Detection of Nucleoli Using ImageJ

Nico Hochberger

November 2014

Contents

List of Figures

List of Tables

1 Introduction

1.1 Preface

1.2 Objective

1.3 Motivation

Currently nucleoli are detected using an application called CellProfiler[1]. While this application yields reliable results, it also takes pretty long to complete the analysis. Runtimes up to 45 seconds are common.

Due to the fact that CellProfiler is a very general approach, applicable to a large variety of tasks related to detecting nuclei and nucleoli, the results of its analysis have to be checked manually to reduce the amount of false-positives.

- TODO -

Consequently, this leads to a more specialized way of analyzing the cells, which does not do all the analysis performed by CellProfiler, but on the other hand is much faster and thus helps to prevent cells dying before the analysis is finished.

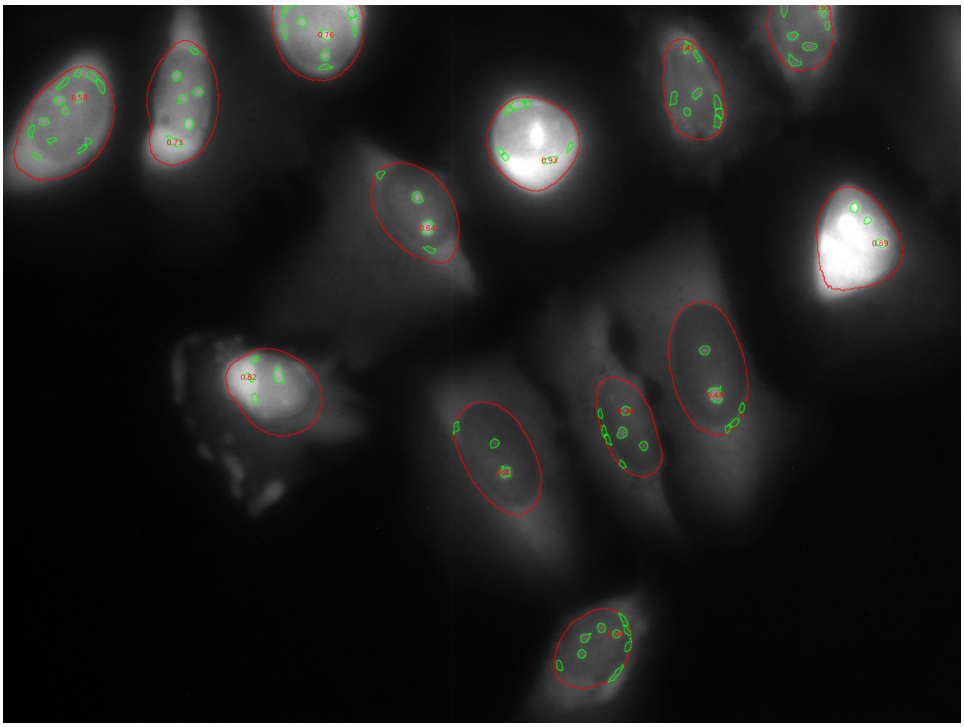

Figure 1: Example analysis performed with CellProfiler

[1] http://www.cellprofiler.org

2 Design and Implementation

2.1 Requirements

The application has to meet the following requirements:

- **Reliable nucleoli detection:** Stable and reliable detection of nucleoli is the main purpose of the application. Hence, it is supposed to detect at least 75% of the nucleoli CellProfiler can detect. This includes a certain degree of stability concerning fuzzy pictures or pictures with inequally distributed or diffuse brightnes.

- **Fast analysis:** As the application is tailored to this single task, it is expected to detect a suitable amount of nucleoli in only a small fraction of the time required by CellProfiler. The topmost time that the application may require to complete the analysis of one image is five seconds.

- **Fallback in case of empty nuclei:** Each nucleus is expected to containg at least one nucleolus. Yet, this expectation cannot always be fulfilled due to potentially damaged nuclei, fuzzy images, or other reasons. In this case, the center of the nucleus has to be provided as fallback target.

- **Visualizer:** In order to quickly check the results directly after running the analysis and to provide a way to quickly present the results to a potential audience, the application has to provide the possibility to be configured so that it shows the results as an image. This image has to contain all detected neuleoli targets, fallback targets and the regions of interest, e.g. the nuclei.

- **Versatility:** Since the appearance of different specimen can vary in various ways, all analysis parameters have to be configurable. Among others, this includes the minimum and maximum sizes of nuclei and nucleoli. The configuration is supposed to be achieved via an understandable, interchangeable, text-based file[2].

- **Statistics:** To determine the most suitable parameters for different kinds of specimen, another feature may be configured. This statistics feature has to include:

  - The amount of detected nuclei
  - The amount of detected nucleoli
  - Nuclei to nucleoli ratio as percentage

[2]Configurable parameters are explained in detail in the User's Manual section

  - The distance of each detected nucleolus to the center point of the containing nuclei and the average distance in pixels
  - The area of each detected nucleus and the average area in square pixels
  - The area of each detected nucleolus and the average area in square pixels

- **Serialization of the results:** All results have to be stored in their accordant files in a subfolder *results* of the folder containing the original data. The name of each result file has to contain a the timestamp indicating the application's execution in the format `<year>_<month>_<day>_<hour>_<minute>_<second>`.
  E.g. `targets_2014_11_20_14_50_35.txt`.

  In the following, the accordant formats and files are described.

  - **Targets:** Real targets and fallback targets sre to be saved in one txt-file named `targets_<timestamp>.txt` in the following format:

```
# nucleoli targets
<target number> : [<x-coord>, <y-coord>]
...
# targets in center of empty nuclei
<target number> : [<x-coord>, <y-coord>]
...
```

Listing 1: Format of results txt-file

Example:

```
# nucleoli targets
1 : [468, 43]
2 : [1183, 14]
# targets in center of empty nuclei
3 : [87, 174]
4 : [769, 198]
```

Listing 2: Example of results file

  - **Statistics:** The statistics as mentioned above have to be stored in a txt-file named `statistics_<timestamp>.txt`. An example of the statistics file can be found in the appendix.
  - **Result image:** The image as it would be displayed by the visualizer has to be stored to a file named
    `targets_<timestamp>.<original image filetype>`. See Figure 2 for an example the image.

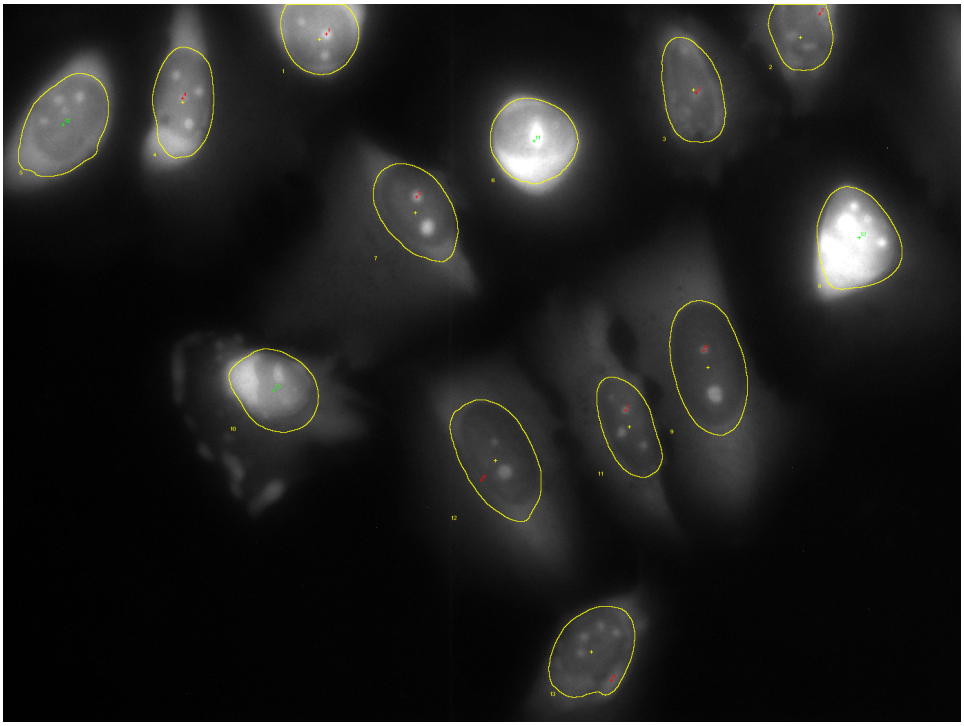

Figure 2: Example of the image containing the results

- **Quick and easy deployment:** The application is supposed to be deployed and run as easily as possible. Since the environment and the machines this will happen on may vary drastically, a versatile and independent way to do this is required. The Java framework delivers this possibility. Hence, the application will be developed using Java.

3 User's Manual

3.1 System Requirements

The application requires the Java Runtime Environment 1.7 or higher. For Microsoft Windows 64bit operating system this results in the following minimum system requirements [3]:

Memory:	128MB
Processor:	Pentium 2 @ 266MHz
Disc space:	124 MB + 2 MB

Table 1: Minimum system requirements of Java 1.7 on Windows 64bit

As it is good practice and vital to the security of any system, the Java framework should be updated on a regular basis. Note that the requirements may change with any update.

3.2 Starting the Application

Starting the application is acoomplished using the command line interface via Java's `java -jar` or `javaw -jar` command[4]. Furthermore, the application expects the configuration file to be passed as single parameter.

Considering a scenario in which the configuration file (see 3.3) is stored as `configuration.txt` in the same folder as the `NuFi.jar`, the following command will properly start the application:
`java -jar NuFi.jar configuration.txt`.

3.3 Configuration

Configuration of the application is performed using a simple text file containing the information listed in the following. Note that a missing parameter will result in a fatal error, preventing the application from properly working.

A complete example of the configuration file can be found in the appendix.

3.3.1 General Parameters

This section contains information on filetypes, source folders and naming convention of source files.

[3] Further details on the system requirements can be found at: https://docs.oracle.com/javase/7/docs/webnotes/install/

[4] See https://docs.oracle.com/javase/7/docs/technotes/tools/windows/java.html for further details.

- **Source folder:** The folder containing the source files (e.g. image files) on which the analysis will be based. The source folders are expected to be provided as absolut path or relative to `NuFi.jar`.

  Note that the application expects exactly three files of the type defined as channel filetype. If there are more or less than three files of that type, a fatal error will occur.

  Key:
  `source.folder`

  Examples:
  `source.folder = C:/example/images`
  `source.folder = ../../example/images`
  `source.folder = example/images`

- **Used channels:** Each specimen is photographed three times using different colorization. This results in three different images referred to as different channels. Channels 1 and 3 are used in the process of detecting targets, thus, the correct specification and order of the files is vital for image analysis. Position one determines channel 1, position 3 determines channel 3. The source folder is scanned for files containing these channels determination. If these are not found, the application will terminate with a fatal error.

  Key:
  `used.channels`

  Examples:
  `used.channels = Kanal1, Kanal2, Kanal3`
  `used.channels = channel1, channel2, channel3`

- **Channel filetype:** Though using the png-file format is advised, the application is built to be able to analyze various image formats. If it is necessary to analyze images that are not provided in the png-format, the accordant format can be provided using this key.

  Key:
  `channel.filetype`

  Examples:
  `channel.filetype = png`
  `channel.filetype = jpg`

3.4 General Picture Analysis Settings

This section contains information of the expected size and shape of nuclei and nucleoli, as well as the width of the in depth-analysis.

- **Size settings:** Since the size of the nuclei and nucleoli in the specimen may vary and since the magnification of the microscope used in taking the pictures may also change, the size in the photographs may need to be adapted. This can be achieved by giving an average size in sqaure-pixels and the minimum and maximum sizes as multiples of that size.

  Keys:
  `nucleus.average`
  `nucleus.min.factor`
  `nucleus.max.factor`
  `nucleolus.average`
  `nucleolus.min.factor`
  `nucleolus.max.factor`

  Examples:
  `nucleus.average = 10000`
  `nucleus.min.factor = 0.5`
  `nucleus.max.factor = 1.5`
  `nucleolus.average = 100`
  `nucleolus.min.factor = 0.75`
  `nucleolus.max.factor = 1.8`

- **Minimum circularity:** This parameter pays respect to the variation of the objects in their shape. A value of 1 indicates a perfect circle, while a 0 indicates that there are no requirements to the shape of the object. Since nuclei typically are elliptical in shape, a value of approximatly 0.6 is advised. Nucleoli are typically close to a perfect circle. Thus, a value of 0.9 includes most of them while it excludes a variety of anomalies and prevents false-positives on nuclei borders.

  Keys:
  `nucleus.min.circularity`
  `nucleolus.min.circularity`

  Examples:
  `nucleus.min.circularity = 0.6`
  `nucleolus.min.circularity = 0.9`

- **In-depth width:** This parameter defines the variation in thresholding used during in-depth anylsis.

  Key:
  `indepth.range`

  Examples:
  `indepth.range = 25`
  `indepth.range = 10`

3.4.1 Improved Image Detection Parameters

In the standard setting the application will use the improved image detection algorithm. In this section, the parameters required for this algorithm can be defined. These include setting for brightness correction (`<object>.background.blur` and `<object>.thresholding.blur`) and the value used for correcting discrepancies between channel 3 colorization and the acutal size of nuclei (`nucleus.boundary.width`).

Keys:

`nucleus.background.blur`

`nucleus.thresholding.blur`

`nucleus.boundary.width`

`nucleolus.background.blur`

`nucleolus.thresholding.blur`

Examples:

```
nucleus.background.blur = 100
nucleus.thresholding.blur = 3
nucleus.boundary.width = 5
nucleolus.background.blur = 10
nucleolus.thresholding.blur = 1
```

3.5 Structure of Files and Folders

4 Conclusion and Prospect

4.1 Conclusion

4.2 Prospect

References