修士論文

免疫細胞の遊走及び形態変化の時系列解析ツールの開発

Development of time series analysis tool for immune cell migration and morphological change

報 告 者

学籍番号:1205064

氏名:井上　智哉

指 導 教 員

星野　孝総　准教授

平成 30 年 2 月 5 日

高知工科大学 大学院工学研究科

基盤工学専攻 電子・光システム工学コース

目次

第 1 章　序論……1
1.1　背景……1
1.2　本研究の目的……1

第 2 章　免疫細胞の解析……3
2.1　マクロファージ……3
2.2　免疫細胞の解析手法……3

第 3 章　開発環境……6
3.1　Java ……6
3.1.1　Swing……6
3.1.2　Java の開発環境 ……7
3.2　OpenCV ……7
3.2.1　開発に用いた主な関数……7
3.3　Eclipse……8
3.3.1　Eclipse による開発手順……8
3.3.2　WindowBuilder ……10
3.3.3　WindowBuilder による開発……10

第 4 章　免疫細胞解析ツール……13
4.1　ツール概要……13
4.2　画像ファイルの閲覧……13
4.3　動画像から静止画へのコンバート……14
4.4　動画像を対象とした免疫細胞の自動追跡 ……15
4.5　画像輝度断面図の表示……20
4.6　細胞の輪郭情報の取得……21
4.7　輪郭情報の時系列処理……22

第 5 章　ツールの使用方法……24
5.1　画像閲覧ツール……24
5.2　動画像から画像への変換機能……25
5.3　免疫細胞の自動追跡機能……26
5.4　輝度断面図の表示ツール……28
5.5　細胞の輪郭情報の取得ツール……29

5.6　輪郭情報の時系列処理機能……31

第 6 章　結論 …… 33

謝辞 …… 34

参考文献 …… 35

研究業績 …… 37

第１章　序論

1.1　背景

近年，免疫学の分野では，免疫細胞の一種であるマクロファージに関する研究が盛んに行われている．このマクロファージ細胞は，形状を変化させながら遊走移動を行う．大学の医学部などで免疫細胞であるマクロファージの動きの解析を行う場合，顕微鏡によって撮影された動画像に対して目視による解析作業を行うという手法をとることが多い．　医学関係の解析作業では正確性が重要とされており，この要件は医学を専門とする人間の視覚に頼って作業を行うことで正確性を担保できるとしている．専門医の手作業で解析を行うには確認しなければいけない画像の枚数が多く，1 回の解析作業に多くの時間がかかってしまうという問題点がある．この問題に関しては，画像処理を用いたシステムによって作業時間を短縮させる手法が取られている[1]．その場合は，一般の画像処理システムでは，免疫細胞の形状を変化させるという特徴から正確なデータを得ることが難しいため，専用の画像処理システムを導入する必要がある．専用のシステムを利用することで解決することが可能ではあるが，そのツールを使用するためには専用の顕微鏡が必要な場合やツール自体の価格が高価である場合があり，簡単に導入することが出来ないという問題がある．　また，顕微鏡を用いた免疫細胞の撮影を行う場合は，免疫細胞自体の動きが非常に遅く必要なデータを取得するためには撮影時間が数十時間を超える．その間休みなく撮影をしていては格納される画像データの量が膨大となり，それら全てを対象に専門医による手作業を行うのは現実的ではない．そのため，一般的な医学部で導入されている設備で解析作業を行う場合は，免疫細胞の動きがわかる程度の時間間隔に間引き撮影を行う．したがって，実時間のレートよりも荒い動画像を対象に解析作業を行うことになる．目視による解析ならば問題にはならないが，何らかのツールを用いて解析を行う場合には，自動的な判別が難しくなってしまう場合がある．そこで，専門医の解析を補助するツールやインタフェースシステムの構築が求められている[2]．

1.2　本研究の目的

本研究ではマクロファージの目視解析を支援するための画像解析ツールの開発を目的とする．マクロファージの画像解析ツールの開発は各所で行われており，中島らは 2 光分子顕微鏡画像を用いた解析ツールの開発を行っている[3][4][5][6][7][8]．この研究は，2 光分子顕微鏡で撮影された画像を対象として，二次元動画像と三次元視覚化ツールによってマクロファージの振る舞いを解析するというものである．このように，マクロファージの振る舞いを画像解析ツールの使用によって解析するという手法は，マクロファージを解析する上で有効であると知られている．しかし，開発が行われているツールはこの研究のように特殊な

顕微鏡や機材を使用する場合があり，一般的な医学部での解析作業に利用することが難しい．このことから，一般的な医学部で使用されている顕微鏡画像を対象としたマクロファージの画像解析ツールを開発することができれば，より多くの大学でマクロファージの振る舞いを解析する研究を行うことができるようになると考えた．そのため，本研究では，一般的な医学部で使用されている顕微鏡画像を対象とした解析ツールの開発を行うこととした．本研究で開発している支援ツールは大学の医学部などで研究されている免疫細胞の解析作業に利用することを考えている．人の手で行うには時間がかかってしまう作業を支援ツールの導入によって自動化することで作業効率の大幅な向上を目指す．更に，本研究で開発を目指すツールは，標準的な顕微鏡で撮影した動画像を解析の対象とするため，新たな機材の購入や環境の構築をする必要がない．このため，ツール導入にコストがかからないメリットがある．

本稿では，開発を行ったツールの処理と仕様について説明する．第 2 章では，対象とする免疫細胞のマクロファージについての説明と実際に行われている解析作業について解説する．第 3 章では，支援ツールの開発環境について解説する．第 4 章，第 5 章ではそれぞれ各ツールの処理内容と機能についての解説を行う．

第 2 章　免疫細胞の解析

2.1　マクロファージ

マクロファージは白血球の 1 種の遊走性細胞である．遊走とは仮足と呼ばれる細胞の移動を担う突出を用いて組織内を移動することを言う．遊走するマクロファージの例を図 1 に示す．マクロファージの主な役割は大きく分けて 2 つ存在する．1 つは貪食作用と呼ばれ，もう一つは抗原提示という．貪食作用とは，死んだ細胞や侵入した細菌などの異物を取り込む作用である．抗原提示とは，取り込んだ抗原物質から抗原情報を他の細胞に提供する作用の事である．この 2 つの作用から人間の免疫機能が維持され人間の健康に大きく寄与しており，マクロファージは重要な細胞であると言われている．このことから，特定の疾患とマクロファージが持つ免疫作用の関連性の研究も行われている[9].

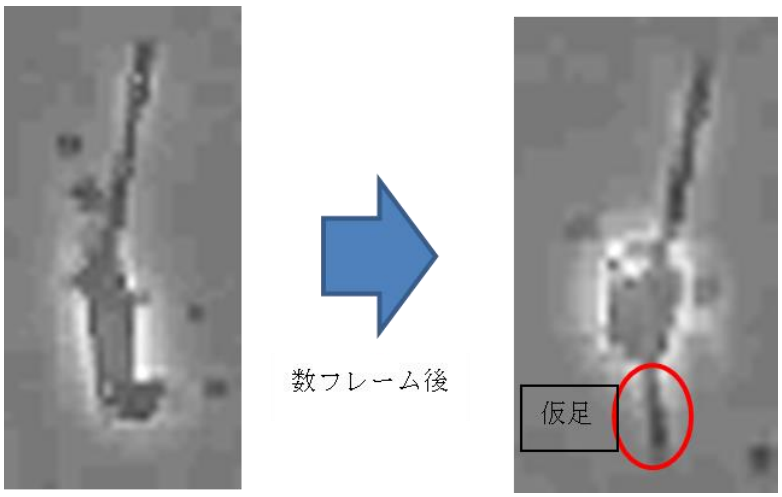

図 1　仮足を用いた遊走の様子

2.2　免疫細胞の解析手法

目視による免疫細胞の解析手法の大まかな流れを図 2 に示す．まずは，採取した体組織を顕微鏡で撮影する．免疫細胞は動きが非常に遅く，解析に必要な量のデータを集めるには数十時間に及ぶ長時間の撮影を行う必要がある．しかし，長時間の顕微鏡撮影によって得られる膨大な量の画像データ全てを格納するにはそれ相応の大容量のストレージが必要になる上に，膨大な量の画像データを一つ一つ人の手で作業していくのは現実的ではない．そのため，一般的な設備を用いて解析作業を行う場合は，撮影した画像データを免疫細胞の動きが見える程度まで間引いて動画像にまとめるシステムを使用することで上記の問題を解決している．解析対象となる動画像の用意が完了したら，次に遊走や形態変化の

データを取得する対象となる免疫細胞を選択する．動画像をコマ送りにしながら対象の免疫細胞の画像内での位置や輪郭形状などのデータを手作業で記録していく．この作業を対象の細胞が画面外に出るか動画像が終了するまで繰り返し行う．解析に必要なデータを収集し終えたら，それらのデータをフレーム毎の区間に区切って時系列毎に解析していく．しかし，この一連の作業には，多くの時間を要することと作業自体を専門医の手で行わなければならない性質上，作業担当者への負担が大きい．この問題の解決には，近年技術が発達し，医療分野での活用が可能になった画像処理技術を利用したソフトウェアを導入する方法がある．専用のソフトウェアを利用すれば時間と労力のかかる作業の効率化が容易になると考えられる．そこで，免疫細胞の解析作業に必要な機能を実装した画像処理を用いた支援ツールの開発を行うこととした．

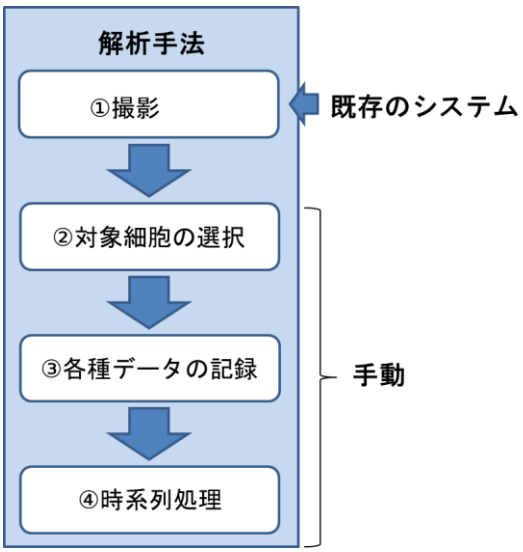

図 2　免疫細胞解析手法の例

上記の作業手順を精査し解析ツールに必要と思われる主な機能に関して考えた結果，支援ツールに実装する機能は以下のようになった．

- 動画像をコマ送りで確認していく
  →動画像をフレーム毎に画像に切り分ける．
- 各フレームで免疫細胞の各種情報を記録
  →細胞を選択するだけで動画像内を自動的に追跡する[10]．
  →細胞の輪郭情報を取得する．
- 収集したデータを時系列毎に解析する．
  →フレームの範囲を選択することで自動的に必要な値を算出する．

これらの機能を実装したツールを開発する事ができれば，免疫細胞の解析作業を支援す

ることができると考えた．また，出力ファイルの閲覧や輝度断面図の表示などの機能を追加し，より完成度の高いツールの開発を目指した．

第３章　開発環境

解析ツールの開発には，プログラミング言語の Java を使用した．内部の処理で様々な画像処理を利用するため，オープンソースの画像処理ライブラリである OpenCV を使用している[11]. ツールのソースコードの作成に統合開発環境でありかつ優秀な Java 開発環境でもある Eclipse 上で開発を行った[12]．さらに円滑な開発を行うために Eclipse のプラグインである WindowBuilder を利用した[13].

3.1　Java

Java とは，オブジェクト指向のプログラミング言語であり，1995 年にサン・マイクロシステムズ社が開発を行った．特徴としては，開発したアプリケーションがどの OS でも動作させることが可能であることと，オブジェクト指向であることなどが挙げられる．

3.1.1 Swing

Swing とは，Java で GUI(グラフィカルユーザーインタフェース) アプリケーションの開発を行う際に，様々なコンポーネントを提供するツールキットである．ウィンドウのフレームやパネルなどからボタンやテキストフィールドなどが用意されている．表 1 に Swing で用意されている主なコンポーネントを示す．これらのコンポーネントを組み合わせることで様々な GUI アプリケーションを開発することができる．

表 1　Swing の主なコンポーネント

コンポーネント名	コンポーネントの種類
JFrame	フレーム
JPanel	パネル
JLabel	ラベル
JTextField	テキストフィールド
JButton	ボタン
JCheckBox	チェックボックス
JList	リスト
JSlider	スライドバー

3.1.2 Java の開発環境

Java の開発には，統合開発環境(IDE : Integrated Develop Enviroment)と呼ばれる開発ツールを利用することが一般的である．統合開発環境とはエディタ，コンパイラ，インタプリタ全てを内蔵した開発ソフトウェアであり，1 つの画面でソースコードの編集からコンパイル，実行に至るまでの開発作業を一通り行うことができる．また，ソースコードの色分けや，キー入力の補完などの機能を備えていることが一般的である．代表的なものとして，Eclipse や NetBeans がよく知られている．本研究では，開発環境として Eclipse を採用した．

3.2 OpenCV

OpenCV とは，インテルによって開発されたオープンソースの画像処理ライブラリであり，多彩な画像処理機能を実装している．このライブラリを利用することで，様々な画像処理を関数呼び出しによって実行することができるようになる．

3.2.1 開発に用いた主な関数

ツールの開発に使用した OpenCV の主な関数とその簡単な説明を表 2 に示す[14]．本研究では画像ファイルの入出力や二値化，輪郭の検出などの処理に OpenCV の関数を採用した．各関数の詳細な説明は第 4 章で行う．これらの関数を利用することで，様々な画像処理を簡単に実行する事ができる．

表 2　ツール開発に使用した主な関数

関数名	説明
imread	ファイルから画像を読み込む
imwrite	指定したファイルに画像を保存
threshold	配列の要素に対して,
findContours	2値画像中の輪郭を検出する
drawContours	輪郭線を描画する
boundingRect	点群に外接する矩形を求める
arcLength	輪郭線の周囲長を求める
resize	画像のサイズを変更する
line	線を描画する
rectangle	矩形を描画する
circle	円を描画する

3.3　Eclipse

Eclipse は，IBM が開発した IDE であり，現在はオープンソースとして公開されているため，無償で利用することが可能となっている．本研究では，Eclipse 4.6 Neon によってツールの開発を行った.

3.3.1 Eclipse による開発手順

Eclipse では，開発するアプリケーション毎にプロジェクトと呼ばれる管理単位を作り，その中にソースコードを作成していく．Java のプログラムを開発するためには，以下の手順で「Java プロジェクト」を作成する必要がある.

1　ファイルメニュー→「新規」→「Java プロジェクト」を選択する.
2　図 3 の画面で任意の「プロジェクト名」を入力し完了ボタンを押す.

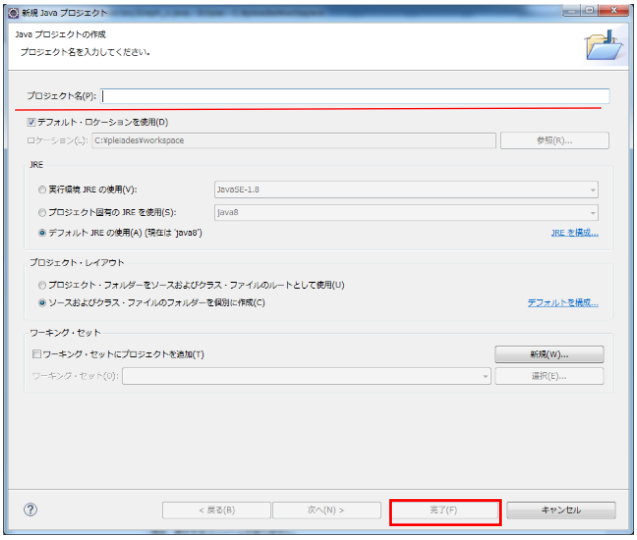

図 3　プロジェクト作成画面

Eclipse のプロジェクト内では，下記の手順でクラスを作成する．

1　パッケージ・エクスプローラービューでクラスを作成したい Java プロジェクトを選択し右クリック→「新規」→クラスを選択する．
2　画面でクラスが所属するパッケージとクラス名を指定し，完了ボタンを押す．

Eclipse には上述の通り Java のコンパイラが含まれており，エディタでコードの編集や上書き保存を行うと自動的にコンパイルが行われるようになっている．その際にコンパイルエラーが発見されると，図 4 のように，行の先頭にエラーを表すマークとコンパイルエラーが発生した箇所に赤色の波線が表示される．コンパイルエラーが発生した箇所にマウスカーソルを合わせることで，発生しているエラーの内容とその修正方法の候補が表示される．その中から 1 つを選択すると，自動でその処理が行われエラーが解消される．main メソッドを持つクラスは，ファイルを選択し右クリックメニューから「実行」→「Java アプリケーション」を選択することで実行することができる．

図 4　エラー表示の例

3.3.2 WindowBuilder

WindowBuilder とは， GUI アプリケーションの開発を補助するための Eclipse のプラグインである．通常，Eclipse で Java の GUI アプリケーションを開発する場合は，そのアプリケーションを実行するまでデザインを確認することが出来ない．また，アプリケーションのフレームのレイアウトを設定する場合，選択したレイアウトによっては，複雑で記述量の多いソースコードになり，開発に時間がかかる可能性がある．WindowBuilder を利用して開発されたアプリケーションは，「ソース」と「Design」という 2 種類の形式でコードの編集を行うことができる．「ソース」は通常のソースコードベースでアプリケーションの開発を行う形式で，「Design」は図 5 のように，アプリケーションのフレームを直接編集しながらアプリケーションを開発する形式である．フレームの編集は，ドラッグアンドドロップによるコンポーネントの追加や移動，サイズの変更などを直感的な操作で行うことができる．編集した箇所は自動的にソースコードに反映されるため，複雑なレイアウトの設定をソースコードから書き起こす必要がなくなる．これにより，編集や確認に時間のかかる GUI アプリケーションの開発を効率化し，開発時間の短縮を図ることができる．

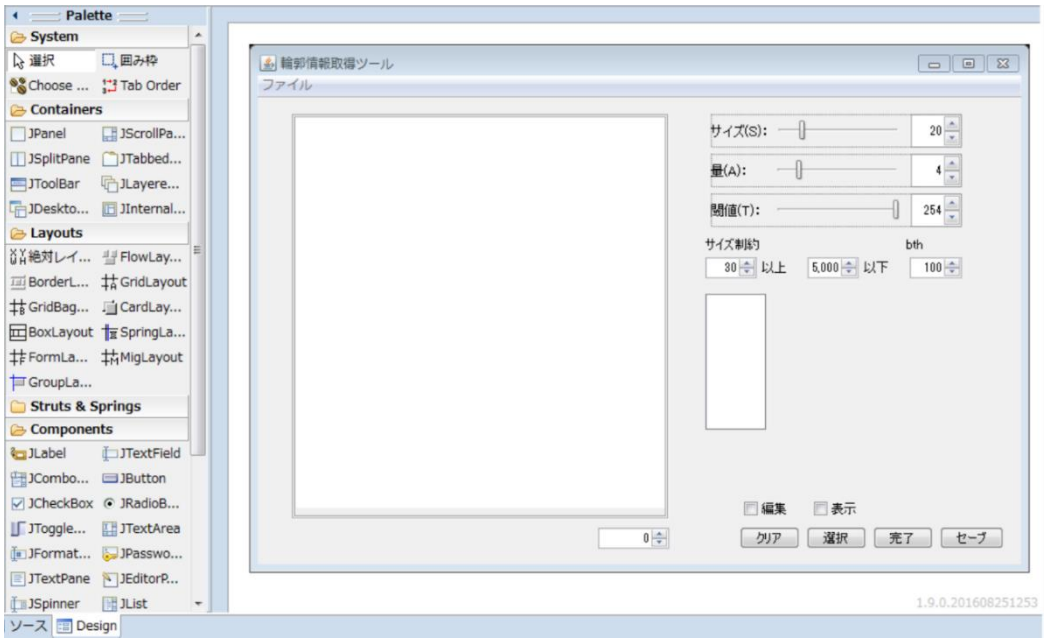

図 5　WindowBuilder の使用例

3.3.3　WindowBuilder による開発

WindowBuilder を用いた Java アプリケーションの開発は以下の手順で行う．

1.　ファイルメニューから「新規」→「その他」を選択すると，ウィンドウが表示される．

2. ウィンドウ上のツリーから「WindowBuilder」→「Swing デザイナー」→「JFrame」を選択し，「次へ」をクリックする．
3. 進んだ先の画面でクラス名を入力し「完了」をクリックすることで WindowBuilder に対応した Java の新規クラスが作成される．
4. 新規クラスを作成すると，ソースコードと形式選択のタブが表示され，「Design」タブを選択すると形式が切り替わり，新規作成したクラスのフレームが表示される．

新規作成した JFrame のレイアウトは「BorderLayout」に設定されている．レイアウトを変更したい場合は，図 6 に示すように「Palette」の「Layout」から適応したいレイアウトを選択し，contentPane や JPanel 上でマウスクリックをする．これにより，クリック操作を行った contentPane や JPanel に選択したレイアウトが適応された状態となる．レイアウトの変更を行うと，その変更に合わせてソースコードが自動的に書き換わる．

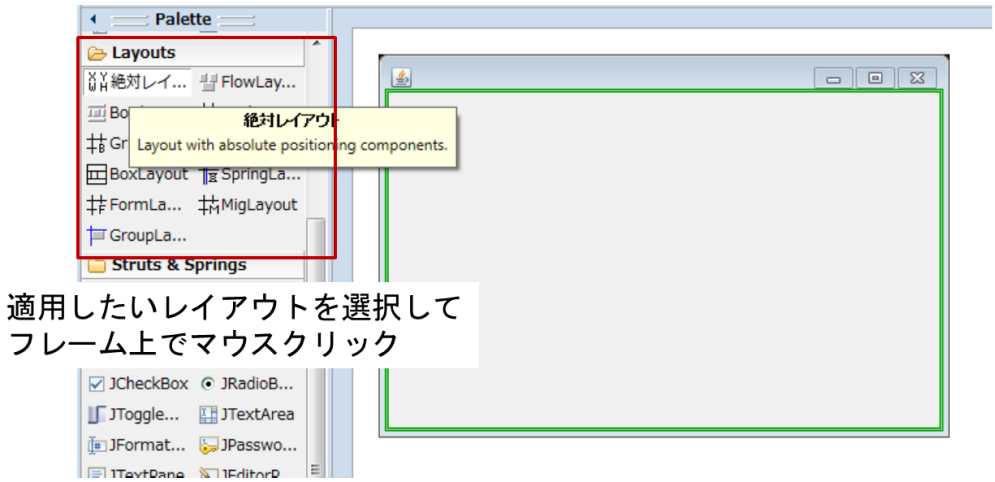

図 6　レイアウトの設定

図 7 のように「Palette」の「Components」から配置したいコンポーネントを選択し，配置したい箇所までドラッグアンドドロップすることで，フレーム上に任意のコンポーネントを配置することができる．配置したコンポーネントをクリックすることで，任意の箇所に移動させることや，サイズの変更，コンポーネント上に記述されたテキストの編集を行うことができる．また，ウィンドウ左下部にあるプロパティーからも同様の変更を行う事ができる．

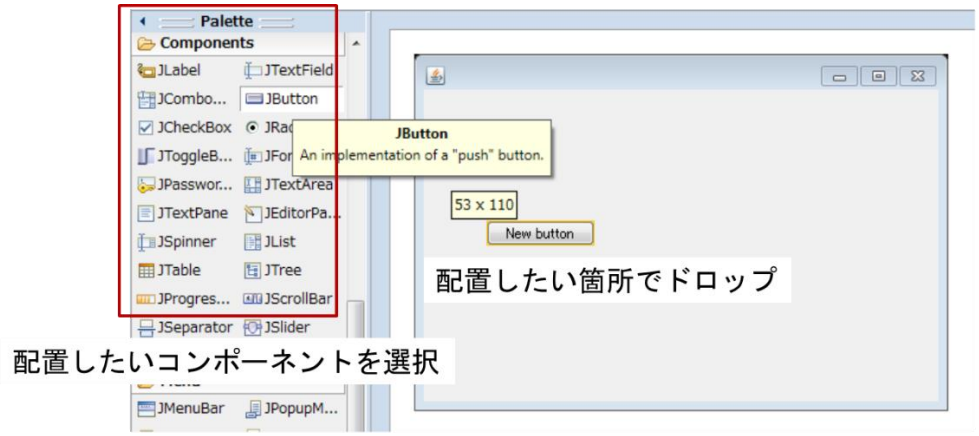

図 7　コンポーネントの配置

第 4 章　免疫細胞解析ツール

4.1　ツール概要

本研究で作成した免疫細胞解析ツールには以下の機能を実装した.

- 画像ファイルの閲覧
- 動画から静止画へのコンバート
- 免疫細胞の自動追跡
- 輝度断面図の表示
- 細胞の輪郭情報の取得
- 輪郭情報の時系列処理

各機能の解説及び詳細な処理の説明は以下に詳述する.

4.2　画像ファイルの閲覧

画像ファイルの閲覧機能は，免疫細胞解析ツールで生成される様々な画像ファイルを閲覧するための機能である．この機能は，メインとなるフレームクラスと画像描画用のキャンバスクラスに分かれている．処理としては，フレームクラスでツール使用者の操作を受け付け，その操作に従ってキャンバスクラスの描画内容を変化させるようになっている．フレームクラスのファイル選択部分は，Swing のファイル選択ダイアログを表示するためのクラスである JFileChooser クラスを使用した．このクラスのオブジェクトを作成し，showOpenDialog メソッドを使って「ファイルを開く」ダイアログを表示する．このダイアログ上でファイルの選択が行われた場合，選択されたファイルの絶対パスを引数に File クラスのオブジェクトを作成する．File クラスは，ファイルの読み込みや書き込みなどの操作を行うためのクラスで，ファイルやフォルダのパスを引数にオブジェクトを作成することで，オブジェクトと対象のファイルが対応付けされる．このオブジェクトを抽象パス名といい，他のファイル操作クラスの引数として使用することができる．ここでは，作成したオブジェクトと File クラスの getParent メソッドを使って，対象のファイルが格納されている親フォルダのパスを取得し，これを引数として File クラスのオブジェクトを作成する．File クラスの listFiles メソッドを使って，この抽象パス名が示すフォルダ内のファイル全ての抽象パス名が格納された File クラスの配列を作成する．この処理によって，選択したファイルが格納されているフォルダ内のファイル全ての抽象パス名を配列によって管理することができる．その後，選択されたファイルのパスを引数としてキャンバスクラスの画像ファイル読み込みメソッドを呼び出す．画像ファイル読み込みメソッドでは，画像ファイル格納用の Mat クラスオブジェクトの初期化，画像ファイルの読み込み，画像ファイルのサイズの取得

を行い，repaint メソッドでコンポーネントの再描画を行っている．画像ファイルの描画には paintComponent メソッドを使用した．paintComponent メソッドはコンポーネントの描画が必要になったときに描画処理を行うメソッドである．paintComponent では，Mat クラスのオブジェクトをそのまま描画することが出来ないため，対象のオブジェクトを一度 Mat から drawImage メソッドで描画する事ができる BufferedImage へ変換する必要がある．これを実現するために Mat クラスのオブジェクトを引数として受け取り，BufferedImage クラスのオブジェクトへの変換を行った上で戻り値として返すというメソッドを作成した．このメソッド内では，まず get 関数を使って Mat から byte クラスの配列（これを mByte とする）への変換を行う．更に戻り値として用意した BufferedImage を byte 型の配列へと変換したオブジェクト（これを biByte とする）を用意する．この byte 型の配列の要素を書き換えることで BufferedImage が更新されるようになる．Java の arraycopy メソッドを使い，mByte の要素を biByte にコピーする．biByte の要素が書き換えられたことで BufferedImage 側も内容が更新され，結果的に Mat から BufferedImage への変換が完了する．戻り値として帰ってきた BufferedImage オブジェクトを drawImage メソッドを使用してコンポーネント上に描画することで，選択した画像ファイルがウィンドウ上に表示される．

4.3　動画像から静止画へのコンバート

動画像から静止画へのコンバート機能は，図 8 のように顕微鏡で撮影された動画像を入力ファイルとし，１フレーム毎に静止画に変換し画像ファイルを生成する事ができる機能である．現在実験で使用している顕微鏡によって撮影された動画像は AVI 形式の動画ファイルである．この形式に対応するように機能の開発を行った．主な処理には OpenCV のビデオファイルやカメラのキャプチャを行うための VIdeoCapture クラスを利用した．コンバートする対象の AVI 形式の動画ファイルの絶対パスを引数として VideoCapture クラスの変数を作成することで，変数に動画ファイルが格納される．VideoCapture クラスの関数である grab 関数で処理を行うフレームを指定し，retrieve 関数で指定されたフレームを Mat クラスの変数に格納することができる．この処理を行うことで，OpenCV の画像を保存する．この処理を行うために OpenCV の imwrite 関数を利用した．この処理によって動画ファイルの 1 フレームを画像として保存する．この処理を動画ファイルの１フレーム目から最後のフレームまで繰り返し行うことで，対象の動画ファイルの全てのフレームを画像に変換する．この機能により生成される画像のファイル名は，以後の処理で使用することを考え，全て「fimg_フレーム数.bmp」の連番ファイルとした．生成された画像ファイルは全て，ツール使用者が指定したフォルダに格納される．解析に使用する AVI ファイルはフレームレートが 30fps で，動画時間は 1-2 分であるため，この機能により一度に生成される画像ファイルの枚数は 2000 から 3000 枚程度となる．

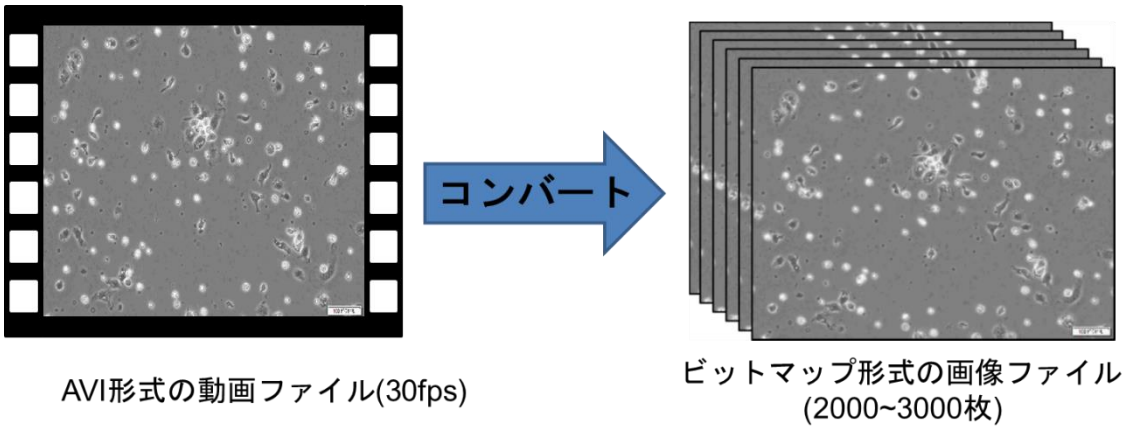

図 8　動画像から静止画へのコンバート

4.4　動画像を対象とした免疫細胞の自動追跡

免疫細胞の自動追跡機能は，コンバート機能により生成された画像群を対象に免疫細胞の自動追跡を行う．まずコンバートされた画像が格納されたフォルダの絶対パスを入力し，追跡処理を行うフォルダを決定する．選択したフォルダ内の動画像の 1 フレーム目に対応する画像内で追跡対象となる免疫細胞を選択することによって，図 9 のように対象の細胞の全体を囲む矩形の左上の XY 座標，幅，高さを細胞の位置情報とし，この情報を記録したテキストファイル（以下，「位置情報ファイル」という）を生成する．この位置情報を利用し免疫細胞の自動追跡を行う．次に自動追跡処理の流れを説明する．まず，処理の対象となるフレームの画像ファイルとその 1 フレーム前の画像ファイルと位置情報ファイルを対応する変数に格納する．画像ファイルは OpenCV の imread 関数を用いて対応する Mat クラスの変数に読み込む．この時，引数として画像ファイルの絶対パスと読み込まれる画像のカラータイプを指定する．カラータイプは，=0 でグレースケール画像，>0 でオリジナル画像となる．処理の対象となる画像に関しては，後の処理に利用するためにオリジナルとグレースケール両方の画像を用意する．位置情報ファイルは Java の文字読み込みクラスである FileReader クラスを用いた．位置情報ファイルと対応付けした File クラスのオブジェクトを引数として FileReader クラスのオブジェクトを作成し，位置情報ファイルの読み込みを行った．読み込んだだけでは，位置情報として記述された 4 種類の情報が並べられた 1 つの文字ストリームとして変数に格納されているため，１つ１つの情報を分けて変数に格納するためにこれを分割する必要がある．そこで，入力ストリームを構文解析しトークンとして切り出す事ができる StreamTokenizer クラスを用いた．作成した StreamTokenizer クラスのオブジェクト を While ループを利用してトークン１つ１つに対して処理を行う．この時，While ループの条件文は，(token.nextToken() != StreamTokenizer.TT_EOF)と記述した．nextToken とは StreamTokenizer クラスの「入力ストリームの次のトークンを構文解析する」という働きを持

つメソッドで，TT_EOF はストリームの終わりが読み込まれたことを示す定数である．これにより，入力ストリームの終わりになるまでトークンに構文解析することができる．それぞれのトークンは，StreamTokenizer の数値を扱うためのフィールドである nval を使用し，各変数に格納する．この 2 種類のファイル入力によって得られた情報をもとに免疫細胞の自動追跡を行っていく．

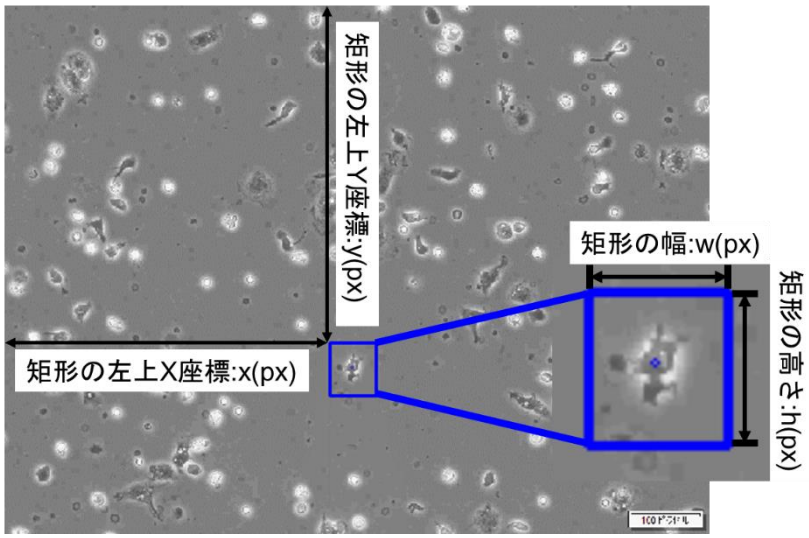

図 9　位置情報ファイルの記述内容

この時，追跡範囲の削減のため，グレースケールで読みこんだ現フレームの画像全体から，前フレームで免疫細胞が存在した位置のみを切り出した Mat クラスのオブジェクトを作成し，引数として自動追跡メソッドに渡す．自動追跡メソッドの大まかな処理の流れは，引数として渡された画像を threshold 関数で二値化し，findContours 関数で，二値化画像内の輪郭を検出する．抽出した輪郭の中で最もサイズの大きいものを追跡対象の免疫細胞とし位置情報の取得を行う．threshold 関数は入力された配列の要素に対して指定した閾値で二値化処理を行う．この関数を呼び出すためには，入力配列：src，入力配列と同じタイプの出力配列：dst，閾値：threshold，最大値：max_value，閾値処理の種類：threshold_type の 5 種類のオブジェクトを引数として渡す必要がある．この時，入力配列はシングルチャンネルである必要があり，画像の場合はグレースケール画像でなければならないため，自動追跡メソッドにはグレースケールの画像を引数として渡した．入力配列としてグレースケール画像用いた場合，入力された画像を引数として渡した閾値で二値化処理した二値化画像が生成される．二値化処理は 5 種類あるが本ツールの開発には，「THRESH_BINARY」，「THRESH_BINARY_INV」の 2 種類を使用した．「THRESH_BINARY」は配列の要素が閾値以上なら最大値に，それ以外は 0 という条件で出力配列の要素の値を決定する．「THRESH_BINARY_INV」はその逆で配列の要素が閾値以下なら最大値に，それ以外は 0 という条件で出力配列の要素の値を決定する．以下，本論文では，「THRESH_BINARY」を

用いた閾値処理を二値化,「THRESH_BINARY_INV」を用いた閾値処理を反転二値化と呼ぶ. この threshold 関数を使用して免疫細胞画像を二値化することになるが，基本的に免疫細胞は背景よりも輝度値が低い部分と高い部分に別れており，単純に二値化をするだけでは細胞全体を捉えることができない．そこで，入力画像に対してマスク処理を施すことでこの問題を解決した．図 10 のように 元画像を指定の閾値で反転二値化したマスク画素データ，元画像と同じサイズ，タイプで要素全てが 0 の画素データを生成する．この時使用する閾値は後述する輝度断面図表示ツールを利用して動画像内の免疫細胞の輝度の傾向を確認した上で決定した．生成した 2 つの画素データを引数として OpenCV の copyTo 関数を用いた元画像のコピーを行う．この時，マスク行列の要素が 0 以外の数値の場合は，元画像の対応する要素がコピーされる．要素が 0 の場合は画素データのコピーが行われず，コピー先の要素は変更されない．つまり，この処理を行うことで，元画像の輝度値が閾値以下の部分がそのままコピーされ，それ以外の部分では黒塗りの画素データが結果として出力されることになる． 結果として出力された画素データに二値化処理を行うことで，追跡対象の免疫細胞全体を捉えることができるようになる．

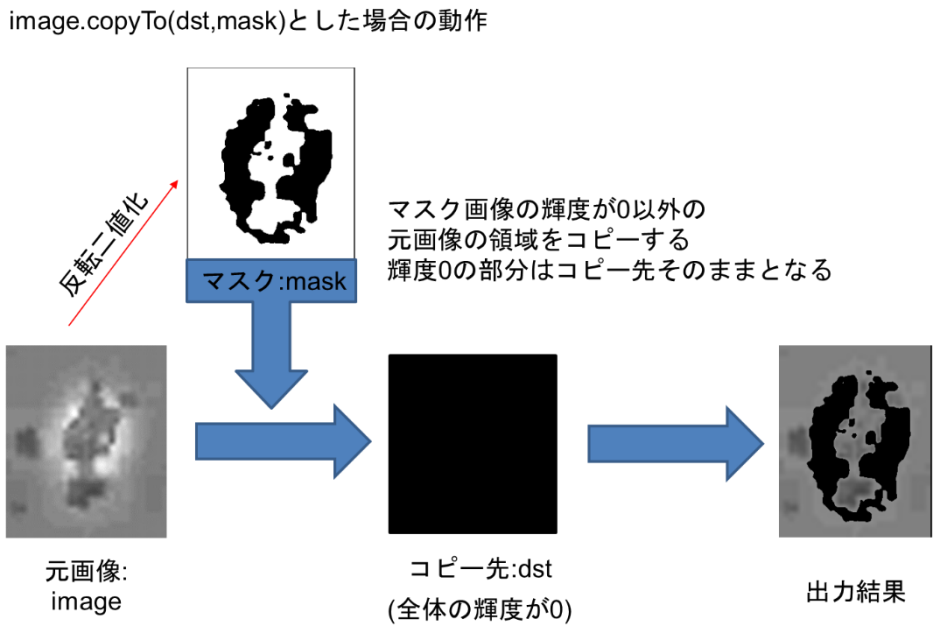

図 10　免疫細胞追跡時のマスク処理

次に，OpenCV の二値化画像の輪郭を検出するための関数である findContours 関数を用いた免疫細胞の輪郭の検出を行う．findContours 関数は Suzuki85 のアルゴリズムを利用して二値画像から輪郭を抽出する[16]．この関数は，検出対象の二値画像，検出された輪郭の座標を格納するための配列，輪郭の階層に関する情報を格納するための配列，輪郭抽出モード，輪郭の近似手法の 5 種類の引数が必要である．輪郭抽出モードとは，この関数が輪郭を抽出

する際に輪郭の構造をどの程度詳細に保持するかを決定する引数である．本研究では輪郭の階層構造までを保持する必要がないため，最も外側の輪郭のみを抽出するモードである RETR_EXTERNAL を選択した．輪郭の近似手法には，図 11 に示すように全ての輪郭点を格納する CHAIN_APPROX_NONE と水平・垂直・斜めの線分を省略し端点のみを格納する CHAIN_APPROX_SIMPLE 等がある．より正確な追跡を行うために本研究では全輪郭点を格納する CHAIN_APPROX_NONE を採用した．輪郭の検出を行うための定義として，二値画像の黒を 1，白を 0 とし，輪郭の親子関係を保持するためのラベルを 0 にする． 左上から順方向にラスタースキャンを行っていき，各ピクセルに対して以下の手順を実行する．

(1) 次の処理のいずれかを選択する．
  (a) f(i,j)=1 かつ f(i,j-1)=0 のとき，f(i,j)を外側の境界の始点であると判断し，ラベルを +1 する．このときの f(i,j-1)を $f(i_1,j_1)$とする．
  (b) f(i,j)≧1 かつ f(i,j+1)=0 のとき，f(i,j)を内側の境界の始点であると判断し，ラベルを +1 する．このときの f(i,j+1)を $f(i_1,j_1)$とする．
  (c) それ以外は(4)に進む．
(2) 新しく見つかった境界が既に検出されているどの境界と親子なのかを，ラベルで決定する．
(3) 以下の(3.1)~(3.4)の処理を実行する(ラベル値を L とする)．
  (3-1) (1)で検出された(i,j)を中心として，(i_1,j_1)から時計回りに近傍画素(8 もしくは 4 近傍)を見ていき，初めの 0 ではない画素を(i_2,j_2)とする．近傍に 0 ではない画素が存在しない場合は，f(i,j)=－L として (4)に進む．
  (3-2)(i,j)を中心として，(i_2,j_2)から反時計回りに近傍画素を見ていき，初めの 0 ではない画素を(i_3,j_3)とする．
  (3-3) 以下の様に f(i,j)の値を変更する．
    (a) f(i,j+1)=0 ならば，f(i,j)=－L とする．
    (b) f(i,j+1)≠0 ならば，f(i,j)=＋L とする．
    (c) それ以外は f(i,j)の値を変更しない．
  (3-4) (i,j)を(i_2,j_2), (i_3,j_3)を(i,j)として, (3-2)の処理へ戻る．始点の画素まで戻った場合は，(4)に進む．
(4) (i,j+1)方向へラスタースキャンを進める．

この処理をスキャンが右下隅に到達するまで繰り返し行うことで，図 12 に示すような輪郭の検出を行うことができる．処理の仕様上，輪郭の各座標は反時計回り配列に格納される．

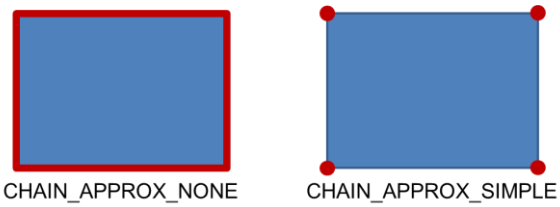

図 11　輪郭の近似手法の比較

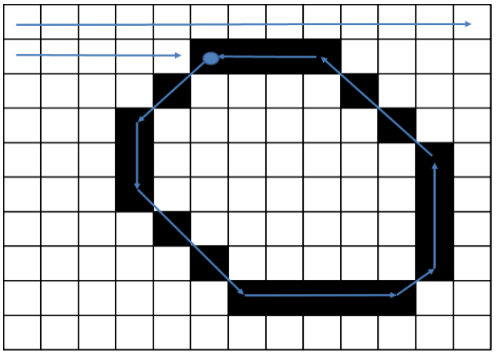

図 12　findContours 関数による輪郭検出

findContours 関数による輪郭の検出が完了すると，引数として渡した二値画像内の全ての輪郭の全輪郭点が OpenCV の MatOfPoint 型の配列に格納される．MatOfPoint 型は N 行 1 列の行列で，要素一つに double の配列(x,y)が格納される．この配列を利用して，各輪郭のサイズを調べていく．輪郭のサイズの取得には，OpenCV の arcLength 関数を用いた．この関数は，輪郭線の周囲長を求めることができる．これにより，輪郭同士のサイズの比較を行うことができる．上記の処理の結果，検出された輪郭の中で最もサイズの大きいものを，追跡対象の細胞の輪郭とし，その位置情報を記録する必要がある．そのために，OpenCV の boundingRect 関数を用いる．この関数は，MatOfPoint 型を引数として，その点群に外接する矩形を求める関数である．求められた矩形は，OpenCV の矩形を表現するためのクラステンプレートである Rect クラスの変数に格納され，その位置情報を利用することができる．細胞の位置情報の記録が終わると，対象の画像を変更して初めの処理に戻る．指定したフォルダ内の最後の画像の処理を終えるか，対象の免疫細胞が画面外に移動した段階で追跡処理を終了する．追跡処理によって生成される出力ファイルは，全体画像内の追跡対象細胞を枠線で囲んだ画像，追跡対象の細胞のみを切り取った画像（以下，細胞画像），細胞画像を 8 倍に拡大した画像，位置情報ファイル，全フレームの細胞の位置情報が記録された csv ファイルがある．細胞画像の拡大には，OpenCV の resize 関数を利用した．resize 関数を利用するためには，入力と出力それぞれの画像と x，y 方向それぞれの拡大倍率，補間手法を 引数で指定する必要がある．補間手法には，ニアレストネイバー法，バイリニア補間，バイキュービック補間などがある．上記 3 種類の補間法を使用し，画像の拡大を行った結果を図 13

に示す. 比較してみると，ニアレストネイバー法で補間された画像が最もジャギーが見られ，バイリニア法はバイキュービック法に比べて画像がぼやけて見えることが分かった．以上のことから本研究では，補間手法として他の手法よりも滑らかではっきりとした出力画像が得られるバイキュービック法を選択した.

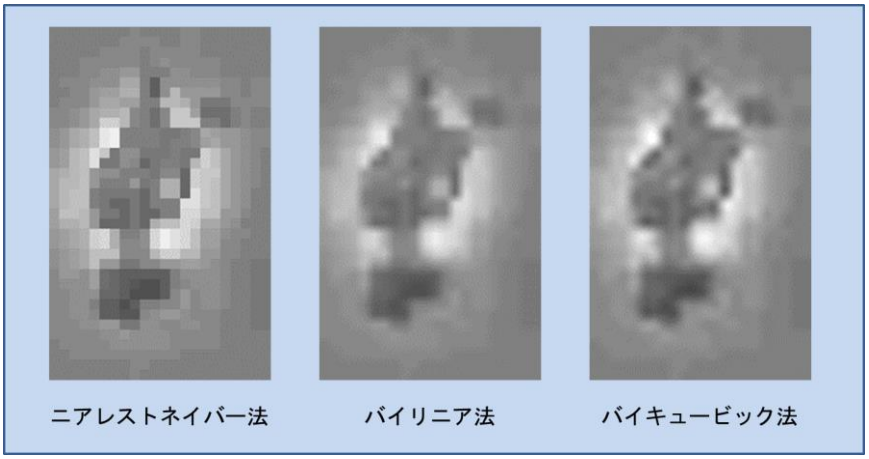

図 13　各種補間法の比較

4.5　画像輝度断面図の表示

画像輝度断面図の表示機能は，入力された画像ファイルの輝度断面図をグラフとして表示することができ，画像の輝度の傾向を視覚的に確認することができる．入力ファイルとして，bmp 形式の画像ファイルとなっている．このツールはフレームクラス，画像描画クラス，基準線描画クラス，X 軸方向の輝度断面図プロットクラス，Y 軸方向の輝度断面図プロットクラスの 5 つのクラスに別れている．ツール使用者がメインとなるフレームクラスで輝度断面図を表示する対象画像の選択を行うと，そのファイルの絶対パスを引数として画像描画クラスの画像読み込み用のメソッドが呼び出される．このメソッドで，画像ファイルの読み込み，画像サイズの取得を行ったあとで，画像をペイントコンポーネント上に描画する．基準線描画クラスでは，drawLine メソッドによって，フレームクラス上で指定された座標に十字線を描画している．各方向の輝度断面座プロットクラスには，画像ファイルの絶対パスと基準線の座標を受け渡す．受け取った座標から，図 14 のように OpenCV の get 関数を利用した各ピクセルの輝度値の取得を行う． get 関数は行と列を指定することで，Mat オブジェクトの対応するピクセルの輝度を返す変数である．この関数を x 軸なら行，y 軸なら列の値を固定して，画像の始点から終点まで for ループで繰り返し呼び出すことで，基準線に示された座標の輝度値全てを取得することができる．ここで得られた輝度値をペイントコンポーネント上にプロットすることで輝度断面図を作成している．表示された画像上をマウスがドラッグした場合，MouseMotionListener によってその動作に合わせて，マウスカーソルの座標を基準線描画クラスと各方向の輝度断面図プロットクラスに受け渡す．マウ

スカーソルの座標を受け取った各クラスでは，それに合わせてオブジェクトの値を更新し，repaint メソッドを呼び出してペイントコンポーネントの再描画を行う．

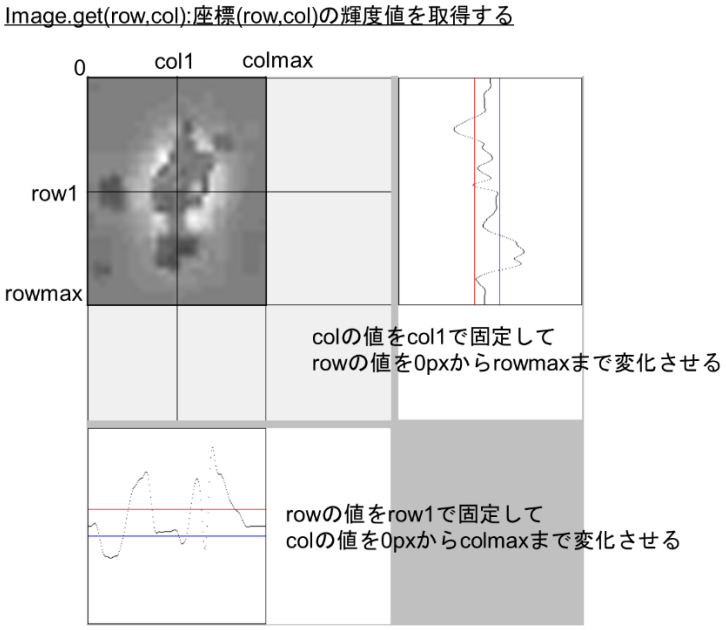

図 14　各ピクセルでの輝度値の取得

4.6　免疫細胞の輪郭情報の取得

輪郭情報取得ツールは，手動操作による細胞の輪郭情報の取得するために使用する．ツールの基本的な処理の流れについて解説する．使用者が選択した画像に対してアンシャープマスク処理を施して画像の鮮鋭化を行った上で，findContours 関数による輪郭の検出を行う[17]．この時，各輪郭に対して，boundingRect 関数を用いて輪郭に外接する矩形を求め，その座標を配列に格納する．検出された輪郭全てを drawContours 関数によって色設定を(255,0,0)で元画像上に描画した Mat クラスオブジェクトを作成し，ツール画面上に表示する．この状態で画面上に描画された輪郭をクリックすると，その輪郭のみ drawContours 関数の描画時の色設定を(0,0,255)に変更する．また，画面上でドラッグ操作を受け付けた場合，ドラッグの開始地点と終了地点の座標から仮想的な矩形を考え，これを選択範囲とする．検出された輪郭に外接する矩形が選択範囲に含まれているかを調べ，含まれている場合は描画時の色設定を(0,0,255)に，それ以外の場合は(255,0,0)に変更して再描画させる．上記の操作を輪郭の選択とする．この時，元画像と同じサイズで全体が白色の Mat クラスオブジェクトを作成し，選択が行われた輪郭のみを描画する．この画像に対して，上下左右から走査を行っていき，輝度が 0 の点に到達した場合，新規の Mat オブジェクトに点を描画し，上下の走査なら x 軸方向，左右の走査なら y 軸方向に+1 走査を進める．この処理を行うこと

で，図 15 のように輪郭の最も外側の点のみが描画された画像(以下，輪郭線画像とする)が出来上がる．輪郭線画像に複数の輪郭線が存在する場合は，boundingRect 関数によってそれぞれの線に外接する矩形を求める．求めた矩形の中で画像内の最も高い座標に存在するものを始まりとして，それぞれの位置関係をもとに時計周りに番号を割り当てていく．全ての輪郭線へ番号の割り当てが完了したら，図 16 に示すように欠けている部分を直線で補間しながら順番通りに輪郭線をつなげて描画する．これにより，免疫細胞の輪郭を取得することができる．

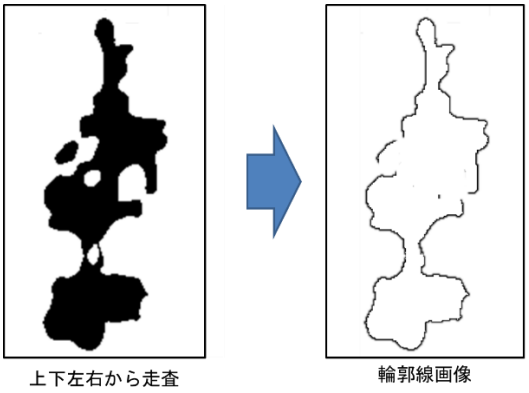

図 15　細胞の輪郭線の取得

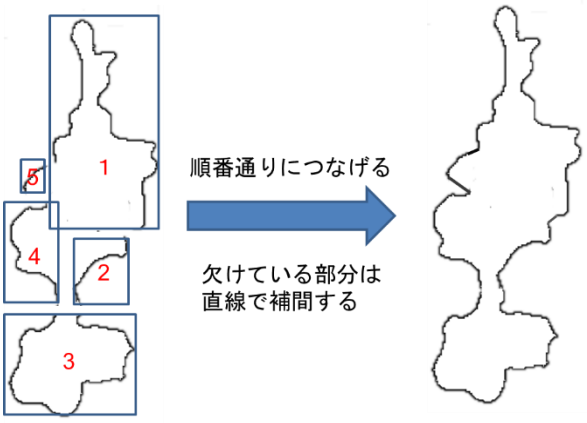

図 16　細胞輪郭の描画

4.7　輪郭情報の時系列処理

輪郭情報の時系列処理機能は指定した範囲のフレーム間での免疫細胞の輪郭情報の平均と標準偏差を求めることができる．初めに使用者が指定したフォルダ内にある輪郭画像全ての絶対パスを Java の JList に格納する．JList は格納された要素をリスト状に表示するコンポーネントである．

JList は ListCellRenderer インタフェースの getListCellRendererComponent メソッドに任意の処理を書き込むことでリストとして表示する内容を変更することが可能である．本ツールでは，図 17 のように JList に格納された絶対パスを引数に imread 関数で輪郭画像ファイルを呼び出し，その画像をリスト上に表示するように設定している．画面上に輪郭画像ファイルのリストが表示された状態で，各画像をクリックするとその画像のフォルダ内での順番が変数に格納されるようになっている．この方法で，時系列処理を行いたい範囲の始点と終点の位置を引数として平均と標準偏差を求めるためのメソッドを呼び出す．算出した結果を指定した csv 形式のファイルに imwrite 関数を用いて保存した．

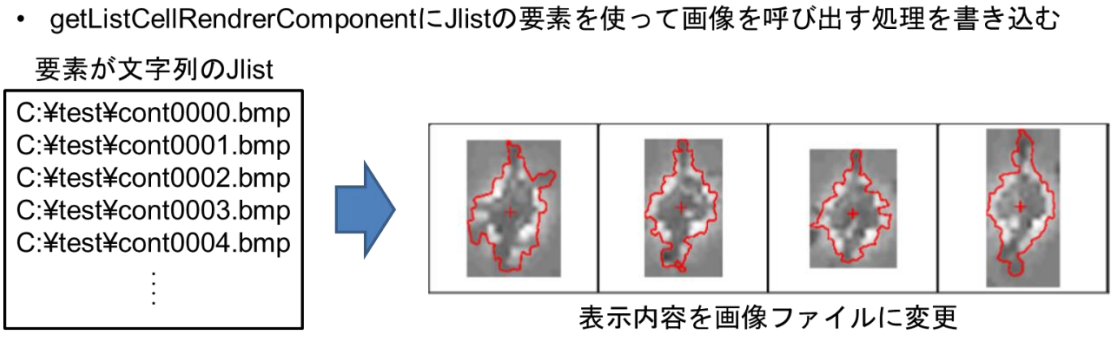

図 17　JList を利用した複数画像の表示

第５章　ツールの使用方法

5.1　画像閲覧ツール

画像の閲覧を使用する場合は，メニューバーの「ファイル」→「開く」を選択することで「ファイルを開く」ダイアログが表示される．このダイアログ上で表示したい画像を選択し，「開く」ボタンを押すことで，選択した画像が図 18 のようにウィンドウ上に表示される．ウィンドウ上に画像が表示された状態で以下の操作を行うことで表示される画像を，最初に選択した画像が保存されているファルダ内の別の画像に切り替える事ができる．

- 画面下部の「Next」，「Prev」ボタンを押す．
  →選択された画像の次，後の画像を表示する．
- 画面下部のテキストフィールドに数値を入力し，「Jump」ボタンを押す．
  →フォルダ内の入力された数値番目の画像を表示する．
- 画面上部のスライドバーを移動させる．
  →スライドバーの移動に従って，画像を切り替えていく．
- マウスホイールを回転させる．
  →マウスホイールの回転数によって，画像を切り替える．

また，画像が表示された状態で，キーボードの「Ctrl キー」を押しながらマウスホイールを回転させることで画像の拡大・縮小表示をすることができる．その他の操作として，画像が表示された状態で，メニューバーの「スライドショー」→「スライドショー開始」を選択することで，表示されている画像が保存されているフォルダの画像を順に表示していくスライドショーが開始される．スライドショーが開始された状態で，メニューバーの「スライドショー」→「スライドショー停止」を押すことでスライドショーを停止させることができる．

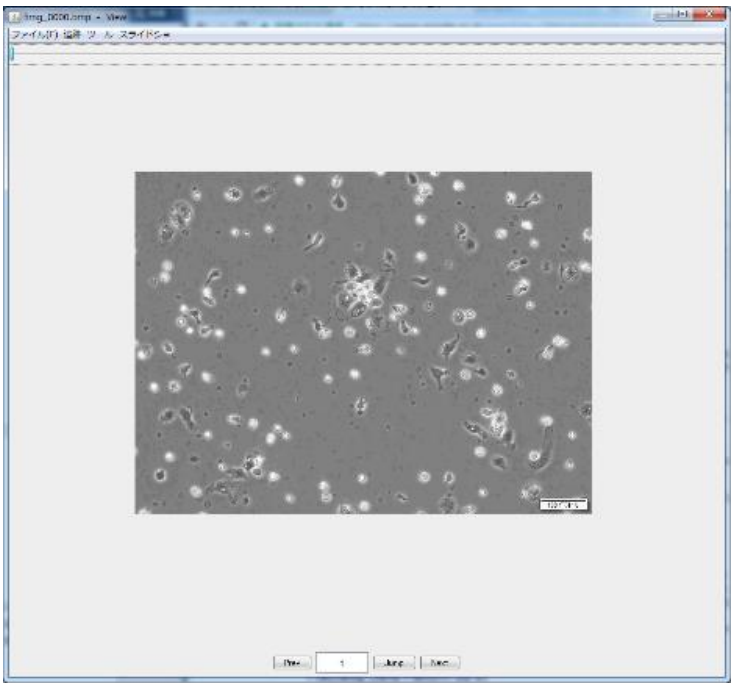

図 18　画像閲覧ツール

5.2　動画像から画像への変換機能

動画像から静止画像へ変換するために，まずメニューバーの「ファイル」→「新規」を選択する．この操作を行うと図 19 に示すウィンドウが表示される．ウィンドウ上の入力ファイルに変換を行いたい AVI 形式の動画ファイル，出力先に生成された画像群を保存したいフォルダの絶対パスをそれぞれ入力する．この時，それぞれの絶対パスを入力するためのテキストフィールドの右側のボタンを押すことで「ファイルを開く」ダイアログが表示される．このダイアログ上で対象のファイルを選択すると，テキストフィールドに自動的にファイルの絶対パスが入力されるようになっている．入力ファイルと出力先の絶対パスがテキストフィールドに入力された状態で「OK」ボタンをクリックすることで対象の動画ファイルから画像ファイルへの自動変換が行われる．この時，出力先として指定されたフォルダの直下に「fimg」，「oimg」，「cimg」，「position」，「rimg」というフォルダが生成される．この機能によって生成された画像は，「fimg_フレーム番号.bmp」というファイル名で，変換処理で生成された画像ファイルを保存するための「fimg」フォルダに保存されていく．動画変換の進捗は，図 20 のようにプログレスバーによって確認することが可能となっている．変換中にキャンセルボタンをクリックすると変換処理を中止することができる． 変換が終了した段階で図 19 と図 20 のウィンドウは自動的に閉じられる．

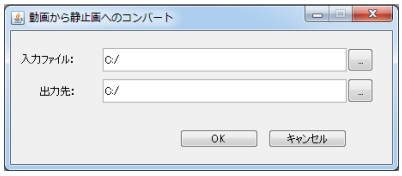

図 19　ファイル選択ウィンドウ

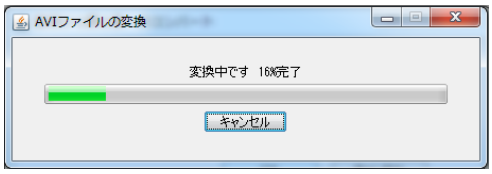

図 20　プログレスバー

5.3　免疫細胞の自動追跡機能

免疫細胞の自動追跡を開始するためにまず，追跡対象となる免疫細胞の選択を行う．メニューバーの「追跡」→「免疫細胞の選択」を選択すると，「ファイルを開く」ダイアログが表示される．ここで，動画像から静止画像への変換機能で出力先として選択したフォルダを選択する．フォルダが選択されるとその直下にある「fimg」フォルダ内の 1 枚目の画像すなわち動画像の一番初めのフレームが表示されたウィンドウが立ち上がる．このウィンドウ上でマウスをドラッグさせると，図 21 のようにドラッグした範囲が青枠で囲われる．更に，選択した範囲を 8 倍に拡大した画像を表示したウィンドウが立ち上がる．立ち上がったウィンドウを確認し，間違いが無いならキーボードの「s」キーを押す．そうすると，選択した範囲の位置情報が記録されたテキストファイルが生成され，位置情報ファイルを保存するための「position」フォルダに保存される．保存が完了すると「保存されました」と表示されたダイアログが立ち上がるので，ダイアログ上の「OK」ボタンをクリックする．ここまでの操作で追跡対象となる免疫細胞の選択が完了する．追跡対象の免疫細胞の選択が完了した状態で，メニューバーの「追跡」→「免疫細胞の追跡」を選択することで免疫細胞の自動追跡が開始される．画像の変換機能と同じようにプログレスバーで進捗の確認とキャンセルボタンで処理の中止をすることができる．自動追跡を行うことによって生成される主な出力ファイルを図 22 に示す．追跡処理の結果として生成される「全体画像+対象細胞をマーク」，「細胞画像」，「拡大画像」，「位置情報ファイル」はそれぞれ，「oimg」，「cimg」，「rimg」，「position」フォルダに保存される．位置情報が記録された csv ファイルから図 23 に示すようなプロット図を作成することができる．このプロット図から追跡対象の免疫細胞の遊走の距離や範囲を視覚的に確認することができる．

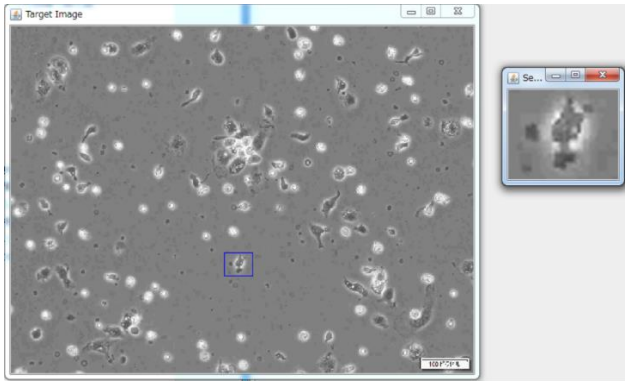

図 21　追跡対象細胞の選択画面

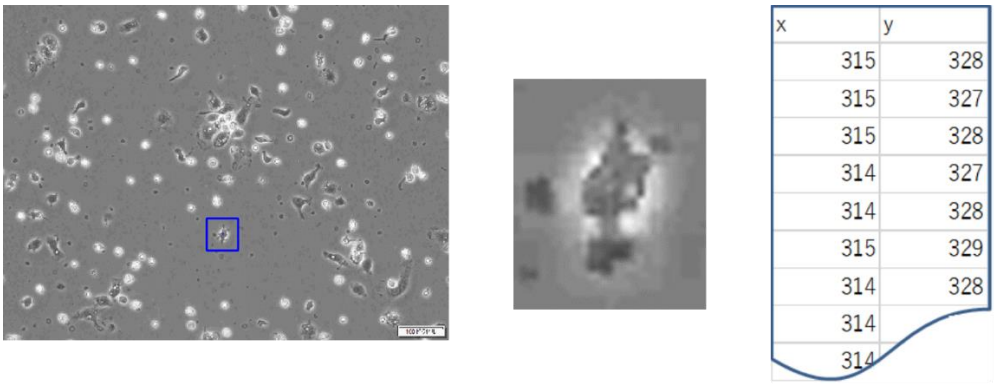

図 22　自動追跡処理後の出力ファイル

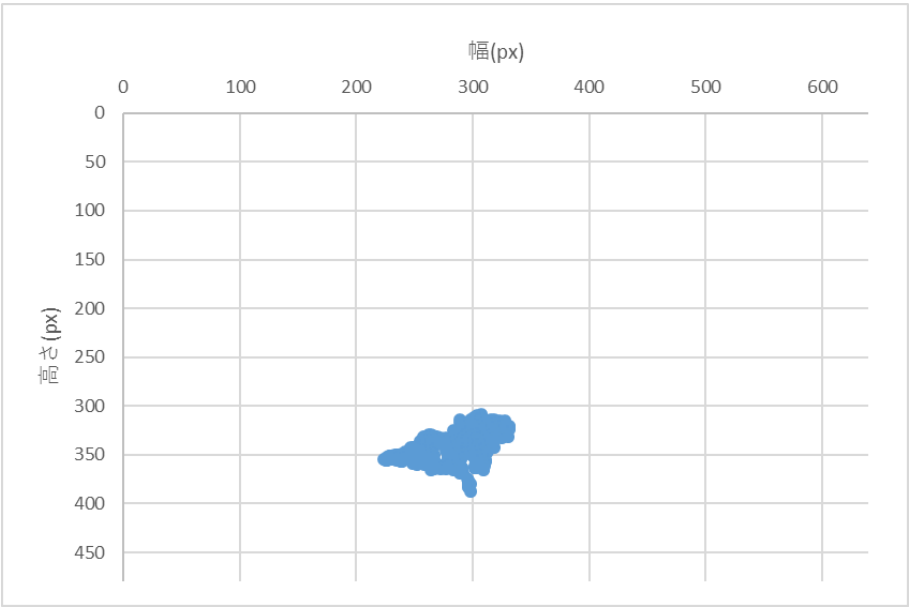

図 23　細胞の遊走データ例

5.4　輝度断面図の表示ツール

輝度断面図表示ツールは，メニューバーの「ツール」→「輝度断面図表示ツール」を選択することで起動させる事ができる．メニューバーの「ファイル」→「開く」選択すると，「ファイルを開く」ダイアログが立ち上がる．このダイアログ上で，輝度断面図を表示させたい画像を選択すると，図 24 のように選択した画像と輝度断面図が表示される．表示された画像上の基準線が通っている箇所の輝度値が画像下部と右部にプロットされる．画像上でマウスをドラッグさせることで基準線の位置を任意の場所に移動させることができる．プロットされた輝度断面図上でマウスをドラッグさせることで，各プロットの座標と輝度値がウィンドウ上に表示される．閾値 1 と閾値 2 の値を変化させることで，輝度断面図上に表示されている赤色と青色の線を上下させることができる．閾値が表示されているテキストフィールド上部の「+」,「-」ボタンをクリックすることで閾値を増減させる事ができる．表示モードは，対象画像の表示方法を変更するためのものである．表示方法の種類は，「元画像」,「二値化」,「反転二値化」,「元画像+二値化」,「元画像+反転二値化」の 5 種類がある．元画像以外の 4 種類の表示モードを選択したときに表示される画像を図 25 に示す．閾値 1 の値を用いて二値化処理を行っている．マウスホイールを回転させることで，表示されている画像を切り替えることができる．このツールを利用することで，対象の画像内の免疫細胞の輝度の傾向を確認することができ，他のツール使用時の輝度値の設定の補助ができる．

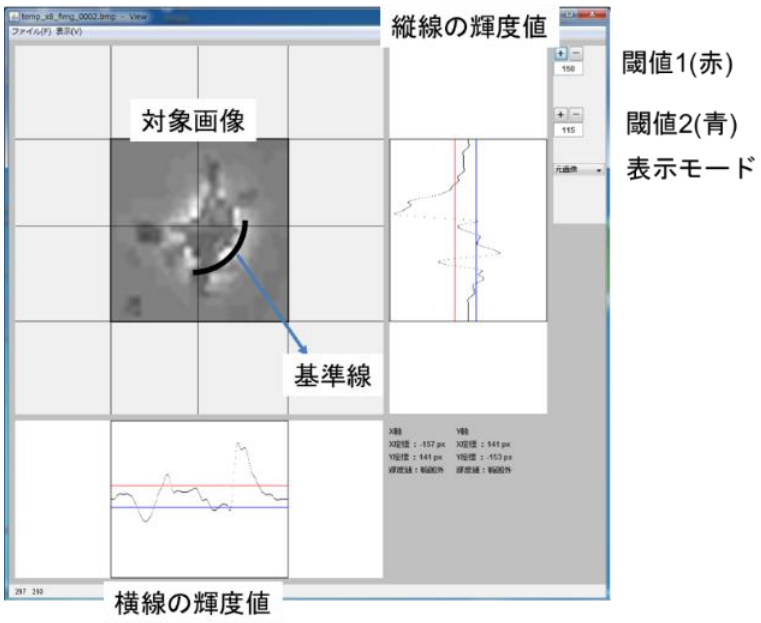

図 24　輝度断面図表示ツール

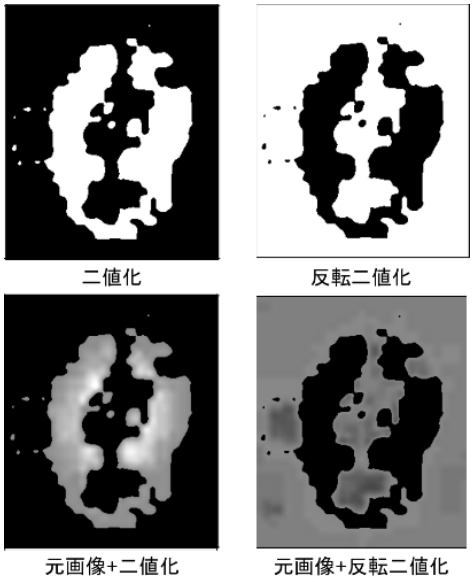

図 25　輝度断面図の各表示モード

5.5　細胞の輪郭情報の取得ツール

輪郭情報取得ツールは，メニューバーの「ツール」→「輪郭情報取得ツール」を選択することで起動させる事ができる．メニューバーの「ファイル」→「開く」を選択すると「ファイルを開く」ダイアログが立ち上がる．ダイアログ上で bmp 形式の画像ファイルを 1 枚選択すると，図 26 のように画面上に表示される．画像内で輪郭として検出された部分が自動的に青線で囲まれる．「サイズ」，「量」，「閾値」，「bth」の値を変更することで，アンシャープマスキングに用いる変数が変更され，表示が変化していく．サイズ制約の値を変更すると，表示する輪郭をサイズで制限することができる．ツール下部のテキストフィールドに付属したボタンをクリックすることで表示する画像の切り替えを行える．画像が表示された状態で，青線で囲まれた箇所をクリックすると，その箇所が赤線で囲まれる．この操作を対象の免疫細胞を構成する部位であると思われる輪郭全てに行う．また，別の選択法として，画像上でマウスをドラッグすると，その範囲が青枠で表示される．その状態で，ドラッグを終了させると，枠で囲まれていた範囲にある輪郭が全て選択された状態になる．全ての輪郭の選択が完了した状態で，「選択」ボタンをクリックすると図 27 のような画像が表示される．この状態で「セーブ」ボタンをクリックすると，赤線で囲まれた部分を免疫細胞として，その輪郭情報を記録することができる．もし，細胞の輪郭を表示した段階で，輪郭が綺麗に表示されていない場合は，「編集」チェックボックスにチェックを入れることで，各輪郭をつなげる際の順番を変更することができる．この操作を行うと，図 28 のように輪郭として認識された線を囲む青色の矩形と，それらの線をつなげる順番を示す数値が表示される．更に，ツール右中央にそのリストが表示される．このリスト内で数値をクリックし，ドラッグさせると選択した数値を移動させることができる．順番の修正を行った状態で，再度「選択」ボ

タンをクリックすると，先の操作で並べ替えた順番通りに輪郭同士をつなげた画像が新たに表示される．数フレーム分の免疫細胞の輪郭を記録した状態で，メニューバーの「ファイル」→「保存」を選択することで，「ファイルを保存する」ダイアログが立ち上がる．ダイアログ上でファイルを保存したいフォルダを選択することで，それまでに記録した免疫細胞全ての輪郭情報が記録された csv ファイルと元画像上に輪郭が描画された画像ファイルが生成される．それとは別に細胞の頂点を 0° として，-180° ~180° までの角度ごとの重心から輪郭までの距離が記録された csv ファイルも生成される．この csv ファイルから図 29 のようなグラフを生成することができ，フレーム毎の免疫細胞の形態変化を比較することもできる．

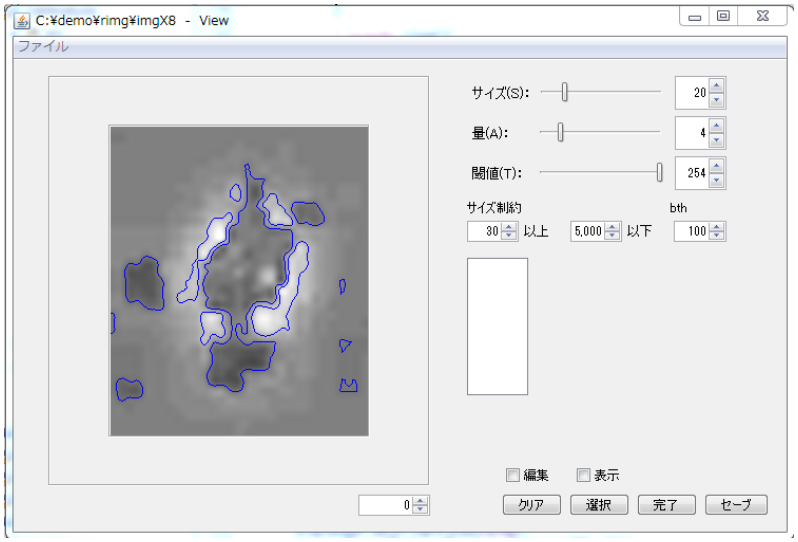

図 26　輪郭情報取得ツール

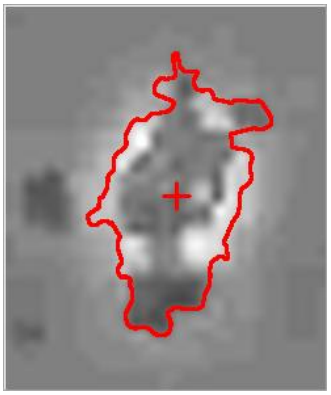

図 27　輪郭画像

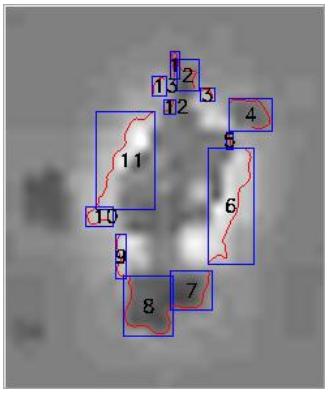

図 28　輪郭番号の変更

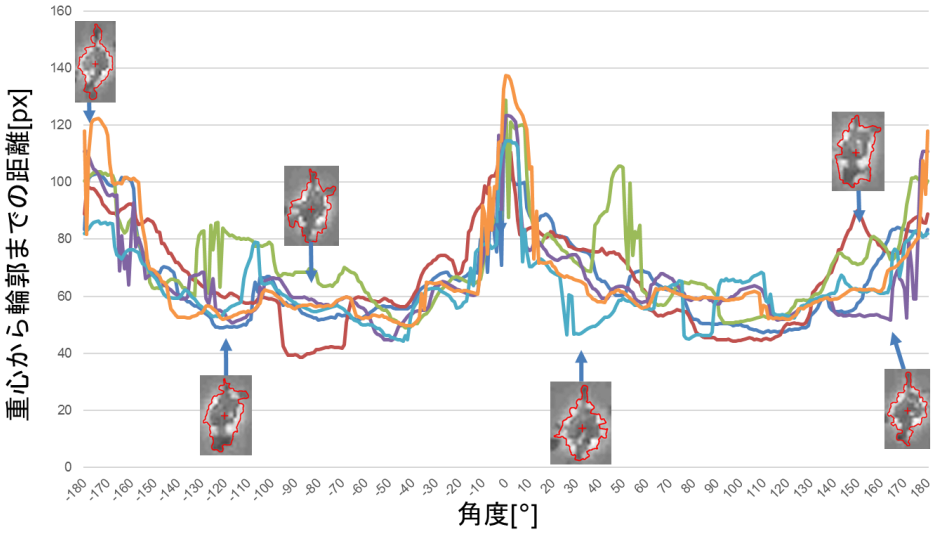

図 29　1 フレーム毎の形状変化の比較

5.6　輪郭情報の時系列処理機能

メニューバーの「ファイル」→「入力ファイル」を選択すると図 30 のウィンドウが立ち上がる．ウィンドウ上で輪郭画像が格納されたフォルダと輪郭情報が記述された csv ファイルの絶対パスを入力し，「OK」ボタンをクリックすることで選択したフォルダ内に存在する輪郭画像が図 31 のように表示される．画像下部のスライドバーを移動させるか，マウスホイールを回転させることで表示されている画像を切り替える事ができる．キーボードの「Shift」キーを押しながら表示されている画像をクリックすると，選択した画像が赤枠で囲われる．この状態を選択された状態とする．時系列処理を行いたいフレーム群の始点と終点でこの操作をそれぞれ行うことで，その区間の画像全てが選択された状態となる．処理を行う区間の選択が完了した状態で「記録」ボタンをクリックすると，その区間の重心から仮足の輪郭点までの距離の平均と標準偏差が計算される．計算した結果は，メニューバーの「ファイル」→「保存」から「ファイルを保存する」ダイアログ上で出力先を指定することで，csv ファイルとして保存する事ができる．この機能を利用することにより，選択したフレーム間での細胞の仮足の形態変化を記録することができるようになる．複数区間で形態変化のデータを記録すれば，対象の免疫細胞の仮足の形態変化の度合いを確認できる．

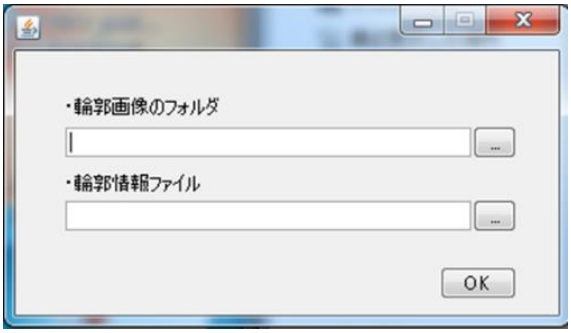

図 30　入力ファイル選択画面

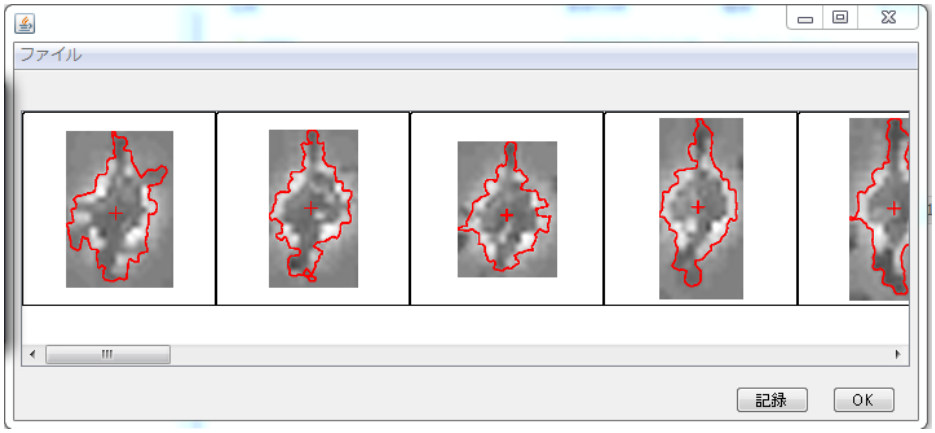

図 31　時系列処理時の画面

第６章　結論

本研究では，一般的な医学部で行われている免疫細胞の解析作業を支援するツールの開発を行った．実際に行われている解析作業をもとにツールに実装する機能を決定した．各種ツールの生成画像を閲覧するためのツールを開発した．AVI 形式の動画像ファイルをフレーム毎に切り出し，画像ファイルへ変換し出力する機能を実装した．手動で選択された免疫細胞を自動追跡し，対象細胞の遊走距離を記録する機能を実装した．追跡結果のファイルから免疫細胞の遊走を視覚的に確認できることを示した．入力された画像の輝度断面図を表示するツールを開発した．追跡を行った免疫細胞の輪郭情報を取得するツールの開発を行った．このツールを利用することで免疫細胞の形態変化に関するデータを取得できることを示した．免疫細胞の形態変化に関するデータの時系列処理機能を実装した．

今後の検討内容としては，機械学習などを利用した免疫細胞の自動追跡精度の向上や各ツールのインタフェースの改良などが挙げられる．

謝辞

本研究を進めるにあたり，常に適切かつ的確なご指導を賜りました高知工科大学大学院工学研究科基盤工学専攻電子・光システム工学コース星野孝総准教授に厚く御礼申し上げます.

そして，本研究に関して多忙にも関わらずご助言をくださいました同コースの山本真行教授，綿森道夫准教授に深く感謝申し上げます.

そして，研究活動全般において御協力，御指導して下さいました星野研究室の諸先輩方に心より感謝の意を表します.

最後になりましたが，卒業するまで本研究活動を辛抱強く・温かく見守ってくれた両親，家族一同に深く感謝します.

参考文献

[1] Scotti, F. “Robust segmentation and measurements techniques of white cells in blood microscope images.” In: Instrumentation and Measurement Technology Conference, 2006. IMTC 2006. Proceedings of the IEEE. IEEE, pp. 43-48, 2006.

[2] 永田毅, 二田晴彦, 前川秀生,“生物・医療，材料・機器分野向け「高度画像処理ソリューション」”, 画像ラボ, 25(5), pp.48-51, 2014.

[3] Benjamin Culeux, Tomoharu Nakashima, Manabu Nii, Toshinobu Hayashi, and Yutaka Komai, “A Macrophage Simulator for Understanding Its Dynamic Behaviour from Video Images, Proc. of 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC2013), pp. 1795-1798, 2013.

[4] Tomoharu Nakashima, Benjamin Celeux, Toshinobu Hayashi, Manabu Nii, Yutaka Komai, “A Study on Reproducing the Dynamic Behavior of Macrophages from Video Images,” Proc. of The 14th International Symposium on Advanced Intelligent Systems (ISIS2013), 2013.

[5] Benjamin Culeux, Tomoharu Nakashima, Toshinobu Hayashi, Manabu Nii and Yutaka Komai, “A Three-Dimensional Macrophage Visualizer”, Proc. of First IIAI International Conference on Advanced Information Technologies 2013 (IIAI-AIT2013) , 2013.

[6] Benjamin Culeux, Tomoharu Nakashima, Manabu Nii, Toshinobu Hayashi, and Yutaka Komai, “A Three-Dimensional Macrophages Movement Analyser,” Proc. of Joint 7th International Conference on Soft Computing and Intelligent Systems and 15th International Symposium on Advanced Intelligent Systems (SCIS&ISIS2014), pp.1424-1427, 2014.

[7] Benjamin Culeux, Tomoharu Nakahima, Toshinobu Hayashi, Manabu Nii, and Yutaka Komai, “Finding Patterns to Characterize the Macrophages Behaviour,” Proc. of the 11th International Conference on Knowledge Management (ICKM2015), pp.374-379, 2015.

[8] 中島 智晴，Benjamin Culeux，新居 学，駒井 豊，“二光分子顕微鏡画像に対するマクロファージ追跡手法の一検討”，第 30 回ファジィシステムシンポジウム講演論文集， pp.408-409， 2014.

[9] 前田長正, 泉谷知明, 楠目智章, 益本貴之， 深谷孝夫.，” 子宮腺筋症の病態に関わる NK レセプターと腹腔マクロファージの免疫学的検討 (第 27 群 子宮内膜症・腺筋症 1)”. 日本産科婦人科學會雜誌, 56(2), 401,2004.

[10] Meijering, E, O. Dzyubachyk, and I. Smal. "9 Methods for Cell and Particle Tracking." Methods in enzymology 504.9, pp. 183-200, 2012.

[11] Laganiere, R. “OpenCV 2 Computer Vision Application Programming Cookbook”, pp.90-94, 2009.

[12] 中山清喬，国本大悟，”スッキリわかる Java 入門　第 2 版”，インプレス，pp.618-626, 2015.

[13] WindowBuilder, www.eclipse.org/windowbuilder/, 2018 年 1 月 31 日参照

[14] 奈良先端科学技術大学院大学 OpenCV プログラミングブック制作チーム, “OpenCV プログラミングブック”, 毎日コミュニケーションズ, pp.124-155, 2007.
[15] 谷口慶治編，”画像処理工学基礎編”，共立出版，pp.80-96，1996.
[16] Polesel, A., Ramponi, G., & Mathews, V. J. "Image enhancement via adaptive unsharp masking." IEEE transactions on image processing 9.3, pp. 505-510, 2000.
[17] S.Suzuki and K.Abe, “Topological structural analysis of digitized binary images by border following”, Computer Vision, Graphics, and Image Processing. 30(1) 32-46, 1985

.

研究業績

学会発表

1. 第 29 回バイオメディカル・ファジィ・システム学会年次大会，2016/11/26-27，高知工科大学永国寺キャンパス，”免疫細胞の画像解析ツールの試作”，井上　智哉，牛若　昂志，前田　長正，星野　孝総.
2. 第 33 回ファジィシステムシンポジウム FSS2017 in 米沢，2017/9/13-15，山形大学米沢キャンパス，”免疫細胞の顕微鏡画像のツール開発と評価”，井上　智哉，牛若　昂志，前田　長正，星野　孝総.
3. The 18th International Symposium on Advanced Intelligent Systems, 2017/10/11-14, EXCO in Daegu, “Development and evaluation of analysis tool for microscopic images of immune cells”, Tomoya Inoue, Takashi Ushiwaka, Nagamasa Maeda, and Yukinobu Hoshino.