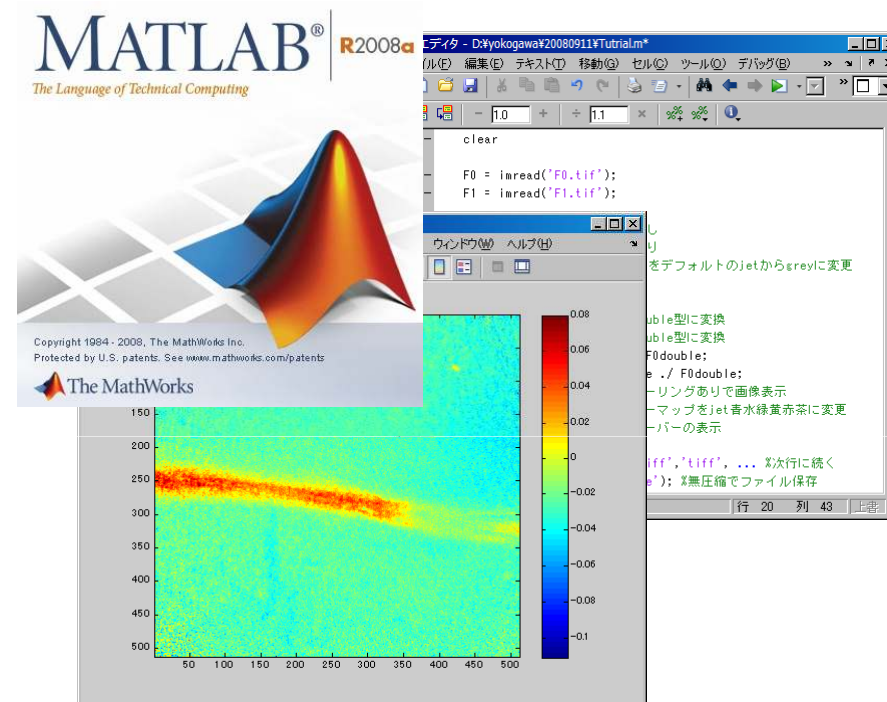
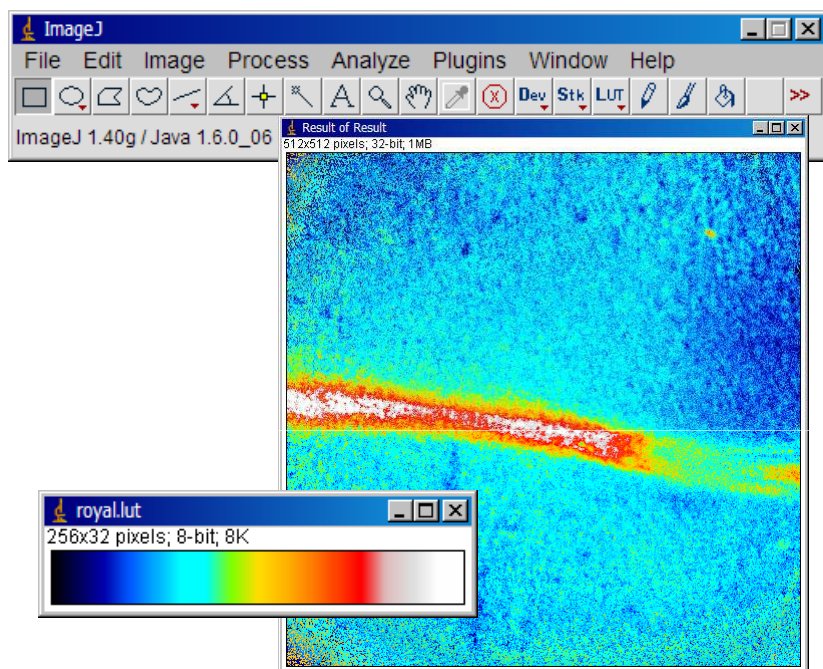


# ImageJとMATLABを用いた 定量的画像解析チュートリアル



大阪バイオサイエンス研究所  
所長研究部 特別研究員 土居智和  
tomdoi@obi.or.jp  
10 January 2009

27 Dec 2008版

# 画像取得と保存時の注意

まずは綺麗なチャンピオンデータを得ないと解析方針が立たない

スケールバーの情報は失われるので、レンズ倍率はメモしておく。

画像ファイルは、無圧縮(TIFF、BMP)または可逆形式(PNG)で保存。  
不可逆圧縮(JPEG、GIF)は定量性が失われる。おすすめはTIFF。

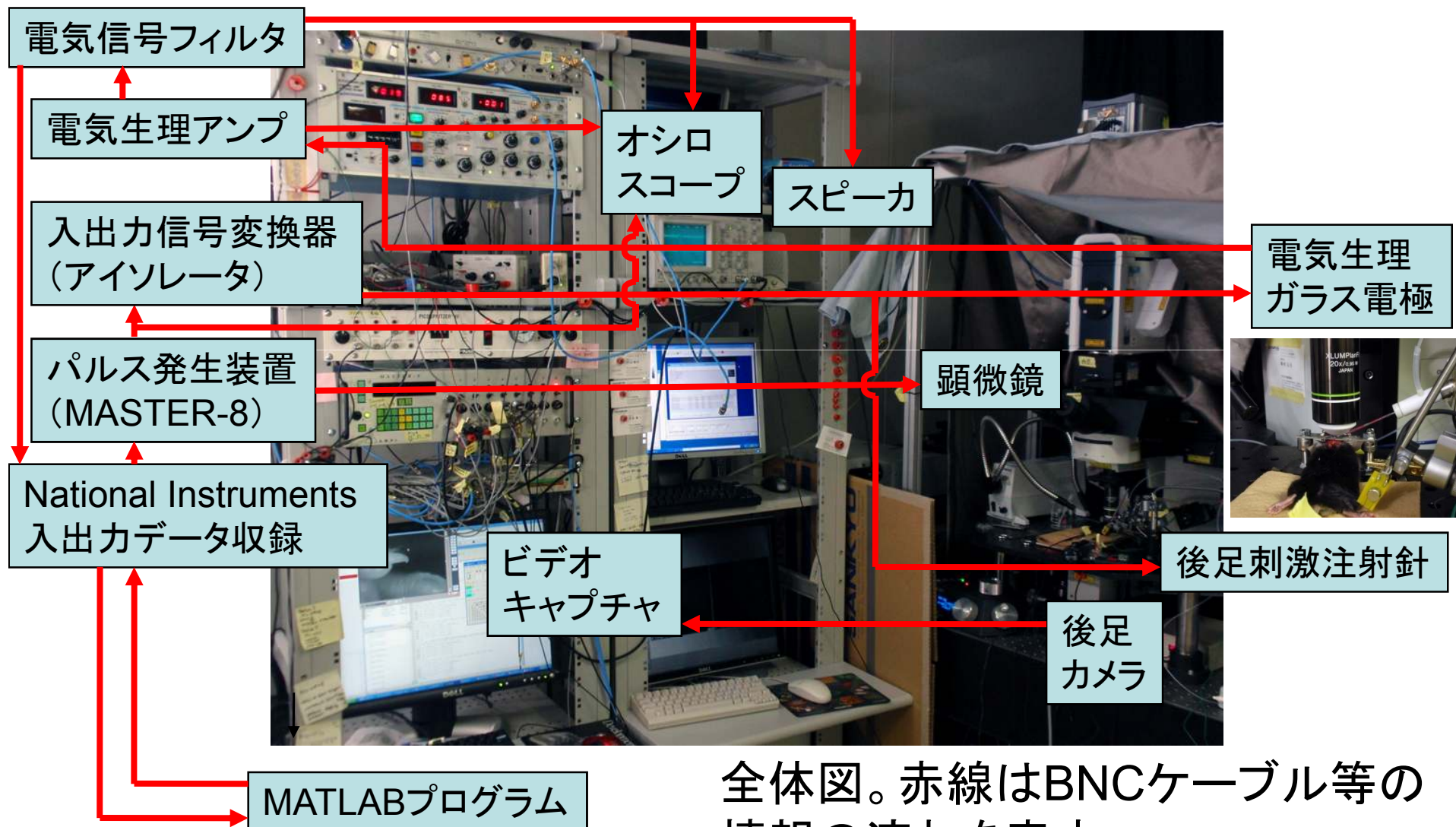
時系列データは1枚1枚のタイムスタンプが正確に記録されるか事前に確認。全速力モードでは取りこぼし、タイムスタンプ詐称の可能性あり。

他の実験装置とタイミングを合わせるときは、  
BNCケーブルをつないでTTL信号(トリガ)が  
来たらスキャン開始するように顕微鏡を設定



BNCケーブル

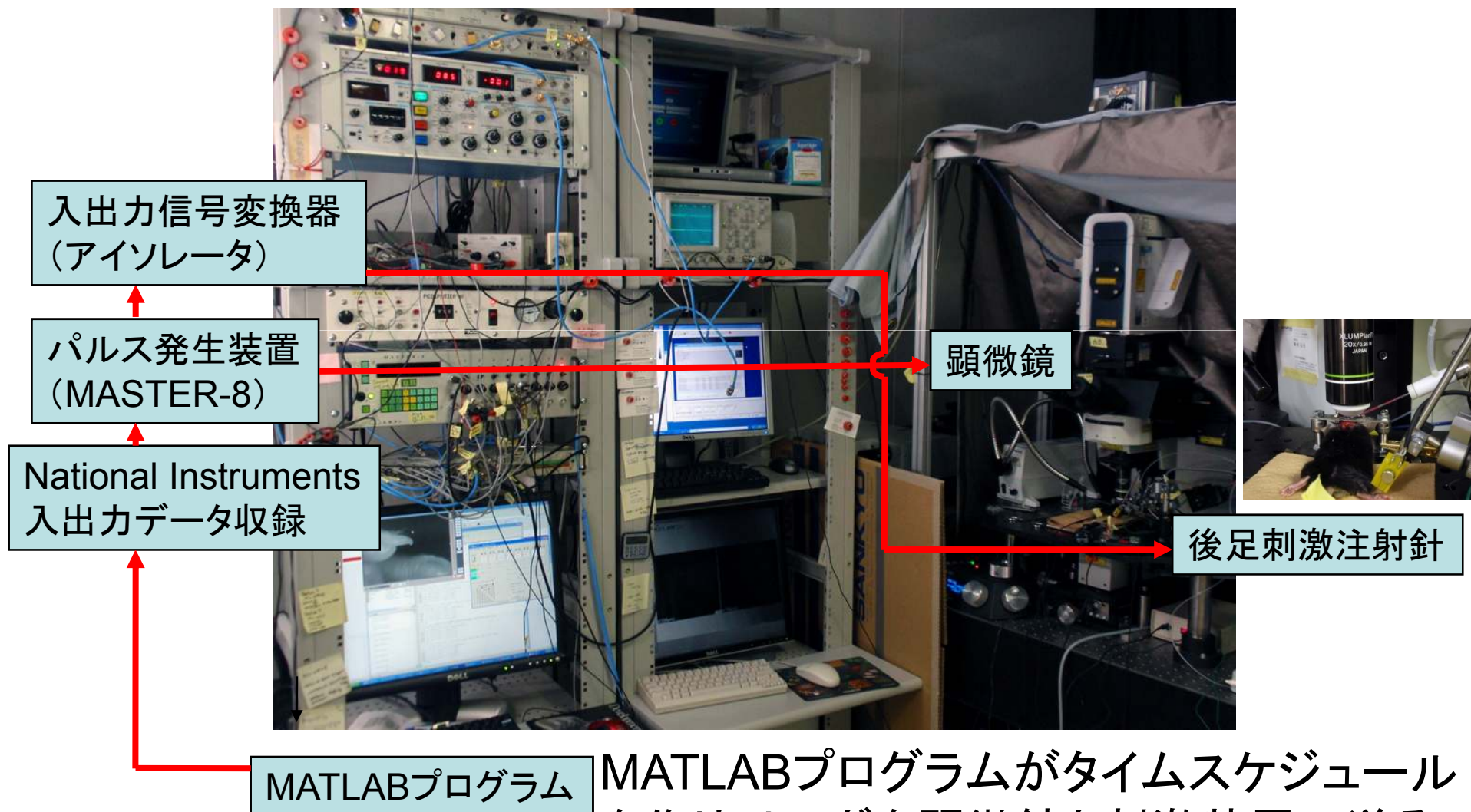
# 実際の画像取得セットアップ



全体図。赤線はBNCケーブル等の情報の流れを表す。

# 実際の画像取得セットアップ

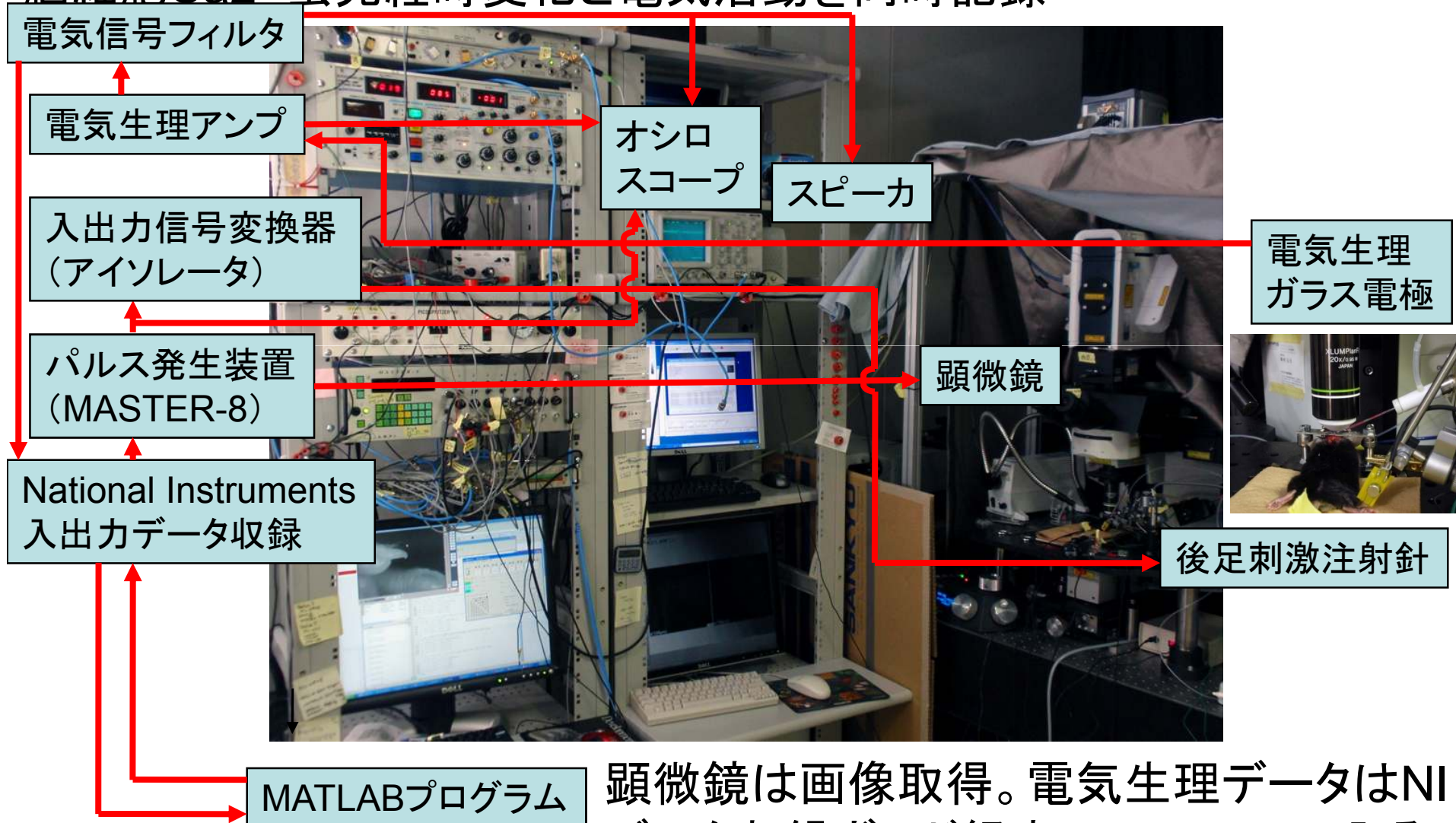
マウス後足に電気ショックを与えながら、  
脳細胞Ca<sup>2+</sup>蛍光経時変化と電気活動を同時記録



MATLABプログラムがタイムスケジュール  
を作り、トリガを顕微鏡と刺激装置に送る

# 実際の画像取得セットアップ

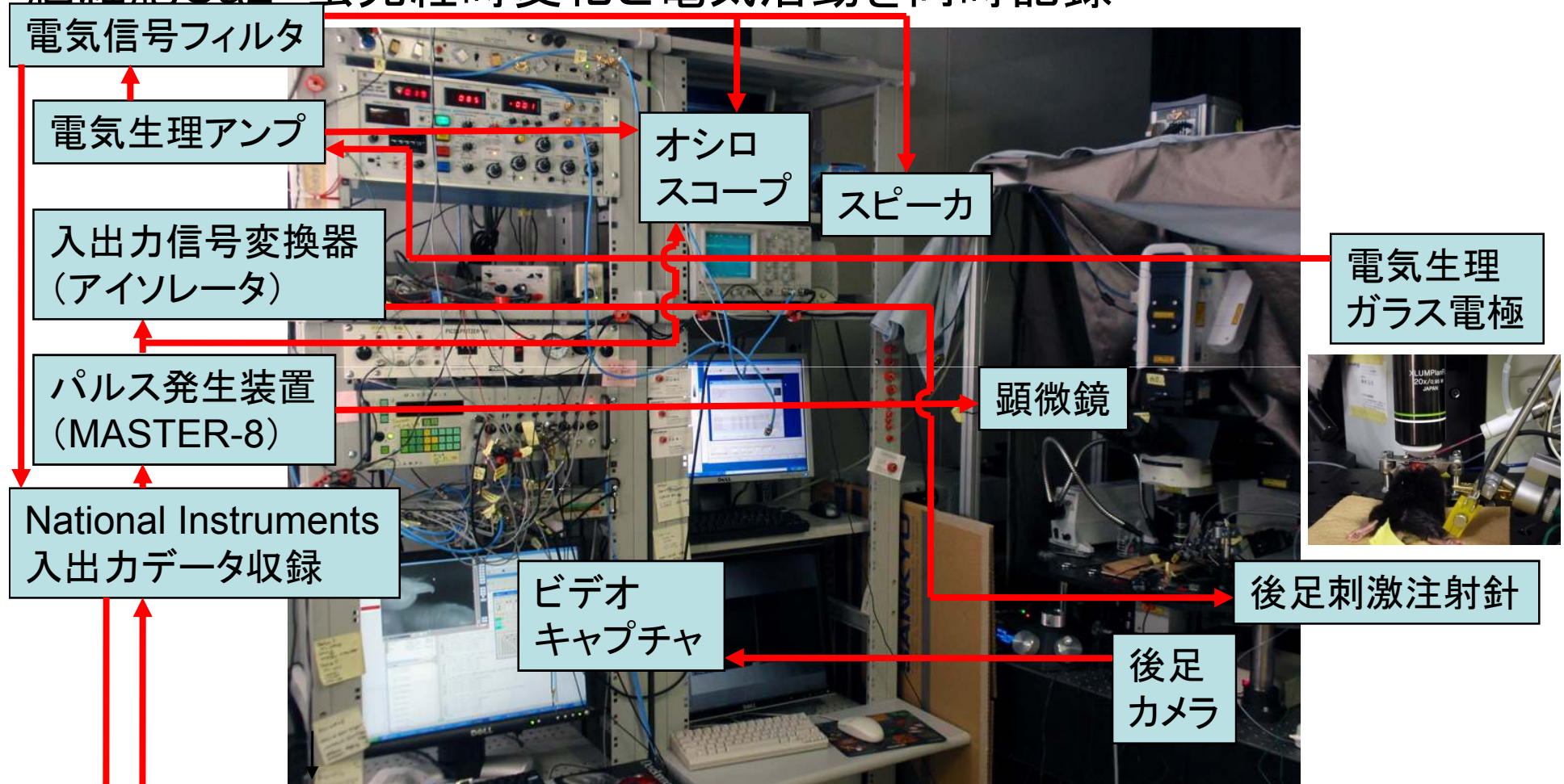
マウス後足に電気ショックを与えながら、  
脳細胞Ca<sup>2+</sup>蛍光経時変化と電気活動を同時記録



顕微鏡は画像取得。電気生理データはNIデータ収録ボード経由でMATLABに入る

# 実際の画像取得セットアップ

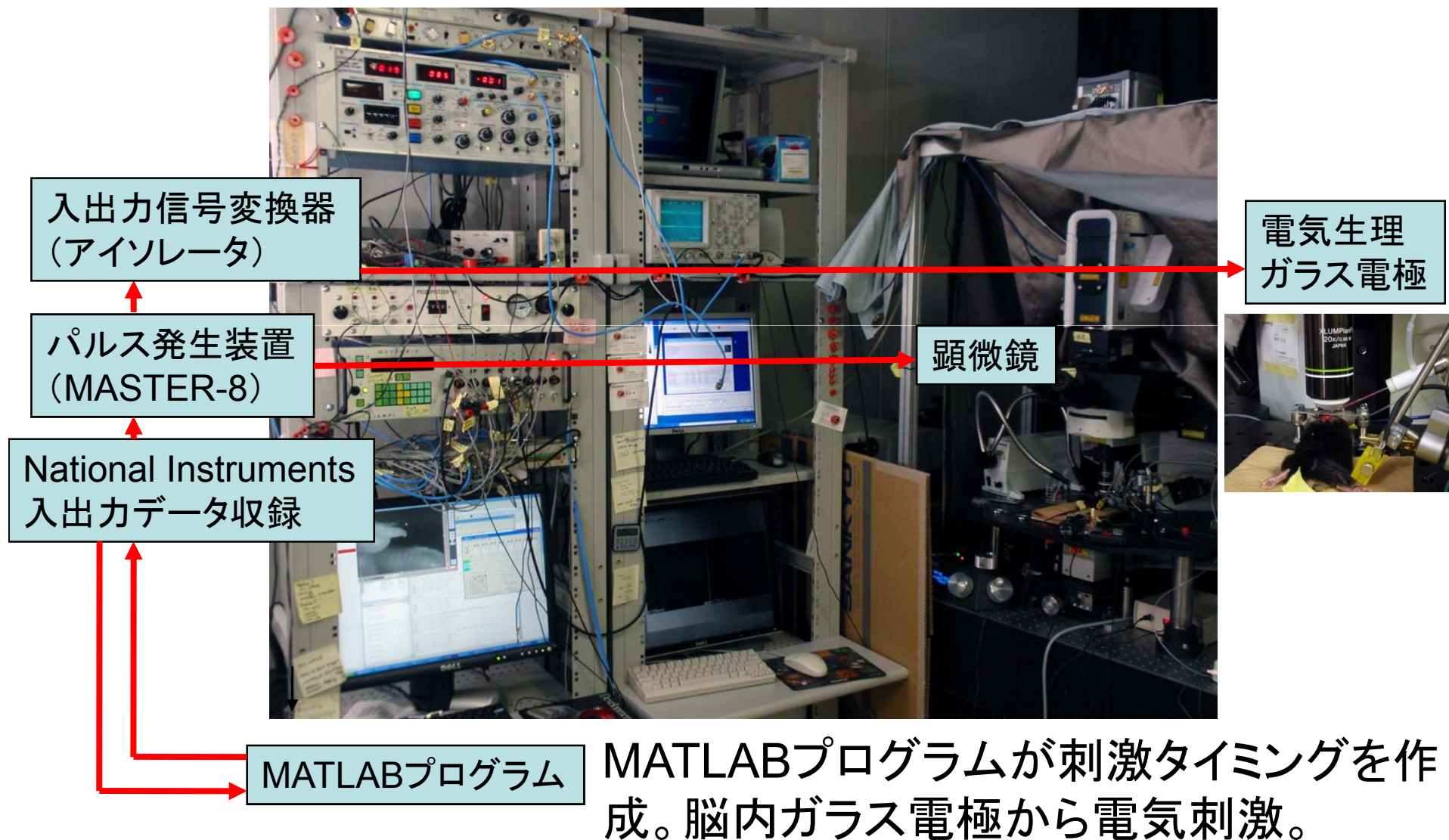
マウス後足に電気ショックを与えながら、  
脳細胞Ca<sup>2+</sup>蛍光経時変化と電気活動を同時記録



後足カメラが後足刺激反射を撮影。  
USBビデオキャプチャで保存

# 実際の画像取得セットアップ

皮質を電気刺激しながら、神経細胞Ca<sup>2+</sup>蛍光経時変化を記録

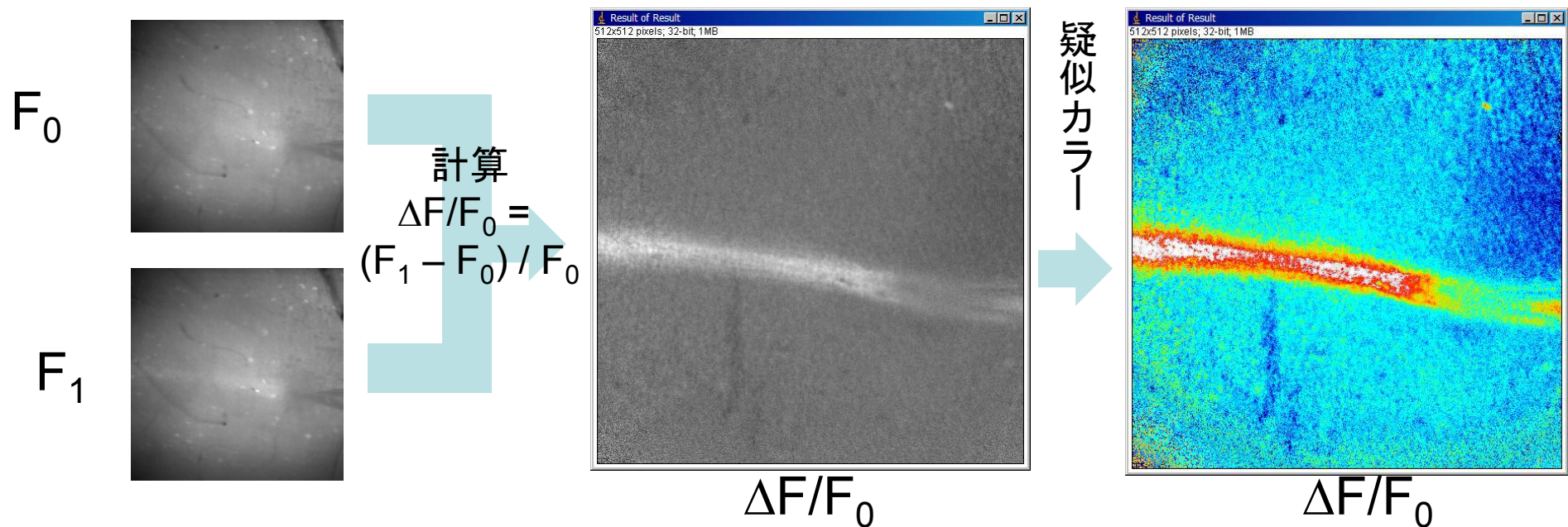


# チュートリアルの課題

刺激を与えると、細胞の蛍光が光った。  
刺激前と刺激後の細胞の画像データから  
蛍光変化率( $\Delta F/F_0$ )を計算して、疑似カラーで表示したい。

$$\Delta F/F_0 = (F_1 - F_0) / F_0$$

変化率 = (後 - 前) / 前





# ImageJ と MATLAB

ImageJ (前身はNIH Image)

無料

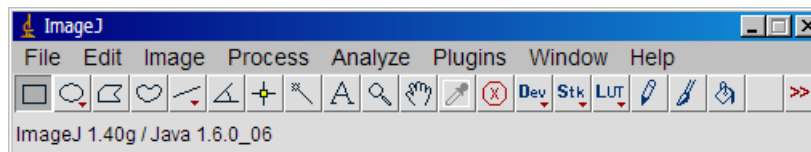
メニューから機能を選ぶ

誰かが公開した新機能をPlug-inで追加できる

操作はマウス主体で易しいが、複雑な計算や自動処理は苦手。

参考書:医学・ライフサイエンス画像解析テキスト

改訂第3版 小島清嗣・岡本洋一 羊土社



MATLAB

有料。毎年契約。大学・研究所単位で契約しているかも。

プログラムを書く。欲しい機能はだいたい用意されている。

画像解析にはオプションのImage Processing Toolbox必須。

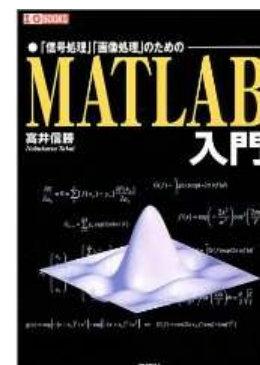
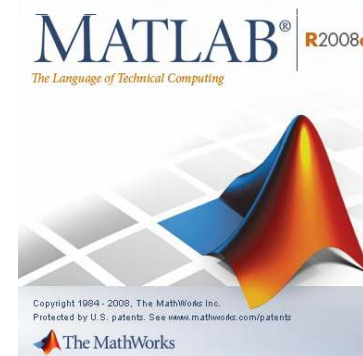
他にもSignal Processing Toolbox、Statistic Toolboxは欲しい。

OctaveやScilabは無料MATLAB互換ソフトだが、機能は限定・保証なし？

プログラムを書くのは難しいが、計算・自動処理・fig作成は得意。

参考書(初級): MATLAB入門「信号処理」「画像処理」のための  
高井信勝 I/O Books

参考書(上級): MATLABによる画像&映像信号処理  
村松正吾 CQ出版社



# ImageJ で画像ファイルの読込

ファイルをImageJメニューバーに  
ドラッグ & ドロップ

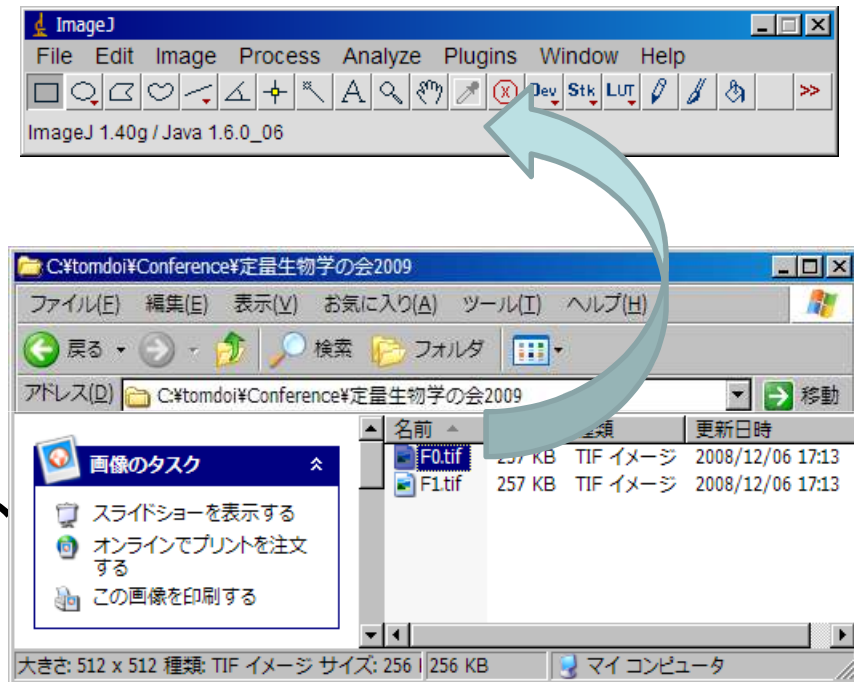
読込成功すると画像が勝手に  
表示される

生物系顕微鏡のファイル形式を網羅し  
ているので確実に読込できるはず

XYZやXYTなどの複数画像ファイルは、  
File – Import – Image Sequence...で  
一気に取込。

スタックで複数画像を1ファイルに  
まとめると操作・ファイル管理が楽。

画像が読めない場合は、顕微鏡ソフトで画像形式を替えて保存し直す。

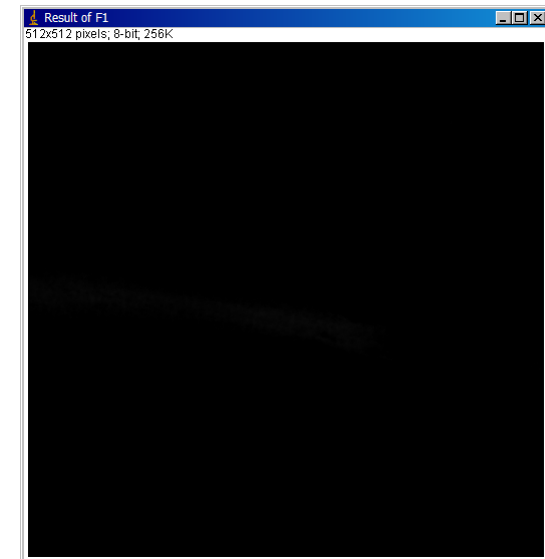
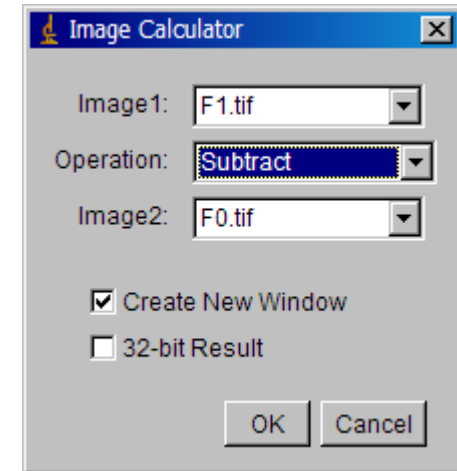


# 刺激後 — 刺激前 の引き算(uint8)

Process – Image Calculatorで、  
Image1: F1.tif  
Operation: Subtract  
Image2: F0.tif

$\Delta F = F_1 - F_0$ の値は小さいので、  
計算結果の画像は暗い。

計算結果は元画像と同じuint8型で、  
uint8型は負値を持たないので、  
 $F_1 \geq F_0$  のとき、 $F_1 - F_0$   
 $F_1 < F_0$  のとき、 0  
となってしまう。  
ちなみに、Operation: Differenceだと、  
 $|F_1 - F_0|$ のように差分の絶対値になる。



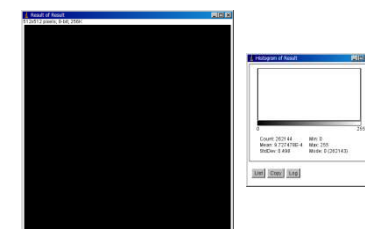
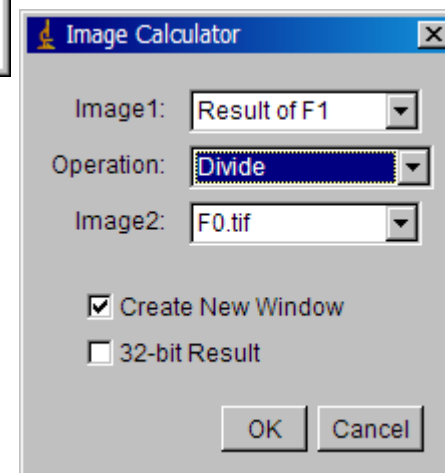
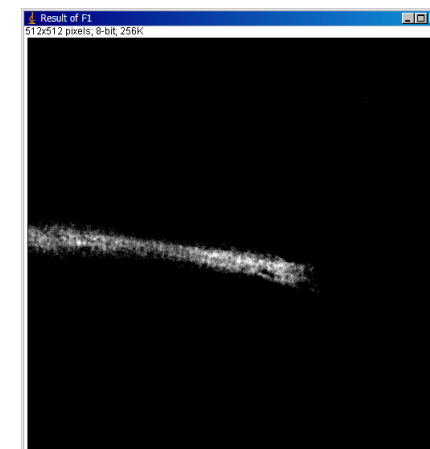
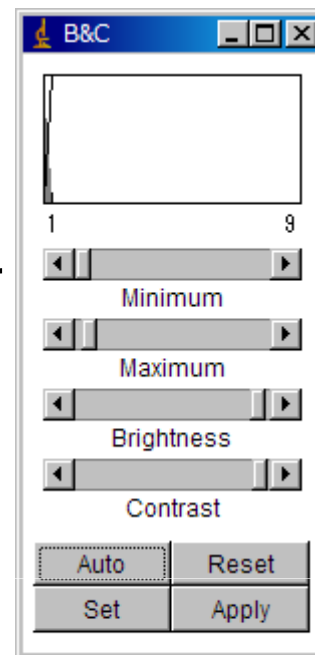
# 蛍光変化率( $\Delta F/F_0$ )の計算(uint8)

Image – Adjust – Brightness/Contrast...  
とりあえず、Auto – Applyを試してみる  
気に入らなければEdit – Undoで元に戻す

Process – Image Calculatorで、  
Image1: Result of F1  
Operation: Divide  
Image2: F0.tif

画像が真っ暗に！ データも0しか入ってない！  
どの点も蛍光変化率( $\Delta F/F_0$ ) < 100% = 1だったので、  
小数点以下の値が0に切り捨て扱いに。失敗。

uint8 = unsigned integer 8-bit  
符号なし 整数 8ビット  
(負値なし)(小数点切捨)( $2^8=256$ 階調)

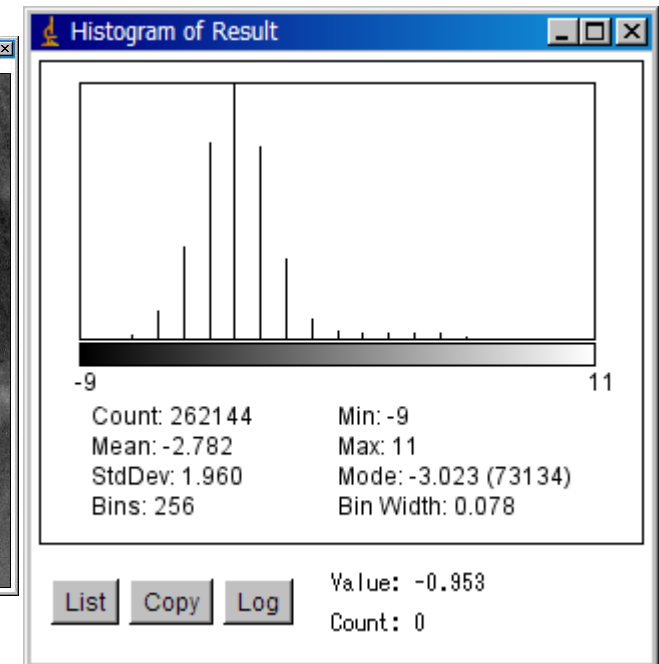
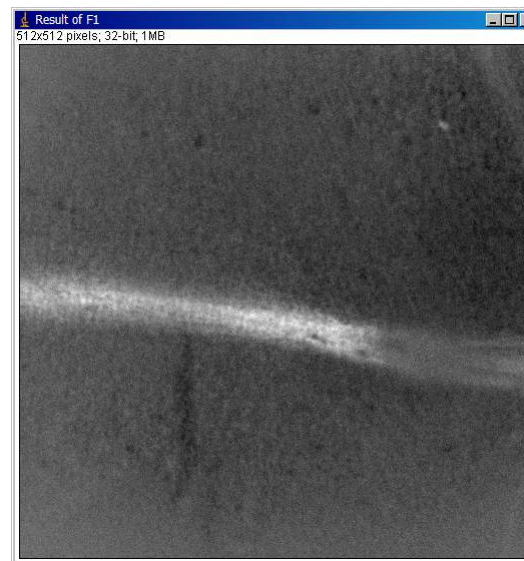
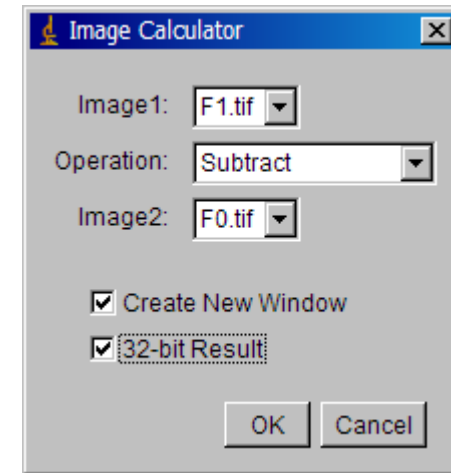


# 刺激後 — 刺激前 の引き算(32-bit)

Process – Image Calculatorで、  
Image1: F1.tif  
Operation: Subtract  
Image2: F0.tif

32-bit Resultにチェックをつける

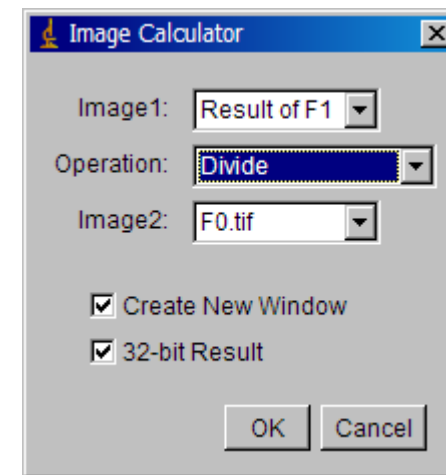
Analyze – Histogramで  
ちゃんと負値もある



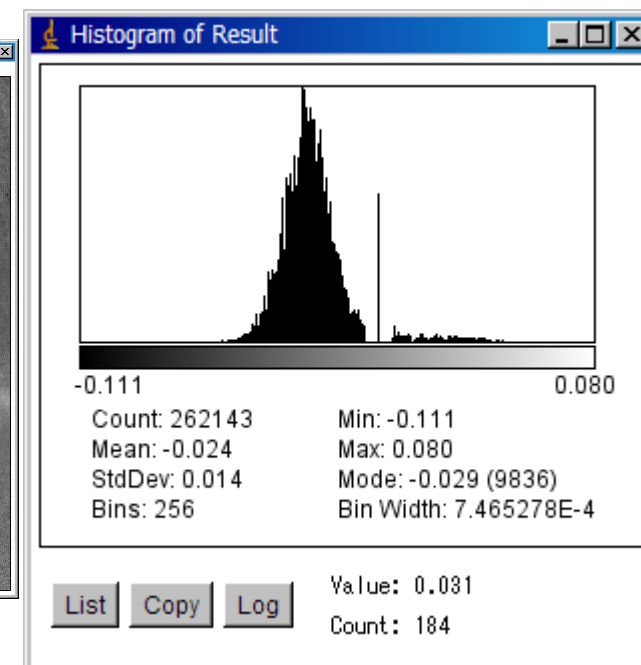
# 蛍光変化率( $\Delta F/F_0$ )の計算(32-bit)

Process – Image Calculatorで、  
Image1: Result of F1  
Operation: Divide  
Image2: F0.tif

32-bit Resultにチェックをつける



Analyze – Histogramで  
ちゃんと小数点以下もある

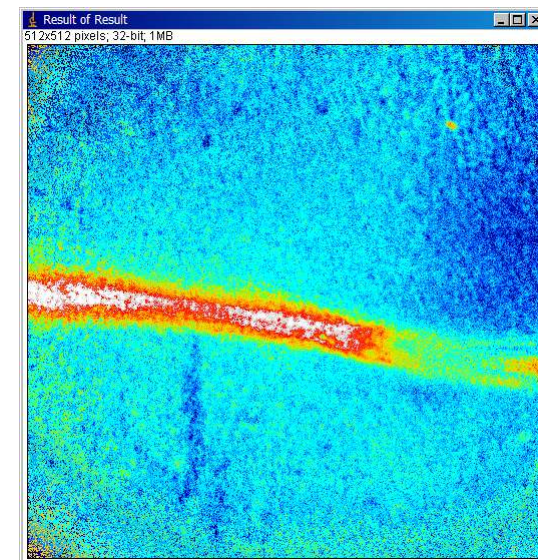
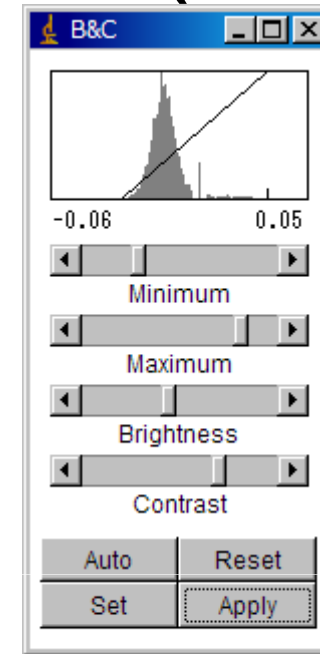
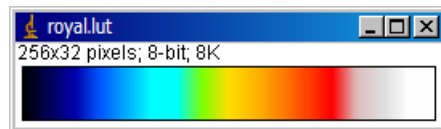


# コントラスト補正と疑似カラー(32-bit)

Image – Adjust – Brightness/Contrast...  
とりあえず、Auto – Applyを試してみる  
気に入らなければEdit – Undoで元に戻す

Image – Lookup Tables – royalで  
黒青水緑黄赤白の疑似カラー表示。  
このまま保存すると白黒で保存されるが、  
Image – Type – RGB Colorに変えると  
疑似カラーの色で保存される。

何も画像を読み込んでいない状態で  
Image – Lookup Tables – royalで  
黒青水緑黄赤白のカラーバー表示。



# MATLAB画像ファイルの読込

imread関数でファイルからMATLABへ画像を読み込む。

```
F0 = imread('D:\Ca2+Imaging\20090110\F0.tif');
```

```
F1 = imread('D:\Ca2+Imaging\20090110\F1.tif');
```

読み込んだ画像を表示させるには、imshowやimagescを使う。

```
imshow(F0) %スケーリングなし
```

```
imagesc(F0) %スケーリングあり
```

```
colormap(gray) %疑似カラーをデフォルトのjetからgreyに変更
```

読めない場合は、ImageJで保存し直すか、別形式に保存。

Tiffのスタック読込はtiffread3関数、動画読込はaviread関数を使う。8-bitのデータならavireadが便利。



# TIFFファイルは便利だが読込難

TIFFファイルには、顕微鏡の設定などの情報を書けるヘッダ一部分が存在する。

Olympus FV300の例→

取得日時、対物レンズ倍率、対物レンズ名、使用チャネル、フォトマル電圧、フォトマルゲイン、フォトマルオフセット、中間倍率、スキャン速度、スキャンモード、レーザーシャッター開閉、レーザーNDフィルタ%、、、

このヘッダ一部分が読込の邪魔になるので、ImageJで開いてから保存し直すとヘッダ一部分が取れる。

```
Dumper XP - T:\tomdoi\#2photon\#20081120\#Z5_20xNA095zoom10.tif
ファイル(E) 表示(V) ヘルプ(H)
終了 開く
Address +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F 0123456789ABCDEF
00000380 20 0D 0A 44 61 74 65 3D 31 31 2D 32 30 2D 32 30 Date=11-20-20
00000390 30 38 0D 0A 54 69 6D 65 3D 31 37 3A 31 30 3A 30 08 Time=17:10:0
000003A0 37 0D 0A 53 79 73 74 65 6D 20 43 6F 6E 66 69 67 7 System Config
000003B0 75 72 61 74 69 6F 6E 3D 46 56 33 30 30 0D 0A 4D uration=FV300 M
000003C0 61 67 6E 69 66 69 63 61 74 69 6F 6E 3D 32 30 58 gnification=20X
000003D0 0D 0A 4F 62 6A 65 63 74 69 76 65 20 4C 65 6E 73 Objective Lens
000003E0 3D 4C 55 4D 50 4C 46 4C 20 32 30 58 57 0D 0A 50 =LUMPLFL 20XW P
000003F0 4D 54 20 56 6F 6C 74 61 67 65 20 43 68 34 3D 35 MT Voltage Ch4=5
00000400 37 35 0D 0A 4F 66 66 73 65 74 20 43 68 34 3D 35 75 Offset Ch4=5
00000410 0D 0A 47 61 69 6E 20 43 68 34 3D 34 2E 30 30 30 Gain Ch4=4.000
00000420 30 30 30 0D 0A 50 4D 54 20 56 6F 6C 74 61 67 65 000 PMT Voltage
00000430 20 43 68 35 3D 36 30 30 0D 0A 4F 66 66 73 65 74 Ch5=600 Offset
00000440 20 43 68 35 3D 35 0D 0A 47 61 69 6E 20 43 68 35 Ch5=5 Gain Ch5
00000450 3D 34 2E 30 30 30 30 30 30 0D 0A 46 72 61 6D 65 =4.000000 Frame
00000460 20 46 69 6C 74 65 72 3D 33 20 66 72 61 6D 65 20 Filter=3 frame
00000470 4B 61 6C 6D 61 6E 20 46 69 6C 74 65 72 0D 0A 5A Kalman Filter Z
00000480 6F 6F 6D 20 53 69 7A 65 3D 31 30 2E 30 30 30 30 oom Size=10.0000
00000490 30 30 0D 0A 53 63 61 6E 20 53 70 65 65 64 3D 38 00 Scan Speed=8
000004A0 30 30 30 0D 0A 53 63 61 6E 4D 6F 64 65 3D 53 74 000 ScanMode=St
000004B0 61 6E 64 61 72 64 20 53 63 61 6E 0D 0A 53 65 63 andard Scan Sec
000004C0 6F 6E 64 73 50 65 72 53 63 61 6E 4C 69 6E 65 3D ondsPerScanLine=
000004D0 30 2E 30 30 35 30 34 38 0D 0A 44 65 6C 61 79 54 0.005048 DelayT
000004E0 6F 46 69 72 73 74 49 6D 61 67 65 50 69 78 65 6C ofFirstImagePixel
000004F0 49 6E 53 65 63 73 3D 30 2E 30 35 31 30 38 38 0D InSecs=0.051088
00000500 0A 4C 61 73 65 72 2D 53 68 75 74 74 65 72 20 41 Laser-Shutter A
00000510 72 3D 43 6C 6F 73 65 64 0D 0A 4C 61 73 65 72 2D r=Closed Laser-
00000520 53 68 75 74 74 65 72 20 4B 72 3D 43 6C 6F 73 65 Shutter Kr=Close
00000530 64 0D 0A 4C 61 73 65 72 2D 53 68 75 74 74 65 72 d Laser-Shutter
00000540 20 48 65 4E 65 2D 52 3D 43 6C 6F 73 65 64 0D 0A HeNe-R=Closed
00000550 4C 61 73 65 72 2D 53 68 75 74 74 65 72 20 49 52 Laser-Shutter IR
00000560 3D 4F 70 65 6E 0D 0A 4C 61 73 65 72 2D 4E 44 20 =Open Laser-ND
00000570 49 52 3D 34 30 25 20 74 72 61 6E 73 6D 69 74 74 IR=40% transmitt
1075654 Byte
```

# 蛍光変化率( $\Delta F/F_0$ )の計算・表示

```
F0double = double(F0); %double型に変換
F1double = double(F1); %double型に変換
DeltaFdouble = F1double - F0double;
DFF0double = DeltaFdouble ./ F0double;
imagesc(DFF0double); %スケーリングありで画像表示
colormap(jet); %カラーマップをjet青水緑黄赤茶に変更
colorbar %カラーバーの表示
```

補足(プログラミング知識がある人用):

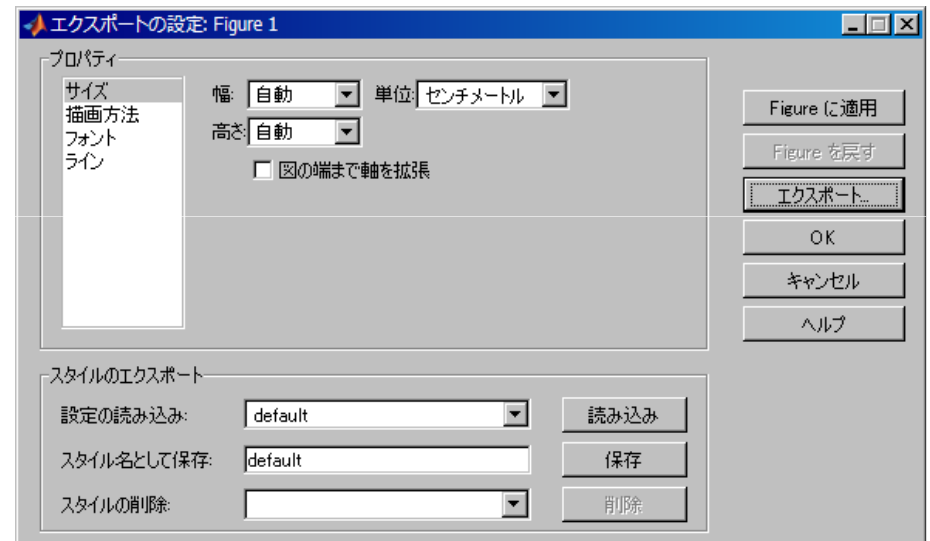
MATLABは行列演算が標準なので、行列A-行列Bの計算にforループを使う必要はない。また、MATLABでは、行列A./行列Bと書くと行列の各要素同士の除算をする。行列A/行列Bでは、行列A x(行列Bの逆行列)の意味になってしまう。これはMATLAB特有。

# 蛍光変化率( $\Delta F/F_0$ )の保存

`imwrite(DFF0double, 'DFF0.tiff', 'tiff', ... %次行に続く  
          'Compression', 'none');        %無圧縮でファイル保存`  
いままでのプログラムは.m形式で保存して使い回しできる

Figureウィンドウから、  
ファイル(F) – エクスポートの設定  
...(R) – エクスポート  
eps形式で保存すれば、Adobe  
Illustratorへの扱いがスムーズ

本チュートリアルでは触れないが、  
MATLABでFigureのタイトルや目盛  
数値などをプログラムで指定できる。  
論文のFigを何回も作るときに便利。



# 画像処理には高性能PCを！

Windows XP 64bit

(32bitでは1アプリケーションでメモリ2GBが上限)

Intel Core2Duo 3.16GHz

(クロック数が計算速度に比例。Quadは効果なし?)

8GB Memory

(読込ファイルの4倍必要、5倍で十分)





HDD 500GB + 1TB + 1TB + 1TB

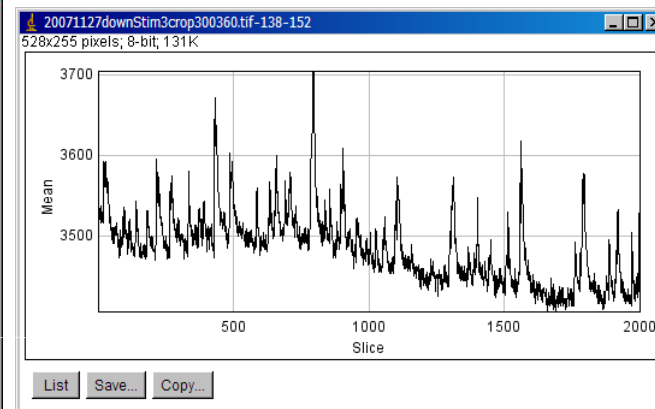
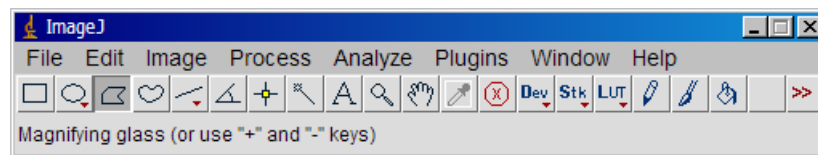
(1TB < ¥10000の時代なのでケチらない)

TeraStationPro 4TB 19インチラック型 RAID5 x 2

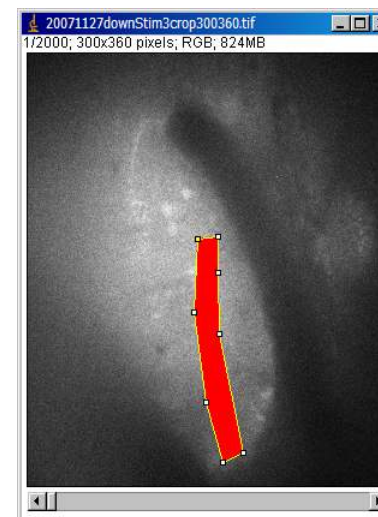
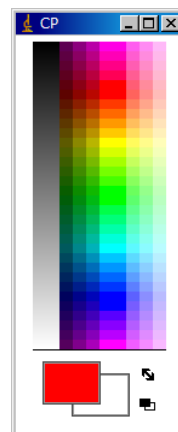
(データが消えると泣く。バックアップは必ず)

# おまけ1: ROIを囲って経時変化

XYT画像で、ROIを囲むには、  
ツールバーから、     
のどれかで囲って、  
Image – Stacks – Plot Z-axis  
Profile  
経時変化のグラフ数値はグラ  
フウィンドウのSaveから保存。



ただし、ROIの位置や形状は保  
存されないのので、Image –  
Color – Color Pickerで好きな  
色を選んで、Edit – FillでROIを  
塗りつぶして画像を保存する。



# おまけ2: ImageJで細胞を数える

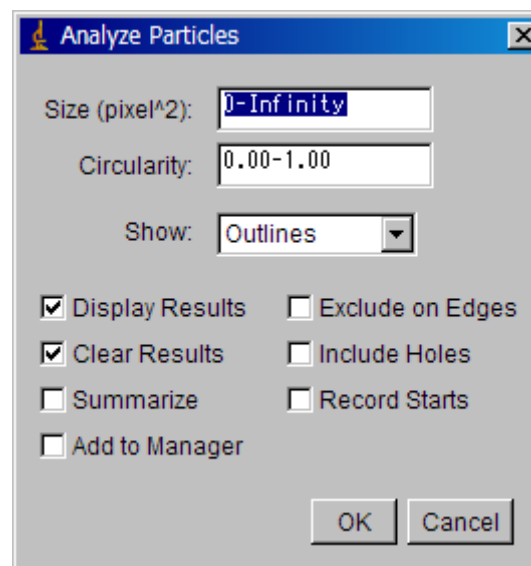
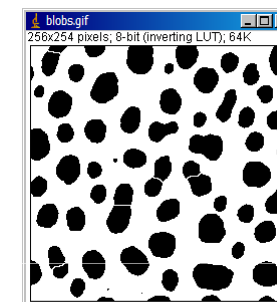
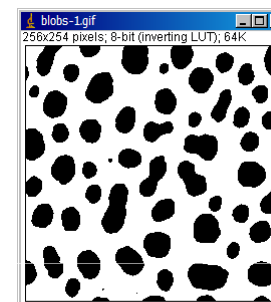
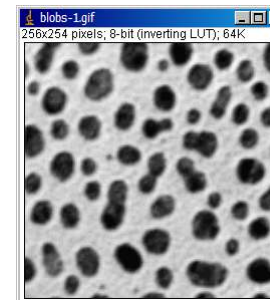
丸い細胞を数えるとき手軽に使える。

Process – Binary – Make Binaryで画像を二値化。結果に満足できなければ事前に画像フィルタを工夫。

Process – Binary – Watershedで密着した複数の細胞を分離して認識

Analyze – Analyze Particles...

で丸を数える。細胞の大きさや丸さを指定して余計なゴミを数えないようにする。



# おまけ3: ImageJでズレやブレを補正

StackReg: スライス連続切片を並べ揃える

<http://bigwww.epfl.ch/thevenaz/stackreg/>

からダウンロードしてPluginsフォルダに入れる  
動作にはTurboRegが必要

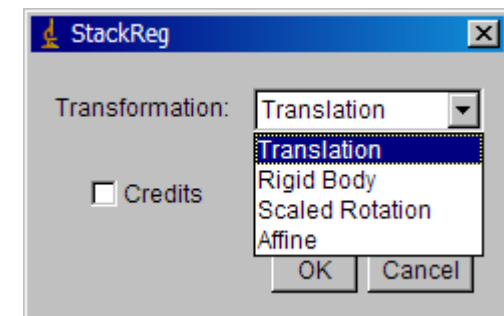
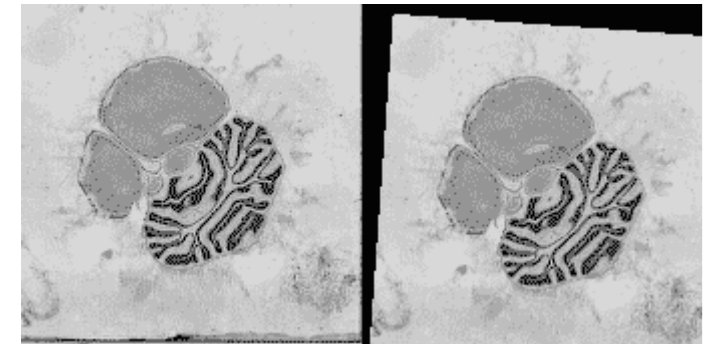
XYスタック画像を読み込んで、Plugin – StackReg

Translation (平行移動)

Rigid Body (平行移動と回転)

Scaled Rotation (平行移動と回転と拡大)

Affine (平行移動と回転と縦横不等拡大)



TurboReg: ブレ(揺れ)のある連続画像を補正する

<http://bigwww.epfl.ch/thevenaz/turboreg/> からダウンロード

XYスタック画像と基準を読み込んで、Plugin – TurboReg

# おまけ4: フィルタ処理で細胞自動認識

