Textbook for 1st Math-Clinic Bioimage Analysis Training Course

using Fiji and ilastik to solve:
2D spots counting,
2D+time tracking,
3D object-based colocalization

(Targeted for biologists with image analysis basics and starting image analysts)

Chong Zhang

Math-Clinic, CellNetworks

University of Heidelberg, Germany

2014

CONTENTS

Contents i

Preface and Acknowledgements iii

1 Fiji and ilastik 1
- 1.1 Aim . . . 4
- 1.2 Fiji . . . 4
  - 1.2.1 Installation and updates . . . 4
  - 1.2.2 Plugins . . . 4
- 1.3 ilastik: the interactive learning and segmentation toolkit . . . 5
  - 1.3.1 Installation and updates . . . 6
    - 1.3.1.1 Install CPLEX . . . 6
- 1.4 Resources . . . 9

2 Counting spots 11
- 2.1 Aim . . . 14
- 2.2 Introduction . . . 14
- 2.3 A Fiji solution . . . 14
  - 2.3.1 Detecting objects . . . 15
    - 2.3.1.1 Detecting and counting nucleus . . . 15
    - 2.3.1.2 Detecting lyososome spots . . . 15
  - 2.3.2 Counting spots . . . 16
    - 2.3.2.1 counting spots in nucleus . . . 17
    - 2.3.2.2 counting spots in regions where mitochondria also presents 17
  - 2.3.3 Convert the recorded commands into a macro script . . . 19
- 2.4 ilastik solution . . . 20
  - 2.4.1 ilastik Density Counting workflow . . . 20
  - 2.4.2 Training and predicting density . . . 20
  - 2.4.3 Counting objects/spots . . . 22
    - 2.4.3.1 Counting using `Box` in ilastik . . . 22
- 2.5 Counting in Fiji from ilastik predictions using arbitrary shapes . . . 25
- 2.6 Tips and tools . . . 27
- 2.7 Groups close by which work on similar problems . . . 27
- 2.8 References . . . 28

3 2D+time tracking **29**
3.1 Aim . . . 32
3.2 Introduction . . . 32
3.3 a Fiji solution . . . 32
3.3.1 Other possibilities to solve tracking problem . . . 34
3.4 ilastik solution . . . 34
3.4.1 Segmentation with ilastik Pixel Classification workflow . . . 34
3.4.2 ilastik Automatic Tracking workflow . . . 35
3.4.2.1 Object extraction . . . 35
3.4.2.2 Tracking . . . 37
3.5 Tracking analysis in Fiji: plot trajectory, displacement, or velocity . . . 37
3.5.1 Plot trajectories on image . . . 40
3.5.2 Plot displacements or velocity . . . 41
3.6 Tips and tools . . . 42
3.7 Groups close by which work on similar problems . . . 43
3.8 References . . . 43

4 3D Object based colocalization **45**
4.1 Aim . . . 48
4.2 Introduction . . . 48
4.3 Segmenting spots using ilastik Pixel Classification workflow . . . 49
4.4 Writing our own ImageJ macro for fully automatic colocalization . . . 50
4.4.1 Filtering objects by size . . . 51
4.4.1.1 Filtering by 3D sizes . . . 51
4.4.1.2 Filtering by 2D sizes . . . 52
4.4.2 Finding spatial overlapping objects in both channels . . . 53
4.4.3 Filtering the colocalization objects by volume overlap percentage criteria . . . 56
4.4.4 Visualizing results . . . 57
4.4.5 Testing the macro on Hela cells with viral replication . . . 58
4.4.6 Setting up a parameter interface . . . 59
4.5 Tips and tools . . . 59
4.6 Groups close by which work on similar problems . . . 60
4.7 References . . . 60

5 Related resources **61**

PREFACE AND ACKNOWLEDGEMENTS

In CellNetworks and other institutes in Heidelberg, the growing demands on bioimage analysis often make a one-to-one analysis support difficult and less efficient. Bioimage analysis training course is a way of offering solutions and ideas on how to use and combine various tools to tackle common and practical problems, in order to narrow the gap between software packages and users' demands by increasing the image analysis literacy of users ...

This course aims at introducing some insights to different open source software tools, and providing an impression of possible solutions for the same biological question using these tools and their applicabilities. Specifically, **Fiji** and **ilastik** are used in this course, the former being a widely used powerful tool and the latter an in-house interactive tool using machine learning algorithms. Three example problems which are rather simple but commonly seen in biological applications are presented, together with a few common analysis and measurements. These exercises could be applied as proof of concept or fast quality check for real problem. Please note that more thorough and accurate analysis tasks often require more specific methods that are likely to involve more complex algorithms and techniques.

The building of this textbook should be a continuous improving process and its contents will hopefully grow as well. Current version is the first attempt therefore you may find some errors or missing part. I apology for this potential inconvenience for you and am grateful to have your comments and help in improving it.

This course is funded by CellNetworks. The instructors are: Dr. Kota Miura, Dr. Jürgen Reymann, Dr. Anna Kreshuk, Dr. Burcin Erocal, Thorben Kröger, and myself. I would like to thank Prof. Fred A. Hamprecht, Prof. Ulrich Schwarz, Prof. Thomas Höfer, Prof. Michael Knop, Prof. Rob Russel for their support and suggestions on initializing this training course, which hopefully will become a regular event with continuous efforts. I would also like to thank Dr. Ulrike Engel, Dr. Holger Erfle for sharing their experiences and practical suggestions, Dr. Peter Bankhead for reviewing the textbook, and Dr. May-Britt Becker, Christine Hermann for their help in administration arrangements. Special acknowledgement goes to EuBIAS Bioimage Data Analysis Course (http://eubias2013.irbbarcelona.org/bias-2-course), which boosted this workshop ...

1

FIJI AND ILASTIK

A problem well put is half solved.

John Dewey

Part of ilastik general description was adapted from ilastik online documentation.

1.1 Aim

The aim of this session is to familiarise you with ImageJ/Fiji and ilastik. This introduction session will be followed by three hands-on practical sessions on how to use these two tools for analysis of microscopy data and we will have a closer look at different Fiji functions and plugins, and several ilastik workflows.

1.2 Fiji

Fiji stands for “Fiji Is Just ImageJ". ImageJ is a public domain image processing and analysis program written in Java. It is freely available (`http://imagej.nih.gov/ij/index.html`) and used by many scientists all over the world. There is a very active ImageJ community and ImageJ is continuously improved. Fiji is a distribution of ImageJ together with Java, Java 3D and a lot of plugins organized into a coherent menu structure.

During this session we will have an introduction on the use of ImageJ/Fiji for digital image processing for life sciences, and also on how to write macros to analyze the data and how to reuse code. So during the practical sessions you will write code using the macro language but programming knowledge is not strictly required to follow the course.

Please refer to the homepage of Fiji for detailed documentation and user manual. Additional handouts for the material used in this course will be distributed during the session.

1.2.1 Installation and updates

After downloading Fiji installer, please follow the instructions from the Fiji page (`http://fiji.sc/Installation`) to install Fiji on your computer.

Fiji has an automatic update system invoked each time when you start Fiji, if there are newly available updates. You can also configure your own updates by calling [`Help -> Update Fiji`], and after checking for details, an `ImageJ Updater` window will appear with the list of items to be updated or installed (Fig. 1.1 *top*). If you would like to exclude/include specific items to be checked for update, or if you would like to add new sites that are not listed, you could click `Manage Update Site` to configure them (Fig. 1.1 *bottom*).

1.2.2 Plugins

Apart from the official plugins, there are many amazing plugins from a large community contribution. Normally you need to download a specific plugin from their website, and then copy it into the Fiji plugins folder. For example, for one plugin we will be using in this course, 3D ImageJ Suite (a package of multiple plugins): we can install it by downloading the bundle from `http://imagejdocu.tudor.lu/lib/exe/fetch.php?media=plugin:stacks:3d_ij_suite:mcib3d-suite.zip` and unzipping it in the plugins folder. And when we restart Fiji, the plugin should be found at [`Plugins -> 3D`].

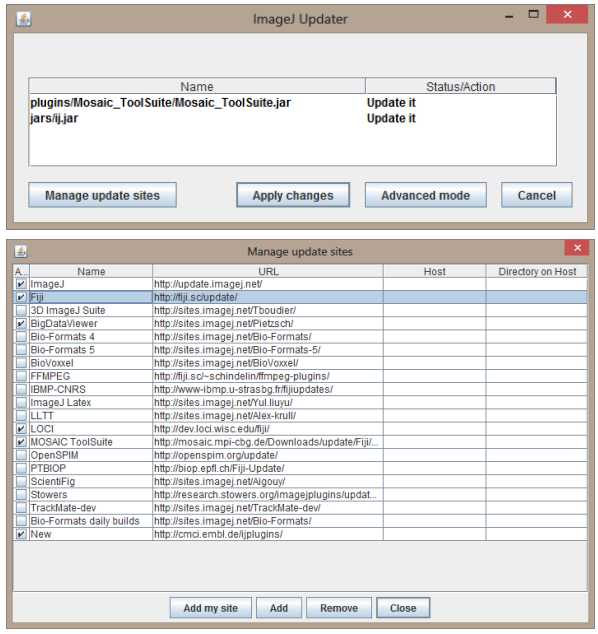


FIGURE 1.1: Fiji Updater.

1.3 ilastik: the interactive learning and segmentation toolkit

ilastik is a simple, user-friendly tool for image classification and segmentation in multiple dimensions. Using it requires no experience in image processing. ilastik has a convenient mouse interface for labeling an arbitrary number of classes in the images. These labels, along with a set of image features, are then used to for a machine learning based method to classifier image regions in different classes. In the interactive training mode, ilastik provides real-time feedback of the current classifier predictions and thus allows for targeted training and overall reduced labeling time. Once the classifier has been trained on a representative subset of the data, it can be used to automatically process a very large number of datasets. The plug-in functionality allows advanced users to add their own problem-specific features, apart from the provided set of features based on color, edges, and textures in the image.

During this session we will have an introduction on the use of ilastik workflows for digital image processing for life sciences. So during the practical sessions you will get familiar with interactively training the computer to process datasets according to your likes with just a few labels.

1.3.1 Installation and updates

You can download the ilastik installers from here: www.ilastik.org/sfn. If you have problems with installation, we will install it during an installation session together. Additional libraries will be needed to be installed for some workflows, such as the tracking workflow we will be using in Session 3.

1.3.1.1 Install CPLEX

Please note that ilastik will run even without IBM CPLEX installed, it is only a requirement for some workflows, e.g. the Counting and the Automatic Tracking workflows we will use. Also, it is not sufficient to download the Trial version of CPLEX since its solver can only handle very small problem sizes. Please make sure, the correct version is downloaded as described here.

IBM CPLEX is a commercial solver which is **free** for academic use. To apply for an ***academic license***, the user first needs to apply for an academic membership at IBM. Details may be found on the **IBM Academic Initiative website** (http://www-03.ibm.com/ibm/university/academic/pub/page/membership). Please note that it might take some days until the application gets approved by IBM. After the academic membership has been approved, the user can download IBM CPLEX. To do so, the steps on this IBM website may be followed. An installation instruction can be found on the ilastik website as well. *Please make sure to download the version with which current ilastik works.*

Windows There are typically no further modifications needed after installing CPLEX by double clicking the “cplex_studio1251.win-x86-64.exe". After a successful installation, the **Automatic Tracking Workflow** is displayed on the Start-Screen of ilastik.

However, if this workflow is not present, something went wrong with the CPLEX installation. First, please double check whether you indeed installed the correct version that your current ilastik uses. If this is the case, as a workaround, you may copy the files **libcplex.dll**, **libilocplex.dll**, and **libconcert.dll** from the CPLEX installation directory into the library folder of ilastik, e.g. C:/ProgramFiles/ilastik/lib.

Linux To install CPLEX, go to the folder where you have “cplex_studio1251.linux-x86-64.bin", then install it with the command line:

```
1 sh cplex_studio1251.linux-x86-64.bin
```

:

Unfortunately, CPLEX packages do not provide shared versions of all required libraries, but only static variants. Some manual work needs to be done here. You have to navigate to the installation folder of CPLEX and therein to the static libraries **libcplex.a**, **libilocplex.a**, and **libconcert.a** (usually located in ILOG/CPLEX_Studio125/cplex/lib/x86-64_sles10_4.1/static_pic and ILOG/CPLEX_Studi

`o125/concert/lib/x86-64_sles10_4.1/static_pic`) to run the following commands to link shared CPLEX libraries from the static versions:

```
g++ -fpic -shared -Wl,-whole-archive libcplex.a -Wl,-no-whole-archive -o
    libcplex.so
g++ -fpic -shared -Wl,-whole-archive libilocplex.a -Wl,-no-whole-archive -o
    libilocplex.so
g++ -fpic -shared -Wl,-whole-archive libconcert.a -Wl,-no-whole-archive -o
    libconcert.so
```

:

Finally, these shared libraries need to be copied into the library folder of the ilastik installation, e.g. `ilastik/lib`.

Mac OSX To install CPLEX, open a **Terminal**, then go to the folder where you have stored "cplex_studio1251.osx.bin", then install it with the command line:

```
bash cplex_studio1251.osx.bin
```

:

Similar to Linux, CPLEX packages for Mac do not provide shared versions of all required libraries, but only static variants. Unfortunately this is the platform that may need more work, because there are different ways to create shared libraries for CPLEX on Mac OSX depending on the operating system version and compiler type. In all cases, you need a compiler installed. You can check whether you have already a compiler installed by running the following command in a terminal :

```
gcc --version
```

:

If no compiler is installed, choose what to do depending on your OSX version:

- For all **OSX < 10.9, so up to Mountain Lion**, this means that you need to install **XCode** from the Apple Store. Then you need to go to XCode's [`Preferences -> Downloads`] tab, and install the **command line tools**.
- For **OSX 10.9 Mavericks** it suffices to install the **command line tools** using the following command lines without installing XCode, but you need to accept the XCode license when running the second line:

```
  xcode-select --install
  sudo gcc

```

:

Now we have the preliminaries to create the shared libraries. This depends on the compiler type and operating system. To find your compiler version, run

```
gcc --version
```

:

and look at the output. This should be either **GCC 4.2.1**, or **Apple LLVM (clang)**. The folders used to specify the location of the libraries in cases involving **clang** depend on whether you installed from “cplex_studio1251.macos.bin" or “cplex_studio1251.osx.bin". So choose the appropriate commands below to create the shared libraries for your laptop configuration. Note that if you installed CPLEX from “cplex_studio1251.macos.bin" , you need to replace “x86-64_osx" by “x86-64_darwin".

- If you have **GCC 4.2.1** and **OSX version < 10.9**

```
cd ~/Applications/IBM/ILOG/CPLEX_Studio1251/concert/lib/x86-64_osx/
  static_pic
g++ -fpic -shared -Wl,-all_load libconcert.a -Wl,-noall_load -o
  libconcert.dylib

cd ~/Applications/IBM/ILOG/CPLEX_Studio1251/cplex/lib/x86-64_osx/
  static_pic
g++ -fpic -shared -Wl,-all_load libcplex.a -Wl,-noall_load -o
  libcplex.dylib

g++ -fpic -shared -Wl,-all_load libilocplex.a -Wl,-noall_load -o
  libilocplex.dylib
```

:

- If you have **Clang** and **OSX version < 10.9**, then you need to supply the symbols of `cplex` and `concert` when building `libilocplex.dylib`

```
cd ~/Applications/IBM/ILOG/CPLEX_Studio1251/concert/lib/x86-64_osx/
  static_pic
g++ -fpic -shared -Wl,-all_load libconcert.a -Wl,-noall_load -o
  libconcert.dylib

cd ~/Applications/IBM/ILOG/CPLEX_Studio1251/cplex/lib/x86-64_osx/
  static_pic
g++ -fpic -shared -Wl,-all_load libcplex.a -Wl,-noall_load -o
  libcplex.dylib

g++ -fpic -shared -Wl,-all_load libilocplex.a -Wl,-noall_load -L
  .:../../../../concert/lib/x86-64_osx/static_pic -lcplex -lconcert -
  o libilocplex.dylib
```

:

- If you have **Clang** and **OSX version 10.9 Mavericks**, you need toto specify the correct C++ standard library to use in addition to the commands above

```
cd ~/Applications/IBM/ILOG/CPLEX_Studio1251/concert/lib/x86-64_osx/
  static_pic
g++ -fpic -shared -Wl,-all_load libconcert.a -Wl,-noall_load -stdlib=
  libstdc++ -o libconcert.dylib

cd ~/Applications/IBM/ILOG/CPLEX_Studio1251/cplex/lib/x86-64_osx/
  static_pic
g++ -fpic -shared -Wl,-all_load libcplex.a -Wl,-noall_load -stdlib=
  libstdc++ -o libcplex.dylib

g++ -fpic -shared -Wl,-all_load libilocplex.a -Wl,-noall_load -L. -L
  ../../../../concert/lib/x86-64_osx/static_pic -lcplex -lconcert -
  stdlib=libstdc++ -o libilocplex.dylib
```

:

Now copy those files to your ilastik installation into the lib folder. On Mac this works by right-clicking on *ilastik.app*, choose `Show package contents`, and place the three libraries in *Contents/Frameworks*. Some last steps are required. Please copy and paste the following lines in Terminal:

```
cd /Applications/ilastik.app/Contents/Frameworks
install_name_tool -id @executable_path/../Frameworks/libcplex.dylib libcplex
    .dylib
install_name_tool -id @executable_path/../Frameworks/libconcert.dylib
    libconcert.dylib
install_name_tool -id @executable_path/../Frameworks/libilocplex.dylib
    libilocplex.dylib
install_name_tool -change libcplex.dylib @executable_path/../Frameworks/
    libcplex.dylib libilocplex.dylib
install_name_tool -change libconcert.dylib @executable_path/../Frameworks/
    libconcert.dylib libilocplex.dylib
```

:

Finally, these shared libraries need to be copied into the library folder of the ilastik installation, e.g. `ilastik/lib`.

1.4 Resources

Fiji related:

- CMCI ImageJ Course page: http://cmci.embl.de/documents/ijcourses
- CMCI Macro programming in ImageJ Textbook: https://github.com/cmci/ij_textbook2/blob/master/CMCImacrocourse.pdf?raw=true
- Fluorescence image analysis introduction:http://blogs.qub.ac.uk/ccbg/fluorescence-image-analysis-intro/
- mailing list: imagej@list.nih.gov

ilastik related:

- ilastik homepage: http://ilastik.org/sfn
- mailing list: ilastik-user@ilastik.org

Bioimage related:

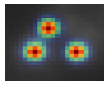
- BioImage Information Index: http://biii.info/

2

COUNTING SPOTS

Not everything that counts can be counted, and not everything that can be counted counts.

William Bruce Cameron



Part of ilastik general description was adapted from ilastik online documentation.

2.1 Aim

In this session, we will count spots in image regions, with two different methods using Fiji and ilastik.

2.2 Introduction

In microscopy images, counting the number of objects is a rather common practice. When the density of objects in the image is low and the objects are well separated from each other, it is possible to count objects by first segmenting the objects and then perform a morphological operation called connected components analysis. However, as the density of the objects increases, such approach underestimates the number of objects due to under-segmentation of clustered objects.

Dataset In this session, we will use the example image, Hela cells, in the Session2 folder of the course material. You can also find it directly from Fiji sample data. This is a 16-bits/channel composite color image of Hela cells with red lysosomes, green mitochondria and blue nucleus. We will count the number of lysosomes in the whole image, find their distributions in e.g. cell nucleus or user-defined regions of arbitrary shape.

2.3 A Fiji solution

Before doing anything with Fiji, let's first turn on the later-to-be your favorite function ***ImageJ command recorder*** *(`Plugins -> Macros -> Record`). It magically records operation you will be doing with Fiji, which can be turned into a macro script either directly or with some modifications.*

If you would like to get the image directly from Fiji, then click on [`File->OpenSamples->HelaCells`]. Since we are interested in counting red lysosomes spots (Fig. 2.1), lets first split the channels using [`Image -> Color -> Split Channels`]. Or, just open the `C1-hela-cells.tif` from the session folder.

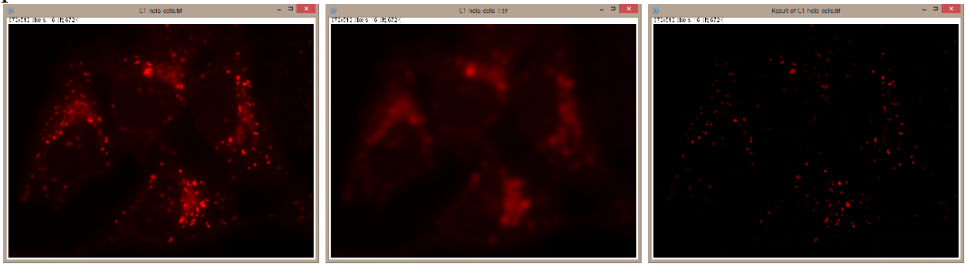


FIGURE 2.1: (*left*) The lysosome channel of the Hela cells image from Fiji sample data. (*middle*) estimated background using Gaussian smoothing. (*right*) The original image after subtracting the background estimate.

2.3.1 Detecting objects

2.3.1.1 Detecting and counting nucleus

Let's first select the nucleus channel image `C3-hela-cells.tif` (Fig. 2.2), since we are interested in the lysosomes inside the cell nucleus. This is a relatively "clean" image, to detect the nucleus we can try just thresholding and play with different methods and choose the best one, e.g. `Triangle`. Notice that the thresholded binary image is most likely not exactly what we want, but with small objects and possibly holes (Fig. 2.3). In such situations, a *morphological opening* operation can be used to clean up small objects. And for filling holes, a *morphological closing* operation can do the job. In this case, we need *morphological opening*, which is completed in two steps: a [`Process -> Filters -> Minimum`] followed by a [`Process -> Filters -> Maximum`]. In order to bring the object shape back to its original size, the `Radius` value for both filters should take the same value, e.g. 5.0 for this image. We can rename the image as "nucleus_mask".
Question: How the operations of *morphological closing* can be done?

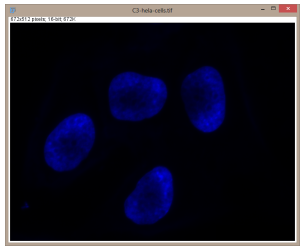


FIGURE 2.2: Nucleus channel

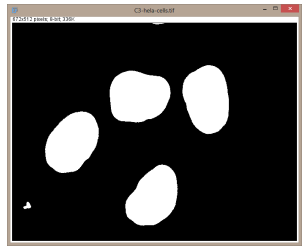


FIGURE 2.3: Thresholded

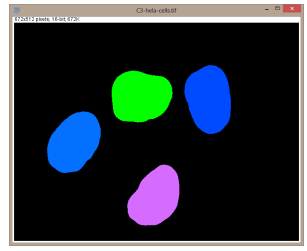


FIGURE 2.4: Cleaned mask

Now we can count the nucleus using [`Analyze -> Analyze Particles`], even though it is pretty easy to check by eye that there are 4 nucleus. We will set the options `Size`, `Circularity`, `Show` to: `0-Infinity`, `0-1`, and `Count Masks`, respectively, and verify that `Add to Manager` is unchecked while `In situ Show` is checked. Then we can click `OK`. In order to visualize the counted objects better, we could change the lookup table (LUT) option, using [`Image -> Lookup Tables`] options. If you like, we can also use the LUT file provided in your Session2 folder, "Random.lut". It should be copied to the ImageJ LUT folder. Once copied you should be able to find it in the `Lookup Tables` list and apply it (see Fig. 2.4). This is because when the option `Count Masks` is chosen, the mask image will result in an image with a different pixel value for each object. If we toggle the mouse cursor in the object regions, we will find their values, or object label or ID, displayed in the main Fiji menu. In this case, they should be 1-4 (and with zero background).

2.3.1.2 Detecting lysosome spots

In the image `C1-hela-cells.tif`, the blob-like highly-contrasted lysosomes spots exist in some cloud-shape background. So the first step is to estimate and subtract the

background. Since we need to subtract images, let's first make a copy of the image by running [Image -> Duplicate]. A typical approximation of such background is a much smoothed version of the image. So we could try the following choices and see which suits the best:

- run a *Gaussian smoothing* ([Process -> Filters -> Gaussian Blur]) with a large Sigma value (e.g. 6.00) on the duplicated image.

- apply a *morphological opening* to the image. This operation is done through a Minimum filter ([Process -> Filters -> Minimum]) followed by a Maximum filter ([Process -> Filters -> Maximum]) with the same Radius (e.g. 6.00). These filters have a Preview mode, which is helpful for choosing the satisfactory parameter(s) for the filters.

Once having a good approximation of the background, we can subtract it from the original image using [Process -> Image Calculator] with Subtract Operation. Alternatively, instead of first estimating and then subtracting background from the original image, we can also run [Process -> Filters -> Unsharp Mask] with a small Sigma value (e.g. 1.00) and large Mask Weight value (e.g. 0.9) directly. And we should be able to remove quite some cloud-like background signal and enhance signals of lysosomes.

Next, we will extract objects from the resultant "background-free" image. A first try can always be getting the binary mask after a thresholding. For example, we can try [Image -> Adjust -> Threshold] with the Otsu method or other ones. And then click Apply after we are satisfied with the thresholding to get a binary mask image. We could see that from this image (Fig. 2.5 *left*), some spatially close spots are connected and thus regarded as one object. So we would need to correct this undersegmentation behavior. One common solution is to run [Process -> Binary -> Watershed]. As we see in Fig. 2.5 *middle*, it separates some clustered spots (indicated in red circles) by a line with one-pixel width. It should be noted that the *watershed* method does not always do the magic, and it works better for objects with more regular shapes and sizes. After we managed to create a mask with one separate object for each spot, we could start identifying each object. First, let's run [Analyze -> Analyze Particles], After verifying whether Size, Circularity, Show options are set to: 0-Infinity, 0-1, and Nothing, respectively, and also Add to Manager is checked, then we can click OK. A new window similar to Fig. 2.5 *right* should appear. The numbers indicate the object (or ROI) ID. Let's rename this image as e.g. "spots" using [Image -> Rename]. And in the ROI Manager window we can find the list of all objects.

2.3.2 Counting spots

We will practice counting the spots that we just extracted from the lysosomes channel in two types of regions: nucleus and regions enclosed by mitochondria.

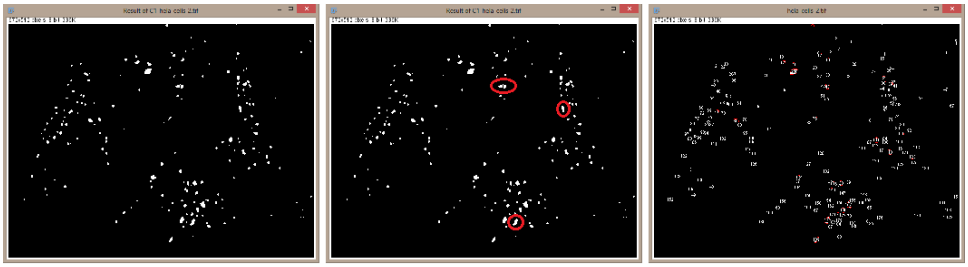


FIGURE 2.5: (*left*) The binary mask obtained from thresholding the background subtracted image. (*middle*) The binary mask after applying a watershed operation so as to separate obvious connected spots. (*right*) The labeled objects in the binary mask.

2.3.2.1 counting spots in nucleus

The trick to achieve this is to **apply the lysosome ROIs to the counted nucleus mask image, and then check the intensity of each ROI in this mask image, since the background and the four nucleus have different IDs.** So let's first go to [`Analyze -> Set Measurements`] and check only the `Min & max gray value` and uncheck the rest. Because we are interested in the maximum intensity value in each ROI.
Question: Why are we interested in the maximum intensity value in each ROI?
We need to select the "nucleus_mask" window, and highlight all the listed lysosome ROIs in the `ROI Manager` window. Then apply [`Image -> Overlay -> From ROI Manager`]. Your nucleus mask image should look like Fig. 2.6.

Then we can measure the parameters we just set, by clicking `Measure` in the `ROI Manager` window. The `Results` window is shown with three columns: object/ROI ID, minimum intensity and maximum intensity. Now if we click [`Results -> Distribution`] in the same `Results` window, and select `Max` as `Parameter`, and set `specify bins` and `range` to 5 and 0-5, respectively, a `Max Distribution` window should appear (see Fig. 2.7) with five bins/bars, whose heights indicate the number of ROIs with their maximum intensity value between the range of each histogram bin. *As we know that the mask has intensity range 0-4, if the histogram bins are chosen such that each bin represents the intensity range for each nucleus and the background (i.e. 1 intensity level per histogram bin in this case), the histogram does the counting for us.* That's the reason for specified values for bins and range.

In order to see the exact number of each counts, we can click `List` in the histogram window, and a two-column table similar to Fig. 2.7 will appear with the intensity `value` column indicating the nucleus ID, and the `count` column showing the number of ROIs.

2.3.2.2 counting spots in regions where mitochondria also presents

Similarly, if we are interested in counting spots in the regions covered by mitochondria, we can perform similar steps but on the mitochondria channel (`C2-hela-cells.tif`).

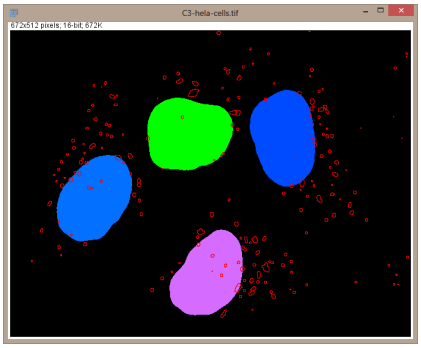


FIGURE 2.6: Nucleus mask & spot ROIs.

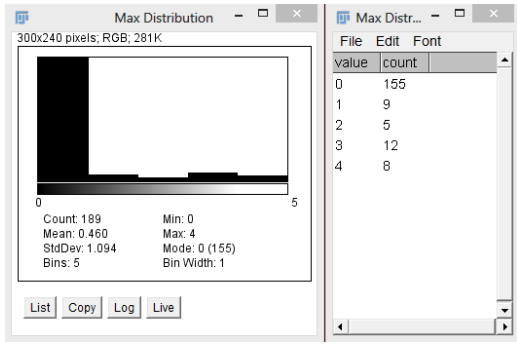


FIGURE 2.7: Histogram & counts in nucleus.

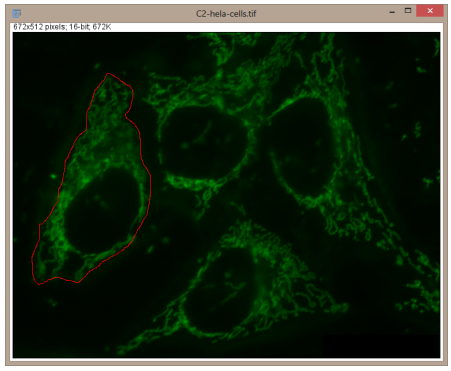


FIGURE 2.8: The mitochondria channel with manual selected region of interest.

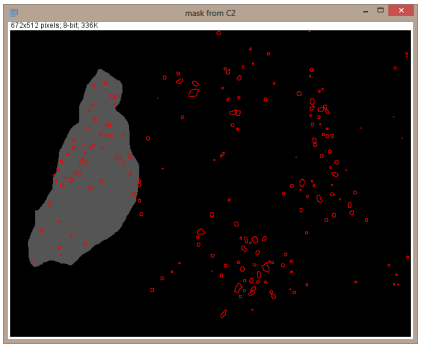


FIGURE 2.9: Nucleus mask & spot ROIs.

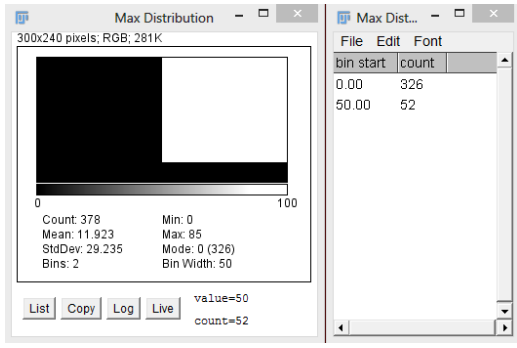
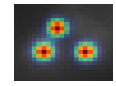


FIGURE 2.10: Histogram & counts in mitochondria.



Instead of using some filters to extract such regions, we will practice with manually drawing a contour of the region of interest using the `Freehand selections` tool (see Fig. 2.8). In previous steps, the counting was done through computing histogram from the labeled mask image. Similarly, a mask image is needed here. So we will create a new mask image from the manual selection. There are many options to do so, and a few of them are listed below:

- A new image of the same size is created using [`File -> New -> Image`] and renamed as “mito_mask". It will create a black image, and when we run [`Edit->Selection->RestoreSelection`], the manually drawn region contour on the mitochondria channel will be “translated" to the new image. Then the pixel values inside this region should be modified to distinguish it from the dark background, using:
  - the command [`Edit -> Fill`]. The filled region will have a value higher than the zero background, e.g. 85 (see this image in Fig. 2.9).
  - the command [`Process -> Math -> Add`], and we can decide which value to add to the new image for the selected region.
- After manually drawn region, we run [`Edit->Selection->CreateMask`], a new binary image with the same dimension is created, with the manually drawn region highlighted (i.e. 255) and the rest being 0.

The next steps are similar to what we have done in the previous section thus please refer to it for detailed descriptions. One example result is shown in Fig. 2.10. Please note that since there are only two values in the image (0 and 85), then we could have many histogram bin distribution options, as long as 0 and 85 are in separate bins.

In case of multiple regions to be measured, we could hold the “shift" key while drawing multiple regions, and then run [`Edit->Selection->CreateMask`]. Or use [`Process -> Math -> Add`] each time after drawing one region. If different values are added for each region, then a labeled mask image is created directly.

2.3.3 Convert the recorded commands into a macro script

We will clean up the commands recorded from the beginning so as it becomes a reusable macro script for other datasets. Actually if it makes things easier, you could clean it parts by parts throughout the recording period, which we will be doing so during the session. Cleaning up mostly involves: deleting mistaken commands, specifying selections like selected windows or images, replacing certain parameter settings by variables with values specified once manually or through programming. We will gradually practice this during the whole course.

The macro code obtained and cleaned up from the ImageJ command recorder (`Plugins -> Macros -> Record`) is provided in the Session2 folder.

2.4 ilastik solution

2.4.1 ilastik Density Counting workflow

We will use the **Density Counting** workflow to solve our problem. The purpose of this workflow is to enable counting the number of objects in crowded scenes such as cells (or in this case, spots) in microscopy images. **Counting is performed by directly estimating the density of objects in the image without performing segmentation or object detection.** This workflow offers a supervised learning strategy to object counting that is robust to overlapping objects. It is appropriate for counting many **blob-like overlapping objects with similar appearance (size, intensity, texture, etc..)** that may overlap in the image plane.

In order to avoid the difficult task of segmenting each object individually, this workflow implements a supervised object counting strategy called **density counting**. The algorithm learns from the user annotations a real valued **object density**. Integrating over a **sufficiently large** image region yields an estimate of the **number of objects** in that region. The density is approximated by a normalized Gaussian function placed on the center of each object. The user gives annotations in the form of **dots (red)** for the object centers and **brush-strokes (green)** for the background. A pixel-wise mapping between local features and the object density is learned directly from these annotations. Only the training image(s) requires manual labeling, and the counting on the entire training image(s) can be predicted. The full prediction on a large dataset can be done via Batch Processing Data Selection. But we will not discuss it here, please follow the steps from the ilastik online tutorial.

It is important to note that:

- The object density is an approximation for the true integer count. The estimates are close to the true count when integrated over sufficiently large regions of the image and when enough training data is provided.
- Contaminations of the image such as debris or other spurious objects may invalidate the estimated density.
- The current version of the Counting workflow is limited to handling **2D data only**.

Please refer to the ilastik website or [1] for further details.

2.4.2 Training and predicting density

As a first step after starting `ilastik` user interface, let us `Create New Project` of `Counting` and add the example image `C1-hela-cells.tif`.

The second step is to define some features at the `Feature Selection` applet. In particular, blob-detectors like the *Laplacian of Gaussians* or line-detectors like the *Hessian of Gaussians* are appropriate for blob like structures. One way of checking this is by viewing the response of each selected feature of specific size. This can be done by clicking the corresponding one in the `Features` panel at the bottom-left of the ilastik

window. The features that can be used in this example image is shown in Fig. 2.11, the `Select Features` window. For further details on feature selection please refer to the online ilastik user manual.

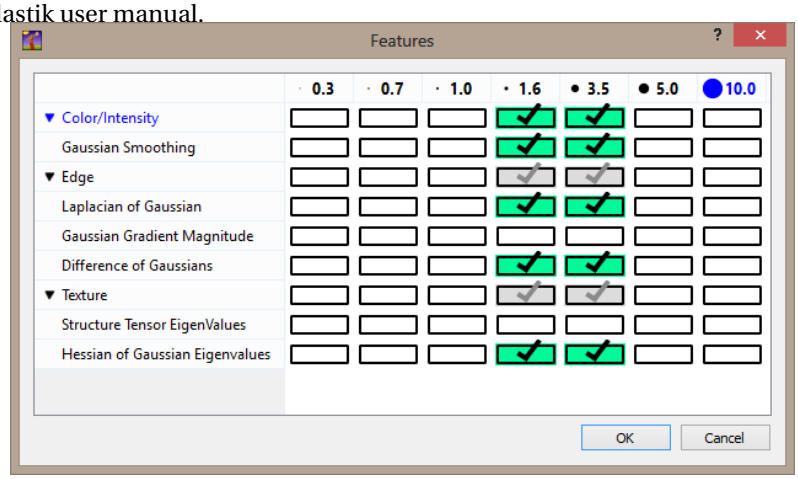


FIGURE 2.11: ilastk `Select Features` window.

The next step is to annotate, or label in the `Counting` panel. Annotations are done by painting while looking at the raw `Input Data` and at the intermediate results of the algorithm. The result of this algorithm can be interactively refined by activating `Live Update` mode. It offers possibilities to add **dots** for the object instances, **brush strokes** over the background, and **boxes** to monitor the count in sub-image regions.

To begin placing dot annotations select the red `Foreground` label and then click on the image. The annotation has to be placed close to the center of an object as in Fig. 2.12.

Given the dotted annotations, a smooth training density is computed by placing a normalized Gaussian function centered at each dot. The scale of the Gaussian is a user parameter `Sigma` which should roughly match the object size. To help deciding an appropriate value for this parameter you will see that the size of the *crosshair-cursor* changes accordingly to the chosen sigma (in the left panel). It can be visually inspected through the density shown in the `LabelPreview` layer in the bottom-left panel. An example of different choices for the parameter `Sigma` is shown in Fig. 2.13. On the left image, the value of sigma is chose too small, while on the right the value of sigma is too large. The center image shows a well chosen sigma. Please note that large values for `Sigma` (e.g. >5) can impact the required computation time, thus it is better to consider using a different approach, such as the `Object Classification` workflow. But this is not the case for our session.

After a few dots have been placed (say around 10 - 20 depending on the data) we can add training examples for the background. To activate the background labeling interaction select the green `Background` label and place broad strokes on the image (Fig. 2.12), marking unimportant areas or regions where the predicted density should be 0.

After some labels for the objects and the background have been given, a first prediction

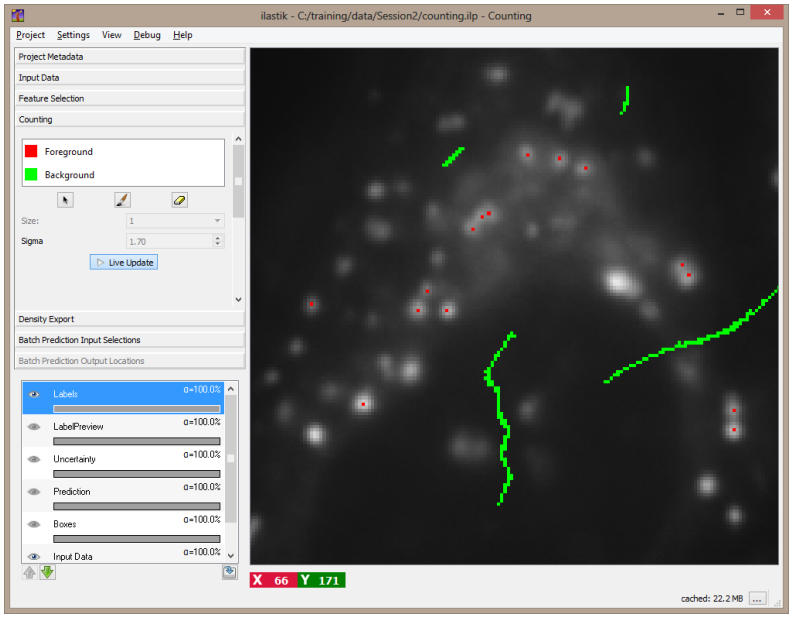


FIGURE 2.12: User annotations as training examples.

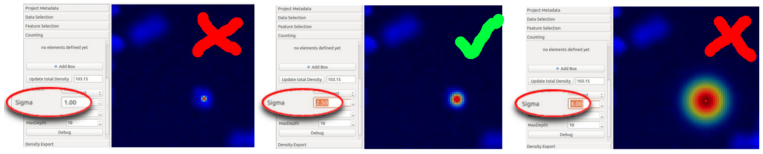


FIGURE 2.13: Select the `Sigma` values.

can be triggered by switching the `Live Update` on, and displayed in the `Prediction` layer (Fig. 2.14). Please note that if the `Live Update` Mode is active, every single change in the training data (e.g. placing new labels or changing parameters) causes a new prediction - thus it may be faster to set it `OFF` again when you plan for extensive modifications.

2.4.3 Counting objects/spots

2.4.3.1 Counting using Box in ilastik

The *boxes* are operator windows that integrate the density over the user-defined **rectangular** image region. Therefore they provide the predicted counts for the objects in that region. This interaction can only take place when the `Live Update` mode is activated.

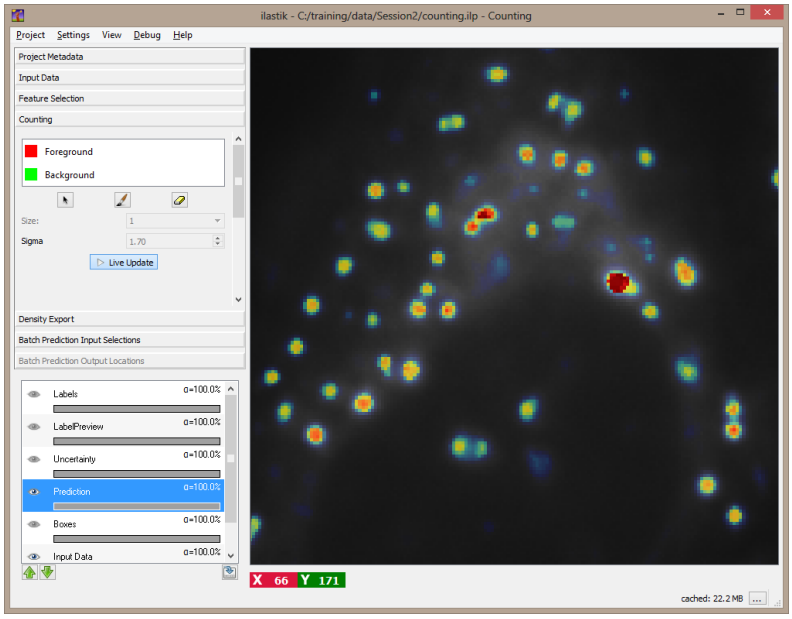


FIGURE 2.14: Prediction.

You can start placing boxes by selecting the `Add Box` button in the `Counting` panel and drawing a rectangular region on the image from the *top left* to the *bottom right*. The new box will be added automatically to the `Box List`. Boxes show the object count for the region on the upper left corner and each of them are displayed in a different color, which correspond to the color tag in the `Box List` (Fig. 2.15).

Boxes can be:

- **Selected and Moved**: you can select a box by its name in the `Box List` or just pass over it with your mouse. For the latter case, its name in the `Box List` is highlighted. The box will change color once selected. A box can be dragged in a different position by clicking and moving the mouse while pressing the `Ctrl` key.
- **Resized**: when selecting a box it will show 2 resize handles at its right and bottom borders.
- **Deleted**: to delete a box either click on the delete button (a red cross) on the `Box List` or press `Del` while selecting the box
- **Configured**: you can configure the appearance (color, fontsize, fontcolor etc...) of each individual box (or of all boxes at the same time), by clicking on the colored rectangle in the `Box List`. The interaction dialog for the box properties is shown below.

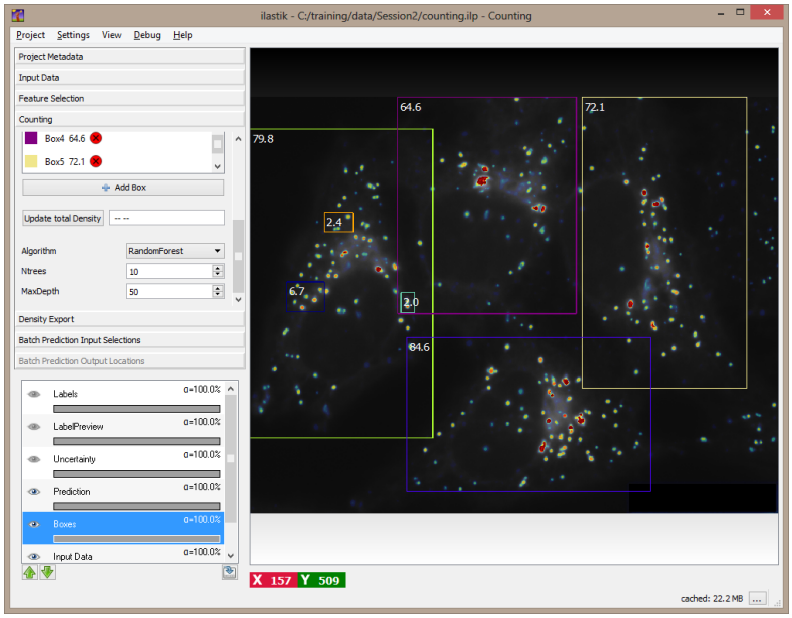


FIGURE 2.15: Counting in multiple user-selected *box* regions.

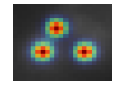
You can repeat the steps of adding boxes and provide new annotations (dots or strokes) for object centers and background until you are satisfied with the counting results for the boxes.

If we have added a set of images of the same kind and size as the training image, we can switch to another image by using the `Current View` menu on the left. Since the algorithm is already trained, then we are ready to compute the density for this new image. Similar as before, it is possible to start the prediction by toggling the `Live Update` button and monitor the results with a large *box* covering the entire image. Or we can also press the `Update total density` button on the left in the `Counting` panel. This button estimates the predicted count for the entire image. If the training labels are sufficient, we should obtain a count similar to what is shown in the image below that matches the number of objects in the images. *In a real world scenario, you may need to distribute several annotations across many images to obtain accurate counts for all the images in the dataset.*

You are now ready to use the workflow on your data! Please continue to read ilastik online manual if you want to know some advanced features.

Additionally, for those who are interested in testing different datasets and batch processing, a few more example datasets are also available in the Session2 folder. In particular, the “SimuCells" folder contain a few highly-realistic synthetic emulations of fluorescence microscopic images of bacterial cells, using the system [2]. Larger numbers

of such images can be downloaded from: http://www.robots.ox.ac.uk/~vgg/research/counting/cells.zip.



2.5 Counting in Fiji from ilastik predictions using arbitrary shapes

You may have already noticed that in ilastik, it is only possible to count in regions defined by a rectangular box. In this section, we will extend counting measurements to arbitrary shapes. This is done by exporting the ilastik density prediction results as e.g. `Tiff` images and then performing the measurements in Fiji.

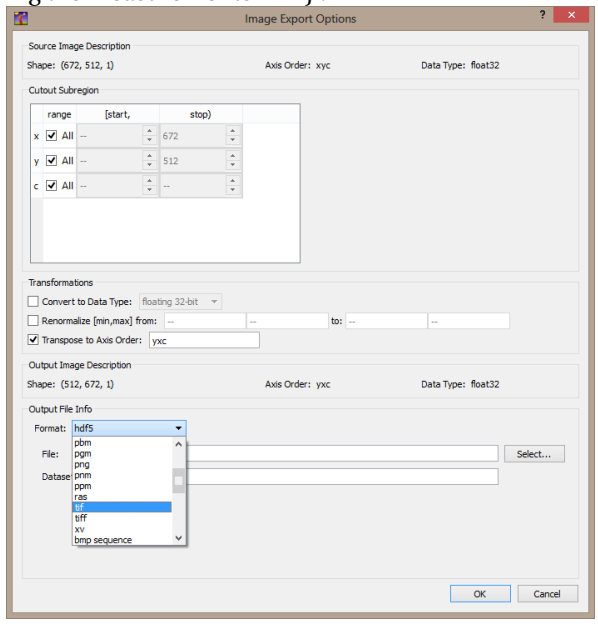


FIGURE 2.16: Exporting option window.

The prediction results can be directly exported by a right click on the `Prediction` layer then choose `Export`. In the popup `Image Export Options` window, set at the `Output File Info -> Format` pulldown list the `tif` option (Fig. 2.16), and then select the desired folder and filename to save it. You can also go to the `Density Export` panel and click on `Choose Settings`, a similar window will show up, but this time all the images will be save at once! So if you have multiple images added in the project and predicted, you should select the `tif sequence` option at the `Output File Info -> Format` pulldown list. In case of batch processing, similar operations could be done at the `Batch Prediction` panels.

Now we can open in Fiji the prediction tiff image you saved just now. Then we could use the selection tools in Fiji to draw the outline of the region of interest in arbitrary shape, with e.g. the Freehand tool. After the region is drawn and selected, the counting can be calculated by integrating the intensity values in the prediction image. We need to run

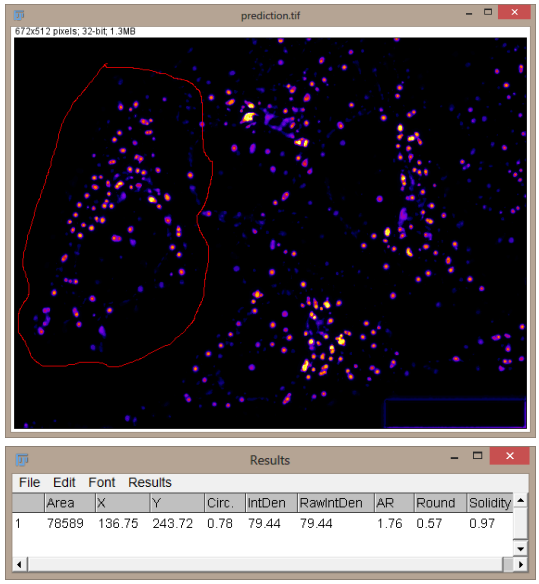


FIGURE 2.17: Counting spots in arbitrary region.

[`Analyze -> Set Measurements`] and select `Integrated density`, and then [`Analyze -> Measure`]. In the `Results` table window, the count value is shown at the column `IntDen`. Fig. 2.17 shows a region containing similar spots as the green large *box* region in Fig. 2.15. As we could see, they have similar counts, 79.44 vs 79.8.

You can also count spots in multiple regions, e.g. the nucleus regions that we have obtained previously from the blue nucleus channel. To take multiple regions from one image and analyze the same region in another, we know that Fiji offers "`ROI Manager`" for that. From the mask image of the blue nucleus channel (we did this in previous section 2.3.1.1), we can run [`Analyze -> Analyze Particles`], select `Show Nothing` and `Add to Manager`. Now we will select the ilastik prediction image, since we will count from it, and then run [`Image -> Overlay -> From ROI Manager`], we should be able to see the ROIs overlaid on the prediction image, as shown in Fig. 2.18. If only one region is to be measured, a quicker option, [`Edit -> Selection -> Restore Selection`], can be used as well. Now we can click `Measure` button in the `ROI Manager` window, the `Results` table window should look like Fig. 2.18, with the four counts in the `IntDen` column.

Compared to the results using Fiji (Fig. 2.7), the counting values obtained from these two methods may not be consistent. One reason is the possible under-segmentation of the objects. But when we look at the images in Fig. 2.7 and Fig. 2.18, we could see that the missing spots in Fig. 2.7 are the ones that have lower density (i.e. blueish) values in Fig. 2.18. This suggests that some dim spots are not present using the method we did

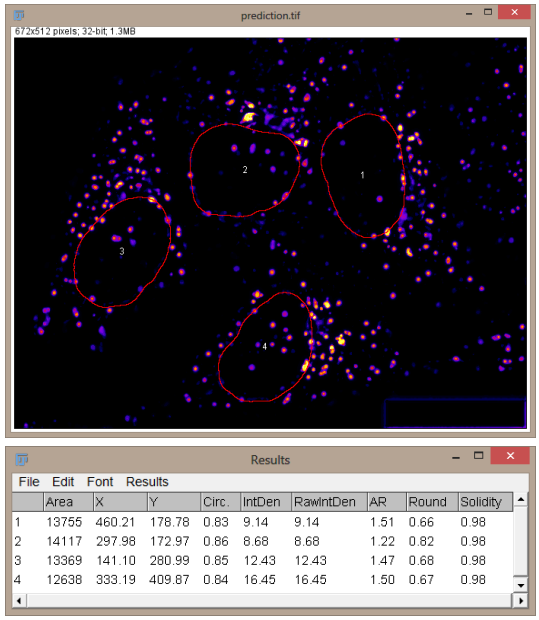


FIGURE 2.18: Counting spots in cell nucleus.

in Fiji. One reason causing this observation could be, by subtracting the approximated background, dim spots remained with even lower intensities and thus likely filtered away after thresholding to get the binary mask.

2.6 Tips and tools

- Search for commands in Fiji: select Fiji general interface, then hit the “L" key (or sometimes in MAC OS “`command`+L" key)
- ImageJ command recorder: `Plugins -> Macros -> Record`
- Official ilastik website: `https://www.ilastik.org` (too be updated very soon)

2.7 Groups close by which work on similar problems

- Prof. Fred Hamprecht’s group, Multidimensional Image Processing Group (`http://hci.iwr.uni-heidelberg.de/MIP/`), located at HCI, Speyererstr. 6

2.8 References

[1] L Fiaschi, R Nair, U Koethe, and F A Hamprecht. Learning to count with regression forest and structured labels. In *Proceedings of the International Conference on Pattern Recognition (ICPR 2012)*, 2012.

[2] A Lehmussola, P Ruusuvuori, J Selinummi, H Huttunen, and O Yli-Harja. Computational Framework for Simulating Fluorescence Microscope Images with Cell Populations. *IEEE Transactions on Medical Imaging*, 26(7):1010–1016, 2007.

3

2D+TIME TRACKING

If it is not fast, we lose interest, after all if we can not become a millionaire in 365 days, we may lose interest in becoming one at all ...

NOT!

Christopher Jansen

Part of ilastik general description was adapted from ilastik online documentation.

3.1 Aim

In this session, we will perform 2D+time tracking on objects of interest with two different methods: Fiji and ilastik. And further tracking results analysis of plotting trajectory on image and drawing displacement or velocity plot will also be done using Fiji.

3.2 Introduction

Time-lapse experiments play a crucial role in current biomedical research on e.g. signaling pathways, drug discovery and developmental biology. Such experiments yield a very large number of images and objects such as cells, thus reliable cell tracking in time-lapse microscopic image sequences is an important prerequisite for further quantitative analysis.

Dataset The dataset we will use ("mitocheck_small.tif") is kindly provided by the publicly available database of the MitoCheck project (`http://www.mitocheck.org/`). One of the focuses of the MitoCheck project is on accurate detection of mitosis (cell division). Please refer to [2] for more details of this project.

3.3 a Fiji solution

Let's load the "mitocheck_small.tif" 2D+time image. The very first step is to segment, or identify, objects (cell nucleus in this case) in each time frames. This step can be done by first smoothing a bit the stack to homogenize a bit the intensity within each object using [`Process -> Filters -> Gaussian Blur`] by a Sigma value of e.g. 2.0. Please note that although we are smoothing the whole stack, this 2D filter applies to each slice, or time frame, independently. Since the background is quite clean, a simple thresholding is probably good enough to segment the objects. You can choose your favorite thresholding method and set the best threshold value, e.g. with the Default or the Moments method, and with min and max threshold value set to 15 and 255, respectively. Probably there will be merged objects, so we can run [`Process -> Binary -> Watershed`] to split the most obviously wrongly-merged ones. We could also run [`Process -> Filters -> Minimum`] with Sigma of 1.0 to shrink a little bit the obtained binary mask. This is to avoid merging of close neighboring objects in 3D in further steps, and also to correct possible dilation of objects due to the Gaussian blur filter. If we wish, a second Watershed operation can be applied afterwards to further split potential wrong merges.

Suppose that we have so far obtained a binary stack containing the segmented cell nucleus. Next, we will track, or link, the same object in consecutive time frames. The final result will be a label stack where each object is identified (filled) by a unique label along time (indicated by a distinct gray level intensity value). **The strategy we employ here is based on the overlap of the same object in temporal space between two consecutive time frames. If we consider time as the third dimension, cell spatial overlap along time translates to the connected parts in the third dimension of a 3D object.** And Fiji's [`Analyze -> 3D Objects Counter`] does the job of finding such connected

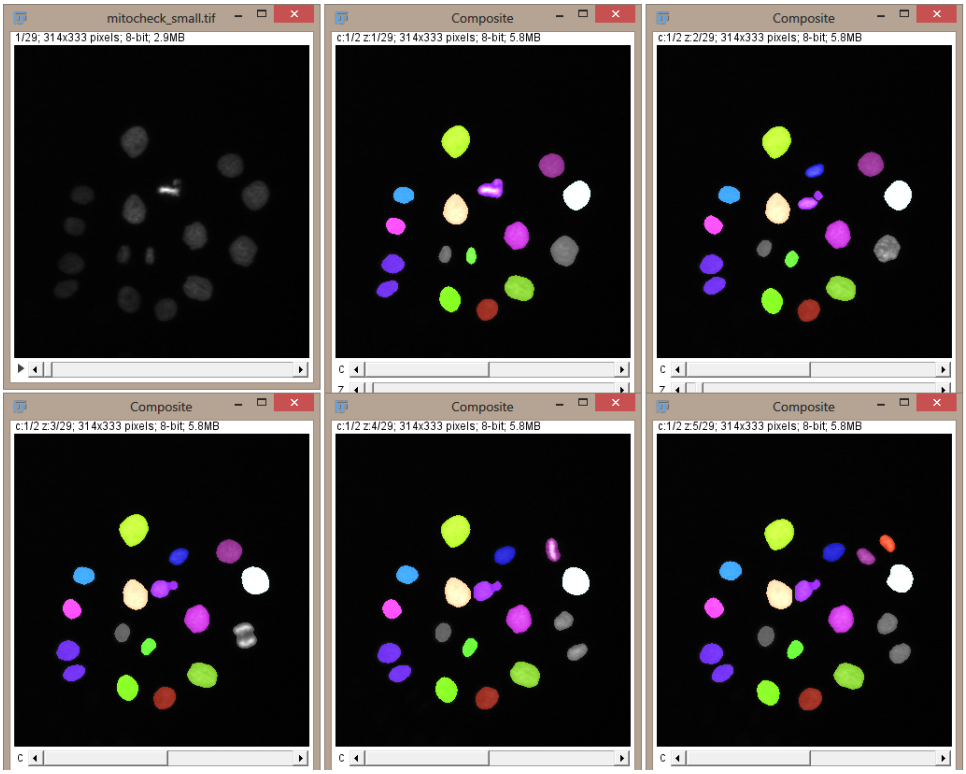


FIGURE 3.1: An example MitoCheck image (first frame) and the first five frames of the tracked objects (in different colors), using our Fiji solution, overlaid on the original image stack.

components in 3D. This function works similarly as the 2D one we are familiar by now - [`Analyze -> Analyze Particles`]. It can identify and count 3D objects in a stack, quantify each object's properties (see the full list in [`Analyze -> 3D OC Options`]) in a table, and generate maps of specified results representations. Once the 3D objects are 3D labeled, we can apply the `Random` LUT in the [`Image -> Lookup Tables`] to visualize better individual objects. We could also merge the label stack with the original stack to check results, using [`Image -> Color -> Merge Channels`]. You can specify the two stacks in two of the available channels. And by unchecking the option `"Ignore source LUT"`, it allows keeping the random LUT in the label channel thus colored IDs after merging. In Fig. 3.1 the first five frames of the objects (with their identity labels shown in random color) are shown.

Please note that the method we use here works properly only in cases: where single objects has spatial overlap in temporal space between any two consecutive time frames; and also a single object at a later time point does not overlap multiple objects at an ear-

lier time point. If there is no overlap, or connection between two consecutive frames, then a new label will be assigned to the same object in the latter frame, since it is considered as another object just showing up. One example can be found in Fig. 3.1. The two objects colored in orange and purple at the upper right corner of the last frame are split from the purple one in the previous frame. So both should haven been assigned the same identify (i.e. purple), but since the orange one has no overlap with its previous time frame, a new identity is assigned instead.

3.3.1 Other possibilities to solve tracking problem

An alternative to the 3D Object Counter could be `Plugins -> Process -> Find Connected Regions`. It sometimes could be faster than the 3D Object Counter and sometimes has better options (like starting from a point selection), but also lacks some flexibility.

There are other advanced tracking tools (included in Fiji or downloadable as Fiji Plugins) available such as [`Plugins -> Mosaic -> Particle Tracker 2D/3D`][1] and [`Plugins -> Tracking -> TrackMate`]. These trackers are however optimized for spot-like particle tracking, the linking is hence performed over a list of spot positions (spots detected in each frame). The tracking can be either straightforward (e.g. linking a spot to the closest spot in the next frame), or with algorithms that can handle cases when splitting and merging events occur.

If the cells, or objects of interest, resemble ellipsoids and are sufficiently separated these trackers might perform well provided the parameters are properly adjusted. For a densely packed scenario, or for more irregular object shapes, a possible strategy is to firstly perform a specific segmentation and generate a map showing for instance the objects centroids, and then apply the spot tracker on the centroids maps. For those who are interested in, an example on multicellular movement in Drosophila with detailed explanation can be found in the BioImage Data Analysis Course at EuBIAS 2013 (http://eubias2013.irbbarcelona.org/bias-2-course).

3.4 ilastik solution

Before we start, **please note that the main ilastik workflow we will be using in this session - the Automatic Tracking workflow - only works on machines where CPLEX is installed additional to ilastik**.

The fully automatic tracking workflow is able to track **multiple (dividing) objects** in presumably big datasets, both **in 2D+time and 3D+time**. But in this session, we will only deal with 2D+time data.

3.4.1 Segmentation with ilastik Pixel Classification workflow

The tracking workflow works on object level (rather than pixel by pixel) thus needs either results from the *Pixel Classification workflow* or segmented images from other sources.

[1] download site: http://mosaic.mpi-cbg.de/Downloads/Mosaic_ToolSuite.jar

We choose to practice the former.

When using *Pixel Classification workflow*, the user segments foreground objects (e.g. cells) from background by defining two labels and providing examples through brush strokes. We will just provide a brief description of this workflow, as we already had a demo in the general ilastik session and are using this workflow throughout the whole course.

Load time-series datasets Essentially, tracking data is a stack of 2D or 3D data at multiple time points. ilastik treats individual file as one dataset, and the time axis should be specified according to the specific data organization. This means, if your tracking data is stored as multiple files, we should use `Add Volume from Stack`; if your tracking data is a single stack file, then we should use `Add one or more separate Files`. In this case, we will use the latter. After the data is added in the project, you will notice that in the `Axes` column of the *top-right* panel it is specified as "xyz" rather than "xyt" (Fig. 3.2), because it considers the data as 3D not 2D+time. We need to correct it by *right* clicking the mouse at the row of our data, and the `Edit properties`, then change the `Axes` information from "xyz" to "xytc". Here "c" stands for channel, and we do have one channel.

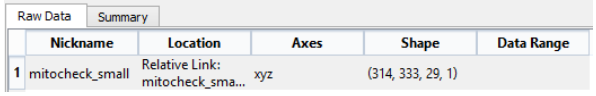


FIGURE 3.2: Data property panel.

In this example, we use all the features of sizes 1.6, 3.5 and 5.0. Then we paint some background pixels with Label 1 (red by default) and cell nucleus are marked with Label 2 (green by default). When happy with the live segmentation, we can apply the learned model to the entire dataset and export the results for further use in tracking. In the `Prediction Export` applet, we can specify the location and file format using `Choose Settings`. It is the same way as described in Section 2.5. However, since the pixel classes prediction will be used as intermediate results to another ilastik workflow, we could save it as an **hdf5** file, an ilastik-friendly format. A copy of the prediction (`mitocheck_small_export.h5`) and the Pixel Classification workflow project (`mitocheck_pixel.ilp`) can be found in the Session3 folder.

3.4.2 ilastik Automatic Tracking workflow

3.4.2.1 Object extraction

Now, the Automatic Tracking workflow can be launched from the start screen of ilastik by creating a new project (Fig. 3.3). To begin, the raw data and the prediction maps (the results from the Pixel Classification workflow or segmented images from other sources) need to be specified in their respective tab. In particular, the file `mitocheck_small.tif` is added as `Raw Data` and the dataset in `mitocheck_small_export.h5` is loaded as `Prediction Maps`.

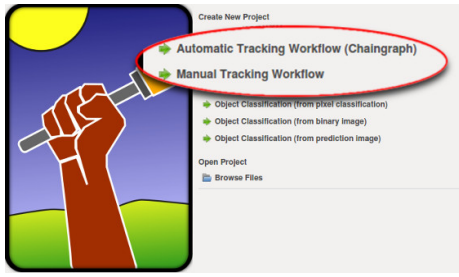


FIGURE 3.3: Tracking workflow staring page.

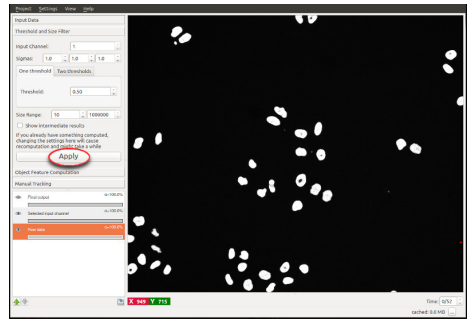


FIGURE 3.4: Foreground objects to be tracked.

Again, the tracking workflow expects the image sequence to be loaded as a time-series data containing a time axis; if the time axis is not automatically detected (as in hdf5-files), the axes tags may be modified in a dialog when loading the data (e.g. the `z` axis may be interpreted as `t` axis by replacing `z` by `t` in this dialog). A detailed description of similar step is in Section 3.4.1.

The prediction maps store a probability for each single pixel/voxel to be specific class defined in the pixel classification. First, the channel of the prediction maps which contains the foreground predictions has to be specified in the `Thresholding and Size Filter` applet. For instance, if in the Pixel Classification workflow, the user chose Label 1 (red by default) to mark foreground, `Channel` will be 0, otherwise, if Label 2 (green by default) was taken as the foreground label, then `Channel` takes value 1. Thus, we choose the Input Channel to be 1. These probabilities can be smoothed over the neighboring probabilities with a Gaussian filter, specified by the `Sigma` values (allowing for anisotropic filtering). The resulting probabilities are finally `thresholded` at the value specified. **The default values for the smoothing and thresholding should be fine in most of the cases.** Please consult the ilastik online documentation of the *Object Classification workflow* for a more detailed description of this applet, including an explanation of the `Two thresholds` option.

As mentioned before, although the tracking workflows expect prediction maps as input files, nothing prevents the user from loading binary segmentation images instead. *In this case, we do **NOT** want to apply the smoothing filter and thresholding to the binary images, so `Sigmas` should be set to 0.0 to keep the original segmentation images.* Finally, objects outside the given `Size` range are filtered out. And we could now proceed by pressing `Apply` (Fig. 3.4).

Please note that all of the following computations and the tracking will be invalid (and deleted) when parameters in this step are changed.

Next, we will go to `Object Feature Computation` applet, which is the most computationally intensive preprocessing step of the tracking workflows. Note that dependent on the size of the datasets, this step might take from minutes to hours. But this is also the less interactive step - all we have to do here is to press the `Calculate Features`

button. Neighboring pixels are then grouped in 2D to define individual objects, those objects are assigned independent and unique identities (indicated by distinct random colors in the `Objects` layer), and features of the objects (e.g. region centers) are computed.

3.4.2.2 Tracking

Now, we can start with the actual tracking of the detected objects. If **CPLEX is installed**, it is possible to launch the automatic tracking workflow (Chaingraph) and – after the same preprocessing steps as described above – we arrive at the automatic tracking applet. To track the objects detected in the preprocessing steps over all time steps, it is enough to press the `Track` button. For detailed explanation of the parameters and the algorithm behind, please refer to the online documentation and [1]. After successful tracking, each object and its children (in case of divisions) should be marked over time in a distinct random color. The results can be visualized directly in ilastik, or we can also export the objects, using the same export steps we have done previously. Since there might be a large amount of objects in the image thus large object label ID, the exported tracking objects images are set to 32bits. In case such exported images do not show object labels properly when using `Import -> Image Sequence` (whole 2D time series) or `Open` (individual exported time frame), we could use `Bio-Formats Importer` instead.

Fig. 3.5 shows the tracking results of the first five frames of `mitocheck_small.tif`. We showed both the ilastik Automatic Tracking workflow results (*top*) and the Fiji 3D Object Counter results (*bottom*). For most of the objects, both methods provided correct tracking results. The box frames highlighted three cases of two consecutive frame where Fiji 3D Object Counter method and ilastik differ. As mentioned before, the underlying condition for using `Fiji 3D Object Counter` for 2D+t tracking is that, the same object from one time frame to the next has some overlapping, and also from this next frame to its own next, and so on. And in situations e.g. when the cell moves too fast, or the time interval between frames is too long, etc, the spatial discontinuity makes this method not applicable. On the other hand, the ilastik tracking workflow can deal with it and give correct tracking results.

3.5 Tracking analysis in Fiji: plot trajectory, displacement, or velocity

Once the objects are correctly identified and tracked along time, we could do some further analysis such as extracting trajectories (Fig. 3.7) and calculating displacements and velocities (Fig. 3.8). We will now find out how to do this in Fiji. In this practice, we will write our own Macro script. By now, we are at ease with the great function `Plugins -> Macros -> Record`. Next we will try to take advantage of some ImageJ built-in functions. Please note that the plotting functionality in Fiji is probably not amongst the most powerful and versatile ones, but it still can manage basic demands. We would like to introduce it as a quick checking means before going for more complex scenes.

Let's treat each object as a rigid body, meaning that everywhere inside the object has exactly the same dynamic behavior. Therefore, we could simplify the problem and look at just one specific point of the object. Good candidates could be the centroid and center of mass. Let's take the centroid as example for illustration. The task now is to obtain a list of

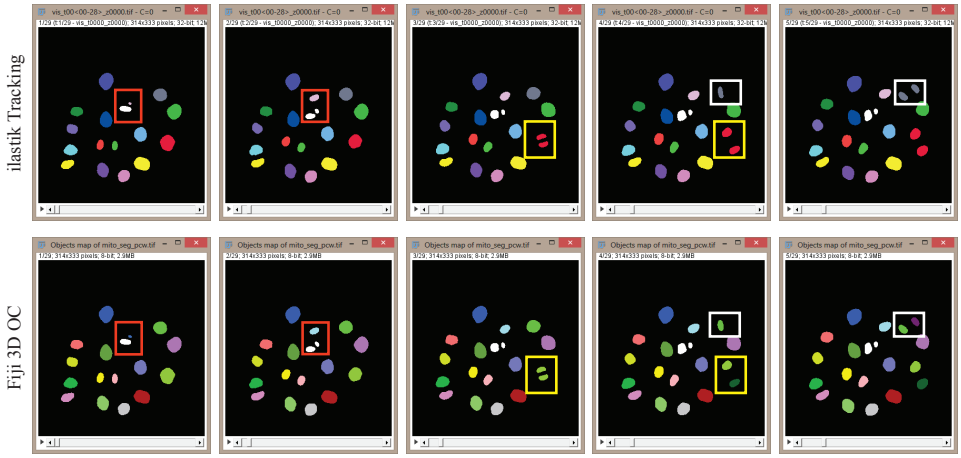


FIGURE 3.5: Results from two methods. The same color indicate the same object across frames. Most of the objects are correctly tracked using both methods. Highlighted colored box frames mark where the two methods worked differently. Note that the two methods may assign a different ID to the same object thus the coloring correspondence is lost.

Results

File Edit Font Results

	Min	Max	X	Y	Slice
1	1	1	128.25	102.02	1
2	2	2	230.62	127.94	1
3	3	3	173.29	144.47	1
4	4	4	257.68	159.91	1
5	5	5	164.56	154.62	1
6	6	6	72.06	159.65	1
7	7	7	127.70	175.38	1
8	8	8	63.39	192.87	1
9	9	9	193.16	203.21	1
10	10	10	244.38	219.36	1
11	11	11	116.74	223.12	1
12	12	12	144.40	224.64	1
13	13	13	60.48	233.12	1
14	14	14	196.10	259.24	1
15	15	15	54.57	260.19	1
16	16	16	121.67	271.15	1
17	17	17	161.52	282.22	1

FIGURE 3.6: Example `Results` window of the first slice/frame showing the requested measurements values from all objects in this slice.

the centroid coordinates in every slice where the object of interest is present. This means that each slice of the original stack should be processed individually. Beware to process the correct slice/time and record it to the correct time indices of centroid coordinates. To this purpose, use the built-in macro function **setSlice()**. A **for** loop could be implemented to go through all the slices, using **nSlices**, a built-in macro function returning the value of the number of slices (or frames) of the active stack. In order to obtain object centroid coordinates, we could use the `Analyze Particles` command to extract objects and check the `Centroid` item in the `Analyze -> Set Measurements` options window. Additionally, in order to know which extracted centroids is the current object we are interested in, we should also check `Min & max gray value` in the same `Analyze -> Set Measurements` options window. Since in previous steps we have identified the objects over time through assigning each object a unique intensity value as its label ID, then by checking the min or max gray value of this label image tells us which object centroid we should be look for in each slice. When we run `Analyze -> Measure` or click `Measure` directly in the `ROI Manager` window, the checked measurements items will be displayed in a table shown in the `Results` window (Fig. 3.6). In order to retrieve the table information, the built-in functions **getResult** and **nResults** can help, where **getResult("Column", row)** returns a measurement from the `Results` table of in total **nResults** rows. For example in Fig. 3.6, **getResult("X", 16)** will return value 161.52, which is the value of the last but also the **nResults**th row (where **nResults**=17 but it is indexed as 16 since the first row has index of 0). Once we identify an object with its ID specified by the intensity value in the label image, we can go through the objects list in the current slice and look for it, using an **if** sentence to check this condition by comparing the ID values - "**Max**" value (in this case "**Min**" or "**Mean**" values are also fine since they have the same value). And once the condition is met, the centroid coordinates can be obtained by getting the Columns "**X**" and "**Y**".

The following piece of the source code shows the corresponding part that implements the steps described above. The mentioned built-in functions are highlighted in brown color. Please note that we clean up the `Results` table and the `ROI Manager` after finishing checking each slice.

```
//The object of interest, specified by "objID"
objID = 16;

//Define and initialize an array with a time flag for each frame, whether
    the object "objID" is shown
objShowTime = newArray(nSlices);
Array.fill(objShowTime,0);
//Define and initialize two arrays to store the centroid coordinates in X
    and Y axes
objCenterX = newArray(nSlices);
Array.fill(objCenterX,-1);
objCenterY = newArray(nSlices);
Array.fill(objCenterY,-1);

//Check the measurements options, i.e. min & max gray value and centroid
run("Set Measurements...", "  min centroid redirect=None decimal=2");

for(i=1;i<=nSlices;i++)
```

```
{
  //Select the binary mask stack image, with image ID "cpID", and get the
    slice i
  selectImage(cpID);
  setSlice(i);
  //analyze the regions of each object in slice i, and add the regions to
    roiManager, to obtain ROI selection
  run("Analyze Particles...", "size=0-Infinity circularity=0.00-1.00 show=
    Nothing add");

  //We would like to measure the min/max gray value in the object label
    image so as to get object "objID"
  selectImage(lblID);
  setSlice(i);
  roiManager("Measure");

  //Going through all the shown objects in the current slice to look for the
     object with "objID"
  for(j=0;j<nResults;j++)
  {
    current_objID=getResult("Max", j);
    if (current_objID==objID)
    {
      print(current_objID);
      objShowTime[i-1]=1;
      objCenterX[i-1]=getResult("X", j);
      objCenterY[i-1]=getResult("Y", j);
    }
  }

  //Clean up, for each slice, the Results table and also the ROI Manager
  run("Clear Results");
  selectWindow("ROI Manager");
  run("Close");
}
```

:

3.5.1 Plot trajectories on image

In this exercise, we aim at practicing ImageJ built-in functions to extract measurements from the `Results` table and further plot object trajectory on image. For illustration, the simplest situation of plotting one trajectory from a single object at a time is considered.

In Fiji, we could use the built-in function **makeLine** in order to plot the trajectory of the centroid positions overlaid on the image (e.g. the first frame or the first frame when the object shows up). Since this function only takes integer pixel positions, the function **floor** is used to always take the integer part of the coordinates and omit the decimal part. The code below realizes the trajectory we want to plot. It should be inside a **for** loop since it prints the trajectory segment by segment between two consecutive centroids. The `Properties` command configures the line appearance. And we need the `ROI Manager` to keep every line segment on display. Is there any special attention we should pay to the `ROI Manager`?

```
makeLine(floor(objCenterX[i-1]),floor(objCenterY[i-1]),floor(objCenterX[i]),
    floor(objCenterY[i]));
run("Properties... ", "name=[] position=none stroke=Red width=2");
roiManager("Add");
```

:

An alternative is the `makeSelection` built-in function with `polyline` type, which does not need to plot line segment by segment but rakes the coordinates arrays.

Fig. 3.7 shows four examples of trajectories overlaid on the mask image. The source code can be found in the Session3 folder ("Tracking_measurements.ijm"). *How to plot multiple trajectories in one goal* is left for those who are interested to practice yourselves after the course. Also, The Fiji built-in functions "Overlay" is an alternative to display trajectories. We will also leave it for your own exploration.

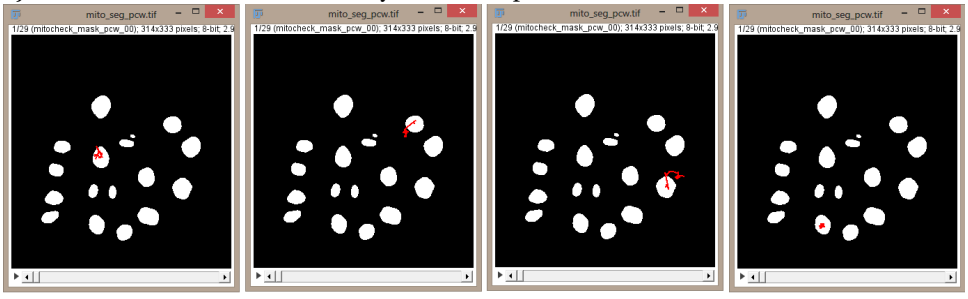


FIGURE 3.7: Example trajectories (marked in *red* lines) of object centroid movements.

3.5.2 Plot displacements or velocity

Examining the object displacements or velocity changes is another interesting measurement for tracking studies. Since we have already obtained the centroid position at each time frame, then the displacement of the centroid point from one time point to the next is the distance between the two positions. The displacement is the square root of the sum of squared difference between each coordinates of the points, which is calculated as shown in line 3 of the code below, in our case. The built-in function `sqrt` calculates the square root of a given number. The displacements over time can be stored in an array, "`disp`", which starts with one array element and keeps growing. This is because we do not expect to know in advance the temporal span of each object. Line 6-13 shows how the array element grows. In the code a flag "`arrayOK`" is used, what is its use?

```
1 disp = newArray(1);
  arrayOK = false;
3
  current_disp = sqrt( (objCenterX[i]-objCenterX[i-1])*(objCenterX[i]-
      objCenterX[i-1]) + (objCenterY[i]-objCenterY[i-1])*(objCenterY[i]-
      objCenterY[i-1]) );
5 //store the current displacement into the arry "disp"
```

```
if (arrayOK)
{
  disp = Array.concat(disp,current_disp);
}else
{
  Array.fill(disp,current_disp);
  arrayOK = true;
}
```

:

And if the time interval is known, the displacement divided by the time interval gives the velocity. ImageJ offers the **Plot** functions to create a window showing a plot. Use it like this:

```
//Set the right time frame - xCoord - for the plot.
//startFr and endFr mark the frames when the current object shows up and
    finishes
xCoord = newArray(endFr-startFr+1);
for(i=startFr;i<=endFr;i++)
{
  xCoord[i-startFr]=i;
}
Plot.create("Fancier Plot", "Frame", "Displacements (pixel)");
//Set the display range of each axis.
//maxY is the maximum displacement of the current object from array disp
Plot.setLimits(0, nSlices, 0, maxY+1);
Plot.setLineWidth(2);
Plot.setColor("lightGray");
Plot.add("line", xCoord, disp);
Plot.setColor("red");
Plot.add("circles", xCoord, disp);
Plot.show();
```

:

And Fig. 3.8 shows four examples of such plots from the corresponding objects shown in Fig. 3.7. Despite of the noisy profile, the differences in terms of amplitude and pattern are discernible.

3.6 Tips and tools

- ImageJ Built-in Macro Functions: http://rsbweb.nih.gov/ij/developer/macro/functions.html
- Built-in functions used: getDimensions, getStatistics, newArray, Array.fill, Array.concatenate, setSlice, nSlices, nResults, getResult, makeLine, Plot
- CPLEX installation guid: http://ilastik.github.io/installation/installation.html
- EuBIAS 2013 - BioImage Data Analysis Course: http://eubias2013.irbbarcelona.org/bias-2-course

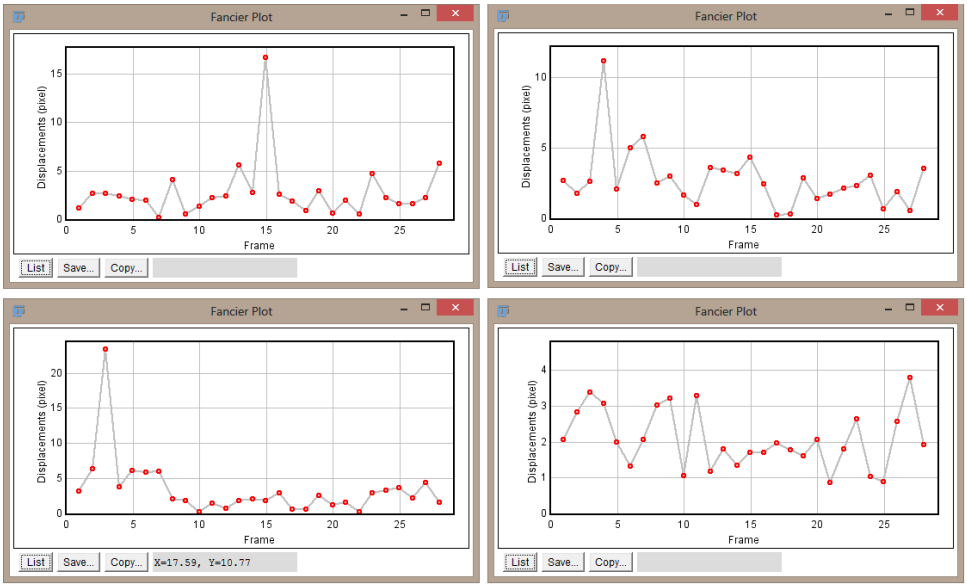


FIGURE 3.8: Example displacements of centroids from the four objects as in Fig. 3.7 (order: *left* to *right* and *top* to *bottom*).

3.7 Groups close by which work on similar problems

- Dr. Karl Rohr's group, Biomedical Computer Vision Group (http://www.bioquant.uni-heidelberg.de/?id=322), located at Bioquant
- Prof. Fred Hamprecht's group, Multidimensional Image Processing Group (http://hci.iwr.uni-heidelberg.de/MIP/), located at HCI, Speyererstr. 6
- Dr. Christian Conrad's group, Intelligent Imaging Group (http://ibios.dkfz.de/tbi/index.php/intelligent-imaging/people/94-cconrad), located at Bioquant

3.8 References

[1] B X Kausler, M Schiegg, B Andres, Lindner M, Leitte H, Hufnagel L, Koethe U, and F A Hamprecht. A discrete chain graph model for 3d+t cell tracking with high misdetection robustness. In *Proceedings of the European Conference on Computer Vision (ECCV 2012)*, 2012.

[2] B Neumann, T Walter, J-K Hériché, J Bulkescher, H Erfle, C Conrad, P Rogers, I Poser, M Held, U Liebel, C Cetin, F Sieckmann, G Pau, R Kabbe, A Wünsche, V Satagopam, M H A Schmitz, C Chapuis, D W Gerlich, R Schneider, R Eils, W Huber, J-M Peters, A A Hyman, R Durbin, R Pep-

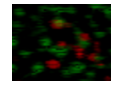
perkok, and J Ellenberg. Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes. *Nature*, 464:721–727, 2010.

4

3D Object based colocalization

All things appear and disappear because of the concurrence of causes and conditions. Nothing ever exists entirely alone; everything is in relation to everything else.

Hindu Prince Gautama Siddharta



We would like to thank Dr. Ke Peng (Virology department, University of Heidelberg) for sharing some of his datasets and accepting to expose part of the methodology involved in his own research work.

Part of ilastik general description was adapted from ilastik online documentation.

4.1 Aim

In this project we will first train a pixel classifier in ilastik to segment object of interest; then we will implement an ImageJ macro to analyze 3D object based colocalization in multiple channels; finally how the colocalization results can be visualized will be discussed.

4.2 Introduction

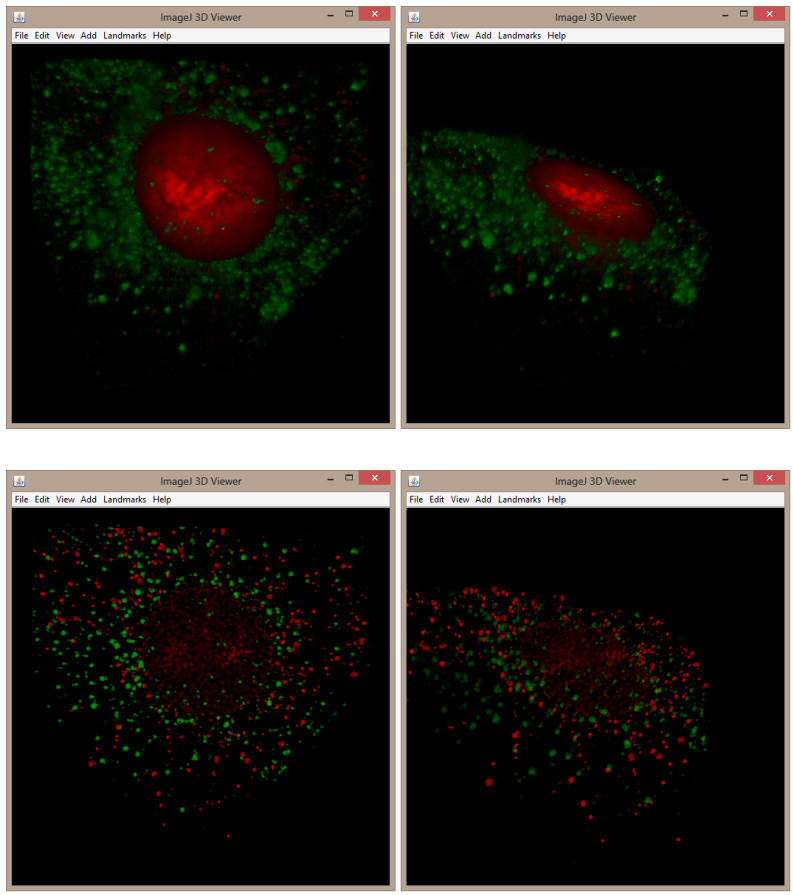
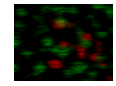


FIGURE 4.1: (*top*) The Hela cells dataset (with two channels) from two views. (*bottom*) The prediction from the trained classifiers using ilastik Pixel Classification workflow, also from two views. Higher intensity in the prediction images indicates higher probability of being the objects of interest.

Subcellular structures interact in numerous ways, which depend on spatial proximity or spatial correlations between the interacting structures. Colocalization analysis aims at

finding such correlations, providing hints of potential interactions. Talking about colocalization, we often also think about deconvolution. Careful image restoration by deconvolution removes noise and increases contrast in images, improving the quality of colocalization analysis results. However, deconvolution is not the focus of this session. Therefore, we would assume that the images to be processed are either already deconvolved or are acquired with high image quality without the need of deconvolution.

Typically, two categories of approaches to colocalization analysis can be found: intensity based correlation methods and object based methods. We will focus on object based methods in this session. Most object-based colocalization methods first segment and identify objects, and then account for objects' inter-distances to analyze possible colocalization. Usually the centroids of these objects are determined and used to calculate the distance [1–3]. Here we will use another criteria, **two objects with certain percentage of volume overlap**, as the indication of colocalization.



Dataset: Virologists often need to answer the questions of when and where the viral replication happens and the relevant virus-host interactions. The dataset (see Fig. 4.1 *top* as an example) we are using in this session is Hela cells imaged with a Spinning disk microscope. Z serial images were acquired in three channels, from which two are used: channel 1 (C1, red) shows viral DNA in high contrast and channel 3 (C3, green) shows viral particles in high contrast (a viral structural protein). The high contrast signals either come from synthetic dyes or fluorescent protein. The goal is to identify the viral particles that have synthesized viral DNA indicating such structures represent replicating viral particles and potentially the viral replication sites. Thus, the identification can be achieved through a colocalization analysis between the objects in these two channels.

4.3 Segmenting spots using ilastik Pixel Classification workflow

There are multiple tools for detecting spot-like structures, including the two methods we have used in the spots counting session. For this session, we will practice a bit more with ilastik, i.e. we will train a spot classifier using ilastik Pixel Classification workflow. This learning based method is especially suitable for training a classifier that can "pick" objects of interest out of other structures in the same image. For datasets like in Fig. 4.1 *top*, we are only interested in highly-contrast spot-like structures, which coexist with others such as nucleus and cloud-like objects.

For this Hela cell dataset, we will use intensity and texture features of sizes 0.7, 1.0, 1.6, and 3.5 (Fig. 4.2 *top*) for both channels. Of course, you can also try using a different set of features if they work better to you. Then we paint some particle voxels with Label 1 (red) and background ones are marked with Label 2 (green) to train ilastik to classify these two types for the whole volume (Fig. 4.2 *bottom*). When happy with the live segmentation prediction, we can apply the learned model to the entire dataset and export the results for further use in colocalization. Fig. 4.1 *bottom* shows the results of the predicted objects from training. In the `Prediction Export` applet, we can specify the location and file format we want to export using `Choose Settings`. It is the same way as described in Section 2.5. We will export the prediction images as a `tif sequence` type in separate

folders for different channels. A copy of the prediction images and the Pixel Classification workflow project can be found in the Session4 folder.

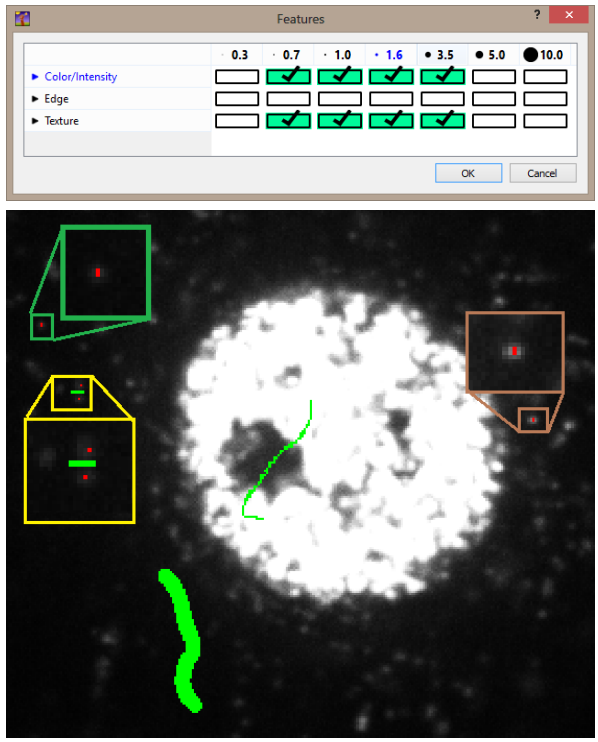


FIGURE 4.2: (*top*) Selected features for training the particle classifier in Hela cells. (*bottom*) Example training labels for particles (*red*) and background (*green*), with insets of zoomed in views.

4.4 Writing our own ImageJ macro for fully automatic colocalization

Many of you may have already been using tools e.g. the ImageJ plugin JACoP [1] for object- or intensity-based colocalization analysis. Here, we will not use any of these, but try to write our own macro for fully automatic colocalization so that we could establish our own protocols and control settings, and at the same time practice more with macro scripting.

We have practiced in the previous sessions with: the function [`Plugins -> Macros -> Record`] to track the sequential operations that have been applied; cleaning up and converting the recorded macro commands to a functional macro file; and even a few ImageJ built-in functions. In this session, we will exploit a little bit more ImageJ Macro scripting, e.g. use more and different functions, construct a parameter setting interface window, etc.

In order to help better understanding the steps during the object-based colocalization, we will first use a synthetic dataset to test and build up the macro. And once our

macro program is working, we will test it on the segmented dataset from the previous step. Fig. 4.3 shows two 3D views of the synthetic dataset with two channels, where channel 1 (*red*) has six objects and channel 2 (*blue*) seven. Each object in channel 1 has different level of spatial overlap with one of the objects in channel 2. The synthetic dataset can be found in the Session4 folder (C1_syn and C2_syn).

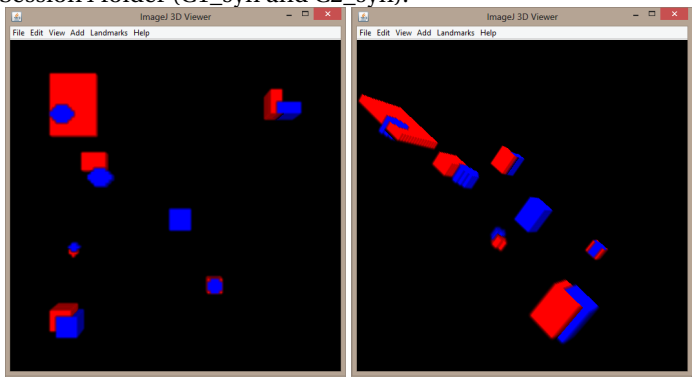


FIGURE 4.3: Synthetic 3D dataset from two views.

4.4.1 Filtering objects by size

Often the segmentation contains objects that are not interesting for us such as noise or other structures. Since object-based methods concern individual objects, then we should apply some filtering criteria in order to discard them for further analysis. Such criteria could be, for example:

- (3D/2D/1D) size range of the object-of-interest (in each channel)
- object shape, e.g. circularity[1], compactness[2]
- object location

It should be mentioned that this step greatly influences the colocalization measurements. We will discuss only size related filtering here. Our strategy consists of two steps: filtering in 3D, and then in 2D.

4.4.1.1 Filtering by 3D sizes

As we have already learned from the previous session, `3D Objects Counter` is able to do this. We can open the macro file "`S4_filtering.ijm`" and add steps in. After

[1] Circularity measures how round, or circular-shape like, the object is. In Fiji, the range of this parameter is between 0 and 1. The more roundish the object, the closer to 1 the circularity.

[2] Compactness is a property that measures how bounded all points in the object are, e.g. within some fixed distance of each other, surface-area to volume ratio. In Fiji, we can find such measurements options from the downloadable plugin in `Plugins -> 3D -> 3D Manager Options`.

running it, we will be asked to specify the directories where the images from two channels are, and also a directory to store results. After that, we will see a window with many parameters to setup. Let's skip these first, and comment on them at later steps. So now let's select the image of channel 1, and then run [`Analyze -> 3D Object Counter`], in the popup window there are two parameters of our interest, `Min` and `Max` in the `Size filter` field. Let's suppose that the object of interest should have a size of minimum 3 voxels and maximum 10 voxels in each of the three dimensions, resulting in object volume of size `Min`=27 and `Max`=1000. This filtering step removes the smallest object from both channels. Although they may seem to overlap with objects in the other channel, they are likely to be e.g. noise and their spatial co-occurrence could be coming from randomly distributed particles/noises that are close to each other by chance. Since in this session we may have to produce many intermediate images, so it might be a good practice to `rename` these intermediate images. And if they will not be used any further, they might as well just be closed by running [`File -> Close`].

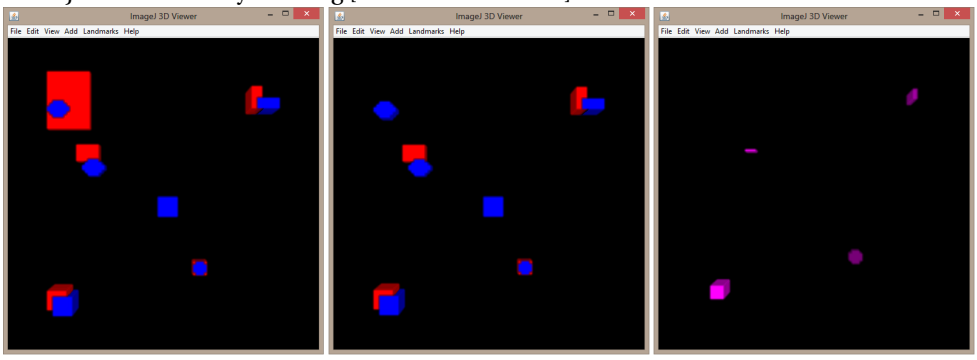


FIGURE 4.4: Synthetic 3D dataset after first filtering in 3D (*left*), then also in 2D (*middle*), and the overlapping regions after the filtering steps (*right*).

4.4.1.2 Filtering by 2D sizes

You may have noticed that this filtering criteria is not sufficient to remove the large object in channel 1 (Fig. 4.4 *left*). This is because e.g. the object is very thin in one or two axes and large in other(s) thus the total 3D size may be within the size range of the object of interest. In this case, it makes sense to have another filtering but in 2D rather than in 3D.

For example, one possibility is to set a size range in the XY plane, and if needed also a maximum spanning size in the Z axis. In order to do this, we will use [`Analyze -> Analyze Particles`] and its `Size` parameter setting. So similarly we could set the value to be e.g. 9-100. Also, distinct size range can be employed for each channel if needed. Additionally, for some applications, the parameter Circularity could also be used. Since we would like to obtain the filtered image, thus "`Masks`" will be chosen to `Show`. Note that the `Analyze Particles` function works on binary images only, then we first need to convert the label image by the 3D Object Counter, which is a label image. A simple thresholding should work, since the label starts at 1. Normally the binary image

after thresholding has value 0 and 255. Sometimes, a binary image of value 0 and 1 makes further analysis easier. To do so, we could divide every pixel by this image's non-zero value, i.e. 255, using [`Process -> Math -> Divide`].

What should we do if the 2D filtering is to be done in the XZ plane? If the axes are swapped, so as the original XZ plane will be the new XY plane, then we could apply the same procedure on the axes-swapped image. This can be done with e.g. [`Plugins -> Transform -> TransformJ -> TransformJ Rotate`], which applies to the image a rotation transform around the three main axes. Please note that if the image to be processed is large, this will not be the optimal solution as it will be both time and computation expensive to apply a transform and interpolate the whole image. A faster option could be [`Plugins -> Transform -> TransformJ -> TransformJ Turn`]. The advantage of using this plugin rather than the more generally applicable [`Plugins -> Transform -> TransformJ -> TransformJ Rotate`] is that it does not involve interpolation of image values but merely a reordering of the image elements. As a result, it generally requires much less computation time, and does not affect image quality.

Another possibility, probably the easier option in many cases, is the erosion + dilation operations in 3D. In this case, we would only want to erode and then dilate in one dimension - the one where object-to-be-removed is thin. Let's try it, `Process -> Filters -> Minimum 3D` and then `Process -> Filters -> Maximum 3D` with the same radius settings in each dimension. We will set both X and Y radius to 0 and try with Z radius being 3.0, because the large object in Channel 1 is thin in Z axis. In general it should work - provided the filter radius is not smaller than the object radius along its smallest dimension, then the object should disappear and not return. This also gives the possibility to filter anisotropically considering the fact that in many microscopy images the Z spacing is larger than pixel size in then XY plane. Note that the erode/dilate alternative in the `Plugins -> Process` submenu will not work, because the filter size is not adjustable). On the other hand, please keep in mind that the erode and dilate operations may modify object shape, especially when objects are not roundish blob-like.

Depending on specific applications, more filtering steps can be applied before or after the previous size filtering, such as shape or location. In this session we will not discuss in details and assume the size filtering is sufficient for our task (Fig. 4.4 *middle*).

4.4.2 Finding spatial overlapping objects in both channels

In order to find colocalized objects, we will first find the overlapping (or shared) parts of the two filtered channels, and then identify the corresponding objects in each channel that contain these overlapping regions. To achieve this, what do we need to do? There are multiple ways, all of which involve:

- in each channel, label the objects so as to identify them;
- in each channel, calculate the volume (in voxels) of each object;
- compute the objects' overlapping regions of the two channels;
- calculate the volume of each overlapping region.

- find the labels of objects in each channel that overlap with some object in the other channel

These tasks could be done with the help of [Analyze -> 3D Object Counter] and [Plugins -> 3D -> 3D Manager].

We could see that in both channels, the overlapping region has value higher than zero; while the rest of the two images have either one or both channels with zero background. Therefore if we multiply the two images, only the overlapping regions will show values higher than zero. So we can run [Process -> Image Calculator], set the object maps of from the two channels as Image 1 and Image 2, and set Multiply as Operation. Let's call the obtained multiplicative image as "SharedMask". Fig. 4.4 *right* shows the 3D view of the overlapping regions after the filtering steps. We can see that the two overlapping but with too large and too small objects (see Fig. 4.3) are now excluded, remaining four overlapping regions of the two channels. Note that the images of both channels that contain objects filtered by size are binary images of values 0 and 1, thus the "SharedMask" is also a binary image of values 0 and 1. Now, if we add the "SharedMask" to the filtered binary images of each channel, the two resultant images would give background zero value, all the objects in each channel value 1, except where the overlaps is, i.e. 2. Then when using "3D hysteresis thresholding" plugin, only regions with value >= a low threshold (i.e. 1) that also contain value >= a high threshold (i.e. 2) are extracted, i.e. the objects containing the overlaps. Therefore, we have obtained also one image per channel, which contains only objects that overlap with some object in the other channel.

Since we define the object volume overlap ratio as the colocalization criteria, objects and their volume values in both channels and the SharedMask should be calculated. We need to make sure in [Analyze -> 3D OC Options], to check the option of "Nb of Obj. voxels" (if we count voxels) or "Volume" (number of voxels multiplying voxel size). If the voxel size is not set, by default it is 1 for each dimension, thus these two parameters give the same value. After running the 3D Objects Counter, the resulted Object map gives each object a unique non-zero label and the background the zero label. Also, the measurements will be shown in a table. You may recall that in 3D OC Options menu, if "Store results within a table named after the image (macro friendly)" is not checked, the results are stored in the Results table. This way we could use the built-in macro functions nResults and getResult to access items in the table. Let's do this again, with the following code:

```
ObjVolume_ch1 = newArray(nResults);
print("ObjVolume_ch1 ("+nResults+"): ");
for(i=0;i<nResults;i++){
  ObjVolume_ch1[i] = getResult("Nb of obj. voxels", i);
  print(ObjVolume_ch1[i]);
}
```

:

This code first creates a new array, *ObjVolume_ch1*, so as to store all objects' volumes in the current channel. It is then filled with the values obtained from the Results table. For checking the code we could also print the array. Similar arrays and object labels should be created for the other channel and the SharedMask.

So far we have obtained the objects' labels and volume values in each channel and the "SharedMask" image. The next task will be to identify the labels of each object pair that overlap, in order to further find out their corresponding overlap volume ratio. Before jumping into the next part, let's ask ourselves a question - does our problem involve analyzing individual colocalized objects, or rather a global measure that tells something about colocalized objects as a whole? If the answer to your problem is the later, then we could skip the remaining of this section and the following one. Instead, probably some nice tools could do the job for us. For example as mentioned previously, the ImageJ plugin JACoP [1] offers measures, e.g. Mander's Coefficients, Overlap Coefficient, and object-based methods. We will not provide detailed descriptions on how to work with them here, please refer to their corresponding online documentation.

If you are interested in studying individual colocalized objects and their further characteristics, such as their spatial distribution, shape, size, etc, we will go on to the next task of identifying in each channel which objects overlap with objects in the other channel. Since we know the overlapping regions, then we just need to look at the same regions in each of the two channels to get these objects that have overlapping parts in the other channel. We have used `ROI Manager` before, now we will use another function from the plugins we installed: [`Plugins -> 3D -> 3D Manager`]. It plays a similar role as the `ROI Manager` but in 3D. So the first thing is to add the 3D overlapping regions into the `3D Manager` so that we could use them for further measurements, and also for inspecting the same regions on other images such as the label images. To do so, we will use the following code[3]:

```
selectImage("Objects map of Shared Mask");
run("3D Manager");
Ext.Manager3D_AddImage();
Ext.Manager3D_SelectAll();
```

:

After adding the 3D overlapping regions into the 3D Manager, we will select the object label image of channel 2 so as to find out the corresponding label values for every overlapping region. Similar to what we have done before, we can measure the maximum intensity value of each region from the label image in channel 2. So we will check `3D Manager Options` and select the `Maximum gray value`. And then click the `Quantif_3D` in the `3D Manager` window. It will give a window named "3D quantif", with a column named as "Max". This column stores the maximum gray values of each region in 3D Manager from the label image in channel 2. Since it is not the usual "`Results`" table, we can't use the `nResults` and `getResult` built-in functions to access the contents of this table. How do we obtain the objects in each channel that have these overlapping regions? There are many ways to achieve this, but let's see how to extract information from any table, in this case a table named "`3D quantif`". **The following code shows an example of how we can get one column's items from a table that is not the Fiji default `Results` table**:

```
shareObjLabelInCh2 = newArray(numSharedObjects);
```

[3]Note that the "`Ext`" is a built-in function added by plugins using the MacroExtension interface.

```
selectWindow("3D quantif");
tbl = getInfo("window.contents");
tblLines = split(tbl, "\n");
idx=0;
for (i=1; i<= numSharedObjects; i++){
  lineA = split(strA[i], "\t");
  shareObjLabelInCh2[idx]=lineA[2];
}
```

:

where (in lines 3-4) "*tbl*" stores everything in the "`3D quantif`" table as *string*, and we can get each item from the table by two "`split`" operations: first into lines through the line break "`\n`" (as in line 4) and then into separate items through the tab or column break "`\t`" (as in line 7). *shareObjLabelInCh2* is an array of size *numSharedObjects*, the number of overlapping regions, which stores the object labels that contain the corresponding overlap part in channel 2. Of course, there is now a built-in function `IJ.renameResults` available (if you have ImageJ version 1.46 or later) that changes the title of a results table from one to another. Thus, we can always change the name of any table to "Results" and then use the easier built-in functions `nResults` and `getResults` to get table contents.

4.4.3 Filtering the colocalization objects by volume overlap percentage criteria

We have finally arrived to the stage that we are ready to select "real" colocalized objects from the candidates. As we will use the volume overlap criteria, let's first calculate the volume overlap ratio. There may be several ways to define the ratio, we will use e.g. the ratio between the overlapping region and the volume of the smaller of the two objects. In order to not complicate the problem too much, we will assume that objects in one of the channels have smaller size. This could be a reasonable assumption in many biological applications. The following code realizes the process of determining the colocalization using such selection criteria, assuming the channel with smaller objects is channel 2. A new image stack, "SharedMask Filtered", is created and filled with the overlapping regions that have volume size larger than the specified percent of the corresponding object in channel 2:

```
selectImage("Objects map of Shared Mask");
run("3D Manager");
Ext.Manager3D_AddImage();
newImage("SharedMask Filtered", "8-bit black", width, height, slices);
for (j=0; j<numSharedObjects; j++){
  objLabelInC2 = shareObjLabelInCh2[j];
  voRatio=ObjVolume_shared[j]/ObjVolume_ch2[objLabelInC2-1];
  print("volume overlap ratio of "+j+"th object in channel 2 is: "+voRatio+"
     = "+ObjVolume_shared[j]+"/"+ObjVolume_ch2[objLabelInC2-1]);

  //select the objects that have volume overlapping higher than a user
    specified ratio, ``volOverlap"
  if (voRatio>volOverlap){
    numColocObjects=numColocObjects+1;
    Ext.Manager3D_Select(j);
```

```
    Ext.Manager3D_FillStack(1,1,1);
15  }
}
```

:

where “numColocObjects" gives the total number of colocalization object pairs, “voRatio" computes the volume overlap ratio, and “volOverlap" is a user-specified threshold that discards objects with lower overlap ratio. `Manager3D_FillStack` fills the newly created image with the selected 3D region. For each region, the values filled can be all the same, or a different one as label. The final colocalized objects in each channel can be obtained using again the `3D Hysteresis Thresholding`, as we have done previously, but with the newly created overlapping regions image, “SharedMask Filter". In the synthetic example, the four candidate objects have volume overlap ratio of: 0.11, 0.51, 0.25, and 1 (see Fig. 4.5 *left*). And if we specify “volOverlap" to be, e.g. 0.2, 0.3, 0.52, the final “real" colocalization results differ, as shown in the three images on the right side, respectively, in Fig. 4.5.

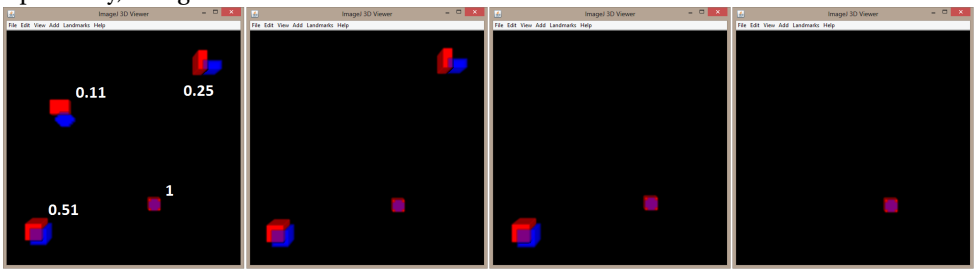


FIGURE 4.5: (*left*) Candidate colocalization objects from the two channels, the number next to each object pair shows the volume overlap ratio compared to the blue channel objects. (*middle-left - right*) Determined colocalization object pairs using specified ratio threshold of: 0.2, 0.3, and 0.52, respectively.

4.4.4 Visualizing results

To better examine 3D data, Fiji offers [`Plugins -> 3D Viewer`] to visualize rendered volumes, surfaces, or orthogonal slices. After opening the “3D Viewer" window, images can be loaded using either [`File -> Open`] (for any image on disk) or [`Add -> From Image`] (for images already loaded in Fiji). Multiple images can be loaded. For overlapping images, we could modify image's transparency by [`Edit -> Change transparency`]. Image brightness can be changed using [`Edit -> Transfer Function -> RGBA`]. There are many more possibilities to control and edit the visualization properties, we will leave this as a homework for you to exploit further. Examples of visualizing 3D data using this viewer can be found in many figures in this session such as Fig. 4.4, 4.5, 4.6.

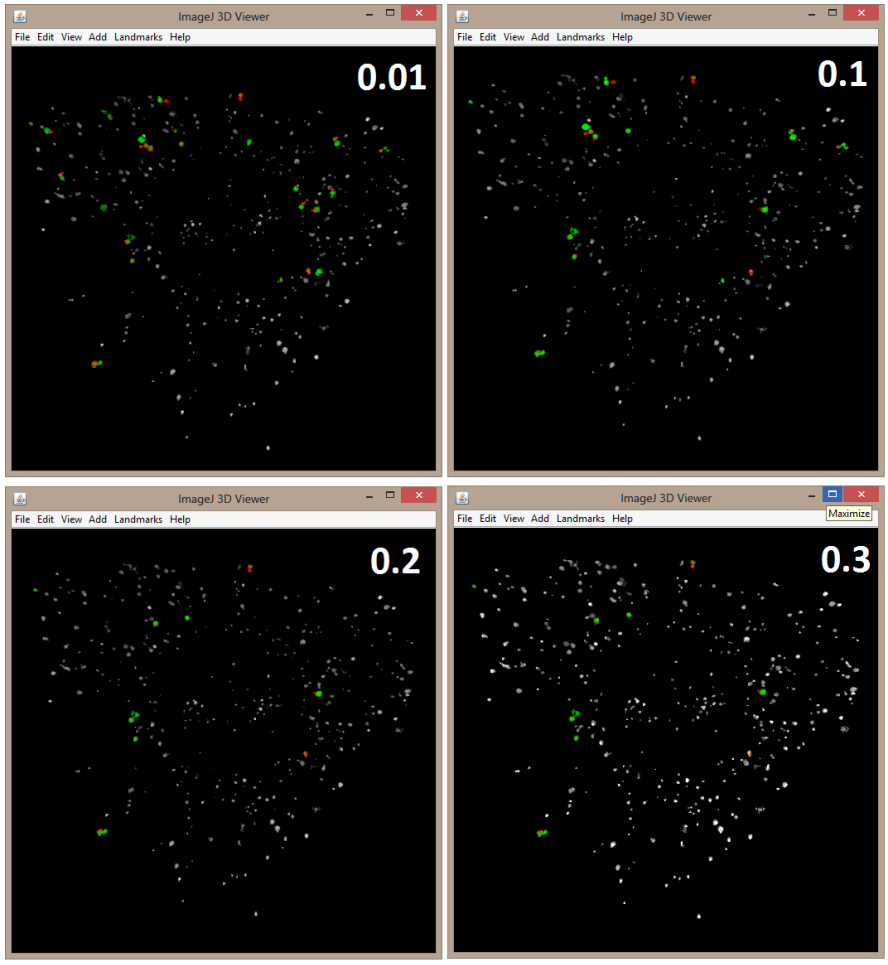


FIGURE 4.6: Colocalized objects in channel 1 (*red*) and channel 2 (*green*), together with all filtered objects in channel 1 (*white with transparency*) using the specified ratio thresholds: 0.01, 0.1, 0.2, and 0.3. Their corresponding number of determined colocalization objects are: 25, 14, 9, and 9, respectively.

4.4.5 Testing the macro on Hela cells with viral replication

The pipeline of operations we came up with can now be assembled to a complete macro to process the original Hela cells stacks. We only miss one step at the very beginning. Because the original Hela cells images are first segmented by ilastik. And we exported the predictions of both channels, which are not binary image since they are the probabilities of each voxel being the object of interest. Thus we would provide a threshold value to

binarize the images. Again, here the threshold value can be set differently for the channels. In order to make things general to work for images with different intensity levels, if we consider a threshold always between 0 and 1, then it can be scaled according to the image intensity range. To do this, the built-in function `getMinAndMax` can be used to get the original image intensity range, and then the threshold value can be calculated accordingly:

```
getMinAndMax(min,max);
setThreshold(min+(max-min+1)*thres,max);
run("Convert to Mask", "method=Default background=Dark black");
```

:

where “thres" is the threshold value between 0 and 1, and we assume the background has low intensity value in this case.

So now when we try to analyze the Hela cell images, the parameters used for the synthetic dataset may not be applicable anymore. Of course we could manually modify each value through the entire macro file we made. But this is not efficient. So it is better to use variables for these parameters, and specify them e.g. in the beginning of the macro code.

After the macro is “parameterized", we can specify values. For this dataset, let’s set the threshold to be 0.5 for both channels, and 3D object sizes to be [5, 500], and maximum 2D object sizes in XY plane are 150 (for channel 1) and 50 (for channel 2). And volume overlap ratio threshold can be any value between 0 and 1. Fig. 4.6 shows a few example results of the Hela cell images.

Please note that for a complete colocalization analysis, further examination steps are needed such as accuracy evaluation, robustness evaluation, and reliability evaluation like comparing to random events. These are out of the scope of this session thus not discussed here.

4.4.6 Setting up a parameter interface

You may find modifying parameters inside the source code of the macro not an elegant usage. If you are willing, constructing a simple user interface window for parameters is made very easy in Fiji. The `Dialog.*` functions offers a practical dialog box/window with just a few lines of code. An example is shown in Fig. 4.7, with the code on the left side and the generated dialog window the right side. Please note that the parameters that fetch the values from the dialog should be specified following the same top-down order as the dialog. Now we can create customized dialog window for the parameters that we would like user to specify.

In case you do not have time to write up the complete macro during the session, a possible solution is provided in your Session4 folder (`coloc.ijm`).

4.5 Tips and tools

- ImageJ Built-in Macro Functions: http://rsbweb.nih.gov/ij/developer/macro/functions.html

```
Dialog.create("Dialog Window Title");
Dialog.addNumber("Probability map threshold: ", 0.5);
Dialog.addMessage("");
Dialog.addNumber("Minimum 3D object size (in voxels): ", 5);
Dialog.addNumber("Volume overlap for colocalization: ", 0.2);
Dialog.addMessage(" ");
Dialog.addCheckbox("Display in 3D Viewer?", false);
Dialog.show();

thres_pm_ch1 = Dialog.getNumber();
min3DObjSize=Dialog.getNumber();
volOverlap = Dialog.getNumber();
Display3D = Dialog.getCheckbox();
```

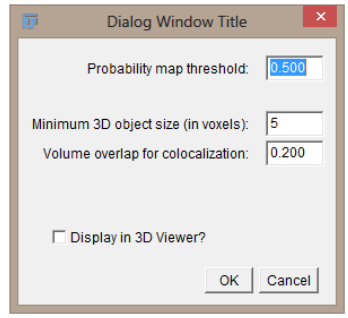


FIGURE 4.7: An example of creating a dialog window for parameters to be specified by users.

- 3D ImageJ Suite: download: http://imagejdocu.tudor.lu/lib/exe/fetch.php?media=plugin:stacks:3d_ij_suite:mcib3d-suite.zip and unzipping it in your plugins folder. Detailed description of the plugin: http://imagejdocu.tudor.lu/doku.php?id=plugin:stacks:3d_ij_suite:start.
- JACoP (includes object-based method): http://imagejdocu.tudor.lu/doku.php?id=plugin:analysis:jacop_2.0:just_another_colocalization_plugin:start

4.6 Groups close by which work on similar problems

- Dr. Holger Erfle's group, ViroQuant (http://www.bioquant.uni-heidelberg.de/technology-platforms/viroquant-cellnetworks-rnai-screening-facility/contact.html), located at Bioquant
- Dr. Karl Rohr's group, Biomedical Computer Vision Group (http://www.bioquant.uni-heidelberg.de/?id=322), located at Bioquant

4.7 References

[1] S Bolte and F P Cordelières. A guided tour into subcellular colocalization analysis in light microscopy. *Journal of Microscopy*, 224:213–232, 2006.

[2] B Obara, A Jabeen, N Fernandez, , and P P Laissue. A novel method for quantified, superresolved, three-dimensional colocalisation of isotropic, fluorescent particles. *Histochemistry and Cell Biology*, 139(3):391–402, 2013.

[3] S Wörz, P Sander, M PfannmÃűller, R J Rieker, S Joos, G Mechtersheimer, P Boukamp, P Lichter, and K Rohr. 3D geometry-based quantification of colocalizations in multichannel 3D microscopy images of human soft tissue tumors. *IEEE Transactions on Medical Imaging*, 29(8):1474–1484, 2010.

5

RELATED RESOURCES

To list just a few...

- Fiji online documentation: http://fiji.sc/Documentation
- ImageJ User Guide: http://rsbweb.nih.gov/ij/docs/guide/146.html
- ilastik online documentation: http://ilastik.github.io/documentation/index.html
- EuBIAS 2013 - BioImage Data Analysis Course: http://eubias2013.irbbarcelona.org/bias-2-course
- CMCI ImageJ Course page: http://cmci.embl.de/documents/ijcourses
- CMCI Macro programming in ImageJ Textbook: https://github.com/cmci/ij_textbook2/blob/master/CMCImacrocourse.pdf?raw=true
- http://www.matdat.life.ku.dk/ia/sessions/session12-4up.pdf
- Fluorescence image analysis introduction:http://blogs.qub.ac.uk/ccbg/fluorescence-image-analysis-intro/
- http://www.encite.org/html/img/pool/7_3D_image_reconstructions_p115-144.pdf
- BioImage Information Index: http://biii.info/
- ...

Notes:

Notes:

Notes:

Notes: