CdTe検出器用のツール

新潟大学・鈴木賢治

2 November, 2018

[1]

1　検出器データから画像を作成する

1.1　イメージ・ファイルからの画像合成

測定された検出器画像 (img_i.txt) を手際よく ImageJ[2]で観察する方法について述べる．なお，マクロプログラムなどは ImageJ の種類の一つである Fiji[3]を利用して確認した[4]．

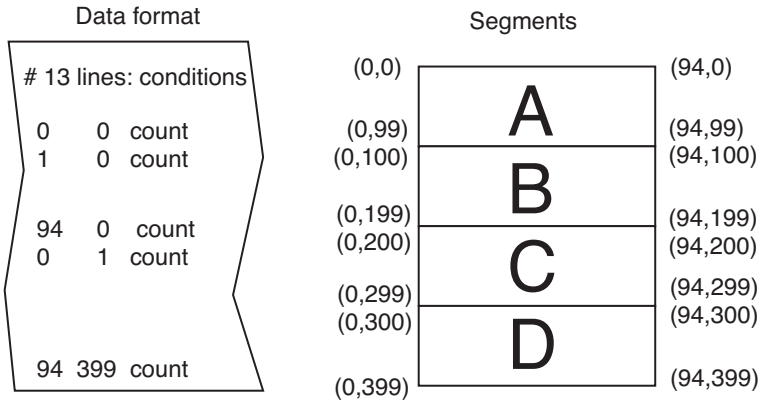


図 1: CdTe 検出器の測定データ img_i.txt の構成

プログラム '**cdte.f90**' は，img_i.txt からデータを読み出し，ImageJ で読めるテキスト画像ファイル Image000i.txt に変換する．CdTe 検出器で測定されたデータファイルは，img_i.txt のテキストファイルである．ファイルのフォーマットと内容を図 1 に示す．このテキスト形式データをそのまま逐次読み取り，その後再構成して，ImageJ のテキストイメージに再構成して，出力ファイル Image000i.txt を作成する．

そのまま読み込んで行くと，1 行目は FileName，2 行目は閾値電圧 (mV) が記録されており，14 行目から読出した各画素の位置 (x, y) とカウントが記録されている (図 1 左)．データは 100 行単位で CdTe センサーのセグメント A, B, C, D に対応する．

ファイル img_i.txt から読み出したデータ群 A, B, C, D を CdTe センサーのセグメントの配置に従ってイメージを再構成しなければならない．そのセグメント A∼D の配置は，図 2 のように各セグメントの向きと位置が異なっている．図の配置に合うようにデータ群 A, B, C, D を配置するプログラムを作成し

[1]作成したプログラムはすべて付録に収録している．また，変更されるたびに最新のツールは以下の URL からダウンロードできる．
http://kikai.ed.niigata-u.ac.jp/CdTe/

[2]ImageJ はアメリカ国立衛生研究所の Wayne Rasband により開発が始められた．Java の仮想マシン上で動作し、プラグインやマクロによる拡張性が高い．次のサイトからダウンロードできる．https://imagej.nih.gov/ij/

[3]J. Schindelin, I. Arganda-Carreras, and E. Frise, et al. (2012), "Fiji: an open-source platform for biological-image analysis", Nature methods 9(7): 676-682, PMID: 22743772.

[4]Mac 用の ImageJ は，Java 6 で作成されており，High Sierra では動作しないので，Fiji を利用すること．

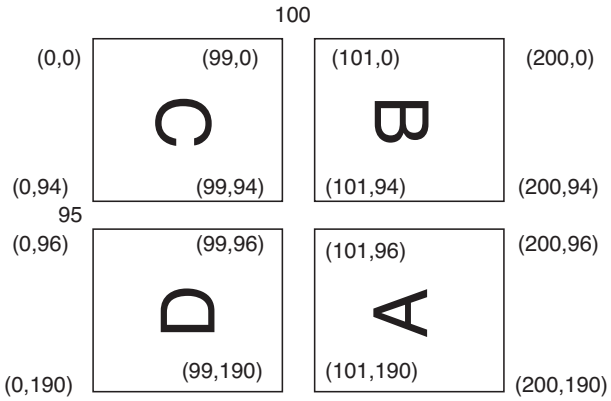


図 2: CdTe 半導体の実際の配置

た．実際の検出された画像は，191 行 201 列のデータとなる．また，95 行目と 100 列目はセンサーの貼り合わせのために画素が抜けている．この補正については，文献[5]を参考にして補正計算をした．

1.2 画像ファイルの貼り合わせ

前節で作成された Image000i.txt は，ImageJ で見ることができる．

読み込み File → Import → Text image

画質調整 ノイズや画素不良により，見たい画像が見れないので，以下の画質調整をすること．
Image → Adjust → Brightness/Contrast → Set

表示形式 白黒から見やすい画像に変換する．以下は，fire により表示した例
Image → Lookup tables → fire

これでは 1 枚ずつ読み込むので，以下では，CdTe 検出器をステージで動かして貼り合わせるために，ImageJ のマクロ'**Composition.ijm**' を作成した．その一例として 3×3 の CdTe 検出画像を図 3 を以下に示す．

マクロの定義 ImageJ の Plugins → Macros → Install を起動して，マクロプログラム"Composition.ijm"を読み込む．

マクロの実行 Plugins → Macros の下側 → Compose 3×3 を選択

2 エネルギー較正用ツール

エネルギー較正の結果を分析するためのツールを開発した"**calibration.f90**"を使用することで，次のような分析用データファイルが表示され，一連の処理を一括して行うことができる．また，表示には Fiji (ImageJ) を利用する．

img_1.txt ∼ img_160.txt 閾値電圧を $-200 \sim -41$ まで 1 mV ピッチで変化させた時の測定結果．

c_map_0001.txt ∼ c_map_0100.txt 閾値電圧 $-200 \sim -41$ に対応する各ピクセルのカウント．1 ファイルに 380 ピクセルずつ記録されている．(gnuplot 用 3 次元データ)

d_map_0001.txt ∼ d_map_0100.txt 閾値電圧 $-200 \sim -41$ に対応する各ピクセルの差分．(gnuplot 用 3 次元データ)

stack_0001.txt ∼ stack_0100.txt

[5]豊川秀訓，Spring-8 における計数型 1 次元, 2 次元検出器の開発とその応用，X線分析の進歩 第 42 集，pp .95-110 (2011)，アグネ技術センター

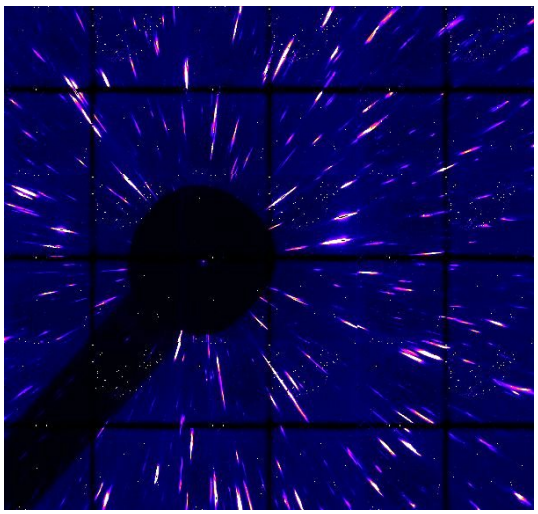


図 3: 3×3 の貼り合わせ画像例

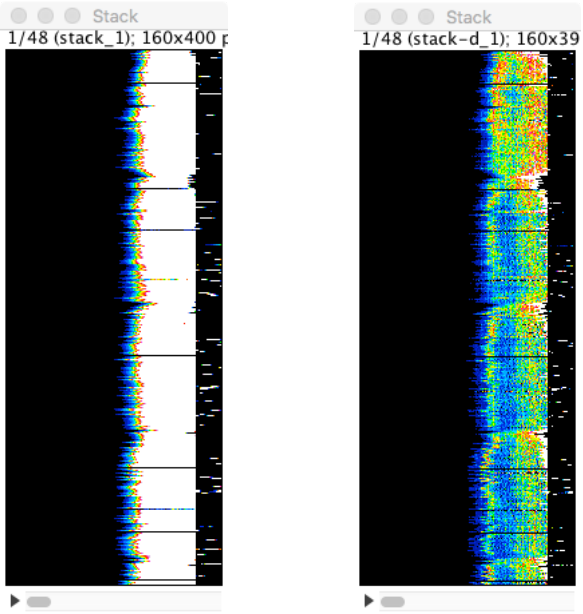
380 個のピクセルの閾値電圧とカウントのイメージ (stck_i.txt) の全ファイル 100 個を重ね合わせたスタック像を表示するために，ImageJ のマクロプログラム”**Stack100.ijm**”を用意した．

stack_i.txt から動画の作成

1. フォルダ stack を作り，閾値電圧のイメージ stack_1.txt ∼ stack_100.txt を stack に移動
   フォルダー内のファイル数を読み取り，ファイルの読み込み回数を計算しているので，フォルダ stack には stack_i.txt 以外を入れてはならない．
2. ImageJ の Plugins → Macros → Install を起動して，マクロプログラム”Stack100.ijm”を読み込む．(Macros を見ると Stack(100) と Stack-d(100) ができている)
3. Plugins → Macros → Stack(100) を選択．フォルダ stack を選択すると，100 個の stack_i.txt ファイルが読み込まれ，すべてが重ね合わされる (図 4 (a))．
4. スタック像の 3 次元表示は，Analyze → 3D surface plot を選択する．
5. スタック像のピクセルのカウントの変化を見るには，ラインツールを選択して Analyze → Plot profile を選択する．
6. スタック像を動画として保存するに，Save as → AVI を選択する．

stack-d_i.txt から stack(重ね合わせ) と動画の作成

1. フォルダ stack-d を作り，閾値電圧微分 stack-d_1.txt ∼ stack-d_100.txt を stack-d に移動
   フォルダー内のファイル数を読み取り，読み込みファイルの回数を計算しているので，フォルダ stack-d には stack-d_i.txt 以外を入れてはならない．
2. ImageJ の Plugins → Macros → Install を起動して，マクロプログラム”Stack100.ijm”を読み込む．(Macros を見ると Stack(100) と Stack-d(100) ができている)
3. Plugins → Macros → Stack-d(100) を選択．フォルダ stack を選択すると，100 個の stack-d_i.txt ファイルが読み込まれ，すべてが重ね合わされる (図 4 (b))．
4. スタック像の 3 次元表示は，Analyze → 3D surface plot を選択する．
5. スタック像のピクセルのカウントの変化を見るには，ラインツールを選択して Analyze → Plot profile を選択する．
6. スタック像を動画で保存 (Save as → AVI を選択)



(a) 各ピクセルのカウント　　(b) 閾値電圧による前進差分像

図 4: エネルギー較正用ツールによる処理．横軸が閾値電圧，縦軸が各ピクセルになる．

3　エネルギー較正

利用する蛍光X線は, W, Sn, Pb, Mo の 4 種類の金属箔であり，その特性X線エネルギーを表 1 に示す．

表 1: 特性X線

Atom	Pb	W	Sn	Mo
Kα (keV)	74.25	58.87	25.20	17.44
Kβ (keV)	84.94	67.24	28.49	19.61
K absorption edge (keV)	88.00	71.31	29.19	20.00

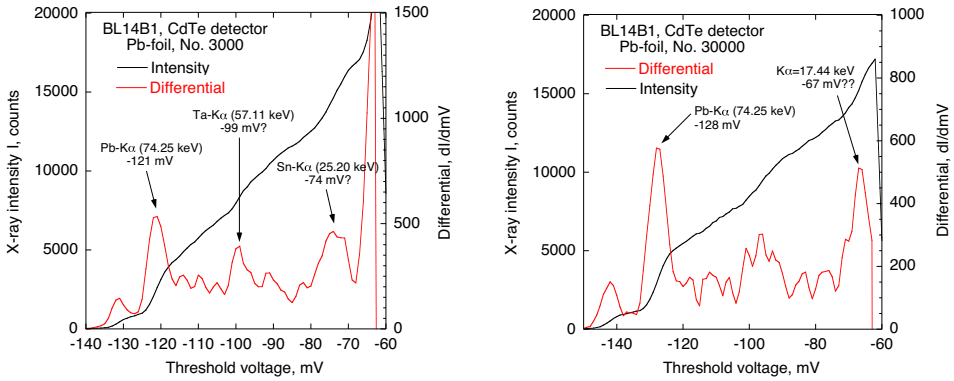
W, Sn, Pb, Mo の 4 種類の金属泊に白色X線を照射して，その蛍光X線を CdTe 検出器で測定した．それらの結果を図 5 (a)∼(d) に示す．各金属箔に対して，ピクセルごとのばらつきを見るために，No. 3000 と No. 30000 の 2 つのピクセルについて測定結果を示している．

各図には，閾値電圧に伴うカウント数の変化を示す．図中のカウントは，ノイズ除去の目的で 3 点単純移動平均の処理を施している．また，差分については，前進差分ではノイズもピークとして現れ，中間差分 ($\frac{x_{i+1}-x_{i-1}}{2\,\Delta x}$) が，ノイズがなくピークが検出しやすかった．

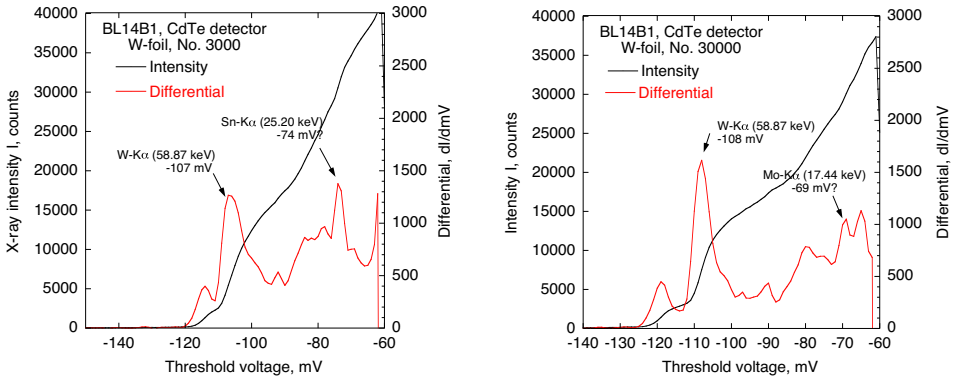
図 (a), (b) の Pb と W の結果では，ターゲット箔以外の特性X線も拾っていることがうかがえる．エネルギー較正をする場合には，ターゲット以外の蛍光X線を拾わないように工夫することが大切である．

図 (c), (d) の Sn および Mo の結果を見ると，明瞭な Sn, Mo の蛍光X線を検出できなかった．検出器の閾値電圧が -60 mV 付近で急激に増加した後，カウントが途絶えてしまう挙動が見られる．また，その付近に Sn, Mo の蛍光X線のエネルギーがあるために，エネルギー較正に難がある．いずれにしても，おおよそ 30 keV 以下のX線エネルギーについては，測定対象としてふさわしいかを再検討する必要がある．Ba-$K\alpha$ のX線エネルギーは 32.07 keV であり，Sn, Mo よりもよい可能性がある．

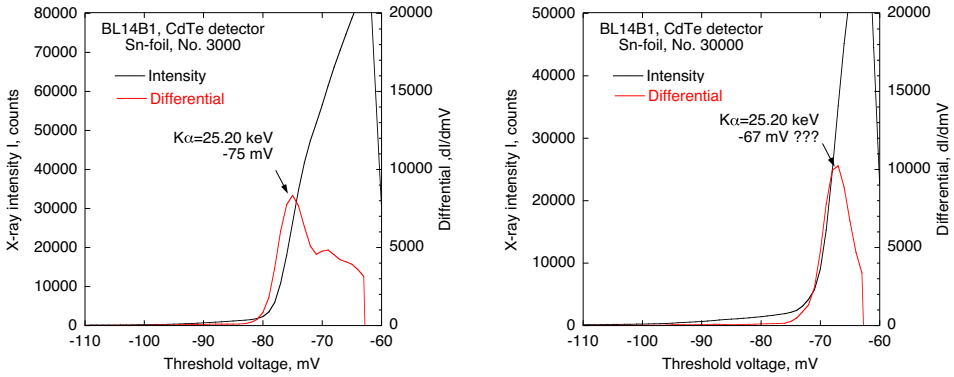
図 6 は，図 5 の結果に基づいて蛍光X線のエネルギーと閾値電圧の関係を表示している．ピクセル No. 3000 と No. 30000 では結果が異なることは，ピクセル依存性に加え，蛍光X線の照射の分布が一様でないことが影響している．また，Sn, Mo の結果が不明瞭なことから，Pb と W を利用して CdTe 検出



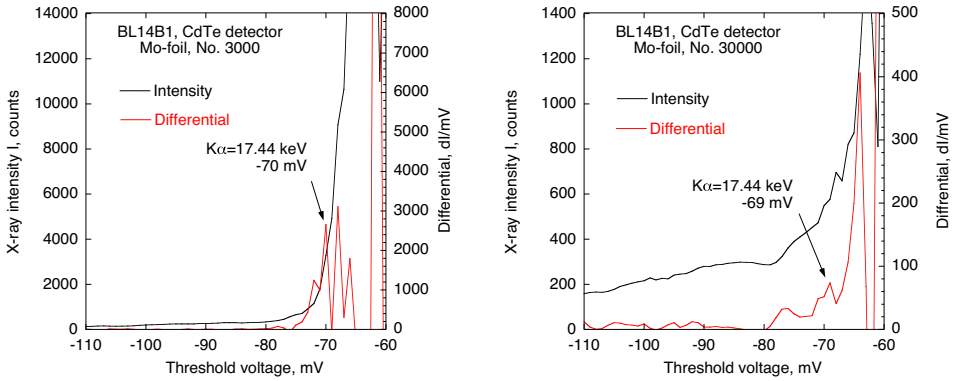
(a) Pb-foil



(b) W-foil



(c) Sn-foil



(d) Mo-foil

図 5: 閾値電圧の変化に伴う各泊についての計数および差分 (No. 3000 and No. 30000 ピクセルの例)

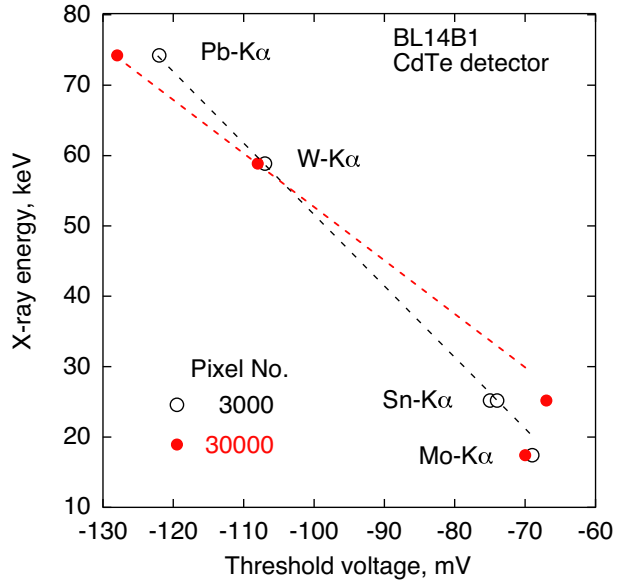


図 6: 閾値電圧の較正結果 (No. 3000 and No. 30000 ピクセルの例)

器のエネルギー較正をする方がよいかもしれない．

エネルギー較正の方針

1. 閾値電圧 vs 計数においては，3 点単純移動平均を用いる．
2. 差分は中間差分を利用する．
3. 差分の値はピクセルに入るフォトンの量に左右されるので，差分の値をカウント数で割って規格化する．この処理は，低エネルギー領域のノイズを減少する効果もある．
4. $K\beta$ のピークは $\frac{1}{I}\frac{\mathrm{d}I}{\mathrm{d}mV} \approx 0.3$，$K\alpha$ のピークは $\frac{1}{I}\frac{\mathrm{d}I}{\mathrm{d}mV} \approx 0.2$ で判別することも可能か．
5. 上記の判定に適合せず，カウントが検出されないピクセル，カウントがあっても計数値が異常なピクセルについては，別途考える．

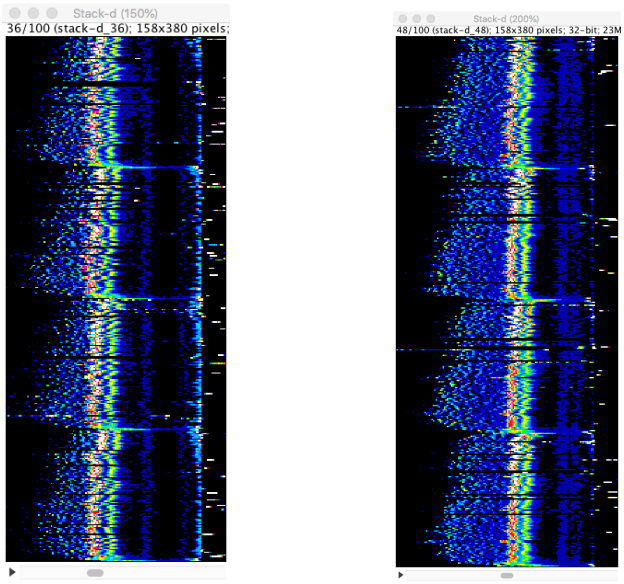
以上の方針に工夫を加えることで，図 7 のように明瞭に $K\alpha$ および $K\beta$ のピークを表示できるようになった．この図を図 4 (b) と比較すると，改良による効果がわかる．ピーク検出ができるようになったことから，以下の方針により，$K\alpha$ および $K\beta$ のピークを検出し，ファイルに出力するプログラム **calibration.f90** を作成した．

1. stack-d (100) の重ね合わせ像を見て，$K\alpha$ および $K\beta$ のピーク範囲の電圧値 v1,v2 を入力する
2. 指定範囲内で差分値の符号が正から負に変化した位置をピークとして検出する．
3. 差分像のピーク値の大きい 2 つのピクセル位置 (閾値電圧)，ピークカウントを記録する．
4. 該当のないピクセルについては，ピクセル位置=0，電圧値 $= -200$ を入力する．

測定結果は，ファイル’**peak.dat**’ として出力される．

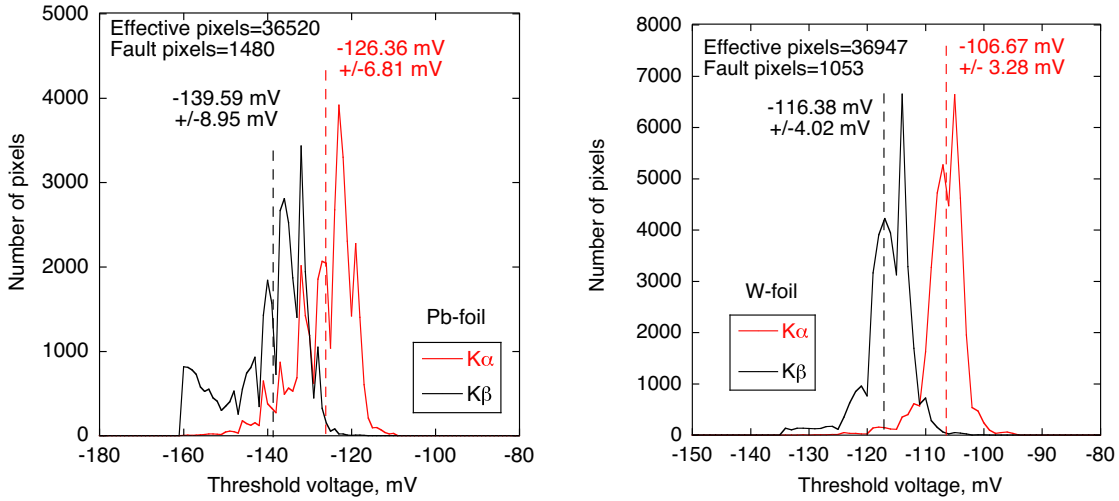
peak.dat を利用して CdTe 検出器のエネルギー構成の統計を得るために，’**evaluate.f90**’ を作成した．evaluate.f90 は，peak.dat を読み込み，38000 ピクセルの $K\alpha$ および $K\beta$ のピーク値からヒストグラム，平均値および標準偏差を計算する．さらに，有効ピクセルと欠陥ピクセルの計数する．Pb 箔および W 箔の蛍光X線を測定した結果を図 8 に示す．

図 8 からわかるように，各検出器によるずれがあり，ヒストグラムには広がりが現れている．高エネルギー側 (低電圧側) にヒストグラムの裾が広がる傾向にある．各図中には統計情報から得られた平均値



(a) Pb 箔　　(b) W 箔

図 7: 改良された差分像 ($K\alpha$ および $K\beta$ のピークが明瞭に見える)



(a) Pb 箔　　(b) W 泊

図 8: $K\alpha$ および $K\beta$ の閾値電圧の分布

(破線) および標準偏差が記されている．Pb 箔と W 箔に対する有効ピクセルと欠陥ピクセルの数を比較すると，38000 ピクセル中おおよそ 1000 ～ 1500 個のピクセルに欠陥がある．

図 8 で得られた各箔の統計情報から CdTe 検出器のエネルギー較正の式を導くために，図 9 を作成した．X線エネルギー y [keV] と閾値電圧 V[mV] の関係がほぼ直線になることから，前述の手法で得られたピーク決定の方法の妥当性を確認することができた．

図 8 の結果から，y [keV] と V[mV] の関係式として，

$$y^{[\mathrm{keV}]} = -0.7837329\, V^{[\mathrm{mV}]} - 24.48635 \tag{1}$$

が得られる．また，目的とするX線エネルギー y に対応する閾値電圧 V を求める場合は，

$$V^{[\mathrm{mV}]} = -1.275945\, y^{[\mathrm{keV}]} - 31.24323 \tag{2}$$

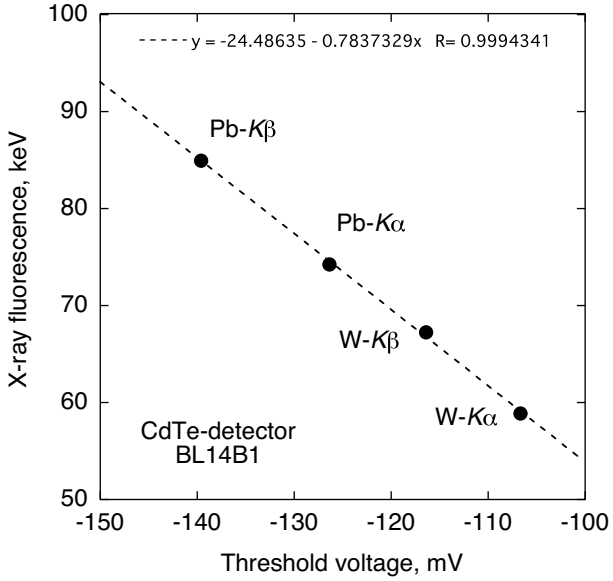


図 9: Pb および W によるエネルギー較正

となる．これらの較正式を利用して，目的とするエネルギー値に対応する閾値電圧を決定することができる．

4 CdTe 検出器により測定した回折像の一括処理

CdTe 検出器により測定すると，回折画像は大量に一連のファイル img_i.txt として保存される．これらの画像は，検出器の各位置にて閾値電圧 ($-210 \sim -60$) を変えて，2×3 の位置で測定し P_1 の画像 (151 枚) を取得する．その後，P_2 位置で 3×5 の測定した画像を同様に取得する．これらのファイルを手作業で扱うことは効率的でない．そこで，以下のツールを開発し一気に P_1，P_2 の位置のデータを処理して，tif 画像のスタックとして保存できる．

作業ディレクトリ ディレクトリを決定し，そこに一括処理する img_i.txt 群を用意する．

閾値電圧フォルダー 閾値電圧ごとのフォルダーを用意する．以下のような bat ファイルを用意して実行するとよい．データの読み書きに支障のないようにパーミッションを考えること．

```
mkdir -m 666 210
mkdir -m 666 209
mkdir -m 666 208
...
```

img_i.txt の読込み・変換 img_i.txt の読み込みと変換するために **cdte2.f90** を利用する．先頭のファイルの番号 (p)．スタートする電圧 mV(符号なし, mV)，停止する電圧 mV(符号なし, mVe)，画像の構成数 (nf)．このパラメータを入力すると 151 間隔でファイルが移動して一巡する．nf の値を入力すれば．$6 = 2 \times 3$，$15 = 3 \times 5$，$9 = 3 \times 3$ にも適用できる．作成された Image000i.txt のセットが閾値電圧ごとのフォルダーに書込まれる．

画像の合成 ImageJ のマクロ **all_image.ijm** をインストールしする．

Compose_all_2x3 および Compose_all_3x5 は，閾値電圧ごとのフォルダーにある Image000i.txt のセットを読込み，画像の合成し，各フォルダに tif で保存する．

Stack_all(2x3) および Stack_all(3x5) を利用して作成された tif イメージのスタック像を作る．

Diff_all(2x3) および Diff_all(3x5) は，中間差分像を作る．

Diff_stack_all(2x3) および Diff_stack_all(3x5) で差分像のスタックを作る．

例　一括処理の流れの一例を以下にリストする．フォルダーやファイル名などは各自の条件に合うようにカスタマイズすること．

1. 作業 dir を作成
2. 処理予定 img.txt を移動,
3. 210 ∼ 60 の dir の作成
4. cdte2.f90 コンパイルして，./a.out で実行
   データ入力
   各 dir に Image000i.txt が作成される．
5. Fiji を起動
6. Pragin → Macros → Install から all_image.ijm を読み込む．目的のマクロ処理を選択するとマクロを実行する．
   (a) Compose_all_3x5 を選択して処理開始，各 dir に貼り合わせた像 P2_xxx.tif が作成されている．
   (b) Stack_all_3x5 P2 の P2_xxx.tif による stack 像が作成される．
   (c) Diff_all(3x5) を選択し，中間差分像を作成する．
   (d) Diff_stack_all(3x5) で中間差分像のスタックを作成する．

5　二重露光法のための一括処理

二重露光法はでは，検出器の閾値電圧を変化させながら 2 つの位置 P_1 および P_2 において画面構成しなければならない．その結果，大量のファイル処理を要する．その対策として，前述のプログラム要素を組み合わせて，以下に示す二重露光法のための一括処理プログラムを作成した．

cdte2.f90　P1 または P2 の連続した img ファイル群を ImageJ のテキストファイルに変換して閾値電圧の各フォルダーごとセットする．

P1-P2.ijm　各フォルダー Image000i.txt から検出像を構成して P1-i.tif と，それらのスタック像も作成する．さらに，中間差分像 P1-$i$diff.tif と，それらのスタック像も作成する．

spot_max.ijm　中間差分像 P1-$i$diff.tif の注目している領域の最大値を閾値電圧の関係を取り出す．

なお，各自の実験に対してプログラムをカスタマイズして利用できる．

5.1　手続き

以下の手続きに沿って，処理を進めるとよい．

1. フォルダ P1 (または P2) を作成する．
2. 関係する img ファイルを P1 に移動．
3. **md.bat** で P1 の下にフォルダー (210 ∼ 60) を作成
4. フォルダー (210sim60) の everyone に読み書きを許可のパーミッションを与える．
5. **cdte2.f90** を起動して Image_000i.txt を作成．
6. ImageJ を起動．
7. Plagin → Macros → Install でマクロプログラム **P1-P2.ijm** を読み込む．
8. Plagin → Macros → P1(2x3) を選択し，処理が開始される．

一括処理された結果，P1 (または P2) 下の各フォルダー (209 ∼ 61) に以下のファイルが生成される．

P1-i.tif　検出画像．ただし，メジアンフィルタおよび移動平均処理済み．

P1-$i$diff.tif　検出画像の中間差分像

が作られる．さらに，P1 (または P2) フォルダーには，

P1_stack.tif　検出画像の閾値電圧に対する動画 (stack)

P1_diff_stack.tif　検出画像の中間差分の閾値電圧に対する動画 (stack)

6 ピクセルごとのエネルギー較正した画像

6.1 ピクセル単位のエネルギー較正のためのプログラム

閾値電圧 mV による画像は，測定結果を概観するには役立つ．しかし，正確な CdTe 検出器によるX線エネルギーによる検出画像を得るためには，ピクセルごとにエネルギー較正をする必要がある．そのためのツールを開発し，CdTe 検出器によるツールの全体としてのシステムを完成した．全体の外観を示すと図 10 のようになる．

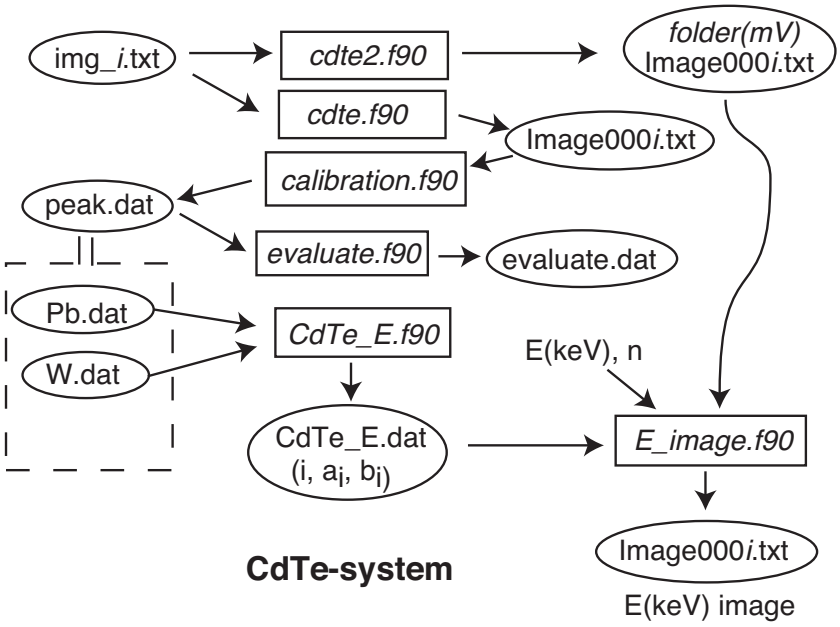


図 10: CdTe-system の構成．これらの Image000i.txt の表示や処理のために，ImageJ のマクロが用意されている．

図に示すように前節までのツールに加え，Pb および W の蛍光X線によるエネルギー較正データ peak.dat を利用して，各ピクセルごとのエネルギー較正直線

$$E_i{}^{[\mathrm{keV}]} = a_i\, V^{[\mathrm{mV}]} + b_i \tag{3}$$

を求めるプログラム **CdTe_E.f90** を作成した．プログラム CdTe_E.f90 は，その結果である 38,000 ピクセルの較正値 (i, a_i, b_i) をファイル CdTe_E.dat に出力する．

さらに，プログラム **E_image.f90** は，要求されたX線エネルギー E(keV) と較正ファイル数 $n(=6, 15)$ に従い，CdTe_E.dat に基づいて各ピクセルごとの対応する閾値電圧を計算して，閾値電圧ごとのフォルダーの image000i.txt から目的のピクセルのカウントを読み出しては，画素データを収集して，X線エネルギー E(keV) に対応した新たな画像構成ファイル Image000i.txt のセットを**ワークディレクトリ**に生成する．[6]

プログラム E_image.f90 で較正するときに重要になるのが，検出器のデータファイル img_i.txt や CdTe_E.dat のフォーマットは，一次元の array(ℓ) で格納されている．このデータフォーマットと Image000i のイメージとの対応が複雑である．データの格納については，図 1 と図 2 ですでに説明しているが，ピクセルごとのエネルギー較正をするには，図 11 にあるデータの格納と変換のしくみを理解しておくとよい．

図 11 に示すように，目的の画像の位置 (i, j) がわかったら，その位置のセグメント (A, B, C, D) を決定し，(i, j) をセグメントの位置 (g, h) に変換する．さらに，セグメントの位置 (g, h) からデータアレイ

[6] ImageJ でについては，これまで作成したマクロで十分対応できる．

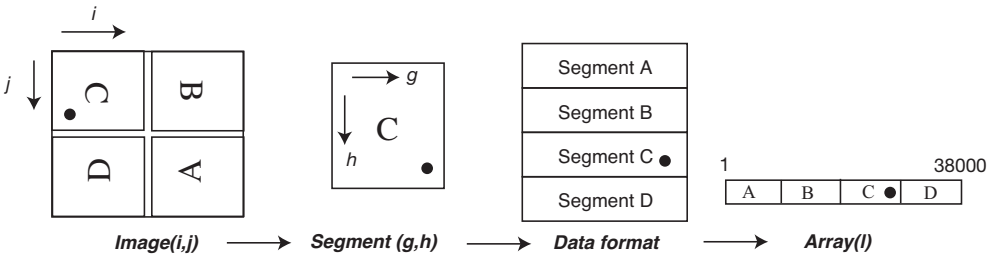
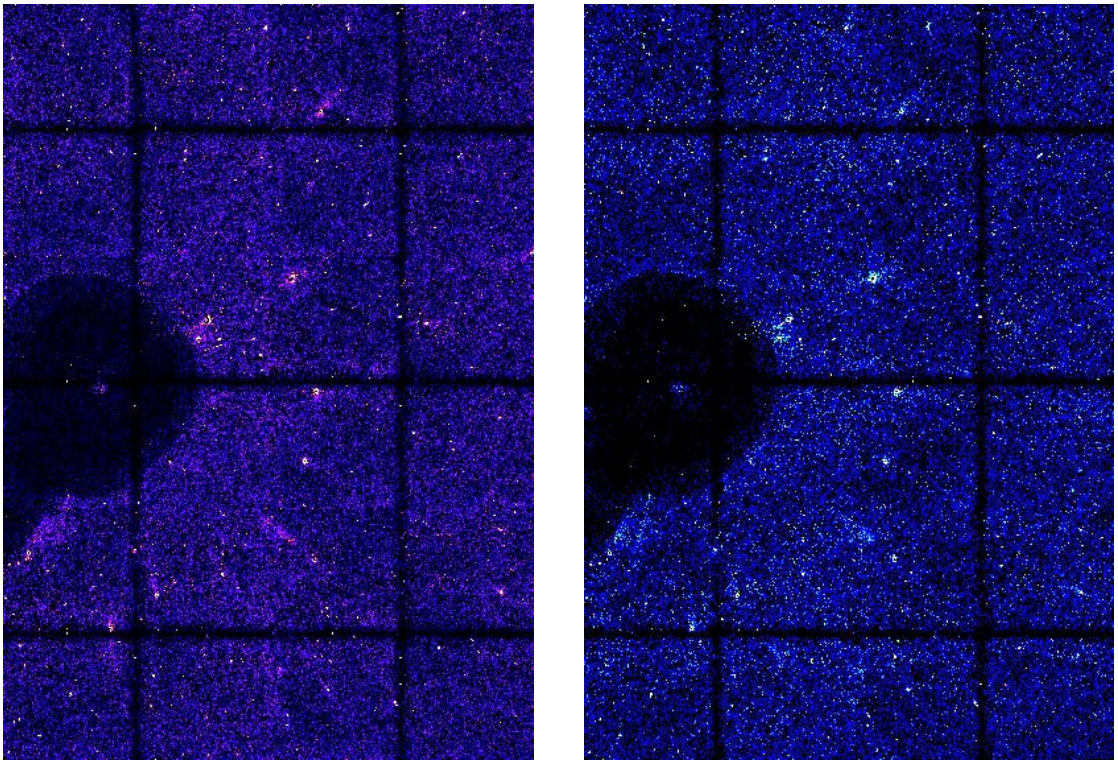


図 11: 検出画像とデータアレイとの変換

の位置 (ℓ) に変換される．その $a_(\ell, b_(\ell$ から，ようやくピクセルに対応する閾値電圧 mV が決定できる．その電圧に対応したフォルダーにある Image000i.txt のピクセル位置 (i, j) のカウントを取り出す．

CdTe-system のプログラムを理解するには，以上のしくみを理解することが必要である．

6.2 ピクセル単位でエネルギー較正した画像の比較



(a) 各ピクセルごとに較正　　(b) 平均閾値電圧で較正

図 12: CdTe 検出器のエネルギー較正による 70keV 相当の画像 (電磁鋼板)

さて，前述の各ピクセルごとのエネルギー構成するツールが完成したので，正確なX線エネルギーによる画像を取得することができるようになった．図 12 (a) は，ピクセルごとの構成をした正確なエネルギー値による画像である．一方，図 12 (b) は検出器の平均，すなわち式 (2) から計算した閾値電圧を利用してX線エネルギーを決定した画像である．

図の (a) と (b) を比較すると，ピクセルごとに較正した図 (a) の方が，図 (b) よりもカウント数も高く，より鮮鋭な画像として捉えられている[7]．測定した結果を素早く概観したいのであれば，cdte2.f90 で処

[7]測定画像を tiff で読み込み，ImageJ の 3D surface plot プロファイルで確認する必要がある．

理して読み取ればよいが，ひずみ解析などに特定のX線エネルギーの画像を使用するには，E_image.f90で処理することが望ましい．

Appendix I: Fortran program

gfortran でコンパイルして利用する．各自の実験に合わせてカスタマイズすることが必要である[8]．

cdte.f90

プログラム cdte.f90 は，img_i.txt からデータを読み出し，ImageJ で読めるテキスト画像ファイル Image000i.txt に変換する．言語は gfortran で記述した．これを使うためには，table.txt に

1. エネルギー較正のデータファイルの個数
2. ファイル名のリスト

を用意しておく．エクセルでファイル名のリストを作っておくと良い．

```
!*************************************************
!   Reading data from CdTe detector
!                     cdte.f90
!               K. Suzuki, Niigata University, July 26, 2018
!*************************************************

program cdte

character filename*128
character  bun*128
character a*128
character cfile*128

  integer :: i,j,k,l
  integer :: nf, loop
  integer*4 :: x(0:94,0:399)    !32bit data
  integer*4 :: y(0:200,0:190)   !32bit data
  integer*4 :: I1,I2,I3,I4

write(6,*) "Do you set table.txt? (y)"
read(5,*) a

open(unit=12,file="table.txt",status='old')
read (12,*) nf

do loop = 1, nf
   read(12,*) filename
   write(*,fmt='(a)',advance='no') filename
   open(unit=10,file=filename,status='old')
     do i=1,13
       read(10,*) bun
     end do
     do j=0,399
       do i=0,94
         read(10,*) k, l, x(i,j)
        end do
      end do
      close(10)

! ----------------------- compose data file
! segment c ->1, B -> 2

  n=0

! m=0-99:C, m=100:non, m=101-200: B
```

[8]Frotran のプログラミングでは，有用なサイト・書籍が用意されている．本ツール開発では，以下のサイト・書籍を活用した．
・Numerical Algorithms Group（NAG) のサイトの URL: http://www.nag-j.co.jp/fortran/index.html
・新井 親夫『Fortran90 入門 — 基礎から再帰手続きまで』森北出版 (1998).

```
    do i=0,94,1
       m=0
       do j=299,200,-1
          y(m,n)=x(i,j)
          m=m+1
       end do
       m=m+1  ! 100 boundary pixel
       do j=199,100,-1
         y(m,n)=x(i,j)
         m=m+1
       end do
       n=n+1
       end do

      n=n+1  ! 100 boundary pixel
      do i=94,0,-1
        m=0
        do j=300,399,1
          y(m,n)=x(i,j)
          m=m+1
        end do
        m=m+1  ! 100 boundary pixel
        do j=0,99,1
          y(m,n)=x(i,j)
          m=m+1
        end do
        n=n+1
        end do

! calculation counts at the boundaries
        do j=0,93
          y(99,j)=int(y(99,j)*2/3)
          y(101,j)=int(y(101,j)*2/3)
          y(100,j)=int((y(99,j)+y(101,j))/2)
        end do
        do j=97,190
          y(99,j)=int(y(99,j)*2/3)
          y(101,j)=int(y(101,j)*2/3)
          y(100,j)=int((y(99,j)+y(101,j))/2)
        end do

        do i=0,98
          y(i,94)=int(y(i,94)*2/3)
          y(i,96)=int(y(i,96)*2/3)
          y(i,95)=int((y(i,94)+y(i,96))/2)
        end do
        do i=102,200
          y(i,94)=int(y(i,94)*2/3)
          y(i,96)=int(y(i,96)*2/3)
          y(i,95)=int((y(i,94)+y(i,96))/2)
        end do

        I1=y(99,94)
        I2=y(101,94)
        I3=y(99,96)
        I4=y(101,96)

        y(99,94)=int(I1*4/9)
        y(100,94)=int((I1+I2)*2/9)
        y(101,94)=int(I2*4/9)

        y(99,95)=int((I1+I3)*4/9)
        y(100,95)=int((I1+I2+I3+I4)/9)
        y(101,95)=int((I2+I4)*4/9)
```

```
        y(99,96)=int(I3*4/9)
        y(100,96)=int((I3+I4)*2/9)
        y(101,96)=int(I4*4/9)

!  Output composed image for  ImageJ

       write (cfile, '("Image", i4.4, ".txt")') loop  !  Image000x.txt を作る
       open(unit=11,file=cfile,status='unknown')
       do j=0,190,1
         write(11,*) (y(i,j),i=0,200,1)
       end do
       close(11)
       write(6,*)  cfile
end do
close(12)
end program cdte
```

calibration.f90

calibration.f90 は，エネルギー較正用プログラムである．蛍光X線を閾値電圧を変えながら測定したデータファイル img_i.txt からデータを読み出し，閾値電圧で整理したカウント像および差分像を表示する．gnuplot 用に c_map_000i.txt (カウント像) と d_map_000i.txt (差分像) 並びに ImageJ 用に stack_i.txt および stack-d_i.txt を生成する．最後に，calibration.f90 はファイル peak.dat を出力する．この中に 38000 ピクセルの $K\alpha$, $K\beta$ の閾値電圧が記録されている．

stack_i.txt および stack-d_i.txt は，それぞれの専用フォルダー stack．stack-d に移動して，次に示す Stack100.ijm のマクロ stack(100) および stack-d(100) で重ね合せ像を表示できる．この stack-d(100) で作成した tiff ファイルを読みながら，$K\beta$, $K\alpha$ のピーク処理を施す x の範囲 v1, v2 (pixel 単位) を調べ，用意しておく．それが完了したら，Please make stack and stack-d folders. Ok?の問に対し yes を入力すると，v1, v2 の入力を求められる．最終的に，エネルギー較正に必要な peak.dat が出力される．

```
!*****************************************
!     calibration.f90          ver 2.0
!         Calibration of CdTe detector
!    by Kenji SUZUKI, Niigata University
!  380x100 files
!*****************************************

program calibration

 character filename*128
 character bun*128
 character a*128
 character cfile*128
 real:: dx,y,z
  integer :: dev(0:200)
  integer :: i,j,k,l,p
  integer :: V0, dV, n, loop, V
  integer*4 :: x(0:170,0:40000)  ! 32 bit data
  integer*4 :: x3(0:170,0:40000)  ! 32 bit data
!   peak search
  integer :: v1, v2, peak(1:100),m,u,p1,p2



write (*,fmt='(a)', advance='no') 'Do you set input.txt? (y)'
write(6,*)
read(5,*) a
open(unit=12,file='calibration.txt',status='old') ! list of reading files
read (12,*) V0,dV, n
```

```
V=V0
do loop = 1, n

 read(12,*) filename
! write(*,fmt='(a)',advance='no') filename

 open(unit=10,file=filename,status='old')
 write(6,*) filename
 do i=1,13
   read(10,*) bun
 end do

 p=0
 do j=0,399
  do i=0,94
  read(10,*) k, l, x(loop,p)
  p=p+1
  end do
 end do
 close(10)
end do
close(12)

! Output calibration files

p=0

do i=1,100

 write (cfile, '("c_map_", i4.4, ".txt")') i     ! ここでファイル名を生成している
 open(unit=11,file=cfile,status='unknown') ! files for gnuplot

 write(11,*) '#calibration map for gnu_plot'
 write(11,*) '# mV  i  counts'

 do j=1,380
   V=V0
   x3(1,p)=x(1,p)
   do k=2,n-1
     x3(k,p)=Int((x(k-1,p)+x(k,p)+x(k+1,p))/3)   ! 3 points  simple moving average
     write(11,*) V,p,x3(k,p)
     V=V+dV
   end do
   x3(n,p)=x(n,p)
   p=p+1
   write(11,*) ''  ! write LF
 end do
 close(11)
end do


! Search peak  using 2nd derivatives

p=0

do i=1,100

 write (cfile, '("d_map_", i4.4, ".txt")') i     ! ここでファイル名を生成している
 open(unit=11,file=cfile,status='unknown') ! files for gnuplot

 write(11,*) '#calibration map for gnu_plot'
 write(11,*) '# mV  i  counts'
```

```
 do j=1,380
   V=V0
   do k=2,n-1
     dev(k)=(x3(k+1,p)-x3(k-1,p))/float(dV)/2.0
     write(11,*) V,p,dev(k)
     V=V+dV
   end do
   p=p+1
   write(11,*) ''  ! write LF
 end do
 close(11)
end do


! Output calibration files for ImageJ

p=0

do i=1,100
 write(cfile,*) i
 cfile='stack_'//trim(adjustl(cfile))//'.txt' !adjustl で左寄せにしてから trim で末尾の空白除去，拡張子等をくっつける
 open(unit=11,file=cfile,status='unknown') ! files with Text Image for ImageJ

 do j=1,380
     write(11,*) (x3(k,p), k=1,n,1)
     p=p+1
 end do
 close(11)
end do


! Output calibration files for ImageJ

p=0

do i=1,100
 write(cfile,*) i
 cfile='stack-d_'//trim(adjustl(cfile))//'.txt' !adjustl で左寄せにしてから trim で末尾の空白除去，拡張子等をくっつける
 open(unit=11,file=cfile,status='unknown') ! files with Text Image for ImageJ

 do j=1,380
   do k=2,n-1
     z=x3(k,p)
     if (z .le. 10.0) then
       z=50.0
     end if
     y=(x3(k+1,p)-x3(k-1,p))/dV/2/z*1000.0
       if (y .lt. 0.0) then
        y=0.0
       end if
     dev(k)=y
   end do
        write(11,*) (dev(k), k=2,n-1,1)
        p=p+1
 end do
 close(11)
end do


! Peak search

  write(6,*) 'Please make stack and stack-d folders. Ok? (yes)'
```

```
  read(5,*) a
open(unit=11, file='peak.dat', status='unknown')

  write(6,*) 'Please input  start and end pixel numbers v1, v2.'
  read(5,*) v1,v2
 do p=1,38000
   do i=1,100
     peak(i)=0
   end do
   do k=2,n-1
     z=x3(k,p)
     y=(x3(k+1,p)-x3(k-1,p))/dV/2/z*1000.0
     if (y .lt. 0.0) then
        y=0.0
     end if
     dev(k)=y
      end do

   i=1
   do k=v1,v2
    if (((dev(k+1)-dev(k)).lt. 0).and.((dev(k)-dev(k-1)).gt. 0)) then  !peak
      peak(i)=k
 i=i+1
! write(6,*) k,dev(k)
    end if
   end do
   m=i-1
!                             sorting
   do j=m,2,-1
    do i=2,j
     if (dev(peak(i)).gt.dev(peak(i-1))) then
      u=peak(i)
      peak(i)=peak(i-1)
 peak(i-1)=u
     end if
    end do
   end do

   do k=v1,v2
     if((x(k,p).gt.1200000).or.(x(k,p).lt. -1200000)) then
      peak(1)=0
      peak(2)=0
     end if
   end do
   if (peak(1).gt.peak(2)) then
    p1=peak(1)
    peak(1)=peak(2)
    peak(2)=p1
   end if

   p1= dev(peak(1))  !K β
   p2= dev(peak(2))  !K α
   write(11,*) p,V0+peak(1)*dV, p1, V0+peak(2)*dV, p2
 end do
 close(11)


end program calibration
```

evaluate.f90

```
!*****************************************************************
! Evaluation of peak.dat     ver. 2
```

```
!     evaluate.f90
!   Statics in calibration of pixels
!                                        by K. Suzuki Niigata University
!******************************************************** Oct. 12, 2018

program evaluate

  character*30 ::a,filename
  integer :: x1(1:210)=0, x2(1:210)=0     ! Kb, Ka
  integer :: i,n,v1(1:39000),v2(1:39000),j,m1=0,fault=0
  real :: mean1=0.0,std1=0.0, mean2=0.0,std2=0.0
  real :: y1,y2,z1, z2
!  1=Kb, 2=Ka
!
  print *, 'Enter file name: '
  read '(A)', filename
  open(11,file=filename,status='old')
  n = 0
  do
    read(11,'()',end=100)
    n = n + 1
  end do
100 close(11)
  print *, n

  open(unit=11,file=filename,status='old')

  m=0
  do i=1,n
    read(11,*) j,v1(i),y1,v2(i),y2
    if (v2(i).eq.-200) then
     y2=0
     fault=fault+1
    else
 m=m+1
 j=-v1(i)
     x1(j)=x1(j)+1
 j=-v2(i)
     x2(j)=x2(j)+1
    end if
  end do
  close(11)

!                                    mean
  do i=40,199
   mean1=mean1-real(x1(i)*i)
   mean2=mean2-real(x2(i)*i)
  end do
  mean1=mean1/real(m)
  mean2=mean2/real(m)

  do i=40,199
    z1=x1(i)*(real(i)+mean1)**2
    std1=std1+z1
z2=x2(i)*(real(i)+mean2)**2
    std2=std2+z2
  end do
  std1 = sqrt(std1/(m-1))
  std2 = sqrt(std2/(m-1))

  write(6,*) 'K-beta Mean =',mean1, 'Std=',std1
  write(6,*) 'K-alpha Mean =',mean2, 'Std=',std2
  write(6,*) ' Number of data=',m, 'Number of fault data=', fault
```

```
  open(unit=10,file='evaluation.dat',status='unknown')
  write(10,*) 'K-beta Mean =',mean1, 'Std=',std1
  write(10,*) 'K-alpha Mean =',mean2, 'Std=',std2
  write(10,*) ' Number of data=',m, 'Number of fault data=', fault
  do i=40,199
   write(10,*) -i,x1(i), x2(i)
  end do
  close(10)

end program evaluate
```

cdte2.f90

```
!*************************************************
!   Reading data from CdTe detector
!                     cdte2.f90, ver. 2.0
!        K. Suzuki, Niigata University, Aug 21, 2018
!*************************************************

program cdte

character filename*128
character  bun*128
character a*4, dir*6
character cfile*128,xfile*20

  integer :: i,j,k,l, mV, p,ix,ie,mVe
  integer :: nf, loop
  integer*4 :: x(0:94,0:399)    !32bit data
  integer*4 :: y(0:200,0:190)   !32bit data
  integer*4 :: I1,I2,I3,I4

!write (*,fmt='(a)', advance='no') 'How many files for CdTe img.txt :   '
write(6,*) "Are you ready? (y)"
read(5,*) a

write(6,*) "Input p of strat img_p.txt="
read (5,*) p

write(6,*) "Input strat mV without sign"
read (5,*) mV

write(6,*) "Input end mV="
read (5,*) mVe

write(6,*) "Number of images for composition="
read (5,*) nf


do ie=1,mV-mVe+1

do loop = 1, nf
 ix=p+(loop-1)*151
 write(filename,*) ix
 filename='img_'//trim(adjustl(filename))//'.txt'
 open(unit=10,file=filename,status='old')
 do i=1,13
   read(10,*) bun
!  write(6,*) bun
 end do

 do j=0,399
   do i=0,94
```

```
  read(10,*) k, l, x(i,j)
! write(6,*) k,l, x(i,j)
   end do
  end do
  close(10)


! ----------------------- compose data file
! segment c ->1, B -> 2

 n=0

!m=0-99:C, m=100:non, m=101-200: B
 do i=0,94,1
 m=0
   do j=299,200,-1
  y(m,n)=x(i,j)
  m=m+1
 end do
 m=m+1  ! 100 boundary pixel
  do j=199,100,-1
   y(m,n)=x(i,j)
   m=m+1
  end do
 n=n+1
 end do

 n=n+1  ! 100 boundary pixel

 do i=94,0,-1
  m=0
    do j=300,399,1
  y(m,n)=x(i,j)
  m=m+1
 end do
 m=m+1  ! 100 boundary pixel
 do j=0,99,1
  y(m,n)=x(i,j)
  m=m+1
 end do
 n=n+1
 end do

! calculation counts at the boundaries
 do j=0,93
  y(99,j)=int(y(99,j)*2/3)
  y(101,j)=int(y(101,j)*2/3)
  y(100,j)=int((y(99,j)+y(101,j))/2)
 end do
 do j=97,190
  y(99,j)=int(y(99,j)*2/3)
  y(101,j)=int(y(101,j)*2/3)
  y(100,j)=int((y(99,j)+y(101,j))/2)
 end do

 do i=0,98
  y(i,94)=int(y(i,94)*2/3)
  y(i,96)=int(y(i,96)*2/3)
  y(i,95)=int((y(i,94)+y(i,96))/2)
 end do
 do i=102,200
  y(i,94)=int(y(i,94)*2/3)
  y(i,96)=int(y(i,96)*2/3)
  y(i,95)=int((y(i,94)+y(i,96))/2)
```

```
 end do

 I1=y(99,94)
 I2=y(101,94)
 I3=y(99,96)
 I4=y(101,96)

 y(99,94)=int(I1*4/9)
 y(100,94)=int((I1+I2)*2/9)
 y(101,94)=int(I2*4/9)

 y(99,95)=int((I1+I3)*4/9)
 y(100,95)=int((I1+I2+I3+I4)/9)
 y(101,95)=int((I2+I4)*4/9)

 y(99,96)=int(I3*4/9)
 y(100,96)=int((I3+I4)*2/9)
 y(101,96)=int(I4*4/9)

!  Output composit image for  ImageJ
      write(dir,'(I4)') mV
 write (xfile,'("Image", i4.4, ".txt")') loop     ! ここでファイル名を生成している
 cfile=trim(adjustl(dir))//"/"//xfile
 write(6,*) cfile

 open(unit=11,file=cfile,status='unknown')
 do j=0,190,1
  write(11,*) (y(i,j),i=0,200,1)
 end do
 close(11)

 write(6,*)  cfile
end do

p=p+1
mV=mV-1
end do
end program cdte
```

cdte2.f90

```
!**************************************************
!   Reading data from CdTe detector
!                     cdte2.f90, ver. 2.0
!        K. Suzuki, Niigata University, Aug 21, 2018
!**************************************************

program cdte

character filename*128
character  bun*128
character a*4, dir*6
character cfile*128,xfile*20


  integer :: i,j,k,l, mV, p,ix,ie,mVe
  integer :: nf, loop
  integer*4 :: x(0:94,0:399)    !32bit data
  integer*4 :: y(0:200,0:190)   !32bit data
  integer*4 :: I1,I2,I3,I4

!write (*,fmt='(a)', advance='no') 'How many files for CdTe img.txt :   '
write(6,*) "Are you ready? (y)"
```

```
read(5,*) a

write(6,*) "Input p of strat img_p.txt="
read (5,*) p

write(6,*) "Input strat mV without sign"
read (5,*) mV

write(6,*) "Input end mV="
read (5,*) mVe

write(6,*) "Number of images for composition="
read (5,*) nf

do ie=1,mV-mVe+1

do loop = 1, nf
 ix=p+(loop-1)*151
 write(filename,*) ix
 filename='img_'//trim(adjustl(filename))//'.txt'
 open(unit=10,file=filename,status='old')
 do i=1,13
   read(10,*) bun
!  write(6,*) bun
 end do

 do j=0,399
   do i=0,94
  read(10,*) k, l, x(i,j)
! write(6,*) k,l, x(i,j)
   end do
  end do
  close(10)

! ----------------------- compose data file
! segment c ->1, B -> 2

 n=0

!m=0-99:C, m=100:non, m=101-200: B
 do i=0,94,1
 m=0
   do j=299,200,-1
  y(m,n)=x(i,j)
  m=m+1
 end do
 m=m+1  ! 100 boundary pixel
  do j=199,100,-1
   y(m,n)=x(i,j)
   m=m+1
  end do
 n=n+1
 end do

 n=n+1  ! 100 boundary pixel

 do i=94,0,-1
  m=0
    do j=300,399,1
  y(m,n)=x(i,j)
  m=m+1
 end do
 m=m+1  ! 100 boundary pixel
 do j=0,99,1
```

```
  y(m,n)=x(i,j)
  m=m+1
 end do
 n=n+1
 end do

! calculation counts at the boundaries
 do j=0,93
  y(99,j)=int(y(99,j)*2/3)
  y(101,j)=int(y(101,j)*2/3)
  y(100,j)=int((y(99,j)+y(101,j))/2)
 end do
 do j=97,190
  y(99,j)=int(y(99,j)*2/3)
  y(101,j)=int(y(101,j)*2/3)
  y(100,j)=int((y(99,j)+y(101,j))/2)
 end do

 do i=0,98
  y(i,94)=int(y(i,94)*2/3)
  y(i,96)=int(y(i,96)*2/3)
  y(i,95)=int((y(i,94)+y(i,96))/2)
 end do
 do i=102,200
  y(i,94)=int(y(i,94)*2/3)
  y(i,96)=int(y(i,96)*2/3)
  y(i,95)=int((y(i,94)+y(i,96))/2)
 end do

 I1=y(99,94)
 I2=y(101,94)
 I3=y(99,96)
 I4=y(101,96)

 y(99,94)=int(I1*4/9)
 y(100,94)=int((I1+I2)*2/9)
 y(101,94)=int(I2*4/9)

 y(99,95)=int((I1+I3)*4/9)
 y(100,95)=int((I1+I2+I3+I4)/9)
 y(101,95)=int((I2+I4)*4/9)

 y(99,96)=int(I3*4/9)
 y(100,96)=int((I3+I4)*2/9)
 y(101,96)=int(I4*4/9)

!  Output composit image for  ImageJ
      write(dir,'(I4)') mV
 write (xfile,'("Image", i4.4, ".txt")') loop     ! ここでファイル名を生成している
 cfile=trim(adjustl(dir))//"/"//xfile
 write(6,*) cfile

 open(unit=11,file=cfile,status='unknown')
 do j=0,190,1
  write(11,*) (y(i,j),i=0,200,1)
 end do
 close(11)

 write(6,*)  cfile
end do

p=p+1
mV=mV-1
end do
```

```
end program cdte
```

CdTe_E.f90

CdTe_E.f90 は，蛍光X線の検出結果を calibration.f90 で処理して得られた各ピクセルの蛍光X線のピーク位置 ($K\beta$, $K\alpha$ の電圧値を格納した peak.dat を使用する．そのため，各 peak.dat のファイル名を W.dat, Pb.dat に rename して読み込ませる．CdTe_E.f90 は，各ピクセルのエネルギー値の較正式のデータ $_1$, a_i, b_i を計算して，CdTe_E.dat として出力する．

```
!------------------------- Energy calibration  of CdTe detector ---------------
!                  CdTe_E.f90        ver. 2.0
!                             by Kenji SUZUKI, Niigata University, 12 October, 2018
!*************************************************************************
!  本バージョンでは Pb,W, Sn, Mo の Ka のデータを読むように変更.

program CdTe_E

  character*30 :: filename
  integer :: i,j,k,l
  real :: mV(1:38000,1:4), value1,value2, Kb,d1,d2,d3,d4
  real :: E(1:4), a(1:38000),b(1:38000)
  real :: m11,m12,c1,c2,n
!  data E/84.94,74.25,67.24,58.87/ ! E(1)=Pb-Kβ, E(2)=Pb-Kα, E(3)=W-Kβ, E(4)=W-K
α
data E/74.25, 58.87, 25.20, 17.44/ ! E(1)=Pb-Ka, E(2)=W-Kα, E(3)=Sn-Ka, E(4)=Mo-K
α

!    read peak.dat
  print *, 'Enter file name for Pb: '
  read '(A)', filename
  open(11,file=filename,status='old')

  do j=1,38000
    read(11,*) i,Kb,value1,mV(i,1),value2
  end do
  close(11)

  print *, 'Enter file name for W: '
  read '(A)', filename
  open(11,file=filename,status='old')

  do j=1,38000
    read(11,*) i,Kb,value1,mV(i,2),value2
  end do
  close(11)

  print *, 'Enter file name for Sn: '
  read '(A)', filename
  open(11,file=filename,status='old')

  do j=1,38000
    read(11,*) i,Kb,value1,mV(i,3),value2
  end do
  close(11)

  print *, 'Enter file name for Mo: '
  read '(A)', filename
  open(11,file=filename,status='old')

  do j=1,38000
    read(11,*) i,Kb,value1,mV(i,4),value2
  end do
```

```
  close(11)

!    Least squares method
  n=4.0
  do i=1,38000
    m11=0.0
    m12=0.0
    c1=0.0
    c2=0.0
    do j=1,4
     m11=m11+mV(i,j)**2
     m12=m12+mV(i,j)
     c1=c1+E(j)*mV(i,j)
     c2=c2+E(j)
    end do

    a(i)=(m12*c2-n*c1)/(m12**2-n*m11)
    b(i)=(m12*c1-m11*c2)/(m12**2-n*m11)
d1=mV(i,1)
d2=mV(i,2)
d3=mV(i,3)
d4=mV(i,4)
    if ((d1.LT.-199.).or.(d2.LT.-199.).or.(d3.LT.-199.).or.(d4.LT.-199.)) then
      a(i)=0.0
      b(i)=0.0
    end if
  end do

!   file out CdTe_E.dat
  open(12,file='CdTe_E.dat', status='unknown')
    do i=1,38000
      write(12,*) i, a(i), b(i)
    end do
  close(12)

end program CdTe_E
```

E_image.f90

E_image.f90 は，要求されたX線エネルギー E に対応するX線回折像を得るために，画像の各ピクセル (i, j) に対応した閾値電圧を CdTe_E.dat に基づいて計算し，それらの閾値電圧フォルダーにある Image000i.txt からピクセルのカウントを取得して，画像データを作成し，ワークディレクトリーに Image000i.txt を出力する.

```
!********************** Image by X-ray energy *********************
!       E_image.f90   ver1.0
!                          Kenji SUZUKI, Niigata University
!---------------------------------------- 20th September, 2018 -----------

program E_image
  real :: a(1:38000),b(1:38000), En, mV
  integer :: i,j,k,l,n,dir
  integer*4 :: y(0:200,0:190),br   !32bit data,
  integer*4 :: I1,I2,I3,I4
  character cfile*128

!   read  a and b for calibration of CdTe-detector
  print *, 'CdTe_E.dat'
  open(11,file='CdTe_E.dat', status='old')
    do i=1,38000
      read(11,*) j, a(j), b(j)
    end do
  close(11)
```

```
  print *, 'Input number of image file set, n='  !6 or 15
  read(5,*) n
  print *, 'Input X-ray energy of image, En (keV)='
  read(5,*) En

  do k=1,n

    do j=0,190
     do i=0,200
      call Pixel (i,j,l)
      if ((l.ne.0).and.(a(l).ne.0.0)) then
        mV=(En-b(l))/a(l)
        dir=nint(mV)
       else
        dir=0
      end if
!      print *, i,j,dir
      if ((dir .ge. -210).and.(dir .lt. -60)) then
        call Bright(i,j,dir,br,k)
        y(i,j)=br
   else
    y(i,j)=0
      end if
     end do
     print *,j
    end do

  ! calculation counts at the boundaries
  do j=0,93
    y(99,j)=int(y(99,j)*2/3)
    y(101,j)=int(y(101,j)*2/3)
    y(100,j)=int((y(99,j)+y(101,j))/2)
  end do
  do j=97,190
    y(99,j)=int(y(99,j)*2/3)
    y(101,j)=int(y(101,j)*2/3)
    y(100,j)=int((y(99,j)+y(101,j))/2)
  end do

  do i=0,98
    y(i,94)=int(y(i,94)*2/3)
    y(i,96)=int(y(i,96)*2/3)
    y(i,95)=int((y(i,94)+y(i,96))/2)
  end do
  do i=102,200
    y(i,94)=int(y(i,94)*2/3)
    y(i,96)=int(y(i,96)*2/3)
    y(i,95)=int((y(i,94)+y(i,96))/2)
  end do

  I1=y(99,94)
  I2=y(101,94)
  I3=y(99,96)
  I4=y(101,96)

  y(99,94)=int(I1*4/9)
  y(100,94)=int((I1+I2)*2/9)
  y(101,94)=int(I2*4/9)

  y(99,95)=int((I1+I3)*4/9)
  y(100,95)=int((I1+I2+I3+I4)/9)
  y(101,95)=int((I2+I4)*4/9)

  y(99,96)=int(I3*4/9)
```

```
  y(100,96)=int((I3+I4)*2/9)
  y(101,96)=int(I4*4/9)

!  Output energy image for  ImageJ
! Image000i.txt は, Composition.ijm のマクロを利用して合成します.

  write (cfile, '("Image", i4.4, ".txt")') k    ! ここでファイル名を生成している
  open(unit=12,file=cfile,status='unknown')
  do j=0,190,1
    write(12,*) (y(i,j),i=0,200,1)
  end do
  close(12)
  write(6,*)  cfile
  end do

end program E_image

!---------------------------------------SUBROUTINE -----------------------------
! transration of parameters, Image(i,j) >  Segment(g,h)  >  Array(l)
subroutine Pixel (i,j,l)
  integer :: i,j,l,g,h

!   translation from Image(i,j) into Segments( A, B, C, D
  if ((i<=99).and.(j<=94)) then ! segment C
    g=94-j
    h=299-i
    l=95*h+g
   else if ((i>=101).and.(j<=94)) then ! segment B
    g=j
    h=200-i
    l=95*h+g
   else if ((i<=99).and.(j>=96)) then ! segment D
    g=190-j
    h=i+300
    l=95*h+g
   else if ((i>=101).and.(j>=96)) then ! segment A
    g=190-j
    h=200-i
    l=95*h+g
   else if ((i==100).or.(j==95)) then  ! boundary
    l=0
  end if
end subroutine Pixel

! Reading brightness from each energy directory
subroutine Bright(m,n,dir,a,k)
  integer*4 :: y(0:200,0:190),a   !32bit data, a: brightness
  integer :: i,j,k,dir,mV, m,n
  character cfile*128,xfile*20,fl*6

!  input file
    if (dir.ne.0) then
      mV=-dir
      write(fl,'(I4)') mV
      write (xfile,'("Image", i4.4, ".txt")') k     ! ここでファイル名 xfile を生成してい
る
      cfile=trim(adjustl(fl))//"/"//xfile   ! cfile= folder+/+Image i .txt を作成.
!  write(6,*) cfile
      open(unit=11,file=cfile,status='old')
      do j=0,n-1,1
        read(11,*) !読飛ばし
      end do
      do j=n,n
        read(11,*) (y(i,j),i=0,m,1)  !読込
```

```
      end do
      close(11)
      a=y(m,n)
     else
      a=0
    end if
end subroutine Bright
```

E2_image.f90

E2_image.f90 は，E_image.f90 と同じであるが，Composition.ijm を利用する必要がなく，X線エネルギー $E_1 \sim E_2$ (keV) の像 1 keV ステップで連続して作成する．

```
!********************** Image by X-ray energy *********************
!       E2_image.f90   ver3.0
!                          Kenji SUZUKI, Niigata University
!---------------------------------------- 18th October, 2018 -----

program E_image
  real :: a(1:38000),b(1:38000), En, mV
  integer :: i,j,k,l,n,dir,m1,n1,keV,e1,e2
  integer*4 :: y(0:200,0:190),br   !32bit data,
  integer*4 :: I1,I2,I3,I4
  character cfile*128

!   read  a and b for calibration of CdTe-detector
  print *, 'CdTe_E.dat'
  open(11,file='CdTe_E.dat', status='old')
    do i=1,38000
      read(11,*) j, a(j), b(j)
    end do
  close(11)

  print *, 'Input number of image file set, m,n (m x n, 2x3, 4x5)='  !
  read(5,*) m1, n1
  n=m1*n1
  print *, 'Input range of X-ray energy, E1, E2 (keV)='
  read(5,*) e1,e2

 do keV=e1,e2
  En=float(keV)

  do k=1,n

    do j=0,190
     do i=0,200
      call Pixel (i,j,l)
      if ((l.ne.0).and.(a(l).ne.0.0)) then
        mV=(En-b(l))/a(l)
        dir=nint(mV)
       else
        dir=0
      end if
!      print *, i,j,dir
      if ((dir .ge. -210).and.(dir .lt. -60)) then
        call Bright(i,j,dir,br,k)
        y(i,j)=br
   else
    y(i,j)=0
      end if
     end do
     print *,j
    end do
```

```
  ! calculation counts at the boundaries
  do j=0,93
    y(99,j)=int(y(99,j)*2/3)
    y(101,j)=int(y(101,j)*2/3)
    y(100,j)=int((y(99,j)+y(101,j))/2)
  end do
  do j=97,190
    y(99,j)=int(y(99,j)*2/3)
    y(101,j)=int(y(101,j)*2/3)
    y(100,j)=int((y(99,j)+y(101,j))/2)
  end do

  do i=0,98
    y(i,94)=int(y(i,94)*2/3)
    y(i,96)=int(y(i,96)*2/3)
    y(i,95)=int((y(i,94)+y(i,96))/2)
  end do
  do i=102,200
    y(i,94)=int(y(i,94)*2/3)
    y(i,96)=int(y(i,96)*2/3)
    y(i,95)=int((y(i,94)+y(i,96))/2)
  end do

  I1=y(99,94)
  I2=y(101,94)
  I3=y(99,96)
  I4=y(101,96)

  y(99,94)=int(I1*4/9)
  y(100,94)=int((I1+I2)*2/9)
  y(101,94)=int(I2*4/9)

  y(99,95)=int((I1+I3)*4/9)
  y(100,95)=int((I1+I2+I3+I4)/9)
  y(101,95)=int((I2+I4)*4/9)

  y(99,96)=int(I3*4/9)
  y(100,96)=int((I3+I4)*2/9)
  y(101,96)=int(I4*4/9)

!  Output energy image for  ImageJ
! Image000i.txt は, Composition.ijm のマクロを利用して合成します.

  write (cfile, '("Image", i4.4, ".txt")') k    ! ここでファイル名を生成している
  open(unit=12,file=cfile,status='unknown')
  do j=0,190,1
    write(12,*) (y(i,j),i=0,200,1)
  end do
  close(12)
  write(6,*)  cfile
  end do

  call Composition(m1,n1,keV)  ! このサブルーチンで keV 画像を合成する.
 end do
end program E_image

!---------------------------------------SUBROUTINE -----------------------------
! transration of parameters, Image(i,j) >  Segment(g,h)  >  Array(l)
subroutine Pixel (i,j,l)
  integer :: i,j,l,g,h

!   translation from Image(i,j) into Segments( A, B, C, D
  if ((i<=99).and.(j<=94)) then ! segment C
```

```
    g=94-j
    h=299-i
    l=95*h+g
   else if ((i>=101).and.(j<=94)) then ! segment B
    g=j
    h=200-i
    l=95*h+g
   else if ((i<=99).and.(j>=96)) then ! segment D
    g=190-j
    h=i+300
    l=95*h+g
   else if ((i>=101).and.(j>=96)) then ! segment A
    g=190-j
    h=200-i
    l=95*h+g
   else if ((i==100).or.(j==95)) then  ! boundary
    l=0
  end if
end subroutine Pixel

! Reading brightness from each energy directory
subroutine Bright(m,n,dir,a,k)
  integer*4 :: y(0:200,0:190),a   !32bit data, a: brightness
  integer :: i,j,k,dir,mV, m,n
  character cfile*128,xfile*20,fl*6

!  input file
    if (dir.ne.0) then
      mV=-dir
      write(fl,'(I4)') mV
      write (xfile,'("Image", i4.4, ".txt")') k     ! ここでファイル名 xfile を生成してい
る
      cfile=trim(adjustl(fl))//"/"//xfile   ! cfile= folder+/+Image i .txt を作成.
!  write(6,*) cfile
      open(unit=11,file=cfile,status='old')
      do j=0,n-1,1
        read(11,*) !読飛ばし
      end do
      do j=n,n
        read(11,*) (y(i,j),i=0,m,1)  !読込
      end do
      close(11)
      a=y(m,n)
     else
      a=0
    end if
end subroutine Bright

!-------------------- SUBROUTINE composition -------------------
subroutine Composition(m,n,keV)
  integer :: i,j,m,n,keV, k,l, No,h
  integer*4 :: x(1:2000,1:2000),y(1:2000) !
  character cfile*128

  No=m*n
  do j=1,n
    do i=1,m
      write (cfile,'("Image", i4.4, ".txt")') No     ! Image000No.txt を用意
      print*,cfile
      open(unit=12,file=cfile,status='old')
      do l=1+(j-1)*190,j*190
       read(12,*) (y(h),h=1,200,1)
       h=1
       do k=1+(i-1)*200,i*200
```

```
          x(k,l)=y(h)
          h=h+1
        end do
       end do
       close(12)
       No=No-1
     end do
   end do


!  xxkeV.txt
   write (cfile,*) keV   ! ここでファイル名を生成
   cfile=trim(adjustl(cfile))//'keV.txt'
    print *,cfile
   open(unit=11,file=cfile,status='unknown')
  do j=1,n*190
   write(11,*) (x(i,j),i=1,200*m,1)
  end do
  close(11)
end subroutine Composition
```

Appendix II: Macro-program for ImageJ

ImageJ のマクロプログラムである．各自の実験に合わせてカスタマイズして利用することが必要である[9].

Composition.ijm

Composition.ijm は，ImageJ マクロであり，ImageJ で Image0001.txt ∼ Image000n.txt まで読み込み．スタックを作成してから，n 個の画像を貼り合わせるマクロである．なお，組み合わせは，2×3, 3×5,3×3 の 3 種類のマクロプログラムを示す[10]．Image000n.txt を格納している作業ホルダーを選択すれば，読み取りを開始します．ImageJ の log の window に読み取ったファイルを順番に示します.

```
macro "Compose 2X3 [1]"{
//input Image files,  Image to stack and then Make stack
  dir = getDirectory("Choose a Directory");
  for (i=0; i<6;i++){
   j=6-i;
   cnt = "0000"+j;
   path1 ="open="+dir+"Image"+substring(cnt,lengthOf(cnt)-4,lengthOf(cnt))+".txt";
   print(path1);
   run("Text Image... ", path1);
   setMinAndMax(0, 1000);
   run("Fire");
  }
// make stack, make montage
run("Images to Stack", "name=Stack title=[] use");
run("Make Montage...", "columns=2 rows=3 scale=1");
}

macro "Compose 3X5 [2]"{
  dir = getDirectory("Choose a Directory");
  for (i=0; i<15;i++){
   j=15-i;
```

[9]ImageJ の利用に際しては有用な書籍やサイトが多数用意されている．本ツール開発には以下の書籍，サイトを活用した．
・三浦耕太，塚田祐基『ImageJ ではじめる生物画像解析』学研プラス (2016).
・ImageJ 日本語情報 URL: http://seesaawiki.jp/w/imagej/

[10]マクロの見出し [1], [2], [3] はキーボードの数値キーでマクロが作動することを意味している．

```
   cnt = "0000"+j;
   path1 ="open="+dir+"Image"+substring(cnt,lengthOf(cnt)-4,lengthOf(cnt))+".txt";
   print(path1);
   run("Text Image... ", path1);
   setMinAndMax(0, 1000);
   run("Fire");
  }
   // make stack, make montage
  run("Images to Stack", "name=Stack title=[] use");
  run("Make Montage...", "columns=3 rows=5 scale=1");
}

macro "Compose 3X3 [3]"{
//input Image files,  Image to stack and then Make stack
  dir = getDirectory("Choose a Directory");
  for (i=0; i<9;i++){
   j=9-i;
   cnt = "0000"+j;
   path1 ="open="+dir+"Image"+substring(cnt,lengthOf(cnt)-4,lengthOf(cnt))+".txt";
   print(path1);
   run("Text Image... ", path1);
   setMinAndMax(0, 1000);
   run("Fire");
  }
// make stack, make montage
run("Images to Stack", "name=Stack title=[] use");
run("Make Montage...", "columns=3 rows=3 scale=1");
}
```

Stack100.ijm

Stack100.txt は，閾値電圧の変化に伴うカウント数と差分の動画を作成する ImageJ のマクロプログラムである．

```
//  カウント
macro "Stack (95) " {
  dir = getDirectory("Choose a Directory");
  list = getFileList(dir);

  for (i=1; i<list.length+1;i++){
    path1 = "open="+ dir+"stack_"+i+".txt";
    run("Text Image... ", path1);
    run("16 colors");
    //run("Brightness/Contrast...");
    setMinAndMax(0, 3000);
  }
 run("Images to Stack", "name=Stack title=[] use");
}

// 差分
macro "Stack-d (95)" {
  dir = getDirectory("Choose a Directory");
  list = getFileList(dir);

  for (i=1; i<list.length+1;i++){
    path1 = "open="+ dir+"stack-d_"+i+".txt";
    run("Text Image... ", path1);
    run("16 colors");
    //run("Brightness/Contrast...");
    setMinAndMax(0, 7000);
  }
 run("Images to Stack", "name=Stack-d title=[] use");
}
```

all_image.ijm

```
//  File name all_image.ijm
// ImageJ macro to make images of 2x3 by CdTe detector
//   (Set up each directory of mV before processing)

macro "Compose_all_2x3"{

 //img のあるディレクトリ dir を指定
 dir = getDirectory("Choose a Directory");

 for (j=210;j>59;j--){
  dir1 = dir+j+"/";
  print(dir1);
  //input Image files,  Image to stack and then Make stack
  for (i=6; i>0;i--){
   cnt = "0000"+i;
   path1 ="open="+dir1+"Image"+substring(cnt,lengthOf(cnt)-4,lengthOf(cnt))+".txt";
   print(path1);
   run("Text Image... ", path1);
   setMinAndMax(0, 1000);
   run("Fire");
  }
  // make stack, make montage
  run("Images to Stack", "name=Stack title=[] use");
  run("Make Montage...", "columns=2 rows=3 scale=1");

  dir1=dir1+"P1-"+j+".tif";
  // 合成画像を.tif で保存して，ウィンドウを閉じる，ファイル名は dir1
  saveAs("Tiff", dir1);
  close();
  selectWindow("Stack");
  close();
 }
}

macro "Compose_all_3x5"{

 //img のあるディレクトリ dir を指定
 dir = getDirectory("Choose a Directory");

 for (j=210;j>59;j--){
  dir1 = dir+j+"/";
  print(dir1);
  //input Image files,  Image to stack and then Make stack
  for (i=15; i>0;i--){
   cnt = "0000"+i;
   path1 ="open="+dir1+"Image"+substring(cnt,lengthOf(cnt)-4,lengthOf(cnt))+".txt";
   print(path1);
   run("Text Image... ", path1);
   setMinAndMax(0, 1000);
   run("Fire");
  }
  // make stack, make montage
  run("Images to Stack", "name=Stack title=[] use");
  run("Make Montage...", "columns=3 rows=5 scale=1");

  dir1=dir1+"P2-"+j+".tif";
  // 合成画像を.tif で保存して，ウィンドウを閉じる，ファイル名は dir1
  saveAs("Tiff", dir1);
  close();
  selectWindow("Stack");
  close();
 }
}
```

```
//  すべて P1-xxx.tif を重ね合わせる．stack は動画として tif で保存し，ImageJ で再生．
macro "Stack_all(2x3)"{

 //img のあるディレクトリ dir を指定
 dir = getDirectory("Choose a Directory");

 for (j=210;j>59;j--){
  dir1 = dir+j+"/";
  path1 =dir1+"P1-"+j+".tif";
  print(path1);
  open(path1);
  setMinAndMax(0, 1000);
  run("Fire");
 }
 // make stack, make montage
 run("Images to Stack", "name=Stack title=[] use");
}

//  すべて P2-xxx.tif を重ね合わせる．stack は動画として tif で保存し，ImageJ で再生．
macro "Stack_all(3x5)"{

 //img のあるディレクトリ dir を指定
 dir = getDirectory("Choose a Directory");

 for (j=210;j>59;j--){
  dir1 = dir+j+"/";
  path1 =dir1+"P2-"+j+".tif";
  print(path1);
  open(path1);
  setMinAndMax(0, 1000);
  run("Fire");
 }
 // make stack, make montage
 run("Images to Stack", "name=Stack title=[] use");
}

// P1-i.tif=P1-i+1.tiff - P1-i-1.tiff
macro "Diff_all (2x3)"{

 //img のあるディレクトリを指定 dir
 dir = getDirectory("Choose a Directory");

 for (j=209;j>61;j--){
  i=j+1;
  k=j-1;
  dir_i = dir+i+"/";
  file_i = "P1-"+i+".tif";
  path_i =dir_i+file_i;
  open(path_i);
  setMinAndMax(0, 1000);
  run("Fire");

  dir_k = dir+k+"/";
  file_k = "P1-"+k+".tif";
  path_k =dir_k+file_k;
  open(path_k);
  setMinAndMax(0, 1000);
  run("Fire");

  // Diffrential
  imageCalculator("Subtract create", file_k,file_i);
  file_k="Result of "+ file_k;
  selectWindow(file_k);
```

```
  //run("Brightness/Contrast...");
  run("16 colors");
  setMinAndMax(-4, 80);

  file_j = dir+j+"/P1-"+j+"diff.tif";
  saveAs("Tiff", file_j);
  print(file_j);
  run("Close All");
 }
}

// P2-i.tif=P2-i+1.tiff - P2-i-1.tiff
macro "Diff_all (3x5)"{

 //img のあるディレクトリを指定 dir
 dir = getDirectory("Choose a Directory");

 for (j=209;j>61;j--){
  i=j+1;
  k=j-1;
  dir_i = dir+i+"/";
  file_i = "P2-"+i+".tif";
  path_i =dir_i+file_i;
  open(path_i);
  setMinAndMax(0, 1000);
  run("Fire");

  dir_k = dir+k+"/";
  file_k = "P2-"+k+".tif";
  path_k =dir_k+file_k;
  open(path_k);
  setMinAndMax(0, 1000);
  run("Fire");

  // Diffrential
  imageCalculator("Subtract create", file_k,file_i);
  file_k="Result of "+ file_k;
  selectWindow(file_k);
  //run("Brightness/Contrast...");
  run("16 colors");
  setMinAndMax(-4, 80);

  file_j = dir+j+"/P2-"+j+"diff.tif";
  saveAs("Tiff", file_j);
  print(file_j);
  run("Close All");

 }
}

//  すべての P1-xxxdiff.tif を重ね合わせる．stack は動画として tif で保存し，ImageJ で再生.
macro "Diff_stack_all(2x3)"{

 //img のあるディレクトリ dir を指定
 dir = getDirectory("Choose a Directory");

 for (j=209;j>60;j--){
  dir1 = dir+j+"/";
  path1 =dir1+"P1-"+j+"diff.tif";
  print(path1);
  open(path1);
  setMinAndMax(-5,80);
  //run("Brightness/Contrast...");
  run("16 colors");
```

```
 }
 // make stack, make montage
 run("Images to Stack", "name=Stack title=[] use");
}

//  すべての P2-xxxdiff.tif を重ね合わせる．stack は動画として tif で保存し，ImageJ で再生．
macro "Diff_stack_all(3x5)"{

 //img のあるディレクトリ dir を指定
 dir = getDirectory("Choose a Directory");

 for (j=209;j>60;j--){
  dir1 = dir+j+"/";
  path1 =dir1+"P2-"+j+"diff.tif";
  print(path1);
  open(path1);
  setMinAndMax(-5,80);
  //run("Brightness/Contrast...");
  run("16 colors");
 }
 // make stack, make montage
 run("Images to Stack", "name=Stack title=[] use");
}
```

P1-P2.ijm

```
macro "P1(2x3)"{
 //img のあるディレクトリ dir を指定
 dir = getDirectory("Choose a Directory");

 for (j=210;j>59;j--){
  dir1 = dir+j+"/";
  print(dir1);
  //input Image files,  Image to stack and then Make stack
  for (i=6; i>0;i--){
   cnt = "0000"+i;
   path1 ="open="+dir1+"Image"+substring(cnt,lengthOf(cnt)-4,lengthOf(cnt))+".txt";
   run("Text Image... ", path1);
   setMinAndMax(0, 1000);
   run("Fire");
  }
  // make stack, make montage
  run("Images to Stack", "name=Stack title=[] use");
  run("Make Montage...", "columns=2 rows=3 scale=1");
  //メジアンフィルター
  run("Fire");run("Median...", "radius=2");
  //移動平均
  run("Mean...", "radius=2");
  dir1=dir1+"P1-"+j+".tif";
  // 合成画像を.tif で保存して，ウィンドウを閉じる，ファイル名は dir1
  saveAs("Tiff", dir1);
  print(dir1);
  close();
  selectWindow("Stack");
  close();
 }

 //img のあるディレクトリ dir を指定
 // dir = getDirectory("Choose a Directory");

 for (j=210;j>59;j--){
  dir1 = dir+j+"/";
  path1 =dir1+"P1-"+j+".tif";
```

```
 print(path1);
 open(path1);
 setMinAndMax(0, 1000);
 run("Fire");
}
// make stack, make montage
run("Images to Stack", "name=Stack title=[] use");

path1=dir+"P1_stack.tif";
print(path1);
saveAs("Tiff", path1);
close();

// P1-i.tif=P2-i+1.tiff - P1-i-1.tiff

for (j=209;j>60;j--){
 i=j+1;
 k=j-1;
 dir_i = dir+i+"/";
 file_i = "P1-"+i+".tif";
 path_i =dir_i+file_i;
 open(path_i);
 setMinAndMax(0, 1000);
 run("Fire");

 dir_k = dir+k+"/";
 file_k = "P1-"+k+".tif";
 path_k =dir_k+file_k;
 open(path_k);
 setMinAndMax(0, 1000);
 run("Fire");

 // Diffrential
 imageCalculator("Subtract create", file_k,file_i);
 file_k="Result of "+ file_k;
 selectWindow(file_k);
 //run("Brightness/Contrast...");
 run("16 colors");
 setMinAndMax(-4, 80);

 file_j = dir+j+"/P1-"+j+"diff.tif";
 saveAs("Tiff", file_j);
 print(file_j);
 run("Close All");
}

for (j=209;j>60;j--){
 dir1 = dir+j+"/";
 path1 =dir1+"P1-"+j+"diff.tif";
 print(path1);
 open(path1);
 setMinAndMax(-5,80);
 //run("Brightness/Contrast...");
 run("16 colors");
}
// make stack
run("Images to Stack", "name=Stack title=[] use");

path1=dir+"P1_diff_stack.tif";
print(path1);
saveAs("Tiff", path1);
close();
print("Complete P1!");
}
```

```
//---------------------------------
macro "P2(3x5)"{
 //img のあるディレクトリ dir を指定
 dir = getDirectory("Choose a Directory");

 for (j=210;j>59;j--){
  dir1 = dir+j+"/";
  print(dir1);
  //input Image files,  Image to stack and then Make stack
  for (i=15; i>0;i--){
   cnt = "0000"+i;
   path1 ="open="+dir1+"Image"+substring(cnt,lengthOf(cnt)-4,lengthOf(cnt))+".txt";
   run("Text Image... ", path1);
   setMinAndMax(0, 1000);
   run("Fire");
  }
  // make stack, make montage
  run("Images to Stack", "name=Stack title=[] use");
  run("Make Montage...", "columns=3 rows=5 scale=1");
  run("Fire");run("Median...", "radius=2");
  run("Mean...", "radius=2");

  dir1=dir1+"P2-"+j+".tif";
  // 合成画像を.tif で保存して，ウィンドウを閉じる，ファイル名は dir1
  print(dir1);
  saveAs("Tiff", dir1);
  close();
  selectWindow("Stack");
  close();
 }

 for (j=210;j>59;j--){
  dir1 = dir+j+"/";
  path1 =dir1+"P2-"+j+".tif";
  print(path1);
  open(path1);
  setMinAndMax(0, 1000);
  run("Fire");
 }
 // make stack, make montage
 run("Images to Stack", "name=Stack title=[] use");

 path1=dir+"P2_stack.tif";
 print(path1);
 saveAs("Tiff", path1);
 close();

 // P2-i.tif=P2-i+1.tiff - P2-i-1.tiff

 for (j=209;j>60;j--){
  i=j+1;
  k=j-1;
  dir_i = dir+i+"/";
  file_i = "P2-"+i+".tif";
  path_i =dir_i+file_i;
  open(path_i);
  setMinAndMax(0, 1000);
  run("Fire");

  dir_k = dir+k+"/";
  file_k = "P2-"+k+".tif";
  path_k =dir_k+file_k;
```

```
 open(path_k);
 setMinAndMax(0, 1000);
 run("Fire");

 // Diffrential
 imageCalculator("Subtract create", file_k,file_i);
 file_k="Result of "+ file_k;
 selectWindow(file_k);
 //run("Brightness/Contrast...");
 run("16 colors");
 setMinAndMax(-4, 80);

 file_j = dir+j+"/P2-"+j+"diff.tif";
 saveAs("Tiff", file_j);
 print(file_j);
 run("Close All");
}

for (j=209;j>60;j--){
 dir1 = dir+j+"/";
 path1 =dir1+"P2-"+j+"diff.tif";
 print(path1);
 open(path1);
 setMinAndMax(-5,80);
 //run("Brightness/Contrast...");
 run("16 colors");
}
// make stack
run("Images to Stack", "name=Stack title=[] use");
path1=dir+"P2_diff_stack.tif";
print(path1);
saveAs("Tiff", path1);
close();
print("Complete P2!");
}
```

spot_max.ijm

```
//-------------------------------- spot_max.ijm -----
macro "Spot_max_P1"{
 //img のあるディレクトリ dir を指定
 dir = getDirectory("Choose a Directory");
 //getNumber("prompt", defaultValue) を使って，数値をキー入力できる.
 //getString("prompt", "default") を利用すると，文字も入力できる.
 x = getNumber("x=",x);
 y = getNumber("y=",y);
 xl = getNumber("lx=",xl);
 yl = getNumber("ly=",yl);

 for (j=209;j>60;j--){
  dir_j = dir+j+"/";
  file_j = "P1-"+j+"diff_F.tif";
  path_j =dir_j+file_j;
  open(path_j);
  setMinAndMax(1, 200);
  run("Fire");
  run("Set Measurements...", "min redirect=None decimal=3");
  makeRectangle(x, y, xl, yl);
  run("Measure");
  close();
 }
 print("Complete P1!");
```

```
}

macro "Spot_max_P2"{
 //img のあるディレクトリ dir を指定
 dir = getDirectory("Choose a Directory");
 //getNumber("prompt", defaultValue) を使って，数値をキー入力できる.
 //getString("prompt", "default") を利用すると，文字も入力できる.
 x = getNumber("x=",x);
 y = getNumber("y=",y);
 xl = getNumber("lx=",xl);
 yl = getNumber("ly=",yl);

 for (j=209;j>60;j--){
  dir_j = dir+j+"/";
  file_j = "P2-"+j+"diff_F.tif";
  path_j =dir_j+file_j;
  open(path_j);
  setMinAndMax(1, 200);
  run("Fire");
  run("Set Measurements...", "min redirect=None decimal=3");
  makeRectangle(x, y, xl, yl);
  run("Measure");
  close();
 }
 print("Complete P2!");
}
```

keV-stack-diff.ijm

keV-stack-diff.ijm は， E_image.f90 でつくられた画像ファイル群 i keV.txt からスタック像や差分像をつくるための ImageJ マクロです.

```
// keV ファイルのスタック像作成 keV-stack.ijm
macro "Make stack for keV" {
  dir = getDirectory("Choose a Directory"); //directry の選択
    X = getNumber("First XkeV, X=",X); //最初のファイル
    n = getNumber("Number of files n=",n); //読込みファイル数
    max=X+n+1; //読み込み終了条件
  for (i=X; i<max;i++){
    path1 = "open="+ dir+i+"keV.txt"; //読込みファイル名
    run("Text Image... ", path1); //テキストイメージの読込み
    run("Fire"); //Image Table として fire スタイルを定義
    setMinAndMax(0, 5000);    //run("Brightness/Contrast..."); の min,max 指定
run("Median...", "radius=1");
run("Mean...", "radius=1");
  }
 run("Images to Stack", "name=Stack title=[] use");
}

// keV ファイルの差分像の作成
macro "Make diff for keV" {
  dir = getDirectory("Choose a Directory"); //directry の選択
    X = getNumber("First XkeV, X=",X); //最初のファイル
    n = getNumber("Number of files n=",n); //読込みファイル数
max=X+n+1; //読み込み終了条件

for (i=X; i<max; i++){
a=i;
b=""+a;// i は数値のため，いろいろ変数を変えて型変換させた
file1= b+"keV.txt";// ""+ これで i を数値から文字列として定義
path1 = "open="+ dir+file1; //i keV 読込みファイル名
     run("Text Image... ", path1); //テキストイメージの読込み
     run("Fire"); //Image Table として fire スタイルを定義
```

```
     setMinAndMax(0, 5000);    //run("Brightness/Contrast..."); の min,max 指定
 run("Median...", "radius=1");
 run("Mean...", "radius=1");

k=i+2;
file2=""+k+"keV.txt"; // ""+ これで k を数値から文字列として定義
path1 = "open="+ dir+file2; //i+2 keV 読込みファイル名
run("Text Image... ", path1); //テキストイメージの読込み
run("Fire"); //Image Table として fire スタイルを定義
setMinAndMax(0, 5000);    //run("Brightness/Contrast..."); の min,max 指定
run("Median...", "radius=1");
run("Mean...", "radius=1");


  // Diffrential
  imageCalculator("Subtract create", file1,file2);
  file1="Result of "+ file1;
selectWindow(file1);
//run("Brightness/Contrast...");
run("Fire");
setMinAndMax(-4, 80);

h=i+1;
file_j = dir+h+"keV_d.tif";
saveAs("Tiff", file_j);
print(file_j);
run("Close All");
  }
}

//keV-stack.ijm
// keV ファイルの差分像の作成
macro "Make stack of diff" {
dir = getDirectory("Choose a Directory"); //directry の選択
V0 = getNumber("Start keV_d.tif, V0=",V0); //最初のファイル
V1 = getNumber("Stop keV_d.tif, V1=",V1); //読込みファイル数

for (i=V0; i<V1+1;i++){
path1 = dir+i+"keV_d.tif"; //読込みファイル名
open(path1); //テキストイメージの読込み
run("Fire"); //Image Table として fire スタイルを定義
setMinAndMax(0, 400);    //run("Brightness/Contrast..."); の min,max 指定
}
run("Images to Stack", "name=Stack title=[] use");
}
```