コンプトン散乱」「対生成」の 3 種類がある。

光電効果とは、光子が持つエネルギーを物質中ですべて失い、光電子を放出する現象で

ある。このとき光電子が得るエネルギー $E_{p.e.}$ は

$$E_{p.e.} = h\nu - W \tag{2.2}$$

である。
ただし h はプランク定数、ν は光子の振動数、W は物質の仕事関数である。光電効果の反応断面積は物質の原子番号の約 5 乗に比例するため、重い元素を標的に使うとはるかに反応が起こりやすくなる。

コンプトン散乱とは、物質中で光子が散乱し、エネルギーの一部を落として電子を放出する現象である。入射した光子が角度 θ の方向に散乱されたとき電子が得るエネルギー $E_{compton}$ は

$$E_{compton} = \left(1 - \frac{1}{1 + \frac{E}{m_e c^2}(1 - \cos\theta)}\right) E \tag{2.3}$$

である。
ただし E は入射した光子のエネルギー、m_e は電子の質量、c は光速である。コンプトン散乱の反応断面積は物質の原子番号の 1 乗に比例するため、重い元素の方が反応は起こりやすいが、この効果は線形にしか効かない。

対生成とは、物質中で光子の持つエネルギーをすべて失い電子と陽電子の組を放出する現象である。電子と陽電子はそれぞれ 0.511 $\mathrm{MeV/c^2}$ の質量を持つので、1.022 MeV のエネルギーを持った光子でなければ対生成は起こらない。対生成の反応断面積は物質の原子番号の 2 乗に比例するため、重い元素を使うとより反応が起こりやすくなる。

これらの三つの反応は、照射される光子のエネルギーによって断面積が異なる。一般にエネルギーが低いと光電効果が、エネルギーが高いと対生成が起こりやすくなる。図 2.2 では標的物質が鉛の時の反応断面積のエネルギー依存性を示している [2]。

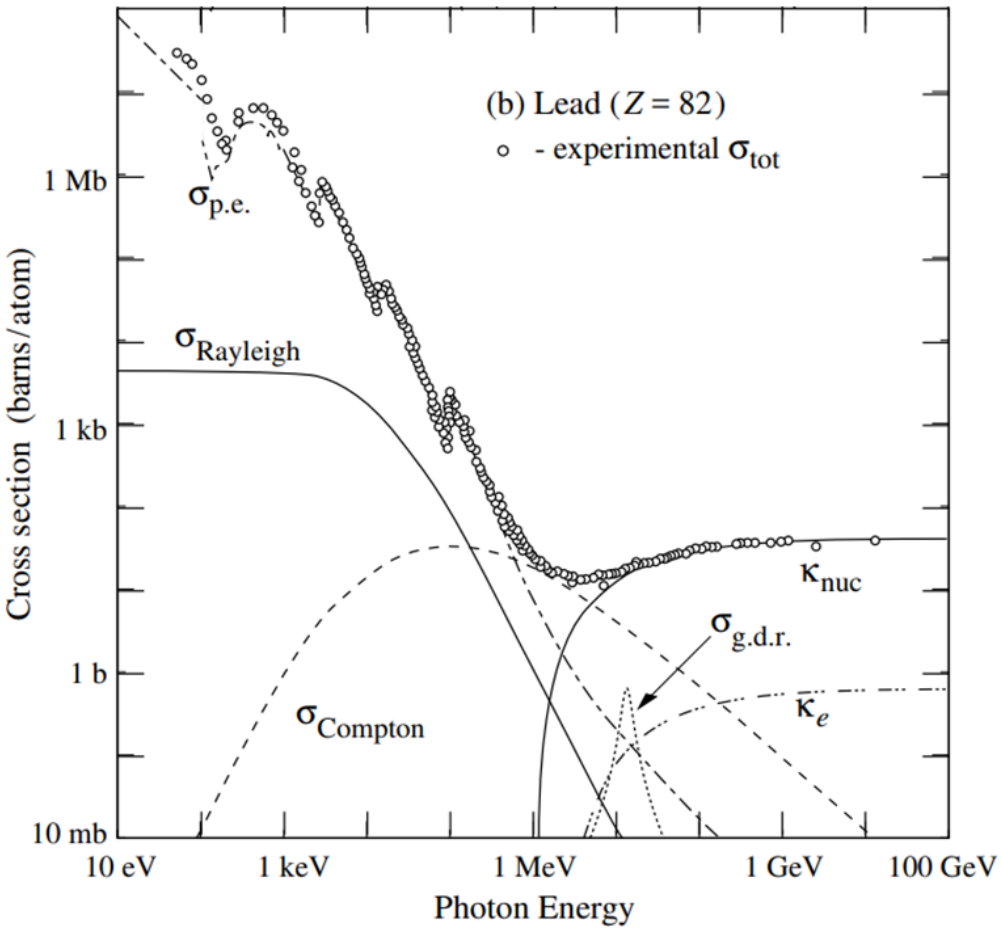

図 2.2　γ 線 (光子) と鉛との反応断面積

第 3 章

霧箱

3.1　霧箱の歴史

1894 年 9 月、ウィルソンはベン・ネヴィス山の気象観測所で見た雲や霧の中の光学現象に興味を持った。そこで 1895 年の初め、この光学現象を実験室で再現することを目的として、湿った空気を膨張させることで雲を発生させるという実験を行った。この時に開発された装置がウィルソンの霧箱 (膨張霧箱) である [3]。1910 年頃になると霧箱は放射線の飛跡の観測に用いられるようになった。1933 年にはホックストンによって拡散霧箱の原型が作られ、1939 年にラングスドルフによって完成された。1932 年、アンダーソンは霧箱に磁場をかけて宇宙線を観測した際に陽電子を発見した [4]。以上のように、霧箱は初期の素粒子物理学に大きく貢献した。

3.2　霧箱の原理

霧箱は、密閉した容器の中にエタノールなどの蒸気が充填された装置である。蒸気量は飽和蒸気量を上回っており、過飽和状態になっている。この過飽和状態では常に凝結と蒸発が繰り返されている。2 章で示したように、霧箱内を荷電粒子が通過すると、凝結した分子を電離させ、飛跡に沿って帯電した液滴が発生する。ここで、液滴は帯電すると式 (3.1) に従って表面の蒸気圧が減少する [5]。

$$\log \frac{P_r}{P} = \frac{V_m}{R\theta}\left(\frac{2T}{r} - \frac{e^2}{8\pi K r^4}\right) \tag{3.1}$$

P_r : 蒸気圧
P : 平坦な表面での飽和蒸気圧
V_m : モル体積
R : 気体定数
θ : 絶対温度
T : 表面張力
r : 液滴の半径
e : 電荷
K : 液滴を取り巻く媒体の比誘電率

表面の蒸気圧が減少すると、周囲の気体を取り込んで液滴の半径が大きくなる。表面張力による液滴の半径を小さくする効果と、イオンによる液滴の半径を大きくする効果が釣り合った時点で成長が止まる。以上のように発生したイオンを凝結核とすることで液滴は大きく成長し、荷電粒子の飛跡が雲のように観測される。

霧箱は過飽和状態を作る方法の違いにより、ウィルソンの霧箱と拡散霧箱の 2 種類に分類される。

ウィルソンの霧箱は断熱膨張を利用して過飽和状態を作り出す。ピストンやゴム膜などを用いて霧箱内の空気を断熱膨張させると、その瞬間温度が下がる。温度が下がると飽和水蒸気量も減少するので、過飽和状態になる。

拡散霧箱は気体の拡散を利用して過飽和領域を作り出す。霧箱の上部を温め下部を冷却すると、上部の蒸気は下部に向かって拡散する。温度が下がるにつれて飽和水蒸気量も減少するので、霧箱の下部に過飽和領域ができる。

ウィルソンの霧箱は飛跡が観測できる時間が短く、再度観測するまでに準備が必要なので、今回は長時間観測できる拡散霧箱を採用した。

飛跡がよく見えるようにするためには霧箱内に十分な過飽和領域を確保する必要がある。ドライアイスの量が多いと霧箱の下部の温度が下がりやすくなり、霧箱内の温度勾配も大きくなる。図 3.2 より、温度勾配が大きいと飽和蒸気量の差も大きくなるので、発生する霧の量が増える。一方霧箱内に充填された蒸気の量が少ないと、十分な量の霧が発生しない。以上のことから、霧箱の動作条件の最適化にあたってはドライアイスの量と霧箱内に充填された蒸気の量が重要になる。

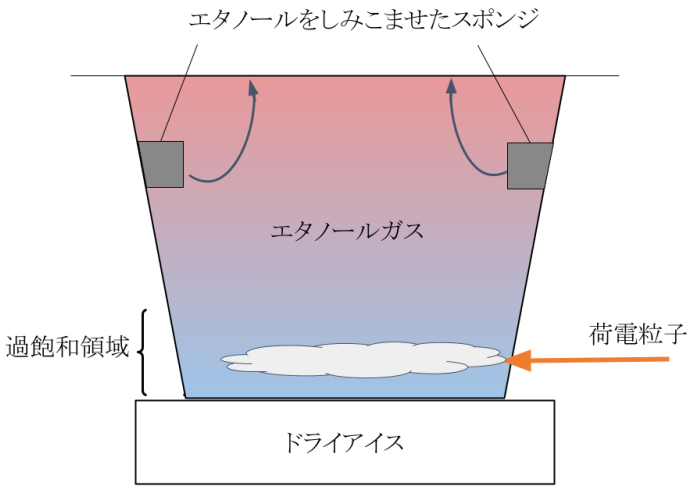

図 3.1　拡散霧箱の例。霧箱上部のスポンジから蒸発したエタノールガスは、上が塞がれているので下部に向かって拡散する。霧箱の下部をドライアイスで冷却することで霧箱内には温度勾配ができている。温度が下がるにつれて飽和水蒸気量も減少するので、霧箱の下部に過飽和領域ができる。

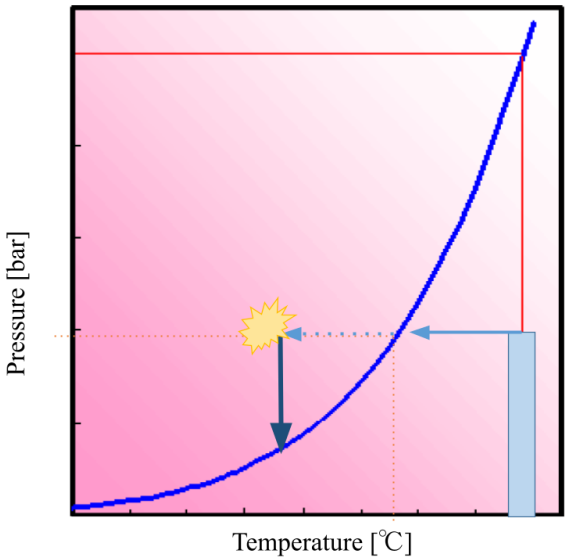

図 3.2　過飽和状態。横軸は温度、縦軸は蒸気圧で、青い曲線は蒸気圧曲線を表している。気体の温度を下げていくと、通常は露点以下の温度で飽和蒸気量を上回った分の蒸気が凝結する。しかし、凝結核が存在しないと露点を下回っても液体にならず、気体の状態を維持することがある。これが過飽和状態である。

第 4 章

霧箱の動作条件の最適化

まず比較的観測が容易な α 線を用いて測定することで、霧箱の動作条件の最適化を行なった。

4.1 霧箱の作成

以下の材料を用いて霧箱を作成した。

- 2L ペットボトル
- ブラックシート
- スポンジ
- 植毛紙
- エタノール
- ランタンのマントル
- ペットボトルのキャップ
- サランラップ
- 輪ゴム
- 黒いビニールテープ
- 発泡スチロール容器
- ドライアイス

作成の手順は以下の要領で行なった。

1. 2L ペットボトルの上部と底部を切り取る。

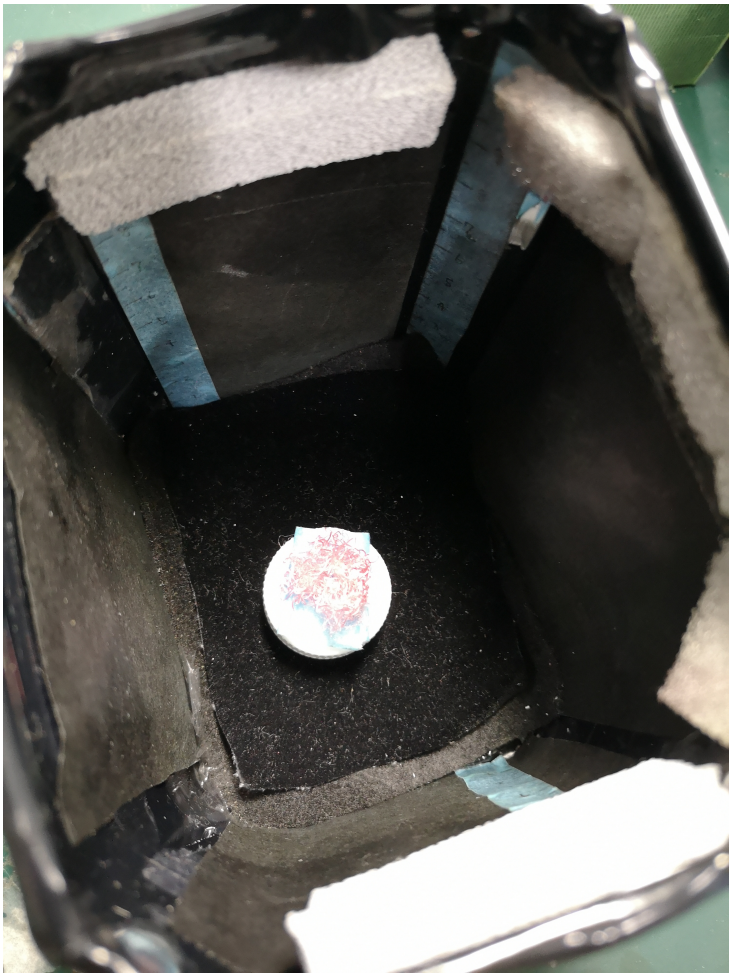

図 4.1　装置の写真

2. 底をブラックシートで覆い接着剤で固定する。
3. 側面を黒いビニールテープで完全に多い外部からの光を完全に遮断する。
4. 上部の内側にエタノールを染み込ませるためのスポンジを張り付ける。
5. 照明の反射を抑えるため植毛紙を底に敷く。
6. 底にマントルを載せたキャップを置く。
7. スポンジと植毛紙にエタノールを染み込ませる。
8. 上部をサランラップで覆い輪ゴムで留める。
9. 発泡スチロール容器にかき氷機で細かい粒子状にしたドライアイスを敷き詰めその上に容器を設置する。

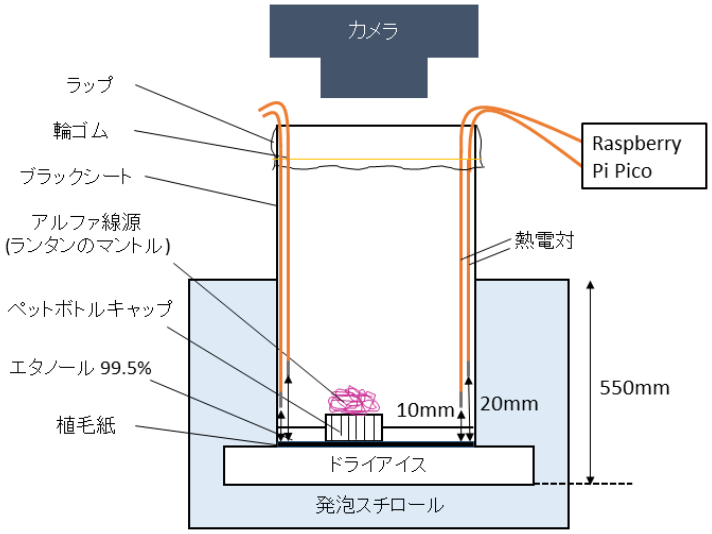

図 4.2　セットアップのイメージ

4.2　セットアップ

目視でα線の飛跡を解析することは難しいのでカメラで動画を撮影し画像解析を行った。また装置内の温度をモニターするために熱電対を合わせて 8 か所に用意した。熱電対はそれぞれ装置の四隅の底から 10 mm, 20 mm の位置に固定させた。図 4.2、図 4.3 のようになった。

4.3　画像解析

Python で OpenCV[6] を用いて、飛跡の発生時刻などを調べる以下のようなアルゴリズムを作成した。

1. 霧箱内の様子を動画に撮影し、フレームごとに分割した。
2. 図 4.4 のように飛跡は白く見えるので、明るさの情報のみにするためにグレースケールに変換した。
3. 飛跡のない 1 秒間を目視で抜き出し、ピクセルごとの明るさ x_i の平均値 μ_i と標準偏差 σ_i を求めた。その後、式 (4.1) のように標準化した。

$$z_i = \frac{x_i - \mu_i}{\sigma_i} \tag{4.1}$$

図 4.3　実際のセットアップの写真

4. 飛跡のみが白く見えるように閾値 t を設定し、$z_i \geq t$ なら白、$z_i < t$ なら黒として二値化画像を作成した。閾値 t を上げるにつれて飛跡以外の細かいノイズは減るが、薄い飛跡は検出されなくなる。今回は目視で最適な閾値を探し、閾値 $t = 15$ に設定した。
5. 飛跡以外のノイズを消すために、メディアンフィルタを用いてノイズを軽減した。

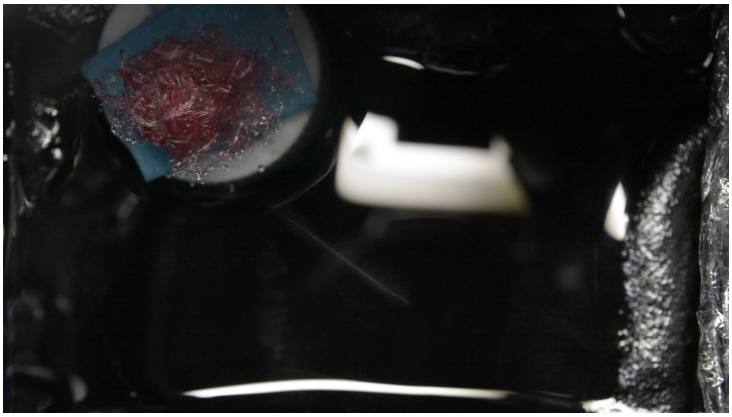

図 4.4　飛跡が観察されたフレームの例

メディアンフィルタはカーネル内の全画素の中央値を出力するフィルタで、細かいノイズの除去に効果的である [7]。

6. Hough 変換 [8] を用いて直線検出を行った。図 4.5 のように同じ飛跡でも複数の直

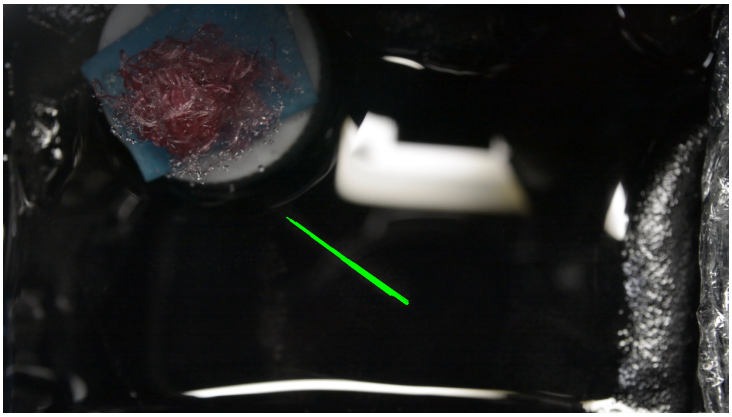

図 4.5　直線検出

線が重なって検出されることがあるので、直線の中心点同士の距離が 200 pixel 以上離れていれば別の飛跡とみなすことで、重複を防ぎつつ飛跡の数を数えた。

以上の手順により、飛跡の発生時刻・長さ・角度が測定できるようになった。

作成した画像解析アルゴリズムの性能評価を行うため、目視との一致率を調べた。その結果、目視で観測された飛跡の数が 850 となったのに対し、画像解析で検出された飛跡の数は 800 であった。図 4.6 の赤い丸で囲んだ飛跡のように、閾値の設定によっては薄い直線は検出されない場合がある。一方目視では確認できない薄い直線や、ノイズの影響により存在しない直線が画像解析で検出される場合もある。このように目視と画像解析どちらか一方でしか検出できない直線もあったが、両者はおおむね一致した。従って、この画像解析アルゴリズムにより、目視に頼らない飛跡の計数が可能になった。

4.4　最適化

本研究ではドライアイスの量とエタノールの量を変化させ、飛跡がもっとも多く見える条件を決定することにした。ドライアイスはかき氷機で細かくした粒子状での体積を 50 mL, 100 mL, 200 mL と変化させた。エタノールは体積を 20 mL, 40 mL, 60 mL, 80 mL, 100 mL と変化させた。

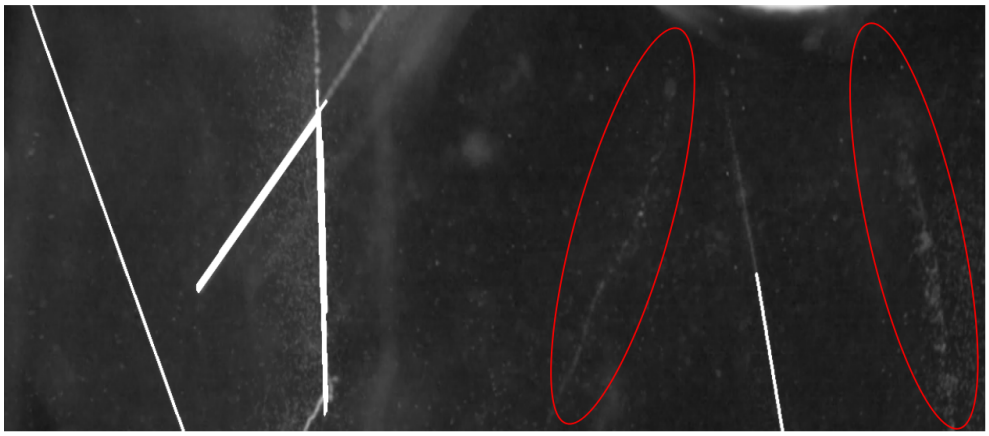

図 4.6　赤い丸で囲まれた飛跡は画像解析で検出されない直線の例。白い直線は画像解析で検出された直線。

4.4.1　ドライアイスの量の決定

ドライアイスの量を変えたときの底から 10 mm 地点と 20 mm 地点での最低到達温度の平均を図 4.7 に表す。ドライアイスの体積が 50 mL では温度が十分に下がり切らないが、100 mL 以上にすることで最低温度を最も低くすることができた。

一方、それぞれの条件で α 線の飛跡の動画 20 分を画像解析し、1 分当たりの飛跡の数をグラフにすると図 4.8 のようになった。ドライアイスの体積 100 mL と 200 mL では、最低到達温度に関しては大きな差はなかったが、飛跡の数は後者のほうがよく見える結果となった。これを踏まえて、ドライアイスの量は 200 mL に決定した。

4.4.2　エタノールの量の決定

ドライアイスの量を 200 mL で固定し、エタノールの量を変化させた。α 線の飛跡の動画 20 分を画像解析し、それぞれの飛跡の総数を求めると図 4.9 のようになった。エタノールの量が 80 mL になると急激に飛跡の数が増えた。量が多いほうが見えやすくなると思われたが、100 mL では数が減ってしまう結果となった。今回は飛跡の数が最も多い条件を最適条件とすることにしたため、エタノールの量は 80 mL に決定した。

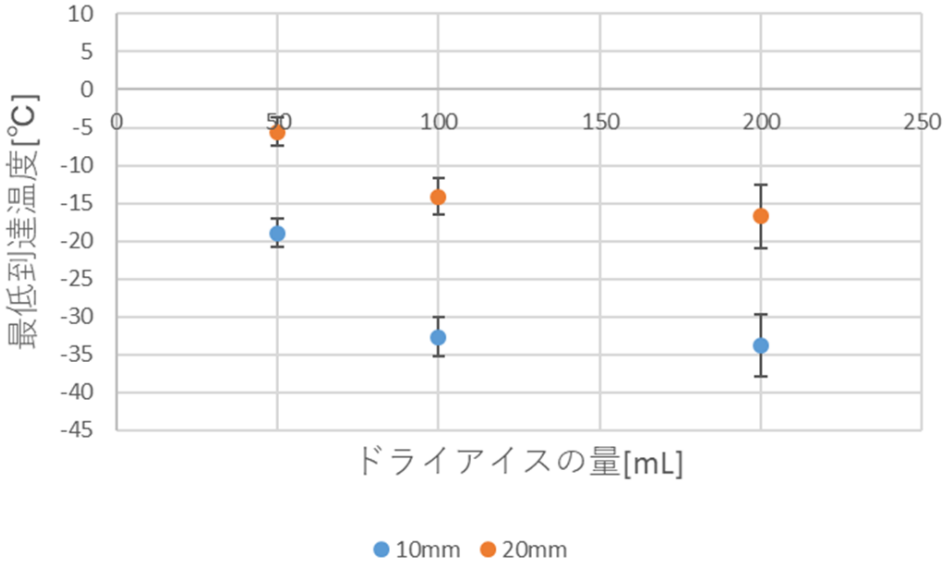

図 4.7　ドライアイスの量と最低到達温度の関係

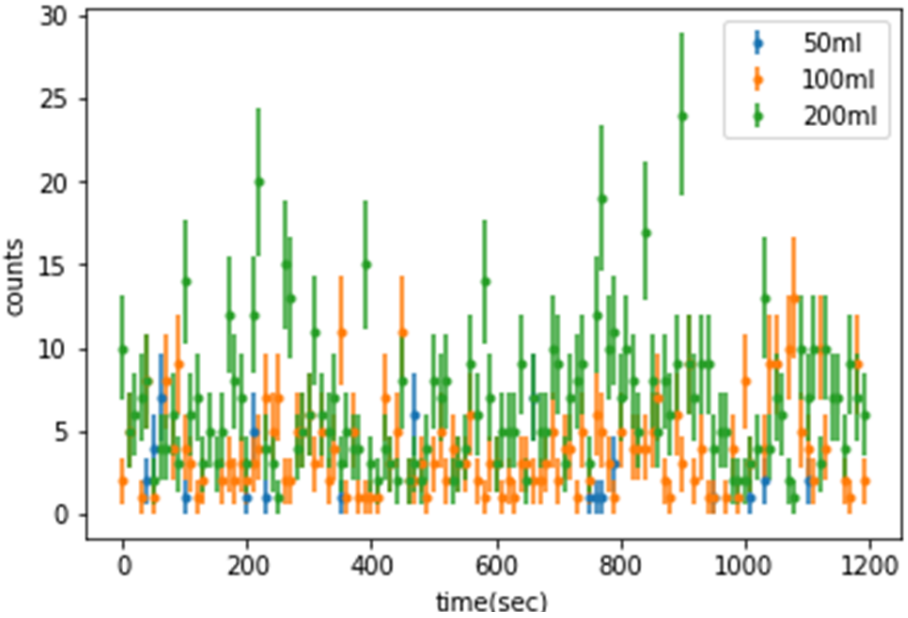

図 4.8　ドライアイスの量と 1 分当たりの飛跡の数の関係

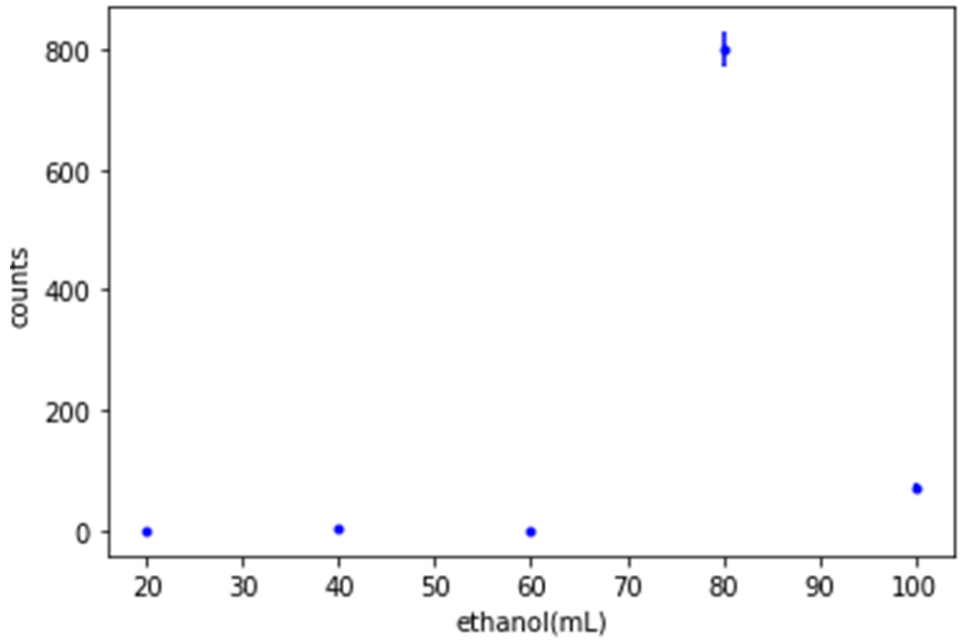

図 4.9　エタノールの量と飛跡の数の関係

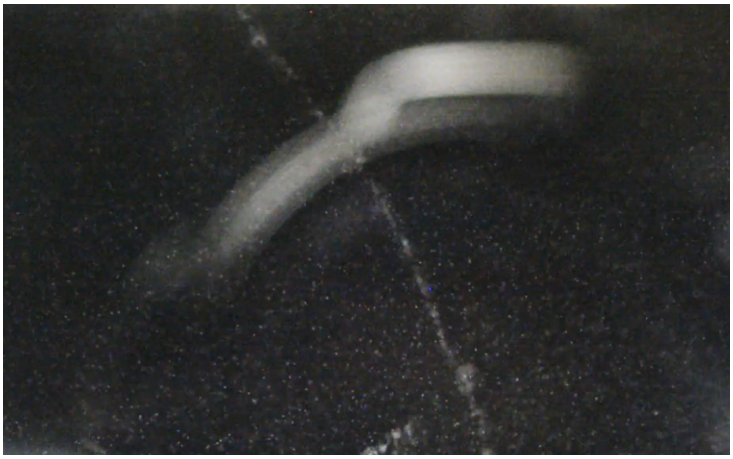

図 4.10　宇宙線ミューオンの飛跡

4.4.3　α 線以外の放射線の観測

最適化した条件で線源を何も置かず観測したところ、飛跡が現れた (図 4.10)。これは宇宙線ミューオンだと考えられる。次に β^- 線源である ^{90}Sr を置くと多くの飛跡が現れた (図 4.11)。最後に β^+ 線源である ^{22}Na を置くとこちらも多くの飛跡が現れた (図 4.12)。

以上のようにこの条件で電子や陽電子の飛跡も十分観測できることを確認できた。

図 4.11　電子線の飛跡

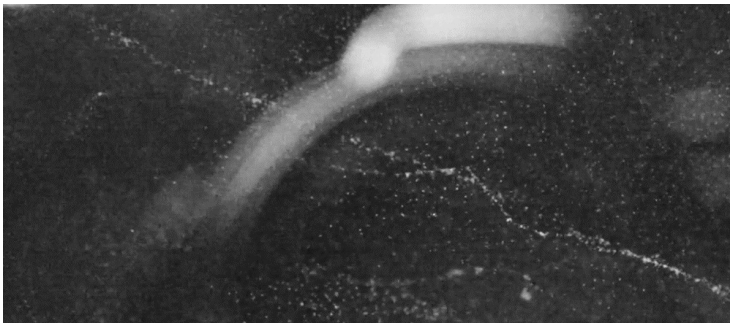

図 4.12　陽電子線の飛跡

第 5 章

γ 線の反応の測定

5.1　線源及び標的物質の選定

対生成が起こるには 1.022 MeV 以上のエネルギーをもつ γ 線源が必要である。研究室で所有しているこの条件を満たす線源の中で、本研究では 1.274 MeV のエネルギーの γ 線を放出する ^{22}Na を用いた。また対生成は標的物質の原子番号が大きいほど反応断面積が大きくなる。これを踏まえたうえで入手が容易であることから今回は鉛をターゲットに選んだ。

γ 線のエネルギーが 1.25 MeV のとき鉛と反応する全断面積は 20.21 barn/atom であり、対生成の断面積は 0.1301 barn/atom である [9]。したがって今回の実験では全反応のうちおよそ 0.6% は対生成が起きていると見積もることができる。

5.2　測定方法

霧箱装置の左側に厚さ約 0.9 mm の鉛板を設置し、そこに向けて装置の外側から γ 線を照射する (図 5.1)。すると鉛板から光電子、コンプトン電子、電子-陽電子対が出てくる。対生成では光電効果やコンプトン散乱と異なり正の電荷をもつ粒子と負の電荷をもつ粒子の組として飛んでくる。そのため磁場をかけることで図 5.2 のような飛跡を描くと予想できる。そこで霧箱を支える発泡スチロール容器の裏側 3 か所にネオジム磁石を固定させた。この時霧箱内の磁場を測定したところ平均的に上向き 5 mT であった。もし電子と陽電子がそれぞれ 0.1 MeV のエネルギーで飛んできた場合、それぞれ半径約 22 cm の円弧を描くはずである。

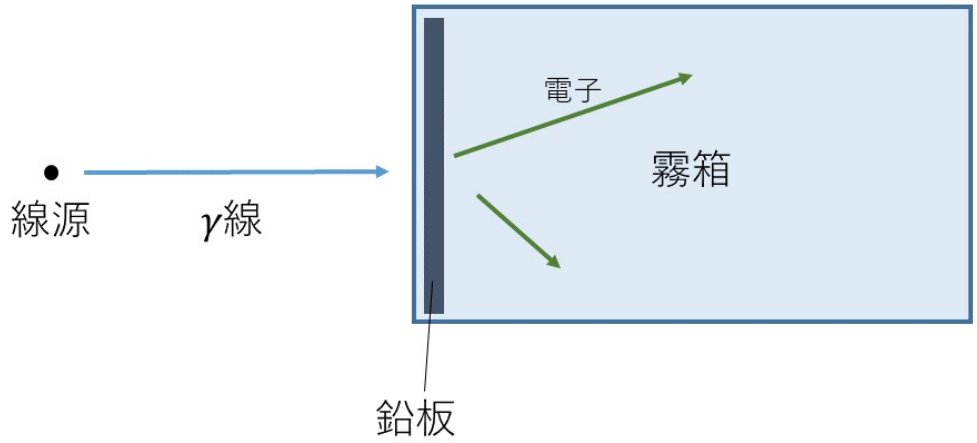

図 5.1　測定方法のイメージ

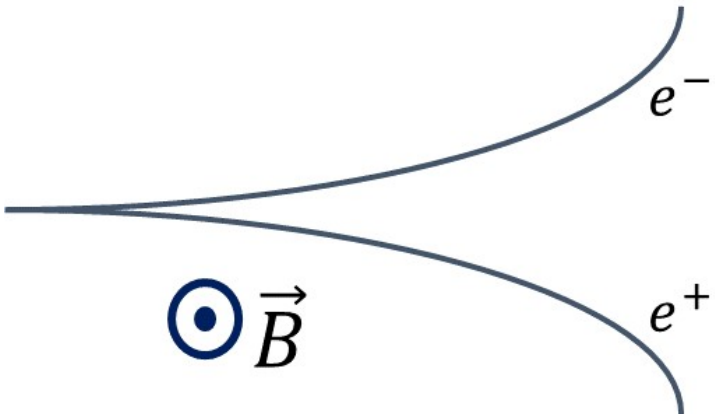

図 5.2　対生成の飛跡のイメージ

5.3　観測結果

20 分間動画で撮影し、図 5.2 のような飛跡がないか目視で確認した。すると図 5.3 のような飛跡がいくつか現れた。このイベントが対生成である妥当性を考察してみる。この飛跡と線源との位置関係は図 5.4 のようになっているため、散乱の角度が大きくなっている。しかしながら散乱 γ 線のエネルギーは

$$E' = \frac{1}{1 + \frac{E}{m_e c^2}(1 - \cos\theta)} E \tag{5.1}$$

で与えられるため、E=1.274 MeV, E'=1.022 MeV となる角度を求めると散乱角 1° 未満になるはずである。これを基にすると候補イベントは実際と矛盾している。したがって

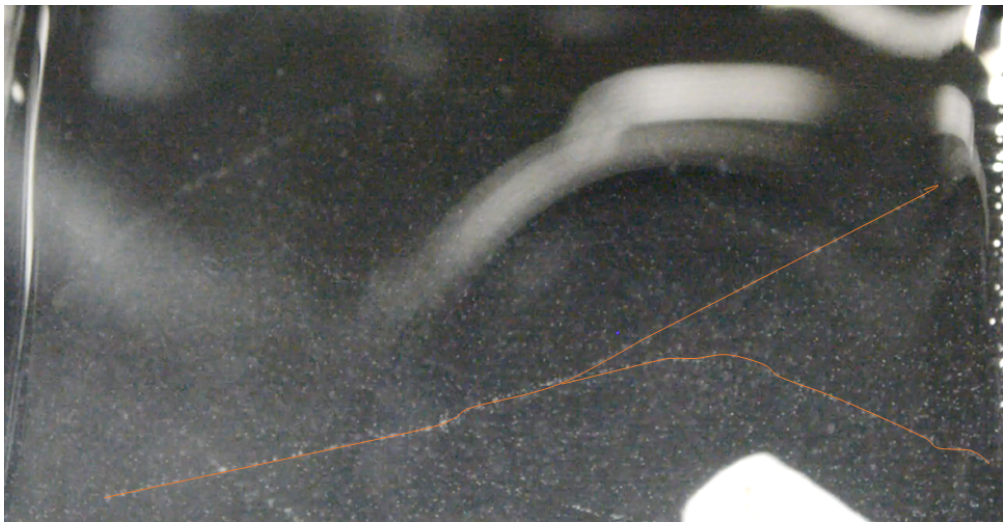

図 5.3　対生成イベントの候補

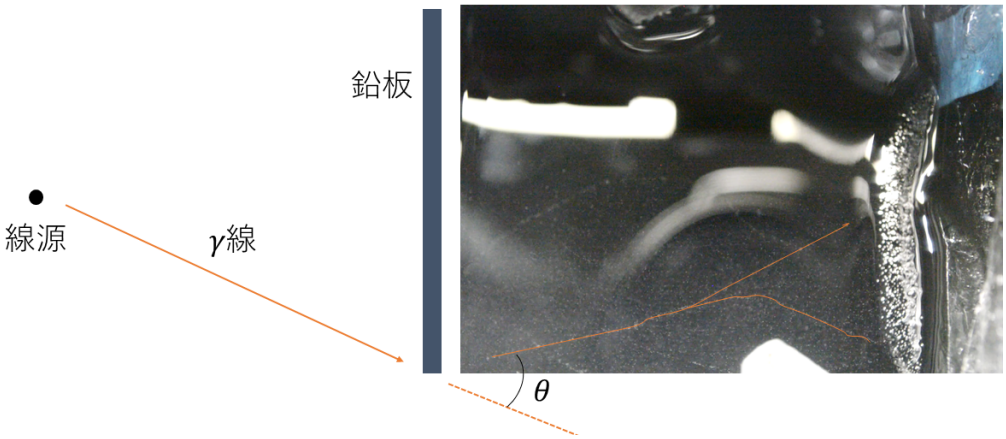

図 5.4　候補イベントの検証

これらは対生成だと断言することができない。

第 6 章

考えられる改善点

今回は対生成だと断言できるイベントを見つけることができなかったが、以下の 3 点を改善することで今後観測につながるのではないかと考えられる。

6.1　線源のエネルギー

^{22}Na ではエネルギーが弱いため対生成の反応断面積が小さかった。もっと強力な線源を使うことで対生成のイベント数を増やすことができ、観測しやすくなると考えられる。

6.2　磁場の強さ

今回用いたネオジム磁石は小型で磁場の大きさが十分ではなった。そのため多重散乱によって電子が曲がっているのか、磁場によって曲がっているのか判断が難しい場面があった。もし霧箱内に 100 mT 程度の磁場をかけることができれば、0.1 MeV 程度のエネルギーをもつ電子や陽電子は 1 cm ほどの半径で円を描き、多重散乱との区別が容易になる。

6.3　測定時間の拡大

今回は時間の制約で 20 分の動画を目視で解析することしかできなかったが、もっと長時間測定を続ければ対生成イベントを見つける可能性は上がるはずである。そのためには目視では限界があるので画像解析で判別できるプログラムの開発も必要になる。

第 7 章

結論

まず α 線を用いて霧箱の動作条件の最適化を行った。最適化を行う際の指標として、飛跡の発生頻度を調べるために OpenCV を用いた画像解析アルゴリズムを開発した。性能評価のために目視との一致率を調べたところ、目視で観測された飛跡の数が 850 だったのに対し、画像解析で検出された飛跡の数は 800 だった。霧箱内で十分な過飽和領域を確保するためにはドライアイスとエタノールの量が重要になるため、これらの量を変化させて飛跡が最も多く観測される条件を調べた。その結果ドライアイスは 200 mL、エタノールは 80 mL が最適であることが分かった。

次にこれらの最適条件で γ 線と鉛の反応の測定を行った結果、$\beta^{\pm}$ 線、光電効果・コンプトン散乱による電子線、ミューオンなどの宇宙線と思われる放射線を観測することができた。その中から目視で対生成イベントの候補を探し、散乱角からその妥当性を評価したが、対生成だと断言できる事象の同定はできなかった。今後はより強力な線源や磁場を使い、測定時間を長くすることで、対生成の観測につながるのではないかと考えられる。

参考文献

[1] William R. Leo, Techniques for Nuclear and Particle Physics Experiments

[2] PDG.33. Passage of Particles Through Matter https://pdg.lbl.gov/2019/reviews/rpp2018-rev-passage-particles-matter.pdf

[3] C.T.R.Wilson, *Nobel Lectures In Physics*, pp.194-214, (1922-1941)

[4] 坂内忠明, 放射線教育, **4**, pp.4-17, (2000)

[5] 宮下晋吉, 科学史研究 第Ⅱ期, **14**, pp.145-153, (1975)

[6] OpenCV team, OpenCV, `https://opencv.org/about/`

[7] Alexander Mordvintsev and Abid K., OpenCV-Python Tutorials documentation, `http://labs.eecs.tottori-u.ac.jp/sd/Member/oyamada/OpenCV/html/py_tutorials/py_imgproc/py_filtering/py_filtering.html`

[8] Alexander Mordvintsev and Abid K., OpenCV-Python Tutorials documentation, `http://labs.eecs.tottori-u.ac.jp/sd/Member/oyamada/OpenCV/html/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html`

[9] NIST. XCOM: Photon Cross Sections Database, https://www.nist.gov/pml/xcom-photon-cross-sections-database

謝辞

本研究にあたりご指導頂いた山中卓教授、南條創准教授、廣瀬穣助教には大変お世話になりました。特に研究全体を監督してくださった廣瀬穣助教には、実験や画像解析の方針に係り多くの助言を頂き、感謝しております。小寺克茂さんには実験器具をお貸し頂きお世話になりました。また秘書の藤阪千衣さん、前田純子さんには実験材料の手配に際しお手伝い頂きありがとうございました。最後に院生の先輩方には助言や激励の言葉を頂き、励みになりました。ありがとうございました。

付録 A

画像解析

放射線の性質を系統的に理解するために、目視で確認することが難しい飛跡の

1. 発生時刻：放射線の発生頻度が分かる
2. 長さ：放射線の飛程が分かる
3. 角度：放射線の種類が分かる

が分かる以下のようなアルゴリズムを作成し、Google Colaboratory で実行した。

```
from google.colab.patches import cv2_imshow サポートパッチ#
import cv2
import numpy as np
import time
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import math
import pandas as pd

H=[]
time= []
f = []
angle = []
X1 = []
Y1= []
X2 = []
Y2 = []
i = 0
```

```
19
20 # 動画ファイルのキャプチャ
21 cap = cv2.VideoCapture('DSC_1184D50E80-2.MOV')
22
23 # 画像範囲
24 roi = (555,925,140,1350)
25
26 # fps
27 fps = cap.get(cv2.CAP_PROP_FPS)
28
29 # 時間指定
30 start_sec = 1170.0
31 stop_sec = 1171.0
32 cap.set(1,int(fps*start_sec))
33
34 while(cap.isOpened()):
35
36     # フレームの取得
37     ret, frame = cap.read()
38     frame = frame[roi[0]:roi[1],roi[2]:roi[3]]
39
40     # グレースケール変換
41     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
42     H.append(gray)
43
44     if cap.get(1) > int(fps*stop_sec):
45         break
46
47 H = np.array(H)
48 mean = np.mean(H, axis = 0)
49 std = np.std(H, axis = 0)
50
51
52 # 時間指定
53 start_sec = 1080.0
54 stop_sec= 1140.0
55 cap.set(1,int(fps*start_sec))
56
```

```
57
58 while(cap.isOpened()):
59
60     # time
61     start = cap.get(1)/fps
62     print('start time:',start)
63
64     # フレームの取得
65     ret, frame = cap.read()
66     frame = frame[roi[0]:roi[1],roi[2]:roi[3]]
67
68     # グレースケール変換
69     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
70
71     # 2 次元ヒストグラム
72     track = np.abs(gray - mean)
73     track = track/std
74     track = track.astype(np.uint8)
75
76     # 二値化してマスク画像を算出
77     th = 15
78     track[track < th] = 0
79     track[track >= th] = 255
80
81     # メディアンフィルタ
82     ksize = 3
83     median = cv2.medianBlur(track,ksize)
84
85     # 直線検出
86     lines = cv2.HoughLinesP(median, 1, np.pi / 180, threshold=100,
           minLineLength=100, maxLineGap=15)
87     if lines is not None: #[......, 1]
88         for x1, y1, x2, y2 in lines.squeeze(axis=1):
89             time.append(start)
90             f.append(i)
91             img = cv2.line(gray, (x1, y1), (x2, y2), (255, 255, 255)
                   , 2)
92             ang = int(math.atan((y2-y1)/(x1-x2))*180/math.pi)
```

```
            angle.append(ang)
            X1.append(x1)
            X2.append(x2)
            Y1.append(y1)
            Y2.append(y2)
            #print(ang)
            #print(x1,y1,x2,y2)
            #leng_2 = ((x2-x1)**2)+((y1-y2)**2)
            #leng = math.sqrt(leng_2)
            #print(leng)
        cv2_imshow(img)

    i = i + 1

    if cap.get(1) > int(fps*stop_sec):
        break

# csv に書き込み
csv_path = "/content/drive/MyDrive/"
csv_name = 'D50.csv'

dict = {'frame': f, 'time': time, 'angle': angle, 'x1': X1, 'y1': Y1
    , 'x2' : X2, 'y2': Y2}
df = pd.DataFrame(dict)
df.to_csv(csv_path+csv_name)

cap.release()
cv2.destroyAllWindows()
}
```

さらに霧箱の動作条件の最適化を行う際の指標として、飛跡の発生頻度を調べる以下のようなアルゴリズムを作成した。

```
import pandas as pd

csv_path = "/content/drive/MyDrive/"
csv_name = 'D50.csv'
df = pd.read_csv(csv_path+csv_name, index_col = 0)
```

```
6
7  import math
8  import numpy as np
9
10 i = 0
11 n = 1
12 C = []
13 N = []
14 L = []
15
16 while(i<3247):
17     cx = (df.iloc[i,5]-df.iloc[i,3])/2 + df.iloc[i,3]
18     cy = (df.iloc[i,6]-df.iloc[i,4])/2 + df.iloc[i,4]
19     if i == 0:
20         C = []
21         C.append([cx,cy])
22         #print(C)
23         n = 1
24     else:
25         if df.iloc[i,1] != df.iloc[i-1,1]:
26             C = []
27             C.append([cx,cy])
28             n = 1
29         else:
30             p = int(len(C))
31             L = []
32             for j in range(p):
33                 #print('C:',C)
34                 len_2 = ((cx-C[j][0])**2)+((cy-C[j][1])**2)
35                 leng = math.sqrt(len_2)
36                 L.append(leng)
37                 #print('len:',L)
38             if all([k > 200 for k in L]):
39                 C.append([cx,cy])
40                 n = n + 1
41     i = i + 1
42     N.append(n)
43
```

```
#print(N)
df['N'] = N
df.to_csv(csv_path+csv_name)

import math
import numpy as np

i = 0
P = []
T = []

while(i<3247):
    if i == 0:
        p = df.iloc[0,7]
        P.append(1)
        T.append(df.iloc[0,1])
    else:
        if df.iloc[i,0] == df.iloc[i-1,0]:
            p = df.iloc[i,7]
        else:
            P.append(p)
            p = df.iloc[i,7]
            if df.iloc[i,0] - df.iloc[i-1,0] > 3:
                T.append(df.iloc[i,1])
            else:
                q = P[-1] - P[-2]
                if q >= 1:
                    for j in range(q):
                        T.append(df.iloc[i-1, 1])
    i = i + 1

# に書き込み csv
csv_path = "/content/drive/MyDrive/"
csv_name = 'D50_hist.csv'

dict = {'time': T}
df = pd.DataFrame(dict)
df.to_csv(csv_path+csv_name)
```

```
82
83 import pandas as pd
84
85 csv_path = "/content/drive/MyDrive/"
86 csv_name = 'D50_hist.csv'
87 df = pd.read_csv(csv_path+csv_name, index_col = 0)
88
89 import math
90 import numpy as np
91 import matplotlib.pyplot as plt
92
93 i = 0
94 j = 0
95 n = [0]*120
96 err = [0]*120
97 t = [0]*120
98
99 while(i<47):
100     for j in range(120):
101         if 10*j <= df.iloc[i,0] < 10*j+10:
102             n[j] = n[j] + 1
103     i = i + 1
104
105 for k in range(120):
106     t[k] = 10*k
107     err[k] = math.sqrt(n[k])
108
109 # ヒストグラム
110 n = np.array(n)
111 n = np.ma.masked_where(n < 1, n)
112 plt.errorbar(t,n,yerr=err,marker=".",linestyle='None')
113 plt.ylabel("counts")
114 plt.xlabel("time(sec)")
115 plt.xticks(np.arange(0, 1300, step=200.0))
116 plt.yticks(np.arange(0, 31, step = 5))
117 plt.show()
```