2012「画像科学」夏の勉強会
- ImageJの基礎からマクロの書き方まで -

三浦耕太

Centre for Molecular and Cellular Imaging
EMBL Heidelberg, Germany

miura@embl.de
http://cmci.embl.de

2012.09.21

ImageJ は誰もがダウンロードしてすぐに使うことのできるパブリックドメインの画像処理・解析ソフトである。GUI を使ったマウスのみによる操作から Java プログラミングによる機能の追加、クラスタを使った大規模な計算まで幅広い使用が可能であること、数多くの研究者・開発者が関わっていることなどから、生物学研究者にとってその習得にはさまざまな利点がある。今回はドイツの EMBL で年に二度ほど開講しているチュートリアルを行う。このチュートリアルは、生物系の院生・ポスドク・研究者の多くが画像処理・解析を学んだ経験をもたない一方、イメージングが日常的になっている研究の現場でとりあえず困らないための実戦的なクラッシュコースとして始まった。直観的な理解をなによりも重視している。こうした手段にすでに熟練している方には、物足りない内容になるかもしれない。

この文書ではチュートリアルの概要と必要事項についてのみ記述する。メインの文書は下記にリンクする CMCI のウェブサイトから入手可能なテキストである。

1　チュートリアルの準備

まずウェブブラウザのブックマークに入れておいて欲しいサイトがある。ImageJ の本家。

http://imagej.net

基本的にすべての情報はこちらからアクセス可能である。

1.1　テキストのダウンロード

チュートリアルではスライドを用いて解説を行い、並行して各自ラップトップで演習を行う。スライドの内容は次のリンク先のページからダウンロードできる 2 冊のテキスト（PDF ファイル）に全て書かれている。

http://cmci.embl.de/documents/ijcourses

印刷されたテキストが必要な方は、各自印刷して持参していただきたい。

1.2 Fiji のインストール

まず次のサイトからそれぞれの OS に適合したインストーラをダウンロードする。

`http://fiji.sc/Downloads`

上記ダウンロードページのインストールの仕方を読み、圧縮ファイルを解凍した後、所定のスッテプにしたがい Fiji をインストールする。完了したら、Fiji メニューの

`[Help > Update Fiji]`[1]

によってプログラムを最新の状態にする。アップデータのモジュール自体が新しくなるなどの場合、何度かアップデートと Fiji の再起動を繰り返すことが必要になる。

1.2.1 Fiji に関する簡単な解説

Fiji は数多くある ImageJ のディストリビューションの一つである。複数の研究者や開発者が Git を通じて継続的に機能を追加・修正しているシステムという特徴がある。このためコードはすべてオープンであるが、科学研究において使われるすべてのコードはオープンでなければ科学的な評価ができない、という立場からもオープンソースを推進している。ユーザーの立場から眺めると、ImageJ に比べて機能がとても多い。まずなんのためにどの機能を使ったら良いのか、という点でつまずくケースを多く見かける。とはいえ、習熟した暁には多様な機能を複合的に使うことができるようになり、更にはプラグインなどを独自に開発する際に、自前ですべてを書くことなくさまざまなライブラリを組み合わせることで効率的に目的の機能の実装を果たすことが可能になる。今回 Fiji を使用する主な理由は、マクロをプログラミングする際に Script Editor が簡便だからである。

[1]以下、ブラケットに囲まれた文章は、メニューのヒエラルキーを示す。

1.3 演習用画像のためのプラグイン

次のページのプラグインをダウンロードし、Fiji に追加する。

`http://cmci.embl.de/downloads/sampleimageloader`

CMCI のトップページ（cmci.embl.de）からは、`Downloads > Sample Image Loader` を辿る。ローカルに保存したファイルの名前は `EMBL_sampleimages.jar` のはずである。これを確認したら、Fiji のメニューから、

`[Plugins > Install Plugin]`

を行い、`EMBL_sampleimages.jar` を選ぶ。インストールが成功するとメニューに”EMBL”という項目が現れる。もしこの項目が現れなかったら、

`[Help > Refresh Menus]`

により、メニューを再読み込みする。うまくインストールの完了をチェックするため、

`[EMBL > Samples > blobs.tif]`

により、画像 “bobs.tif” をネットを通じてロードできることを確認する。

2 画像処理・解析の基礎

生物学で画像を扱う上で必要な知識を実際に手を動かしながら学ぶのがこのチュートリアルの目的である。画像を数値データとして扱うことは日常的な意味での画像の扱いとは異なることを体感し、定量的に処理する技法を学ぶ。

一日目の 6 時間は基礎の演習を行う。項目は以下の通り。詳細はテキスト “Basics in Image Processing and Analysis” を参照。

- 基礎の基礎
- 強度
- フィルタリング
- 分節化
- 時系列の解析

通常は 8 時間ほどかかる内容で更に宿題があるが、経験者が多いことも考え 6 時間に短縮するつもりである。

3　マクロの書き方

ImageJ のマクロは作業を自動化するために簡便なスクリプティング言語である。GUI のメニューと密接に対応しており、メニューを操作し、マクロレコーダでコマンドを抽出しつつ結果を目で確認しながら開発できるので容易である。習得速度が速い。

二日目の 3 時間は ImageJ マクロの書き方を学ぶ。時間がかぎられるので、基礎となる部分を集中しておこなう。内容の詳細、およびチュートリアルで触れる時間のない高度な運用に関してはテキスト “Macro Programming in ImageJ” を参照されたい。

- Script Editor の使い方、Hello World、コマンドリファレンス
- 変数の扱い
- マクロレコーダ
- ループ・分枝
- 配列・輝度プロファイルの例

3.1　他のスクリプト言語に関するノート

ImageJ のマクロは GUI を経由したスクリプトであるため、ヘッドレス（GUI なし）で使用する際にはさまざまな制限が生じる。また、マクロの関数が用意されていないプラグインを使う場合などに不自由を感じることがある。これらの問題があるときには Java の仕様（API と呼ばれる）に直接アクセスできるスクリプト言語を使う必要がある。

JavaAPI を使うには Java についてある程度の知識を持っていることが必須であり、Javadoc と呼ばれる仕様書（ImageJ の場合には ImageJ の Javadoc）を繰りながらスクリプトを書くことになる。ImageJ の Javadoc は次のリンク先にある。

`http://imagej.net/developer/api/`

ImageJ におけるスクリプティング言語としては主に Javascript(Rhino) や、Jython（Java で実装した Python) などがある。Javascript は ImageJ においてそのまま使うことが出来る。また、Javascript はマクロレコーダの記録言語としても実装されており、マクロと同じように記述することができる。ただしこの場合の運用は、マクロと同程度の機能に限られるので、Javadoc を駆使しながらコーディングを行うのでなければマクロでプログラムを書くことをお勧めする。

Jython は Python の文法であること、Jython 自体に実装されているさまざまな機能があることから（特に文字列操作、ファイルシステムへの IO においてさまざまなメリットがある）、Jython を使う研究者が多い。ImageJ で Jython を使うには Jython のライブラリをインストールし、自分で設定を行う必要がある。Fiji では最初から Jython が導入されている。

処理速度を高めるならば、Clojure もしくは Scala を使う。

今回のチュートリアルでは触れないが、テキストの最終章は Javascript の導入について書かれているので必要のある方は参照されたい。