Tutorial: 3D Image Analysis Pipeline with FIJI, Ilastik, and Code::Blocks

This tutorial guides you through a comprehensive 3D image analysis pipeline using FIJI, Ilastik, and Code::Blocks. The pipeline includes preprocessing, classification of nuclei and spots, map reconstruction, and fusion of different 3D volumes.

Prerequisites

- FIJI (ImageJ) installed for image preprocessing and post-processing.
- Ilastik installed on your system for nuclei and spot classification.
- Code::Blocks IDE installed for map reconstruction.

1. Preprocessing with FIJI

Before starting the processing, several directories need to be created. For any given acquisition date, the following folders should be created:

```
acquisition_date (e.g., 180301)
-- cell
-- csv [CSV files obtained with Ilastik, named cell_*.csv]
-- codeblocks [results are saved here]
-- raw [pre-processed images]
-- spot
-- csv [CSV files obtained with Ilastik, named spot_*.csv]
```

*Note: * represents the number of timelapses, 1, 2, etc.*

1.1 Image Cropping

To reduce the size of images for analysis, images are cropped to 1200x1200 pixels. The macro "crop.ijm" in the `macro_fiji` folder is used for this purpose. After modifying the macro, it's best to save the file (Ctrl + S) before running it.

```
makeRectangle(x, y, 1200, 1200);
```

x and *y* correspond to the selection positions.

1.2 Bleaching Correction

For batch processing, it's beneficial to correct for bleaching to have a homogeneous background across all stacks. The macro for this is "crop_bleachcorrect.ijm".

Generally, both cropping and bleaching correction are done simultaneously.

```
dir_in=getDirectory("Choose_input_directory");
dir_out=getDirectory("Choose_output_directory");
list= getFileList(dir_in);
path = dir_in + list[0];
run("Image Sequence...", "open=&path file=Index sort"); // load images containing "Index"
run("Stack to Hyperstack...", "order=xyczt(default) channels=1 slices=150 frames=50
display=Color"); // slices: z number, frames: time number
makeRectangle(580, 522, 1024, 1024); // cropping area
run("Crop");
run("Bleach Correction", "correction=[Simple Ratio]");
for (i=0; i<list.length; i++) {
   Stack.setFrame(i+1)
   run("Reduce Dimensionality...", "slices keep");
   saveAs("Tiff", dir_out+list[i]);
   close();
}
run("Close All");
```

Three parameters need to be changed, which are already commented in the macro:

- *slices* - number in z, here 150
- *frames* - number of time lapses, here 50
- `makeRectangle(580, 522, 1024, 1024)` - cropping area

After this step, the image size is reduced by a factor of 3. The macro language can't distinguish between 1 and 01, so ensure acquisition names are in the correct order.

2. Nuclei Classification with Ilastik

All analyses are performed with version 1.2.2post1. Using the newer version will yield different CSV results, requiring modification of the "lire_csv" function in the C++ program (3d_texture). For instance, "Size in pixels" is the 8th column in the CSV table, so "sizeObject_spot" is 7 (row_spot[7], subtract 1) in the program.

```
sizeObject_spot[lineNumber_spot] = atof(row_spot[7].c_str());

bbMin_x_spot[lineNumber_spot] = atof(row_spot[8].c_str());

bbMin_y_spot[lineNumber_spot] = atof(row_spot[9].c_str());

bbMin_z_spot[lineNumber_spot] = atof(row_spot[10].c_str());

centerObject_x_spot[lineNumber_spot] = atof(row_spot[11].c_str());
```

```
centerObject_y_spot[lineNumber_spot] = atof(row_spot[12].c_str());

centerObject_z_spot[lineNumber_spot] = atof(row_spot[13].c_str());

bbMax_x_spot[lineNumber_spot] = atof(row_spot[14].c_str());

bbMax_y_spot[lineNumber_spot] = atof(row_spot[15].c_str());

bbMax_z_spot[lineNumber_spot] = atof(row_spot[16].c_str());

rayon_x_spot[lineNumber_spot] = atof(row_spot[31].c_str());

rayon_y_spot[lineNumber_spot] = atof(row_spot[32].c_str());

rayon_z_spot[lineNumber_spot] = atof(row_spot[33].c_str());

meanIntensity_spot[lineNumber_spot] = atof(row_spot[113].c_str());
```

Steps for using Ilastik:

1. Create a new project by clicking "Pixel Classification + Object Classification" and add a pre-processed image stack by clicking "Add New..." in the "Raw data" window. Images can be cropped with Fiji to accelerate computation.

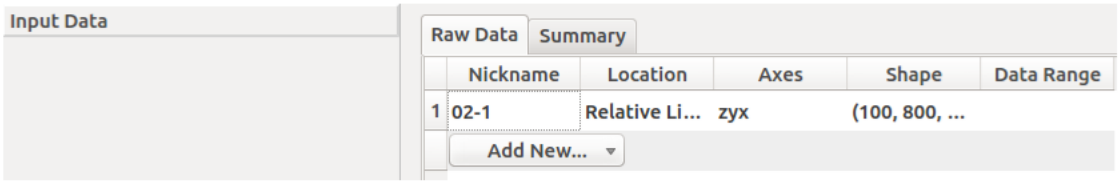

FIGURE 1 –

2. Often, the first stack (T0) is used for training nuclei, duplicating a 600x600x150 section. For spots, the first stack (T0) often has few spots. To improve training, use two time-lapses: one with many spots and one with few spots. Duplicate 75 slices from each stack (15 to 89) and merge them to create a new stack of 150 slices for Ilastik spot classification. Use "Image -> Stacks -> Tools -> Concatenate" for merging.
3. Select "Feature Selection". For nuclei, choose α = 3.5px, 5px, and 10px. For spots, choose α = 1px and 1.6px.

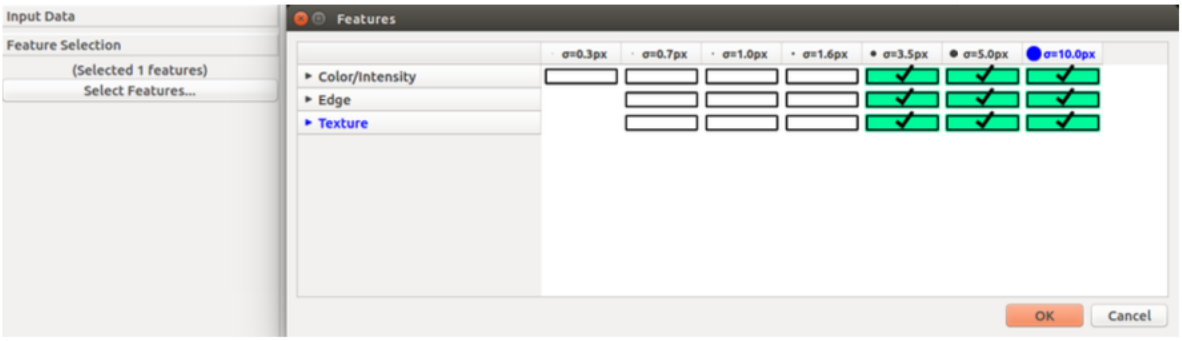

FIGURE 2 –

4. Begin "Training" with label 1: background, label 2: nuclei.

Ilastik Tips:

- To change slices, either modify the number directly or use the middle mouse wheel.
- For zooming in/out, press the left shift key and move the middle mouse wheel.
- To change contrast, right-click on the white area in the lower-left window.

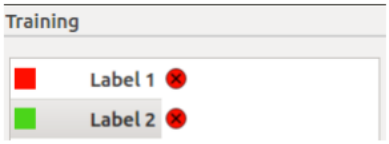

FIGURE 3 –

Manually labeled images are shown (Fig.4 (2)). To save a window preview, click the "camera" icon (Fig.5).

For 150 slices, label every 10 on the xy plane; labels can also be added on the xz and yz planes. The "Live update" function previews results after each label addition. The goal of training is to achieve the best possible result, which may take time depending on image quality.

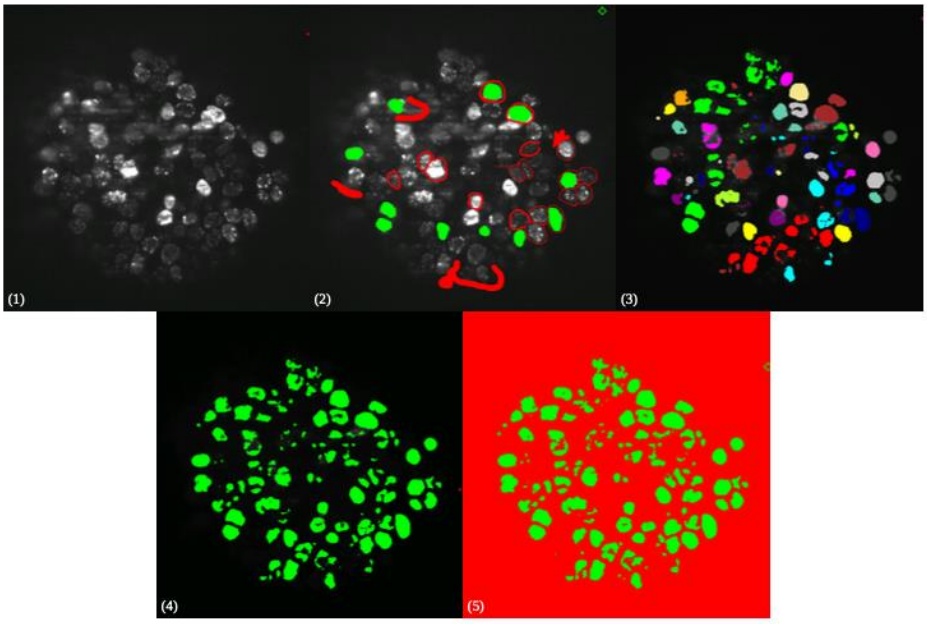

FIGURE 4 – (1) : raw images, (2) : label images, (3) : threshold, (4) : prédiction, (5) : segmentation

FIGURE 5 –

After "Training", move to "Thresholding". Adjust the threshold value of 0.6 slightly for better results (Fig.6). The "Thresholding" result is shown in Figure 4 (3).

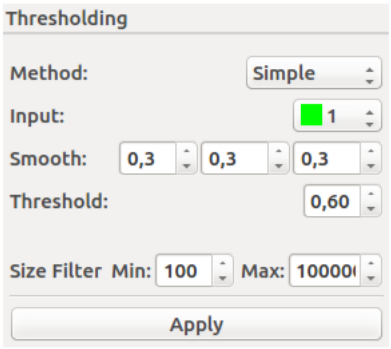

FIGURE 6 –

In "Object Feature Selection", select all features since it doesn't affect computation time. In "Object Classification", create two labels for normal software operation. A simple click with the object label on a nucleus classifies all nuclei.

The final step is exporting the result. The default format is h5, which is smaller and faster to export than tiff. To open h5 format with Fiji, install the "Ilastik HDF5" plugin. Once installed, click "File -> Import -> Ilastik HDF5".

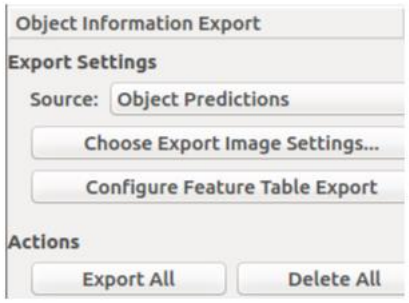

FIGURE 7 –

In "Choose Export Image Settings", modify the image type to 8 bits, 16 bits, etc., or leave the default configuration. In "Configure Feature Table Export", choose CSV (Fig. 8) format and select all features (Fig.9).

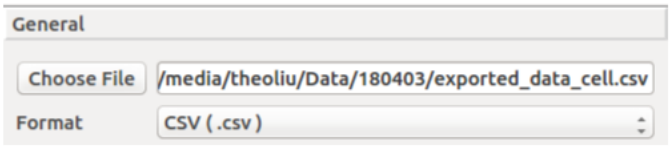

FIGURE 8 –

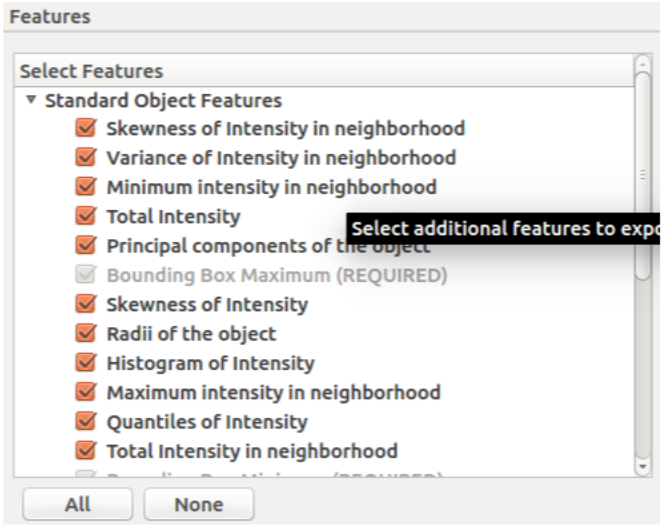

FIGURE 9 –

Save the Ilastik project (Ctrl + S) before starting "Batch Processing". After exporting, proceed with "Batch Processing". Click "Select Raw Data Files" to add images, then "Process all files". The computation time depends on image size and the number of labels used during "Training".

The exported CSV file is used for mapping in the next section, so place it in the "cell -> csv" folder and name it as cell_*.csv.

3. Spot Classification with Ilastik

The process for spot classification is the same as for nuclei. Select α = 1.0px and 1.6px during feature selection. The results of each step are shown in Figure 10.

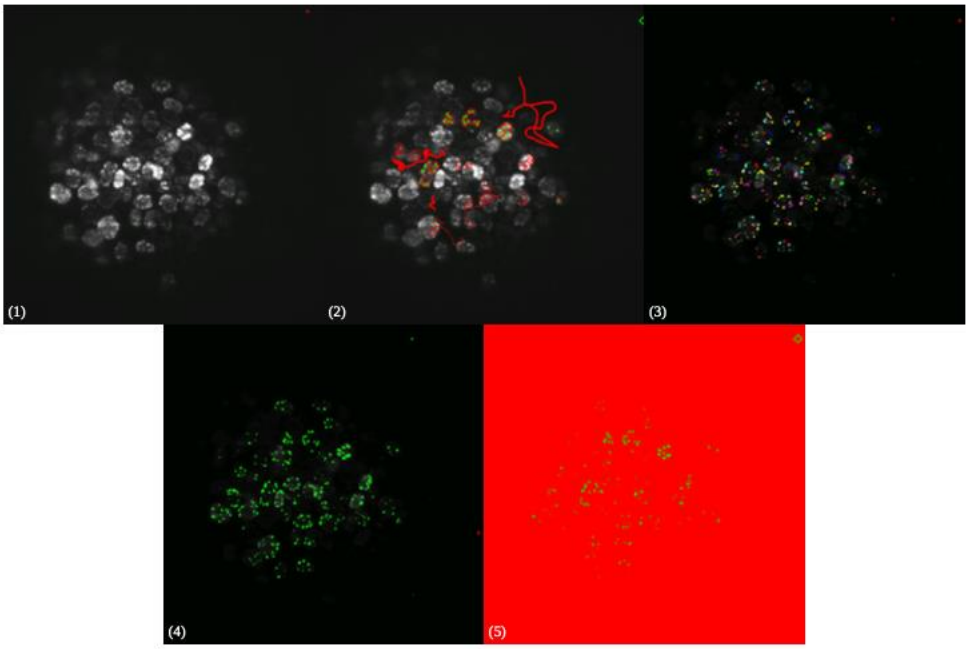

FIGURE 10 –

Another difference is the thresholding. Adjust the threshold value of 0.2 slightly for better results.

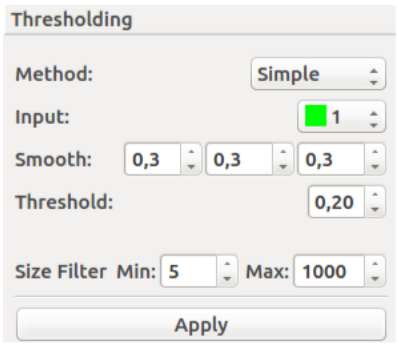

FIGURE 11 –

The exported CSV file is used for mapping in the next section, so place it in the "spot -> csv" folder and name it as spot_*.csv.

4. Data Processing with Code::Blocks

After object classification with Ilastik, you typically get two types of files: `cell_i.csv` and `spot_i.csv`. Each .csv file contains the position (Cx, Cy, Cz) of 3D objects and their corresponding radius (Rx, Ry, Rz).

The mapping program is named "3d_texture". To open it with Code::Blocks, click "File -> Open... -> 3d_texture.cbp". By default, results are saved in the H: partition. If you need to change the save path, search for "H:" and replace it with another partition, e.g., "F:".

The main function for reconstruction is `lire_csv`, which loads .csv files and reads the necessary variables to generate the map.

```
void lire_csv(std::string userName, std::string acquiDate, int frame, Var3D N3D);
```

Parameters:

- `userName` - folder name in the H: partition
- `acquiDate` - acquisition date
- `frame` - number of time lapses
- `N3D` - dimensions (x, y, z) of the images

Once you have the two types of .csv files, you can quickly calculate the number of nuclei and spots segmented with Ilastik by commenting out most of the code in the `lire_csv` function. Save the number in `nombre_cell.csv` and `nombre_spot.csv` files under the `codeblocks` directory.

two main functions can be achieved with the C/C++ program:

1. Generate the mapping of nuclei and spots, testing if a spot belongs to a nucleus, presenting the mapping of nuclei based on the number of spots in each nucleus, giving different sizes and pixel intensities for each nucleus.
2. Calculate the spheroid center, then the Euclidean distance of each nucleus from the center, and classify the number of spots under different layers (10 layers in the program), each representing a distance of 0.1R from the spheroid center.

By default, the mapping of nuclei containing the number of spots per nucleus is generated (the larger the size, the more spots; pixel intensity indicates the exact number of spots).

5. Merging Different 3D Volumes

The macro "fusion_volumes3D.ijm" merges different analyses of the same experiment. It can merge four experiments, but slight modifications may be needed for more or fewer experiments.

First, open several stacks and use `imageCalculator` to merge them. First, open an image stack:

```
run("Raw...", "open=&pathIn1 image=[16-bit Unsigned] width=1200 height=1200 number=150 little-endian");
```

Ensure image stacks are of the same size (e.g., 1200) to avoid errors during merging. Add a size change to 1200 if needed:

```
if(imageSize1!=1200){
   run("Size...", "width=1200 height=1200 depth=150 constrain average interpolation=Bilinear");
}
```

6. Result Management

The result is generated in two folders:

- `codeblocks`: contains the generated map (`cellSfr_*.tif`), number of nuclei (`nombre_cell.csv`), and number of spots (`nombre_spot.csv`).