Implementation and Evaluation of Otsu's Thresholding

Elizaveta Chernova, Veronika Schuler, Laura Wächter, Hannah Winter
Supervisor: PD Dr. Karl Rohr
19.07.2021

Abstract

Thresholding is a useful method that is frequently used in the context of image segmentation. In this project, we used Otsu's thresholding algorithm in order to find the optimal threshold value, that optimizes image segmentation. The algorithm was applied to a number of images from different datasets (N2DH-GOWT1, N2DL-HeLa, NIH3T3). In order to improve the results, several preprocessing methods (mainly filters) were used. The final segmentations were compared to reference images and evaluated with several methods (Dice Score, MSD, Hausdorff Distance). The different datasets are characterized by different features, like reflections or low contrast. For this reason, we hypothesized that for the seperate datasets different preprocessing methods would lead to the optimal segmentation result, which was confirmed by our overall analysis.

Table of Contents

1. Introduction

In computer vision, *image segmentation* is used for a variety of purposes. In general, image segmentation describes the process of dividing a digital image into multiple segments to simplify the image. These simplified images can be used for further analysis. In the field of medicine or biology, image segmentation is often used to locate tumors (Liu et al., 2018) and other pathologies or to measure tissue volume (Lee, Im, Solaiyappan and Cho, 2017). Segmentations of images displaying cell nuclei are often used as an input for algorithms that measure the number or size of the cell nuclei. If time-lapse images are used, it is also possible to track the cell movement.

Thresholding is a simple image segmentation method. Based on a single intensity value, the image is divided into two segments: Pixels with higher intensities than the threshold value are distinguished from the pixels with intensities lower than the threshold. In order to find the threshold value that results in the best possible image segmentation we used 'Otsu's Thresholding' algorithm (Otsu, 1979).

In the course of the project, three different datasets have been used, each of which has different challenges to it: Low contrast, noise and reflections make it difficult to distinguish the cells from the background. In order to achieve good results, despite these difficulties, different preprocessing methods were applied to the images prior to the image segmentation. Due to the differences between the datasets we hypothesized that different preprocessing methods would result in the optimal segmentation for every dataset.

The accuracy of the obtained segmented images was evaluated by using the Dice Score as well as the Median Surface Distance (MSD) and the Hausdorff Distance. These metrics compare the obtained segmented image to a reference image. Beyond that, a cell counting algorithm was implemented, which is one possible way to use segmented images.

2. Dataset description

2.1 N2DH-GOWT1 cells
The dataset N2DH-GOWT1 of the cell tracking challenge (Bártová et al., 2011) contains images of GFP-GOWT1 mouse embryonic stem cells that were captured using time-lapse confocal microscopy (Leica TCS SP5 microscope). The varying brightness of the cells makes it difficult to distinguish the cells from the background. Further, low contrast and the noise in the images present challenges to the segmentation algorithm.

2.2 N2HL-HeLa cells
The dataset N2DL-HeLa of the cell tracking challenge (Neumann et al., 2010) contains images of human epithelial cells of cervical cancer. Those images were captured with an Olympus IX81 microscope used for live imaging of fluorescently labelled chromosomes. The challenge of these images is the variable brightness of the cells.

2.3 NIH3T3 cells
The dataset NIH3T3 (Coelho, Shariff and Murphy, 2009) contains images of several mouse embryonic fibroblast cells. These images were also captured using fluorescence microscopy. The difficulty when segmenting these images lies mainly in certain bright spots (probably reflections).

In [1]:
```
# Import of the Modules
from nuclei_segmentation import metrics, complete_analysis, visualisation,
from skimage import io
from matplotlib import pyplot as plt
from skimage.filters import threshold_otsu
import numpy as np
import pathlib as pl
import json
import pandas as pd
import warnings
import matplotlib as mpl
mpl.rcParams['figure.dpi'] = 120
warnings.filterwarnings('ignore')
```

3. Methods

The following preprocessing methods have been tested with the intention to improve the results of Otsu's thresholding algorithm. They help to reduce noise and other disruptive factors in the input images.

3.1 Preprocessing

Gaussian filter
The Gaussian filter blurs images and therefore reduces noise. In the optimal case, this filter helps to reduce reflections in the images that might disturb the thresholding algorithm. The Gaussian kernel, that is used to compute the weighted average of the pixels, has the shape of the 2D zero-mean Gaussian function G_σ with standard deviation σ. The higher σ is set, the stronger is the blurring effect in the processed image.

$$G_\sigma(x, y) = \frac{1}{\sigma^2 2\pi} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

Median filter
The Median filter is a non linear filter, that is often used to reduce noise in images. The gray-values of all pixels in a pixel neighborhood of a defined size are ranked by size. The intensity value of the central pixel is then substituted with the median of its neighbourhood. This filter is particularly useful to reduce salt and pepper noise in images. Compared to the Gaussian filter, a useful feature of the Median filter is that edges are preserved, while the overall noise is reduced.

Histogram stretching
Histogram stretching is a technique that aims to enhance the contrast of an image and might therefore help to optimize the segmentation results. The range of intensity values is 'stretched' to a desired minimal and maximal intensity value. The first step is to take all intensity values in the image and to select the minimal intensity value c and the maximal intensity value d of the input image as the $2nd$ and $98th$ percent quantiles. The lower and upper pixel intensity limits a and b to which the image is stretched are often times the minimum and the maximum pixel values of the respective image type. Hence, for 8 bit gray-level images a is set to 0 whereas b is set to 255. These values must be adjusted accordingly for other image types. The pixel intensity of the output

image P_{out} is calculated from the pixel intensity P_{in} of the input image based on the following linear function.

$$P_{out} = (P_{in} - c)\frac{(b-a)}{(d-c)} + a$$

3.2 Thresholding

Otsu's Thresholding
In order to adequately separate objects from the background, it is important to select an optimal threshold value. In the optimal case, the image histogram shows a bimodal distribution. In this scenario, the threshold could simply be selected at the bottom of the valley between the two intensity peaks representing foreground and background. However, in most cases it is not obvious where the optimal threshold value lies. In this case, Otsu's Thresholding method allows us to compute the best threshold value automatically. The algorithm computes the in-between-class variance σ_B for all possible threshold values. Every threshold value separates all pixels in the image in two classes: Pixels with a lower intensity and pixels with a higher intensity than the threshold value. The between-class variance σ_B is calculated from the probability of class occurrence ω, the mean intensity values μ of both pixel classes as well as the total intensity value variance μ_T. The intensity level that maximizes the in-between-class variance is selected as the optimal threshold value.

$$\sigma_B^2 = \omega_1(\mu_1 - \mu_T)^2 + \omega_2(\mu_2 - \mu_T)^2$$

In order to receive a segmented, binary image, the pixel intensity $g(x, y)$ has to be changed accordingly. Pixels with a lower or equal intensity value than the optimal threshold k are set to 0. Analogous to this, all pixels with a higher intensity value than the threshold are set to 1.

$$g_{segmentation}(x, y) = \begin{cases} 0 & if \quad g(x,y) \le k \\ 1 & if \quad g(x,y) > k \end{cases}$$

Two-level Otsu's thresholding
Otsu's Thresholding method can be easily extended to multi-thresholding problems. We chose it with the intention of trying to reduce the bright reflection spots in the NIH3T3 dataset, that might otherwise be recognized as cell nuclei by the algorithm. The main idea is to set two threshold values: The higher threshold value separates the reflections from the remaining image, whereas the lower threshold value separates the cell nuclei from the background. The computation of two threshold values results in three different pixel classes. Analogous to the one-level algorithm, the optimal threshold values are computed by minimizing the in-between-class variance between the classes:

$$\sigma_B^2 = \omega_1(\mu_1 - \mu_T)^2 + \omega_2(\mu_2 - \mu_T)^2 + \omega_3(\mu_3 - \mu_T)^2$$

In our case the next step was to set all pixels with a higher intensity than the upper threshold value (ideally the reflections in the image) to 0. All pixels with a lower or equal intensity than the lower threshold value were also set to 0. The remaining pixels were set to 1.

3.3 Evaluation

In order to evaluate the result of the image segmentation based on Otsu's Thresholding algorithm, the segmented images were compared with reference images. For every picture in the dataset, there is a binary Ground truth image, that displays an accurate segmentation of the respective image. The correspondence of our final segmentation and the Ground truth image can be evaluated by using various metrics.

Dice Score
The Dice score (DSC) is often used in order to assess the performance of image segmentation algorithms. The Dice score is a measure of similarity between the segmented image and the Ground truth. It is based on the number of correctly assigned pixels (TP) pixels as compared to the number of pixels wrongly set to itensity 1 (FP) and the pixels that were falsely set to 0 (FN) (Eelbode et al., 2020).

$$DSC = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

The Dice score lies within the interval $[0, 1]$. In the event of a complete match between the segmented image and the Ground truth image, the obtained Dice score is 1. In the opposite case, the obtained Dice Score is 0. In general, higher Dice scores indicate a more precise image segmentation.

Surface Distances
Surface distances are another way to evaluate an image segmentation method by comparing the segmented image with the Ground truth image. The distance D between a point s in the set of pixels S (all pixels with intensity 1 in the segmented image) to the set of pixels G (all pixels with intensity 1 in the Ground truth image) corresponds to the minimal euclidean distance between s and all pixels g in G:

$$D(s,G) = min\{d(s-g)|g \in G\}$$

This distance is computed for every pixel in both sets S and G, whereby the total distances $D(S,G)$ and $D(G,S)$ (two distance vectors) between the pixel sets are obtained. The Mean surface distance (MSD), as the name implies, is the average surface distance between the two pixel sets by adding up all distances and dividing by the total number of pixels n_S and n_G in both pixel sets. For smaller MSD values the segmentation is more accurate (Rhee et al., 2020).

$$MSD = \frac{1}{n_S + n_G}\left(\sum_{p=1}^{n_S} D(s,G) + \sum_{g=1}^{n_G} D(g,S)\right)$$

By contrast, the Hausdorff distance (HD), named after the German mathematician Felix Hausdorff, is the maximal surface distance between the two pixel sets. For smaller HD values, the segmentation is more accurate.

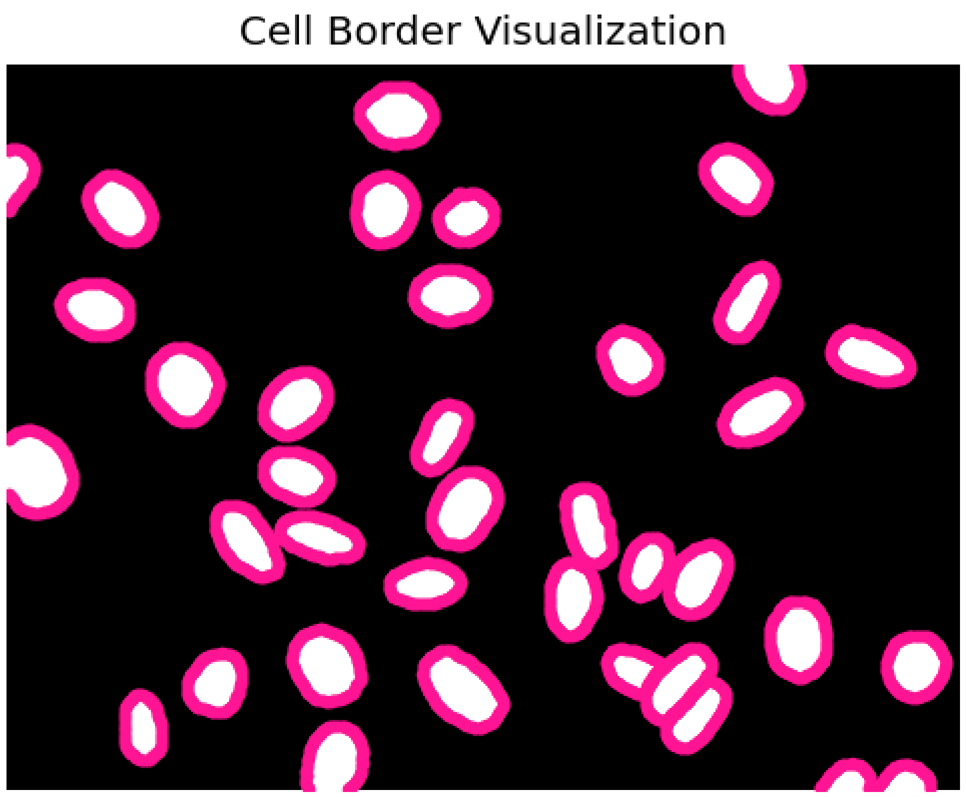
$$HD = max\{D(S,G), D(G,S)\}$$

In general, the Hausdorff distance is very sensitive to outliers and therefore not recommended if outliers are likely (Taha and Hanbury, 2015).

3.4 Cell counting

The basic idea of our cell counting algorithm is to identify all pixels that form the outline of the segmented cells. These border pixels are located by identifying intensity changes from zero to one in the segmented image. By grouping adjacent border pixels, all pixels belonging to the same shape are combined into groups. The count of the obtained groups matches the count of cells in the image. The picture below visualizes the located border pixels.

In [3]:
```
cell_counting_example = io.imread(str(pl.Path('./Data/NIH3T3/gt/42.png')))
border_pixels = metrics.find_border(cell_counting_example)
visualisation.border_image(cell_counting_example, border_pixels)
```



4. Outcomes

Five different preprocessing methods were applied to the images: Histogram stretching, the gaussian and median filter individually as well as two combinations (gaussian filter and histogram stretching as well as median filter and histogram stretching). The results are precalculated and saved in .json files. If desired, they can be recalculated in a few hours.

In [4]:
```
# Set to True if one level recalculations are desired (requires a few hours
data_one_level = complete_analysis.recalculation_desired_one_lvl(recalculat
                                                                 path_to_da
# Set to True if two level recalculations are desired (requires a few hours
data_two_level = complete_analysis.recalculation_desired_two_lvl(recalculat
                                                                 path_to_da
```

The optimal preprocessing method for each dataset was determined based on the mean Dice Scores as well as the mean surface distances of every dataset. In the following, the results of each dataset are discussed individually. Thereby, also the distribution of the evaluation measure values is taken into account.

4.1 N2DH-GOWT1 Dataset

4.1.1 N2DH-GOWT1 Results

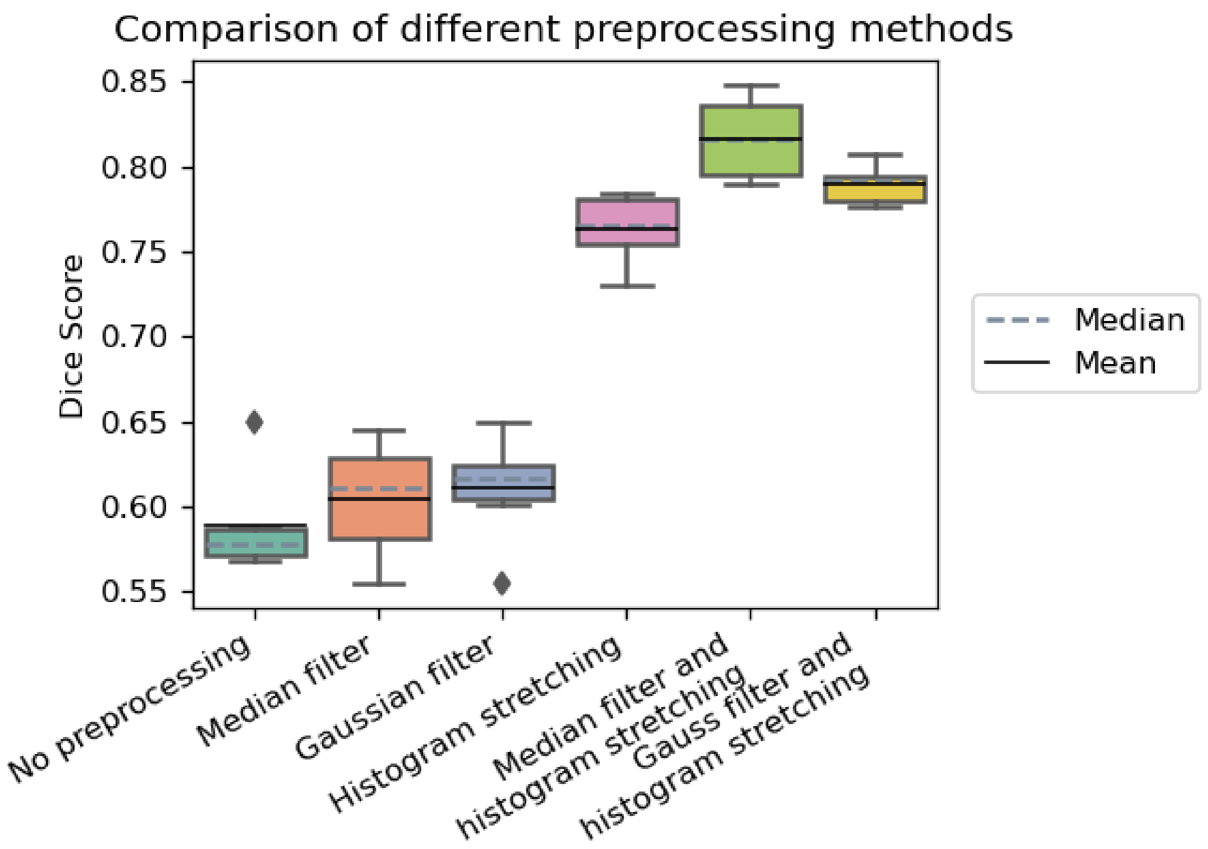
Based on the mean values of the evaluation measures, the following optimal preprocessing methods were obtained:

In [5]:
```
complete_analysis.result_evaluation(str(pl.Path("Results/values.json")), ["
```

```
N2DH-GOWT1 (dice) : 0.817   --->   Median filter and histogram stretching
N2DH-GOWT1 (msd) : 3.389   --->   Histogram stretching
N2DH-GOWT1 (hd) : 221.375   --->   Median filter and histogram stretching
```

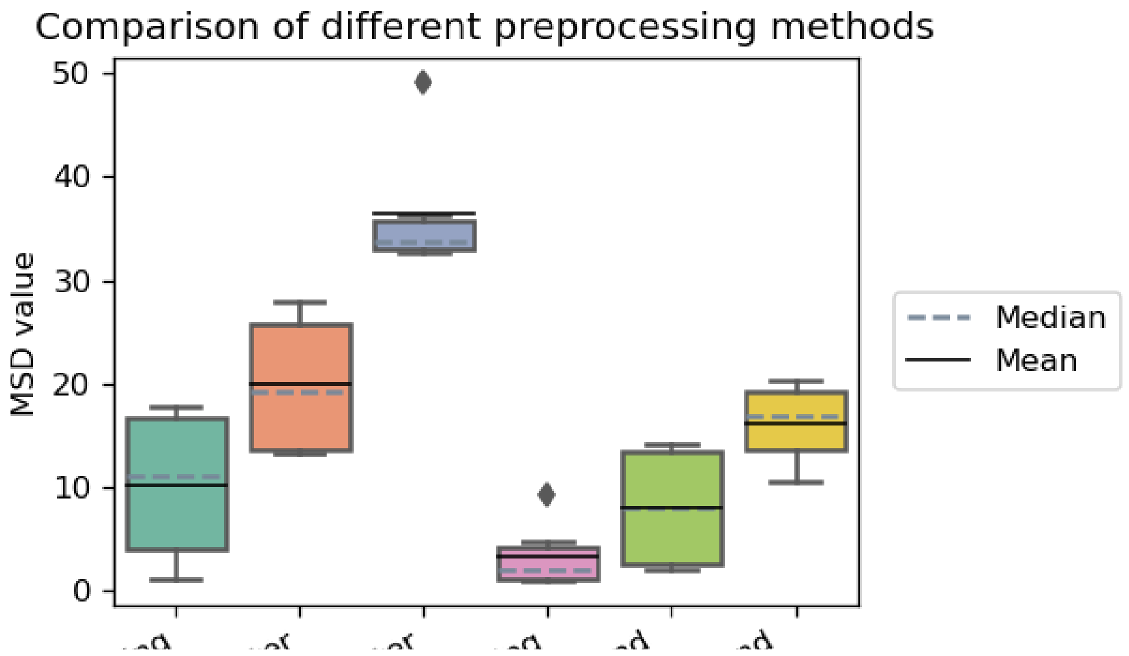
The box plot below visualizes the distribution of the Dice scores for every dataset. A general trend is that preprocessing methods including histogram stretching result in a better segmentation than other preprocessing methods. Filters alone do not seem to have a grat impact on the result.

In [6]:
```
visualisation.comparison_boxplot(complete_analysis.get_one_lvl_dice_scores(
```



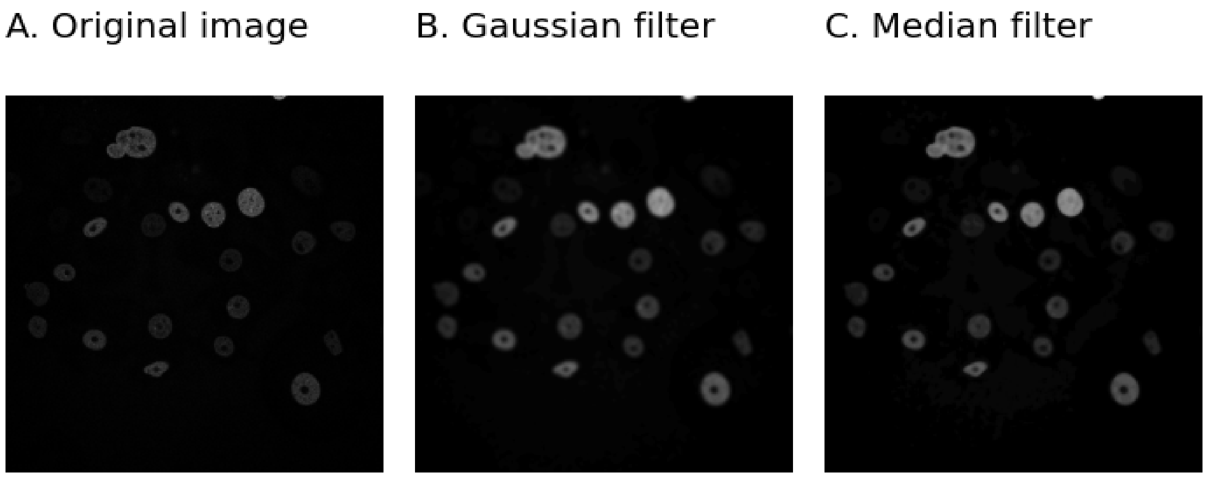
Based on the Mean Surface distances, the methods containing histogram stretching also prove to be the best. However, the difference is not as obvious as in the Dice score distribution. Further, images without preprocessing are rated very high in contrast to the evaluation based on the Dice Score.

In [7]:
```
visualisation.comparison_boxplot(complete_analysis.get_one_lvl_msd(data_one
                                                  y_label = 'MSD value')
```



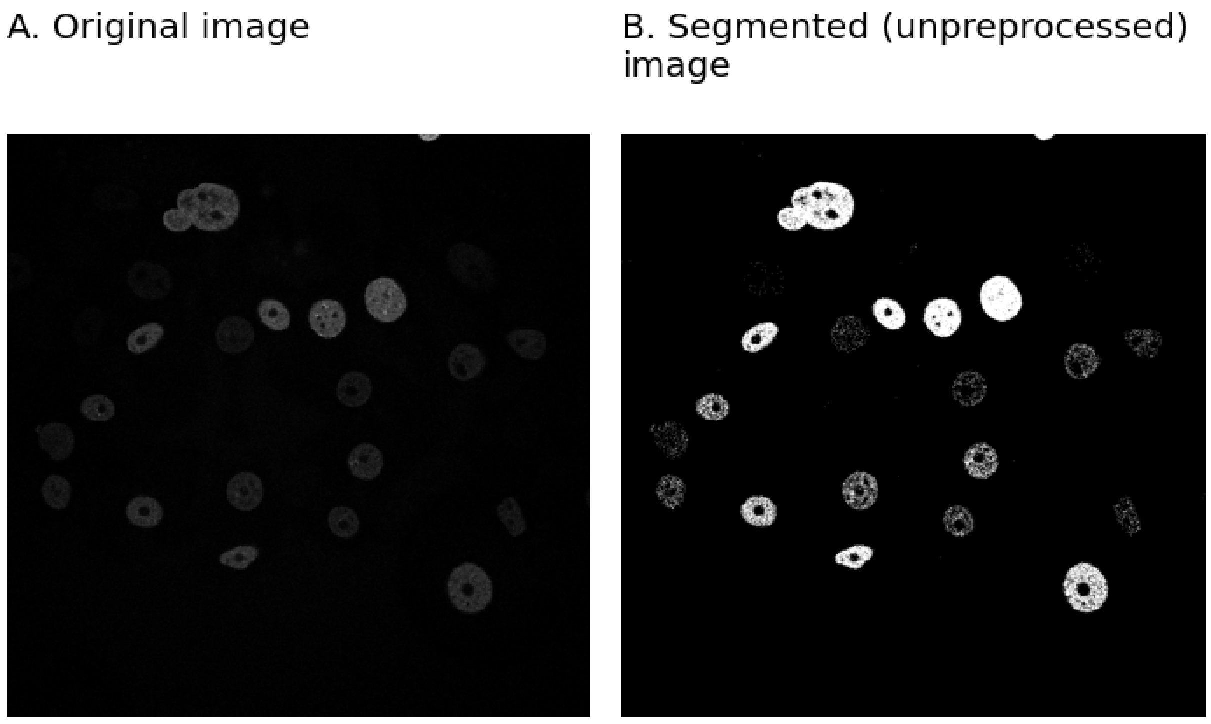
4.1.2 N2DH-GOWT1 Discussion
All images in the first dataset are very low in contrast. Even with the naked eye, some cells are hard to recognize. As the main effect of filters is to blur images, they are not helpful to improve the segmentation result in this context. This explains the minor effect of filters on the mean Dice score. The following figure shows that the filters do not have an obvious effect on the overall appearance of the images.

In [8]:
```
image = io.imread(str(pl.Path('Data/N2DH-GOWT1/img/t01.tif')))
kernel = preprocessing.gaussian_kernel(21, 5)
gaussian_image = preprocessing.convolution(image, kernel)
median_image = preprocessing.median_filter(image,15)
visualisation.three_img_plot(image,gaussian_image, median_image,"A. Origina
```

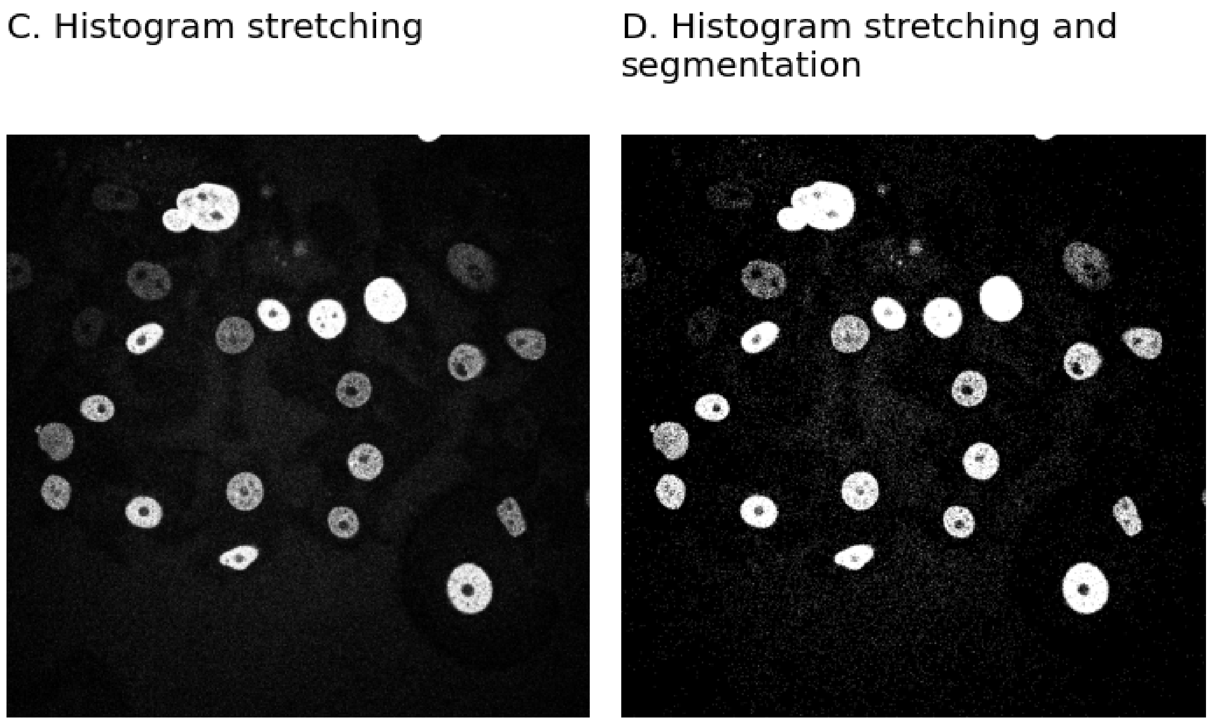


The reason histogram stretching works so well regarding this dataset is, that it increases contrast. This makes it easier to set a threshold that separates the cell nuclei from the background. Based on the following images, it is clearly visible that more pixels belonging to cell nuclei are detected by the segmentation algorithm after applying histogram stretching to the images.

In [9]:
```
stretched_image = preprocessing.histogram_stretching(image, intensity_lvls
segmented_image = otsu.complete_segmentation(image,intensity_lvls = 2**16)
segmented_stretched_image = otsu.complete_segmentation(stretched_image,inte
visualisation.two_img_plot(image,segmented_image,'A. Original image\n','B.
```



In [10]:
```
visualisation.two_img_plot(stretched_image,segmented_stretched_image,'C. Hi
                                        'D. Histogram stretching and\nsegmentation')
```



The reason why the Dice score and the MSD value evaluate some images differently is, that the MSD, being the mean surface distance between the pixels of the cell nuclei, penalizes false positive pixels in the background image much more than the Dice Score. The Dice Score cannot differentiate between false positive pixels near a cell and those scattered in the background.

This explains why the MSD rates the unpreprocessed segmentation results much higher than the Dice Score: As it can be seen in figure B, there are very few false positive pixels in the background (there is no background noise). However, some pixels belonging to cell nuclei were not recognized as such. This results in a lower Dices Score. For the same reason, segmented images preprocessed with histogram stretching are rated higher by the Dice Score than the MSD value. In this case the situation is the opposite. As can be seen in figure D, there are a lot of false positives in the background. However, the number of true positive pixels is higher.

4.2 N2DL-HeLa Dataset

4.2.1 N2DL-HeLa Results

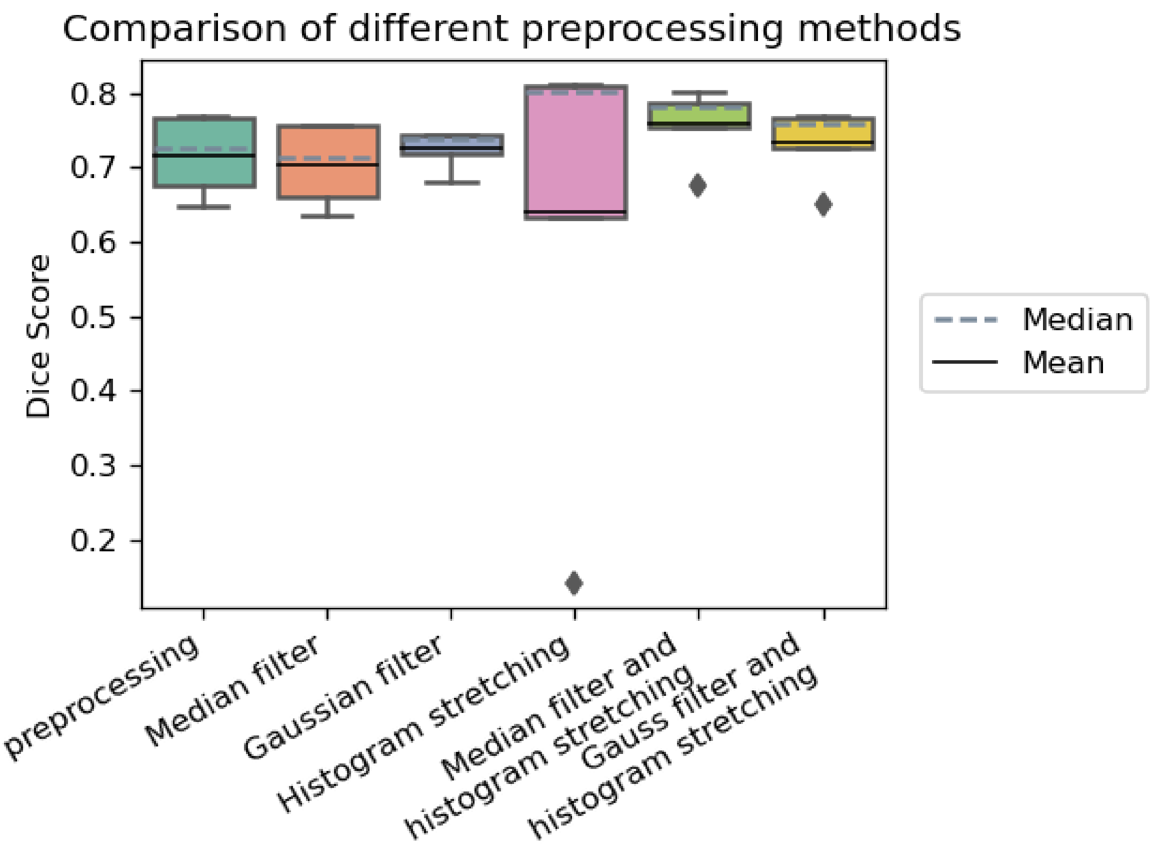
For the second dataset, the combination the Median filter and Histogram stretching resulted in the best segmentations:

In [11]:
```
complete_analysis.result_evaluation("Results/values.json", ["N2DL-HeLa"])
```

```
N2DL-HeLa (dice) : 0.76   --->   Median filter and histogram stretching
N2DL-HeLa (msd) : 5.527   --->   Median filter and histogram stretching
N2DL-HeLa (hd) : 119.439   --->   Gaussian filter
```

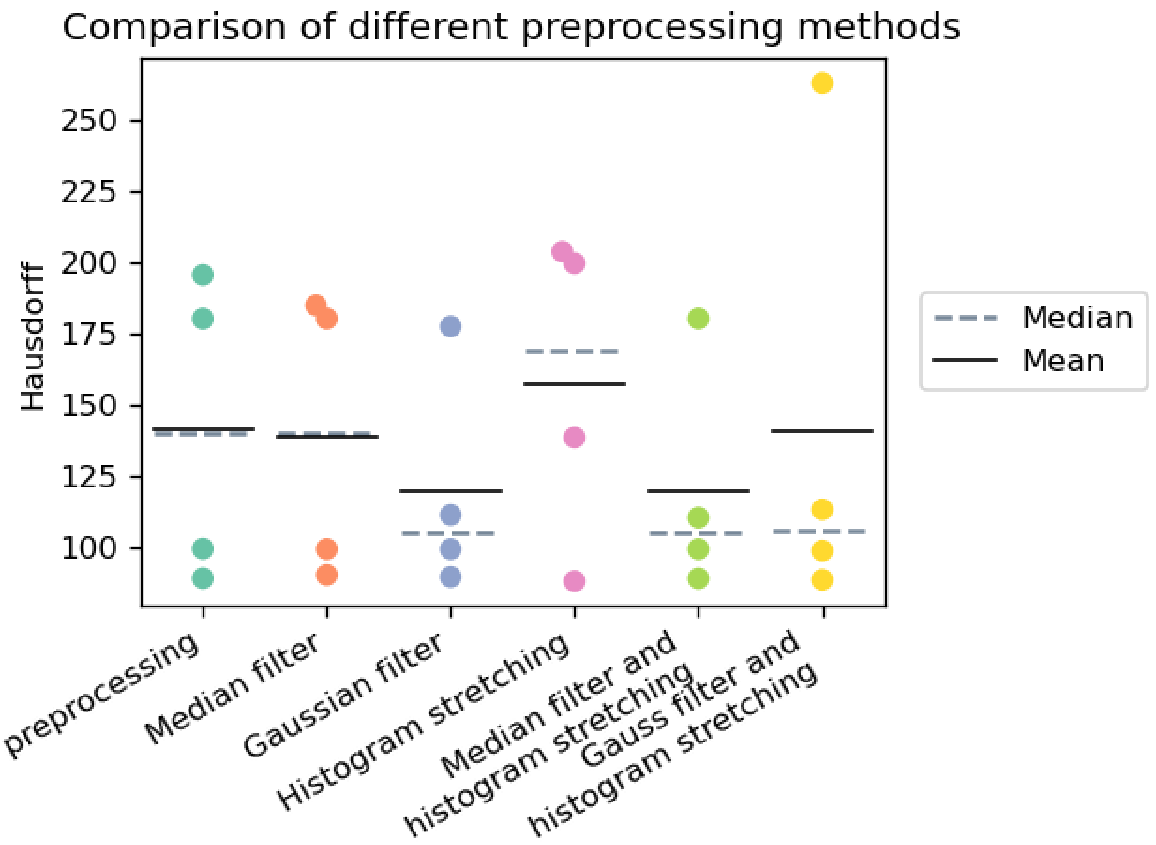
However, the overall distribution of the Dice scores for the images in the dataset shows that there is not a big difference with regard to the performance of the different preprocessing methods. With the exception of one outlier, the segmented images are very accurate in total.

In [12]:
```
visualisation.comparison_boxplot(complete_analysis.get_one_lvl_dice_scores(
```



The evaluation based on the Hausdorff Distance suggests that the Gaussian filter alone results in the best image segmentation. Generally, there is a high dispersion of the Hausdorff Distance values, which is why the results are presented using a swarmplot.

In [13]:
```
visualisation.comparison_swarmplot(complete_analysis.get_one_lvl_hd(data_on
```



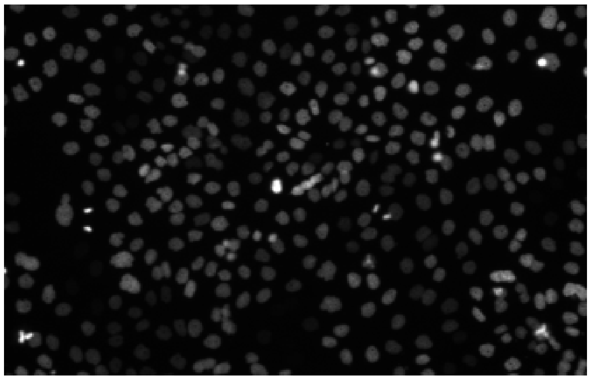
4.2.2 N2DL-HeLa Discussion

The major challenge of this dataset is the strongly varying brightness of the cell nuclei in the images. Some cell nuclei are very difficult to distinguish from the background while others are relatively bright. The degree of this characteristic is different for each image, which explains the 'outlier' within the image segmentation results that were evaluated using the Dice Score.

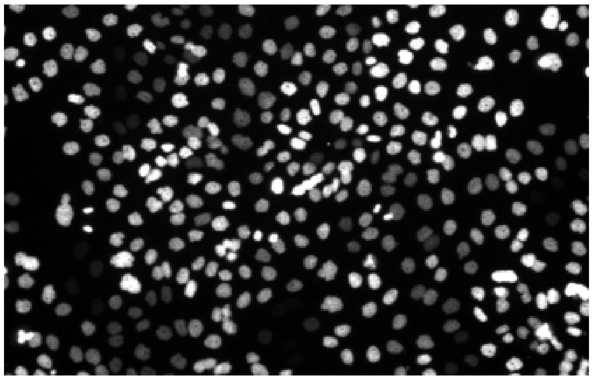
In the respective image (figure A), the variance in brightness between the cell nuclei is very strong. After applying histogram stretching, many cell nuclei are lost into the background in the final segmentation (figure C). Histogram stretching increases the brightness of already bright spots in the image and reduces the brightness of darker spots. If there are large differences in the brightness of the different cells, this characteristic is reinforced (figure B). This makes it even more difficult to find a suitable threshold value that separates the cell nuclei from the background. Therefore, the application of histogram stretching leads to the loss of many cell nuclei in the segmented image.

In [14]:
```
image = io.imread(str(pl.Path('Data/N2DL-HeLa/img/t79.tif')))
stretched_image = preprocessing.histogram_stretching(image, intensity_lvls
segmented_image = otsu.complete_segmentation(stretched_image,intensity_lvls
visualisation.two_img_plot(image, stretched_image, 'A. Original image', 'B.
```

A. Original image



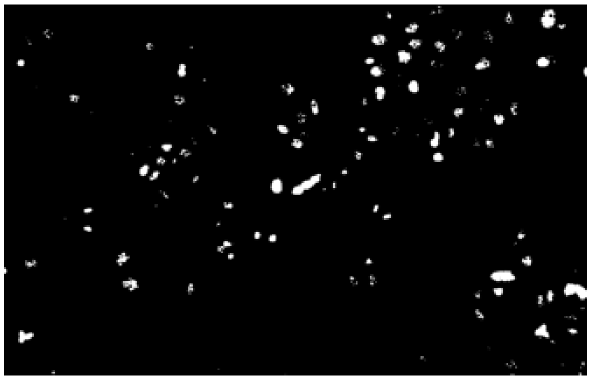
B. Histogram stretching



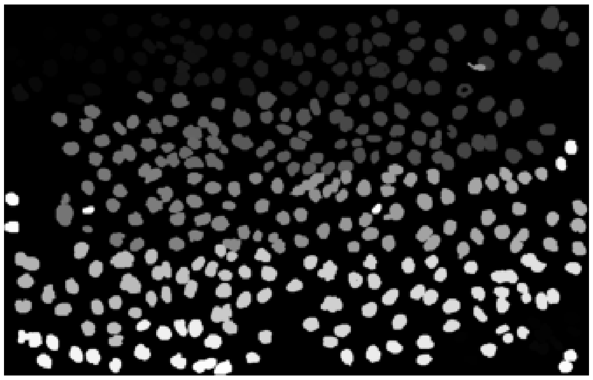
In [15]:
```

ground_truth = io.imread(str(pl.Path('Data/N2DL-HeLa/gt/man_seg79.tif')))
visualisation.two_img_plot(segmented_image, ground_truth, 'C. Segmented ima
```

C. Segmented image

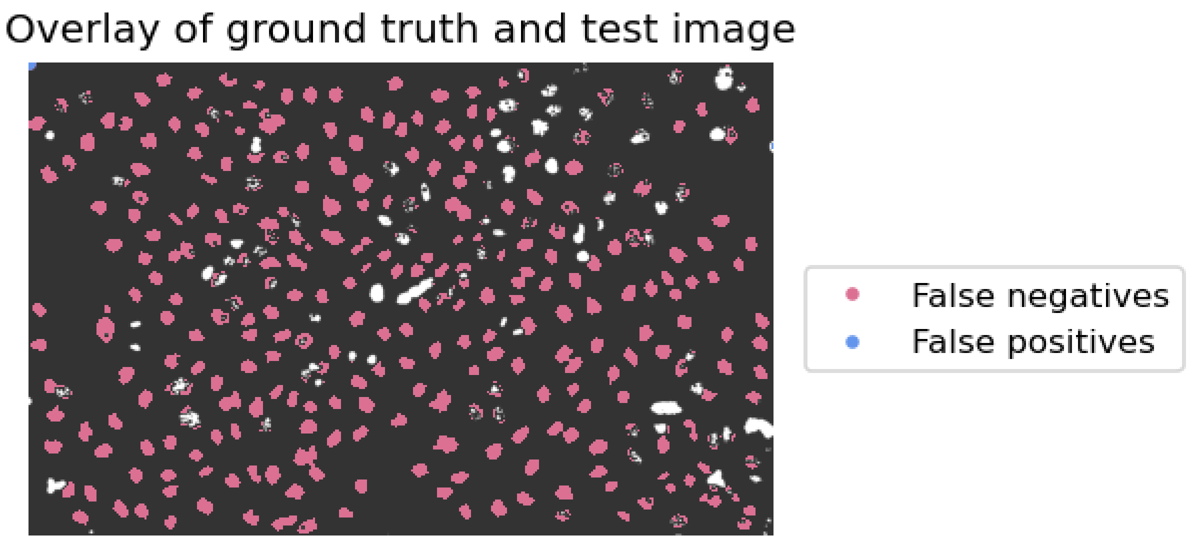


D. Ground truth



The overlay of the ground truth and the segmented image visualises the very high percentage of false negative pixels.

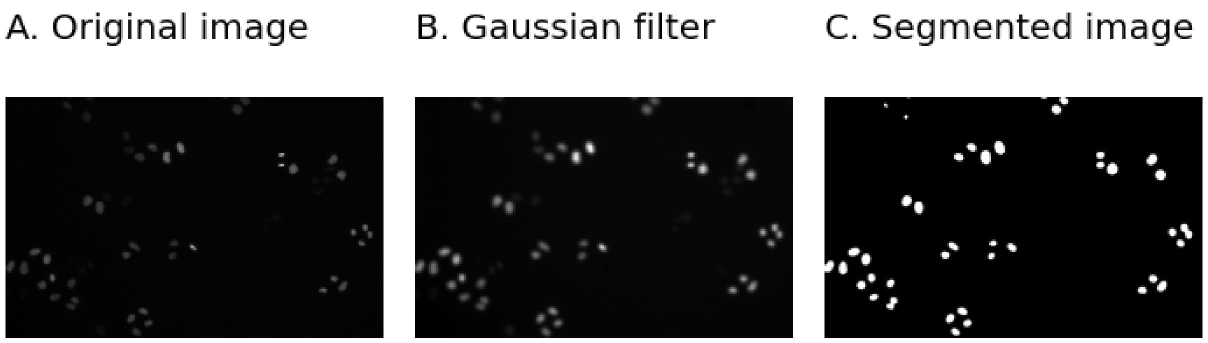
In [16]:
```
visualisation.overlay(segmented_image, ground_truth)
```



As shown by the high Dice scores, the other images of this dataset resulted in relatively accurate segmentations. This is due to less variance in the brightness between different cell nuclei. Therefore, the cells can be separated from the background more easily. As a consequence, histogram stretching does not lead to a massive loss of pixels.

In this dataset, filters generally seem to lead to good segmentation results. This might be due to the fact that the blurring effect of filters reduces the difference of brightness between the cells.

In [17]:
```
image1 = io.imread(str(pl.Path('Data/N2DL-HeLa/img/t13.tif')))
kernel =  preprocessing.gaussian_kernel(21, 5)
gaussian_image1 = preprocessing.convolution(image1, kernel)
segmented_image = otsu.complete_segmentation(gaussian_image1,intensity_lvls
visualisation.three_img_plot(image1, gaussian_image1,segmented_image ,'A. O
```



As already mentioned, the Hausdorff distances form a scattered distribution. Hence, this metric seems not to be a good measure to evaluate the quality of the image segmentation. The reason for that is, that the Hausdorff Distance is the maximal surface distance between the segmented image and the Ground truth. This means that single outliers can dramatically increase the Hausdorff

distance, even if the general segmentation is very accurate.
To conclude, the use of this metric only makes sense if outliers are very unlikely.The segmentation algorithm used here produces results that are probably too imprecise to be evaluable with the Hausdorff metric in a meaningful way.

4.3. NIH3T3 Dataset

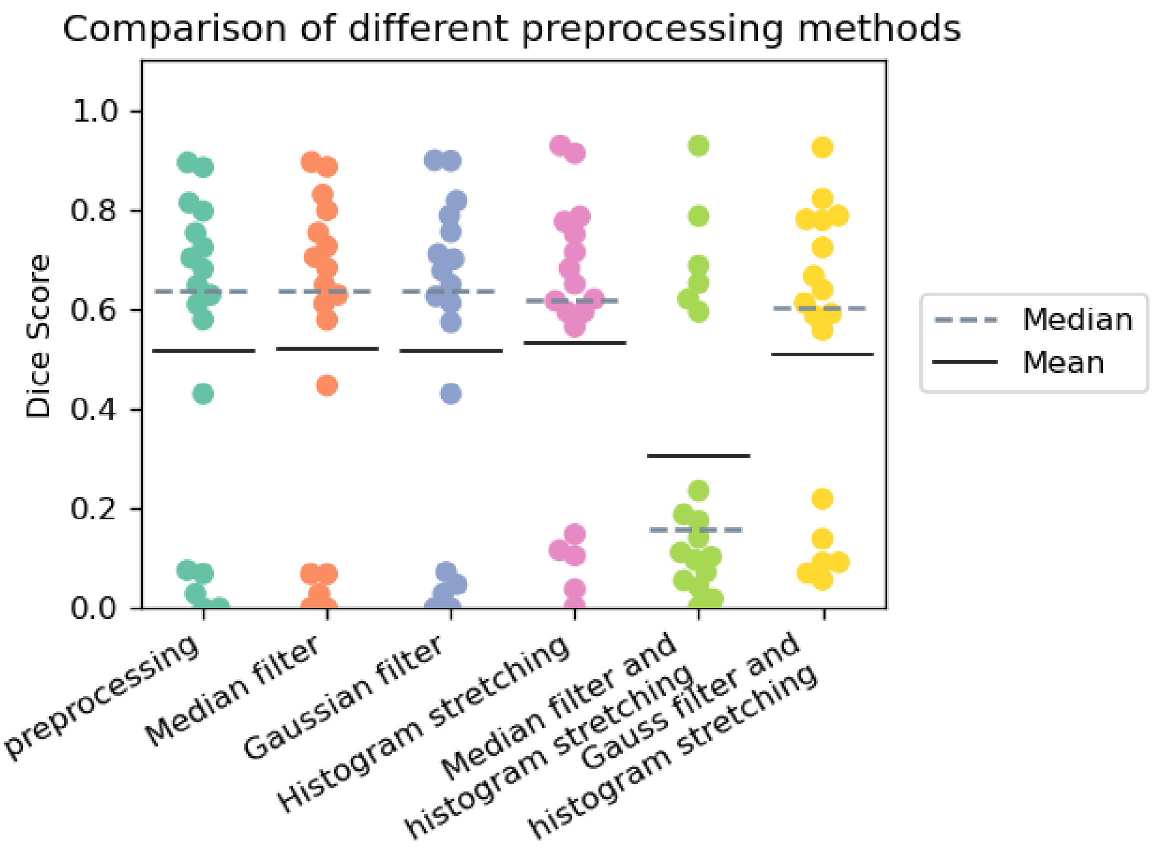
4.3.1 NIH3T3 Results

Since the major challenge of the NIH3T3 dataset is reflections, we applied both, one- and two-level Otsu thresholding on them. Moreover, we tested the different preprocessing methods. The best results for one-level Otsu's thresholding were as follows:

In [18]:
```
complete_analysis.result_evaluation(str(pl.Path("Results/values.json")), ["
```

```
NIH3T3 (dice) : 0.533   --->   Histogram stretching
NIH3T3 (msd) : 27.174   --->   Histogram stretching
NIH3T3 (hd) : 208.439   --->   Histogram stretching
```

In [19]:
```
visualisation.comparison_swarmplot(complete_analysis.get_one_lvl_dice_score
                                   scale_axis=1)
```

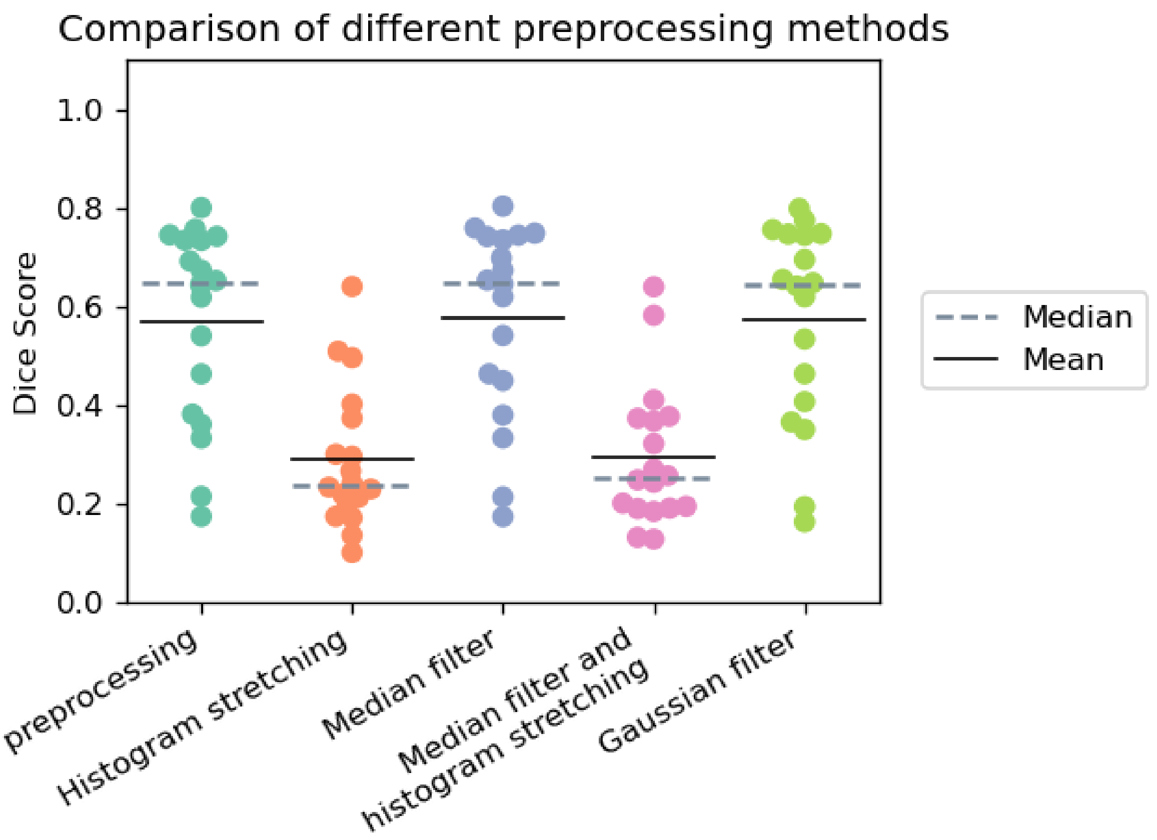


The preprocessing strategy that resulted in the highest mean Dice Score (0.533) is histogram stretching.

Dice Scores resulting from two-level Otsu's Thresholding show a relatively big variance. The best

Dice Scores resulting from two-level Otsu's Thresholding show a relatively big variance. The best Dice Score (0.577) resulted from images that were preprocessed using the median filter as shown in the following figure.

```
In [20]: scores = complete_analysis.get_all_two_level_results(str(pl.Path("Results/t
         visualisation.comparison_swarmplot(scores, scale_axis=1,
                                            x_label=["No preprocessing", "Histogram st
                                                     "Median filter and\nhistogram str
```



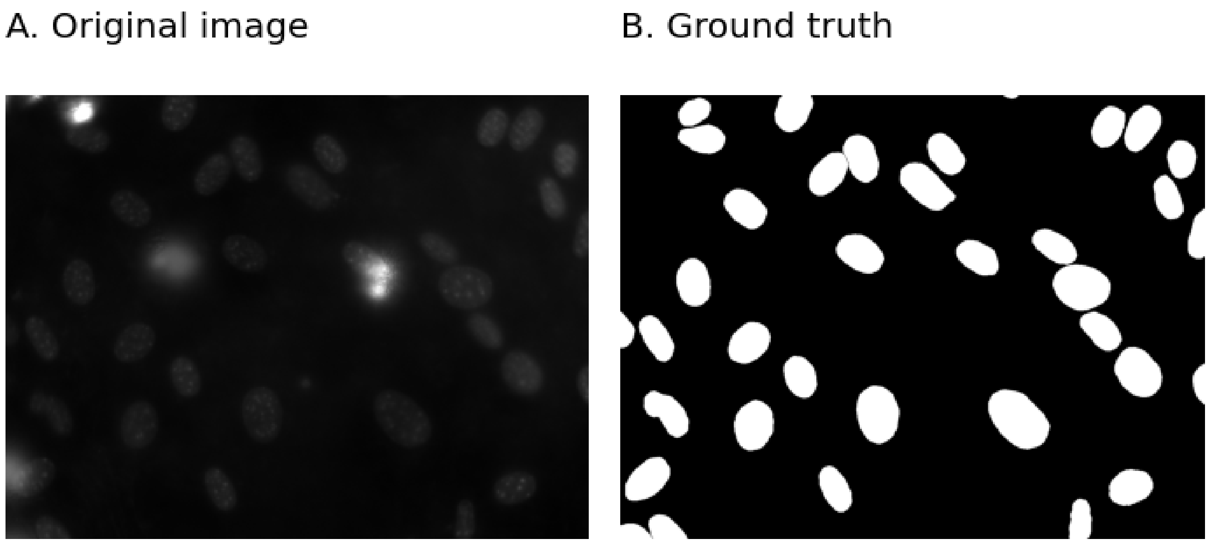
In both cases (one- and two-level thresholding) the mean and the median values are strongly deviating, which means that the distribution of the Dice Score values is not symmetrical.

4.3.2 NIH3T3: Discussion

As mentioned in the results, the distribution of Dice Score values after one-level Otsu segmentation is bimodal. The reason for that is that not all images in the dataset have reflections. Those without reflections are successfully segmented when using one-level Otsu's Thresholding. The two-level approach is not needed on those images and sometimes contra-productive, which explains the high variance of the results.

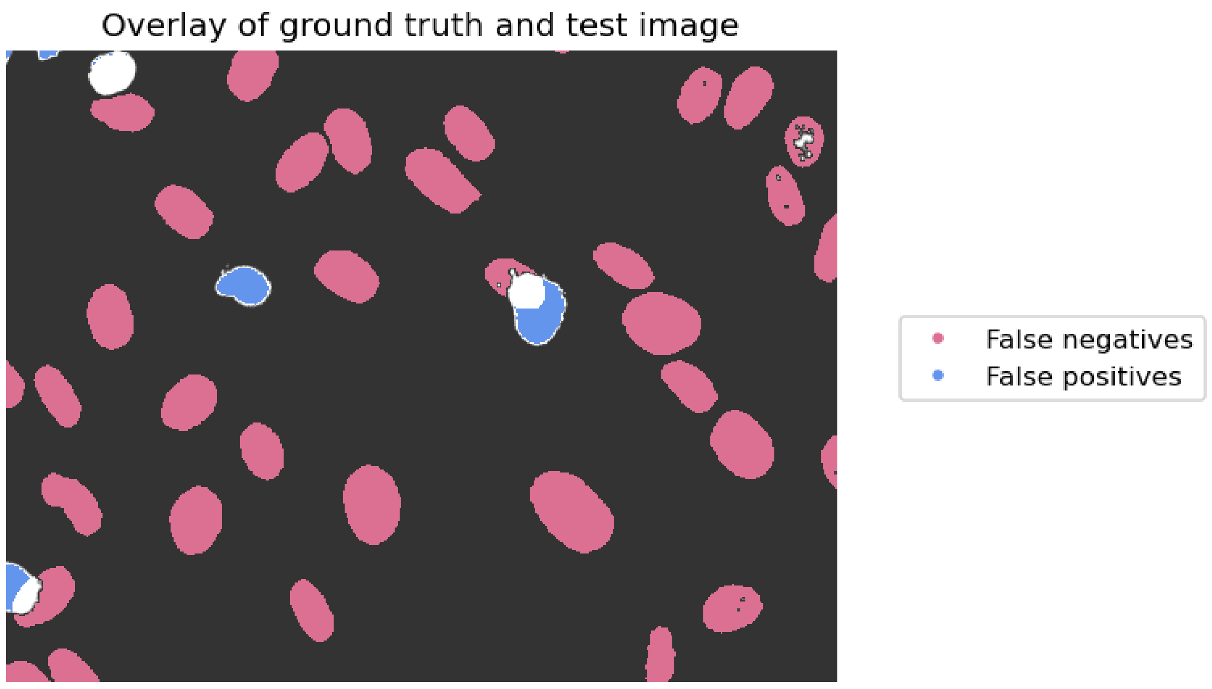
An example of an image with really bright reflections and the respective ground truth image are shown below:

In [21]:
```
reflections_image =io.imread(str(pl.Path('Data/NIH3T3/img/dna-47.png')))
ground_truth = io.imread(str(pl.Path('Data/NIH3T3/gt/47.png')))
visualisation.two_img_plot(reflections_image, ground_truth, "A. Original im
```



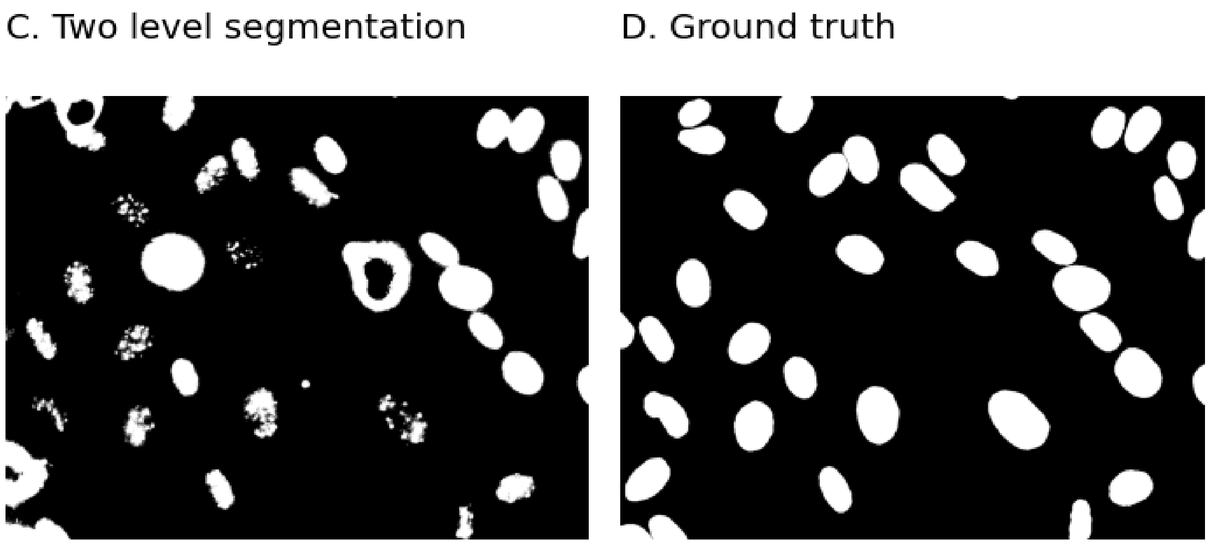
The following image shows the segmentation result of an image with a few very bright reflections. As the intensity values in the reflection regions are much higher than the average image intensity, the one-level algorithm sets the threshold too high. Therefore, most of the cell nuclei are not detected in the segmentation.

In [22]:
```
segmentation_one_level = otsu.complete_segmentation(reflections_image)
visualisation.overlay(segmentation_one_level,ground_truth)
```



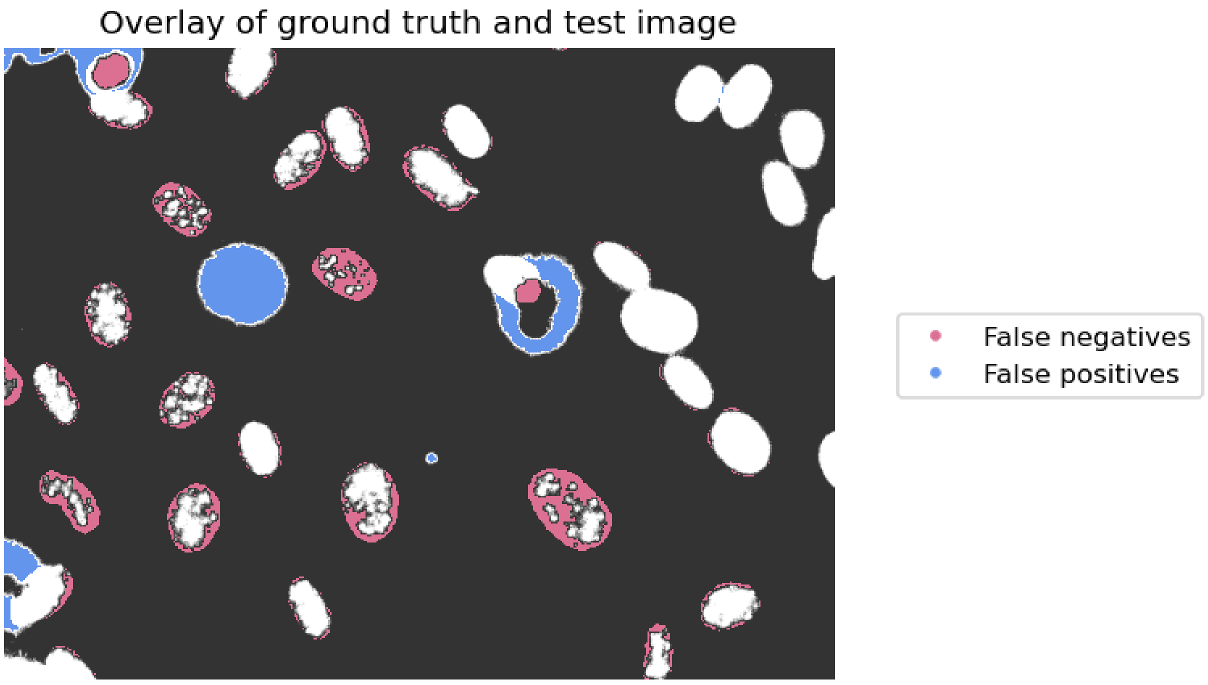
The same image segmented with the two-level algorithm showed to be much more successful than the one-level approach (figure C). The values above the higher threshold were assigned to the background and the values between the two thresholds were assigned to the cell nuclei.

In [23]:
```
segmentation_two_level = otsu.complete_segmentation_twolevel(io.imread(str(
visualisation.two_img_plot(segmentation_two_level, ground_truth, "C. Two le
```



The overlay of the segmented image and the ground truth shows that the number of false negative pixels is greatly reduced.

In [24]:
```
visualisation.overlay(segmentation_two_level,ground_truth)
```



4.4 Cell nuclei counting

4.4.1 Results

The cell nuclei counting algorithm was tested on the ground truth images from the N2DH-GOWT1 and the N2DL-HeLa dataset. The single cell nuclei in the Ground truth images in these datasets have different intensities. Therefore, it is possible to extract the number of cell nuclei from the Ground truth images.

The results are presented in the following tables.

In [25]:
```
with open(str(pl.Path('Results/cell_counting_results.json')), "r") as file:
    cell_counting_results = json.load(file)

table_GOWT1 = pd.DataFrame.from_dict(cell_counting_results["N2DH-GOWT1"])
table_GOWT1 = table_GOWT1.set_axis(["man_seg01.tif", "man_seg21.tif", "man_
                                    "man_seg39.tif", "man_seg52.tif", "man_
table_GOWT1.style.set_caption('Table 1: Results of the cell counting on the
```

Out[25]:

Table 1: Results of the cell counting on the N2DH-GOWT1 dataset.

	Calculated number	Ground truth number	Absolute difference	Relative difference
man_seg01.tif	24	23	1	0.043478
man_seg21.tif	23	24	-1	-0.041667
man_seg31.tif	24	22	2	0.090909
man_seg39.tif	23	25	-2	-0.080000
man_seg52.tif	30	30	0	0.000000
man_seg72.tif	28	28	0	0.000000

In [26]:
```
table_HeLa = pd.DataFrame.from_dict(cell_counting_results["N2DL-HeLa"])
table_HeLa = table_HeLa.set_axis(["man_seg13.tif", "man_seg52.tif", "man_se
table_HeLa.style.set_caption('Table 2: Results of the cell counting on the
```

Out[26]:

Table 2: Results of the cell counting on the N2DL-HeLa dataset.

	Calculated number	Ground truth number	Absolute difference	Relative difference
man_seg13.tif	58	59	-1	-0.016949
man_seg52.tif	107	109	-2	-0.018349
man_seg75.tif	365	349	16	0.045845
man_seg79.tif	329	342	-13	-0.038012

In [27]:
```
mean_error = round((np.abs(np.mean(cell_counting_results["N2DL-HeLa"]["Rela
                             + cell_counting_results["N2DH-GOWT1"]["Relative
standard_deviation = round((np.abs(np.std(cell_counting_results["N2DL-HeLa"
                                   + cell_counting_results["N2DH-GOWT1"]["Re
print("Relative error (datasets combined): {} +/- {}".format(mean_error, st
```

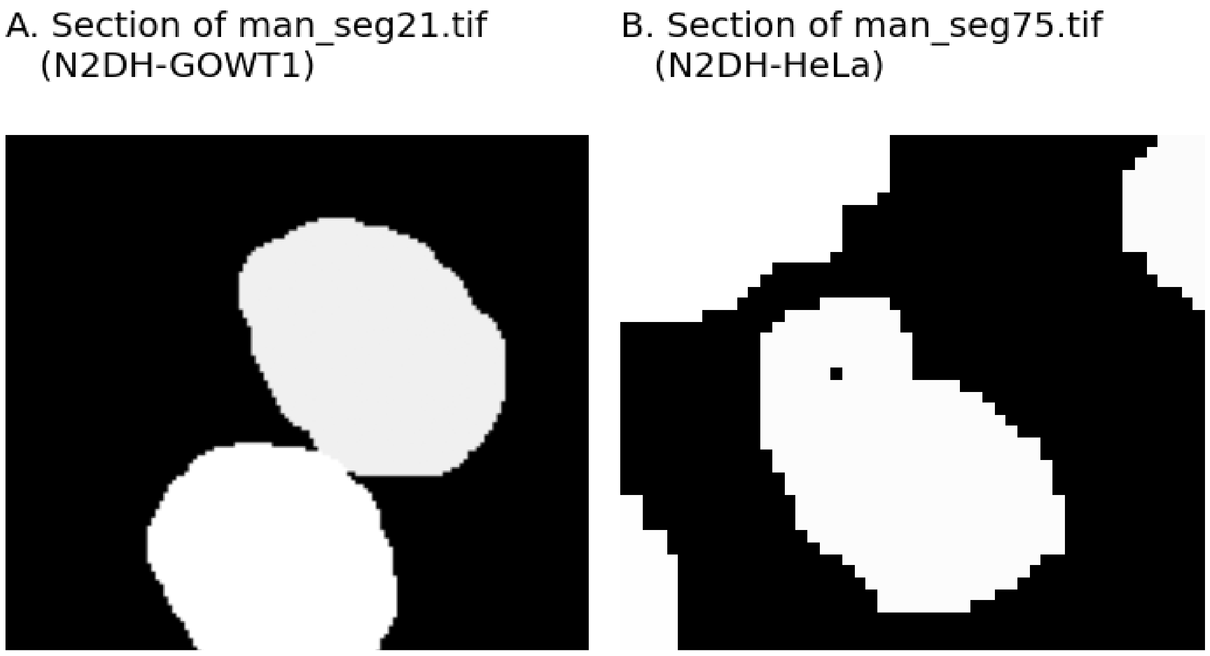
```
Relative error (datasets combined): 0.0015 +/- 0.0474
```

4.4.2 Discussion

Our algorithm calculated the cell nuclei number in 2 of 10 images correctly. The relative error did not exceed 10% and 5% when applied to the N2DH-GOWT1 and on the N2DL-HeLa datasets respectively. By analyzing the images, we determined the possible causes of the errors. Characteristic examples are shown below.

```
In [28]: img_GOWT1 = io.imread(str(pl.Path('Data/N2DH-GOWT1/gt/man_seg21.tif')))
         img_HeLa = io.imread(str(pl.Path('Data/N2DL-HeLa/gt/man_seg75.tif')))

         visualisation.two_img_plot(img_GOWT1[660:800, 800:940], img_HeLa[610:660, 1
                                    "A. Section of man_seg21.tif\n   (N2DH-GOWT1)",
                                    "B. Section of man_seg75.tif\n   (N2DH-HeLa)")
```



As shown in figure A some images contain merging cell nuclei. In this case, the two cell nuclei have a common border. This is why the algorithm detects only one cell.
In other cases there are small black regions inside the cell nuclei (figure B). In this case the algorithm does not only detect the border around the cell, but also the border around the black pixel inside the cell. Since these two borders do not touch, they would be assigned to two different pixel groups and therefore counted as two cell nuclei.

All in all, it can be said that our algorithm works well, considering its low level of complexity. Generally, it is able to detect the number of cell nuclei correctly except for a few predictable exceptions.

5. Conclusion

Our hypothesis, that the segmentation would be more accurate for the different datasets with specific preprocessing methods, was confirmed. For images in the N2DH-GOWT1 dataset, that are mainly low in contrast, histogram stretching improved the segmentation results. Filters had no major impact. On the images in the N2DL-HeLa dataset histogram stretching resulted in bad results in single cases. The reason for that is that histogram stretching enhanced the highly varying brightness of the cell nuclei. In contrast, filters reduced this effect. Applying Otsu's thresholding on the images from the NIH3T3 dataset resulted in a bimodal distribution of the Dice Scores. The poorly segmented images were the ones with bright spots (reflections). Two-level thresholding improved the segmentation results.

The evaluation with the Hausdorff Distance varies very strongly regarding all datasets. Hence, this metric is not useful to evaluate the quality of the finished segmentations in our case. Further improvement of our algorithm and better segmentation results would be necessary in order to use the Hausdorff distance in a reasonable way. The meaningfulness of the MSD and Dice Score relies on the main goal of the segmentation. MSD penalized noise and false positives that are not close to cell nuclei more gravely than the Dice Score. If the goal is to reduce the number of false negative pixels scattered in the image background, even if this means that the number of false negative pixels increases, the MSD value is more fitting to evaluate the segmented images. In contrast, the Dice Score does not take the positions of the false positives/negatives into account. Therefore, if the goal is to recognize as much cell nuclei pixels (true positives) as possible, even though there might be false positive pixels scattered in the background, the Dice Score is more meaningful.

Except for minor deviations, the cell nuclei counting algorithm was generally very accurate. Improvement of the algorithm could include overcoming limitations such as single dark pixels within the cell nucleus and merging cell nuclei.

Bibliography

Bártová, E., Šustáčková, G., Stixová, L., Kozubek, S., Legartová, S., & Foltánková, V. (2011). Recruitment of Oct4 Protein to UV-Damaged Chromatin in Embryonic Stem Cells. Plos ONE, 6(12), e27281. doi: 10.1371/journal.pone.0027281

Eelbode, T., Bertels, J., Berman, M., Vandermeulen, D., Maes, F., Bisschops, R., & Blaschko, M. (2020). Optimization for Medical Image Segmentation: Theory and Practice When Evaluating With Dice Score or Jaccard Index. IEEE Transactions On Medical Imaging, 39(11), 3679-3690. doi: 10.1109/tmi.2020.3002417

Neumann, B., Walter, T., Hériché, J., Bulkescher, J., Erfle, H., & Conrad, C. et al. (2010). Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes. Nature, 464(7289), 721-727. doi: 10.1038/nature08869

Coelho, L., Shariff, A., & Murphy, R. (2009). Nuclear segmentation in microscope cell images: A hand-segmented dataset and comparison of algorithms. 2009 IEEE International Symposium On Biomedical Imaging: From Nano To Macro. doi: 10.1109/isbi.2009.5193098

Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. IEEE Transactions On Systems, Man, And Cybernetics, 9(1), 62-66. doi: 10.1109/tsmc.1979.4310076

Lee, I., Im, H., Solaiyappan, M., & Cho, S. (2017). Comparison of novel multi-level Otsu (MO-PET) and conventional PET segmentation methods for measuring FDG metabolic tumor volume in patients with soft tissue sarcoma. EJNMMI Physics, 4(1). doi: 10.1186/s40658-017-0189-0

Liu, L., Li, K., Qin, W., Wen, T., Li, L., Wu, J., & Gu, J. (2018). Automated breast tumor detection and segmentation with a novel computational framework of whole ultrasound images. Medical & Biological Engineering & Computing, 56(2), 183-199. doi: 10.1007/s11517-017-1770-3

Rhee, D., Jhingran, A., Rigaud, B., Netherton, T., Cardenas, C., & Zhang, L. et al. (2020). Automatic contouring system for cervical cancer using convolutional neural networks. Medical Physics, 47(11), 5648-5658. doi: 10.1002/mp.14467

Taha, A., & Hanbury, A. (2015). Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool. BMC Medical Imaging, 15(1). doi: 10.1186/s12880-015-0068-x