Soft Computing (2020) 24:17847–17862
https://doi.org/10.1007/s00500-020-05033-0

METHODOLOGIES AND APPLICATION

Evolutionary optimization of image processing for cell detection in microscopy images

Andreas Haghofer[1,2,3] · Sebastian Dorl[1] · Andre Oszwald[4] · Johannes Breuss[5] · Jaroslaw Jacak[6] · Stephan M. Winkler[1,3]

Published online: 11 June 2020

Abstract
In this paper, we present a new evolution-based algorithm that optimizes cell detection image processing workflows in a self-adaptive fashion. We use evolution strategies to optimize the parameters for all steps of the image processing pipeline and improve cell detection results. The algorithm reliably produces good cell detection results without the need for extensive domain knowledge. Our algorithm also needs no labeled data to produce good cell detection results compared to the state-of-the-art neural network approaches. Furthermore, the algorithm can easily be adapted to different applications by modifying the processing steps in the pipeline and has high scalability since it supports multithreading and computation on graphical processing units (GPUs).

Keywords Evolutionary algorithms · Image processing · Computer vision · Heuristic optimization · Machine learning

1 Introduction

Microscopy imaging techniques are widely used in modern medicine. Various medical treatments and diagnoses require the analysis of cell images, for example, blood cell counting is regularly used for detection of various diseases (Hawkins et al. 2014). Cell imaging is also used to research the dividing behavior of cells, or observe them as they approach a state of senescence (Yang et al. 1999). In such applications, it is essential to automatically segment the microscopy images and detect the individual cells. Automatic cell detection algorithms are hindered by different shaped cells within the same image as well as varying conditions in image background, illumination, cell size, etc. In Fig. 1, we see examples of microscopy images that show this heterogeneity.

Cell detection in microscopy images is often done by machine learning algorithms using neural networks. Neural network approaches like u-net (Falk et al. 2019) are capable of detecting cells with various shapes. In combination with feature extraction and image processing algorithms (Al-Kofahi et al. 2018), deep learning approaches are the state-of-the-art method for cell detection.

Communicated by V. Loia.

✉ Andreas Haghofer
andreas.haghofer@fh-hagenberg.at;
andreas.haghofer@ffoqsi.at

Sebastian Dorl
sebastian.dorl@fh-hagenberg.at

Andre Oszwald
andre.oszwald@meduniwien.ac.at

Johannes Breuss
johannes.breuss@meduniwien.ac.at

Jaroslaw Jacak
jaroslaw.jacak@fh-linz.at

Stephan M. Winkler
stephan.winkler@fh-hagenberg.at

[1] Bioinformatics Research Group, University of Applied Sciences Upper Austria, Softwarepark 11–13, 4232 Hagenberg, Austria

[2] Austrian Competence Centre for Feed and Food Quality, Safety and Innovation, Technopark 1C, 3430 Tulln, Austria

[3] Department of Computer Science, Johannes Kepler University Linz, Altenberger Straße 69, 4040 Linz, Austria

[4] Clinical Institute of Pathology, Medical University of Vienna, Schwarzspanierstraße 17, 1090 Vienna, Austria

[5] Center of Physiology and Pharmacology, Medical University of Vienna, Schwarzspanierstraße 17, 1090 Vienna, Austria

[6] Department for Medical Engineering, University of Applied Sciences Upper Austria, Garnisonstraße 21, 4020 Linz, Austria

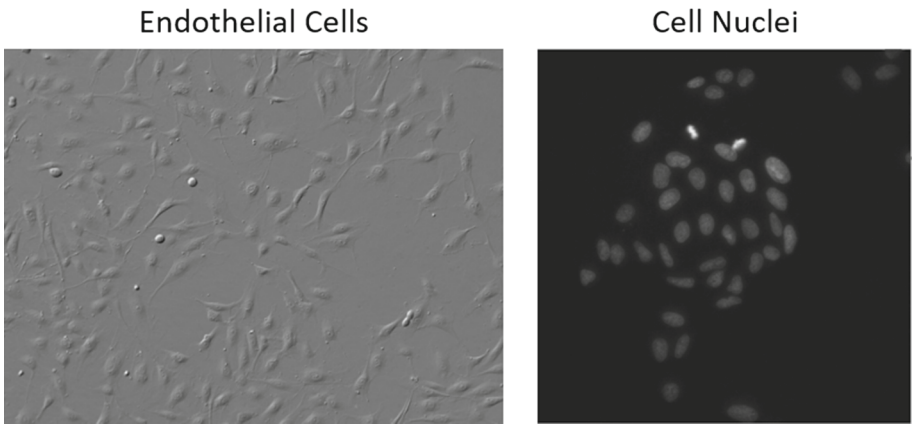

Fig. 1 Left: image of endothelial cells; right: image of cell nuclei. Objects within microscopy images appear in different shapes and intensity and therefore require different preprocessing steps to be correctly detected

However, these model-based approaches generally rely on training data, which usually needs to be labeled manually. This can be done a priori by labeling data where the model can learn to identify each object under different conditions. Alternatively, there are also unsupervised modeling algorithms like W-net (Xide et al. 2017), which can learn how to separate the different parts using two merged U-nets that are supported by conditional random fields (Chen et al. 2018). A different approach is used by the image segmentation tool Ilastik (Sommer et al. 2011) which uses on-the-fly labeling for parts of the image which should be segmented. These labels are further used by a random forest to segment the rest of the image automatically.

The drawback of such supervised approaches is the fact that these models often require large amounts of labeled images to train correctly. Thus, they are rarely applicable on sets of very few images. If the data only include a small number of images, training a neural network is rarely possible.

We have designed and implemented a complete workflow based on a heuristic parameterization of image processing algorithms, which does not require any previously labeled data. By automatically selecting precisely the image processing algorithms which are useful for the current image, our methodology delivers a level of flexibility that cannot be achieved by model-based approaches. Since selected algorithms highly depend on the type of image and information in the image, it is not possible to generate one workflow which can fit every type of image.

Other publications about image segmentation demonstrated that heuristic methods can keep up with machine learning-based approaches in this domain. A lot of them focus on multilevel thresholding using swarm optimization (El Aziz et al. 2016) or use adapted versions of the genetic algorithm (Sun et al. 2016). Our approach differs from these approaches in that we use standard image processing algorithms that are selected and adjusted using heuristic optimization. This also allows the user to preselect a pool of algorithms that might be appropriate for the current segmentation task and let our method do the fine tuning.

Even if our methodology uses algorithms that are selected for cell detection, it is also possible to include additional algorithms if needed, due to the modularity of our approach. For the results presented in this publication, we used the following algorithms:

- Gaussian blur filter
- Bandpass filter
- Adaptive histogram equalization
- Watershed
- Thresholding

Due to the number of possible parameter settings and algorithms, it is not efficient to try every combination of the listed algorithms. Using a heuristic optimization algorithm, we can find a suitable workflow within a limited amount of time. Our algorithm works even on a single image without any labeled training data. The goal of our approach is to enable an in-depth analysis of cell images with minimal need for image processing knowledge or user interaction during the detection process.

2 Methods

In this section, we describe the main workflow of our cell detection algorithm. Based on commonly used image processing algorithms, this workflow is capable of generating processing pipelines for any type of image.

2.1 Image processing

Even though it is possible to use this workflow for various kinds of segmentation problems, the presented algorithm performs especially well in the case of cell detection. For an appropriate detection result, it is necessary to equalize noise

and reduce the differences in illumination between image areas. The following algorithms are used to eliminate these detrimental factors and create a binary image for the final edge detection algorithm.

Gaussian blur The intention of applying a Gaussian filter on an image is to smooth the image intensity differences by applying a kernel representing a Gaussian distribution. This reduces the overall noise in the image (Deng and Cahill 2005). Applying a Gaussian kernel function results in image smoothing as shown in Fig. 2. The standard deviation for the distribution function is a parameter that has to be optimized to improve the detection quality.

Bandpass filter A bandpass filter represents a combination of high- and low-pass filter. These are used to cut off the unnecessary frequencies within the frequency space (Butterworth 1930). Depending on illumination conditions and the type of microscopy image, this method can increase the contrast between cell areas and the background. This filter is used to reduce noise and unneeded intensities which often appear when the range of used intensity values is narrow compared to the color depth of the image.

Fourier transformation By applying a Fourier transformation, it is possible to get a frequency representation of the image as shown in Fig. 3. This enables the use of filters originally invented for within the signal processing domain such as the Butterworth filter. Applying the transfer function of a low-pass Butterworth filter with a set cutoff frequency and order eliminates the high-frequency ranges above the cutoff frequency limit (Butterworth 1930). The combination of this filter with the inverse variant where lower frequencies are cut off eliminates inhibiting frequency ranges, which often contain noise and decrease segmentation quality (Fig. 4).

Adaptive histogram equalization As shown in Fig. 5, adaptive histogram equalization is used to increase the overall image contrast by spreading the intensity values across the whole color depth range. To equalize the intensity distribution, it is necessary to normalize by dividing the frequency of each intensity value through the total number of image pixels (Zuiderveld 2013). The used implementation additionally prevents burnouts of high-intensity ranges. The only decision made during the optimization process is if this algorithm is used or not.

Watershed segmentation Applying watershed segmentation on grayscale images improves the overall separation of image regions. By using a fixed threshold in combination with the so-called sheds, image regions are separated depending on their intensity values (Beucher 1992). By approximating image intensities with a topographic landscape, these sheds are used to separate "hills" and "basins" from each other which can also be used to separate wrongly connected cells from each other as shown in Fig. 6.

The parameters for the shed height and the used threshold are intended to be set automatically. Therefore, the type of neighborhood used to find the connection between two pixels is the only parameter to be set manually. In our implementation, we used an 8-pixel neighborhood to detect all surrounding connections of each pixel within the 2D image. Similar to histogram equalization, the only decision that is to be made during the optimization process is whether watershed segmentation is used or not.

Binary thresholding is used as an obligatory step after each generated workflow to convert the image into a binary representation (Halwa et al. 2013). Each intensity value above the threshold gets replaced by one, and all other values are set to zero.

As shown in Fig. 7, the binary conversion of a cell image leads to separated image regions representing individual cell positions. Independently from the initial image, this binary representation is suitable as a generalized format for the following detection process.

2.2 Cell segmentation

The point of the optimization process is to prepare the input images for the identification of the individual cells by the segmentation. It is part of the optimization to reduce factors like varying illumination or noise which reduces the quality of the segmentation.

To be able to handle each cell individually, it is necessary to know which parts of the input image contain cells, but also which pixels belong to which cell. Based on the flood fill algorithm (Torbert 2013), connected components labeling is able to transform the previously generated binary image into a matrix representation where each entry contains the unique ID of the cell it belongs to or zero if it is part of the background.

As shown in Fig. 8, each cell is marked individually and can be identified by a unique identification number. Due to this simple flood fill approach, there is no need for further feature detection to support the identification of the individual cells. This information is the basis for cell tracking in images series.

2.3 Optional parameter

To create a more robust method, we also implemented the possibility for optional constraints, which can be used to support the evolution strategy by ensuring that unrealistic detection results are not even considered during the optimiza-

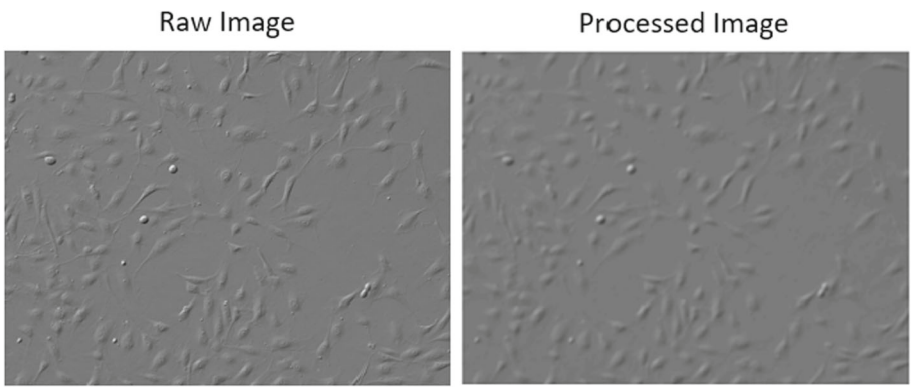

Fig. 2 Comparison of cell images with and without applying Gaussian blur. Left: Raw image of endothelial cells. Right: Image with the applied Gaussian kernel using a standard deviation of 4

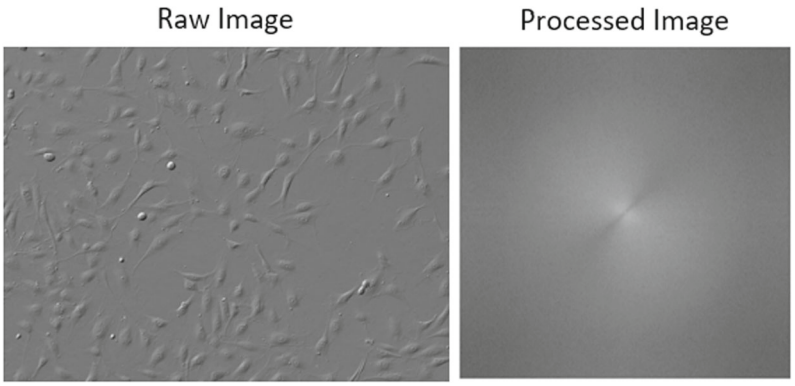

Fig. 3 Demonstration of fast Fourier transformation on raw cell image (left) resulting in the frequency representation (right)

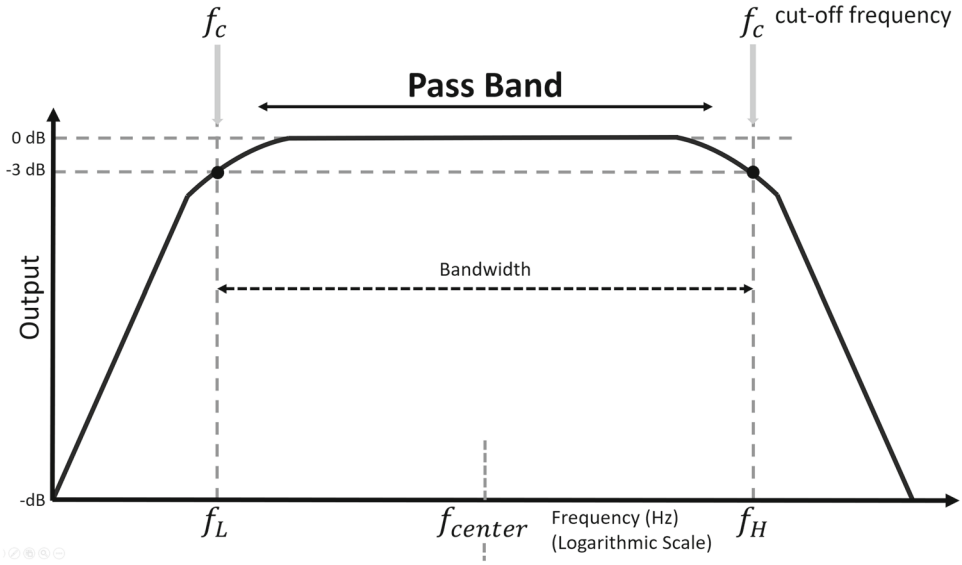

Fig. 4 Graphical representation of a bandpass filter

tion process. In our final implementation, we use a simple limit for the minimum and maximum possible size of a cell. To use this parameter, the user draws a window around the smallest and the largest cell in the current image. The selected area plus a ten percent tolerance is then used as a filter for the detected cells. If a cell is too small or too large, it is not part of the detection result and therefore reduces the number of detected cells, which leads to a lower estimated quality.

In some cases, as shown in Fig. 9, it can also be used to remedy an unsatisfying segmentation result, which can happen if there is a wide range of intensity values within the cell areas. Thus, using the constraint can correct the segmentation result in certain types of images. These parameters are also used for the actual segmentation if the user only wants to mark cells within a specific size, after the processing workflow was generated.

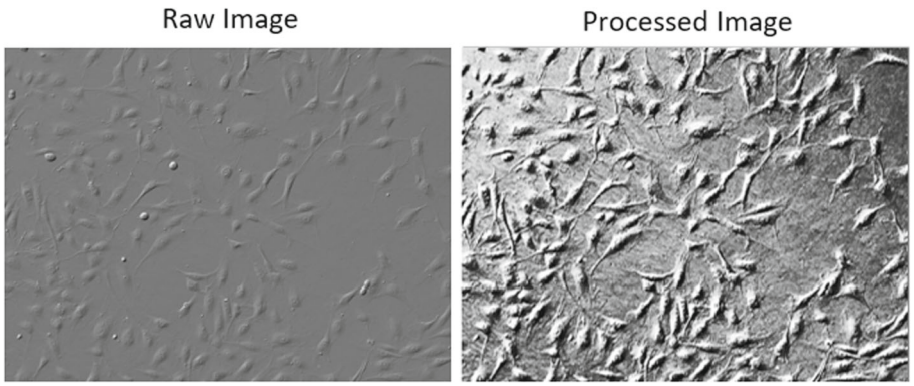

Fig. 5 Comparison of cell images with and without applying adaptive histogram equalization. Left: Raw image of endothelial cells. Right: Image with histogram equalization. The processed image provides an increased contrast between the cell areas and the background by increasing the difference in intensity between these two areas. We see that the resulting intensity differences within the cells are also increased, which should be compensated by a combination with other algorithms

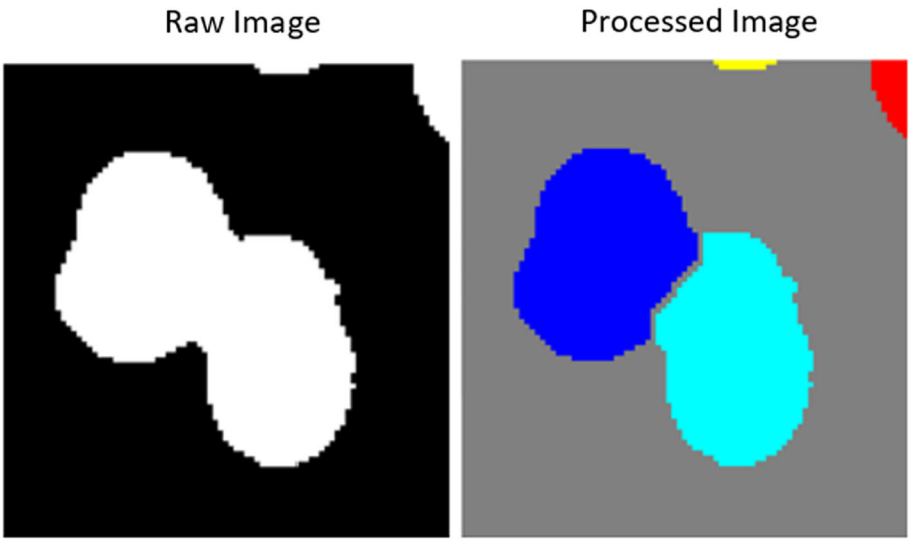

Fig. 6 Example of binary image using watershed segmentation to divide cells which are wrongly connected

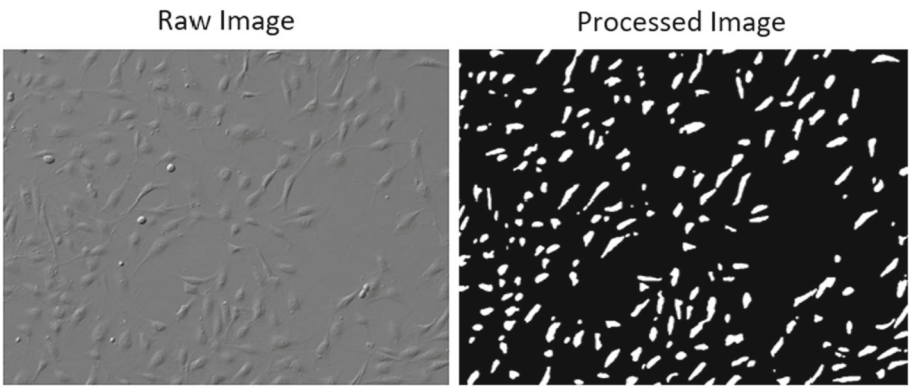

Fig. 7 Example of binary image segmentation of endothelial cells

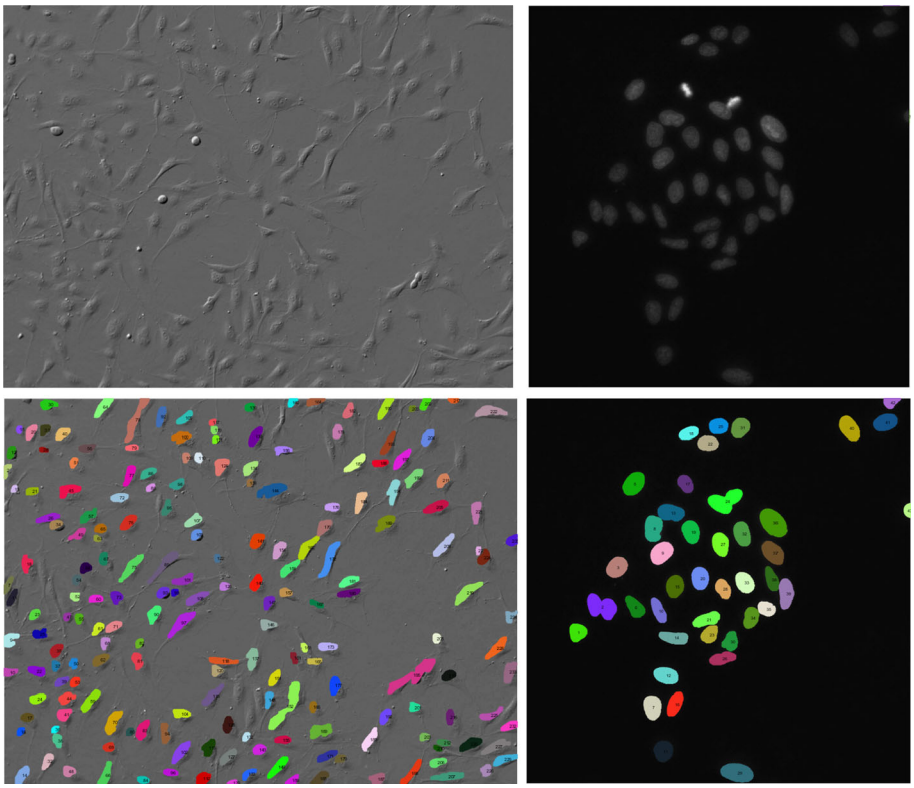

Fig. 8 Segmentation results of endothelial cells and cell nuclei including the individual cell IDs which are needed for further cell analysis

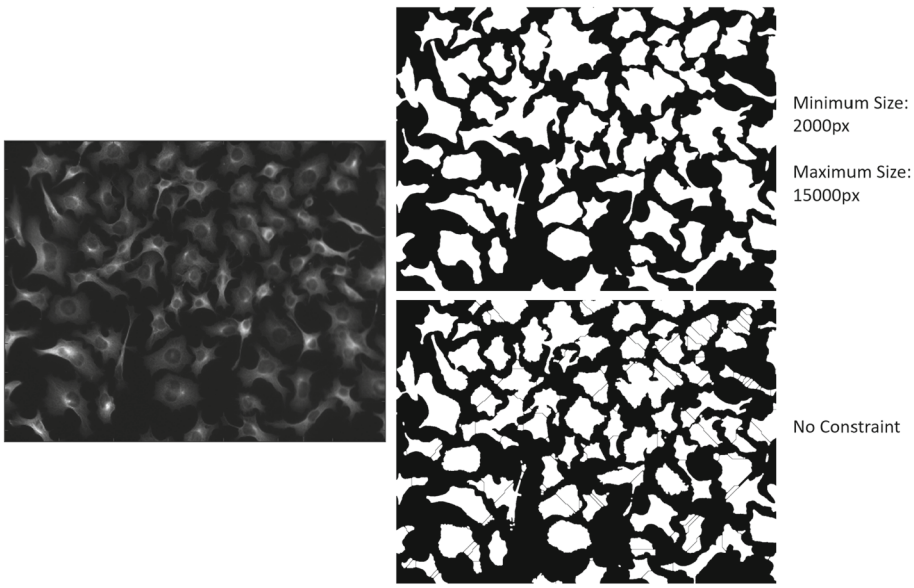

Fig. 9 Cell segmentation with and without the use of a constraint on the minimum and maximum size of detected cells. This neuroblastoma cell images (Yu et al. 2019) show that the result without using the constraint seems to provide a better separation of the cells, but there are also more wrong dividing lines

2.4 Evolution strategy for optimizing image processing workflows

2.4.1 Evolution strategies

The main idea of the original evolution strategy created by Rechenberg is to use the concept of natural evolution to solve heuristic optimization problems. The core element therefore is to introduce random changes of each parameter, following the example of natural mutation. With the addition of simple rules for mutation and selection, this concept is the basis for evolution strategies in heuristic optimization (Back et al. 1991).

As originally described by Rechenberg, it is essential to set the population size, the algorithm uses for each generation (Back et al. 1991). The parent generation describes how many solution candidates are generated in the 'initialize solutions' part of the algorithm (Fig. 10). This population of solu-

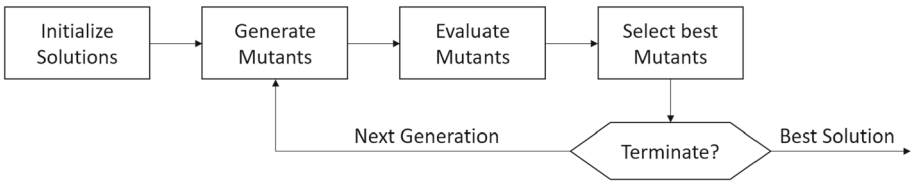

Fig. 10 Schematic visualization of an evolution strategy, starting with an initial solution which is used to generate × amount of mutants. By evaluating the fitness of these mutants, the n best of them were used to create mutants for the next generation until the resulting quality (fitness) converges

tion candidates contains the available gene pool (parameter sets) for future generations. To keep up the genetic diversity, Rechenberg adapted the concept of mutation. By adding a weighted (sigma) random number between zero and one to the current parameter value, a mutant of the initial solution is created (Michalewicz 2013).

If sigma is close to one, the changes caused by the mutation are significant, which leads to a random search. If the fitness increases during the evolution process, sigma is reduced to only allowing slight changes. This process converts the initial randomness into a more directed search to find the optimal value (Affenzeller et al. 2009).

The implementation in our algorithm uses some variations to the original evolution strategy. Our approach includes the parent generation into the selection process. This concept, called offspring selection, is often used in genetic algorithms (Affenzeller and Wagner 2005). For the presented implementation, it is also beneficial to use more than one parent and produce multiple mutants. This increase in population size allows for a more efficient optimization process with a more varied gene pool. Efficiency is further increased by using the success rule mechanism. This mechanism adds the constraint that a fixed percentage of mutants must have higher fitness than the parents from the previous generation; otherwise, the value of sigma is increased. The longer the optimization process is running, the more specific the search process should be. Therefore, the mutations should not cause a switch to completely different solution candidate close to the optimum (Affenzeller and Wagner 2005).

2.4.2 Solution candidate

The presented methodology uses solution candidates with two types of genes. One type represents the actual parameter for the image processing algorithm, whereas the other represents switches which can turn off the use of certain algorithms entirely, as seen in Table 1. The combination of all genes of one solution candidate represents a complete processing workflow, including optimized parameters for all algorithms which will be used.

To reduce complexity, we added artificial limits to ensure that the evolution strategy does not have to test solution candidates who do not make sense. Therefore, the sigma used for the Gaussian blur is limited within the range of 0.1 to 4. This range ensures that the filter is not disabled by using a value near zero or that artificial borders are created due to a very large sigma value. The value range of the bandpass filter parameters is dependent on the image type and will either include 256 possible values for 8-bit images or 65535 possible values for 16-bit images.

2.4.3 Evaluation of solution candidates

Depending on the type of cells, it is possible that cells overlap with each other within the image. Therefore, it is necessary to find an appropriate balance between the ability to differentiate the cell areas from the background and the separation between the cells. This balance is realized by three measures which are used for the quality calculation of each solution candidate. Each of these measures can be weighted differently to vary their influence on the final quality value, which allows the flexibility to adjust to different types of microscopy images.

$$label(p) = \begin{cases} ID(c), & \text{if pixel } p \text{ belongs to cell } c \\ 0, & \text{otherwise, i.d. } p \text{ is background} \end{cases} \quad (1)$$

As described in Eq. 1, connected components labeling provides the information, about how many individual cell areas were detected. Using this information, the following evaluation measures are calculated.

Amount of detected cells The primary goal of a cell detection algorithm is to identify as many cells as possible. After detecting the areas of each cell using the generated pipeline, binary morphology closes gaps within the cells (Soille 1999). A minimum threshold for the size is used to filter out false positives which are created when the algorithm detects dust particles or air bubbles as cells. The final number of detected cells is calculated as the number of individual detected regions and used as one of the quality measures for evaluation.

Table 1 Solution candidates for a cell segmentation workflow

Gene	Datatype	Range
Sigma of Gaussian filter	Float	0.1 <= x <= 4
Lower cutoff frequency of bandpass filter	Int	0 <= x <=(65535 or 255)
Lower cutoff frequency of bandpass filter	Int	0 <= x <= (65535 or 255)
Threshold for binarization	Float	0 < x < 1
Switch for the Gauss filter	Boolean	0,1
Switch for the bandpass filter	Boolean	0,1
Switch for the watershed segmentation	Boolean	0,1
Switch for the histogram equalization	Boolean	0,1

Standard deviation of the detected background area After reducing the influence of light sources and their position, the distribution of the background intensities should be similar to a normal distribution with low standard deviation. The standard deviation of pixel intensities (Eq. 2) within a correctly detected background will be lower than within a background with false negatives. This is because the surface tension of the cells is influencing light rays, resulting in more variation of intensities within the cells, compared to the background.

$$stdBackground = std(intensity(p : label(p) = 0)) \quad (2)$$

Difference in the histogram of cell areas and the background The histograms of background and cell areas are compared. We assume that correct identification of cells as foreground objects will lead to a significant difference in the histograms of the intensities of the background and the detected cells. Thus, we calculate the histograms of the pixels identified as cell pixels (Eq. 3) and the histogram of the background pixels (Eq. 4) and then calculate the relative difference of all entries in these histograms (Eq. 5):

$$histObjects = hist(intensity(p : label(p) > 0)) \quad (3)$$

$$histBackground = hist(intensity(p : label(p) = 0)) \quad (4)$$

$$histDiff = \sum_i \left(1 - \frac{min(histObjects[i], histBackground[i])}{max(histObjects[i], histBackground[i])}\right) \quad (5)$$

Overall fitness After generating the three individual quality measures, we calculate the overall quality of the whole image processing workflow as:

$$quality = \frac{cellCount * histDiff}{stdBackground} \quad (6)$$

This quality value (fitness) is used to compare the mutated solution candidates with each other to find the optimal solution.

2.4.4 Island ES optimization

Our goal is on the one hand to find the optimal set of parameters for the image processing algorithms and on the other hand to find the correct selection of processing algorithms to use. We have developed a parallel evolutionary optimization method that uses islands of individuals, which is based on the concept of island GAs (Whitley et al. 1999). The switch system of our solution candidates leads to 16 different algorithm combinations including the constraint that the thresholding always has to be active. These 16 islands are initialized with randomly set parameters.

Each of these islands gets optimized independently from the others, to provide parameters which are optimized not only for the algorithm itself, but also for the specific combination with the other algorithms. Otherwise, the optimization is distorted when an algorithm gets activated during one of the last iterations and there is not enough time left to optimize its parameters. The results from the individual islands are compared to each other by their fitness, and the worst performing ones are discarded.

Islands are often used to keep up the genetic diversity during optimization, but most implementations exchange candidates between the islands (Affenzeller et al. 2009) which is not useful for our approach because we use the islands to compare the quality of specific combination of algorithms.

As shown in Fig. 11, each island undergoes several optimization steps, but only the best of them are selected for further optimization iterations. The final solution delivered by this pipeline contains the optimal combination of algorithms, including their optimized parameters.

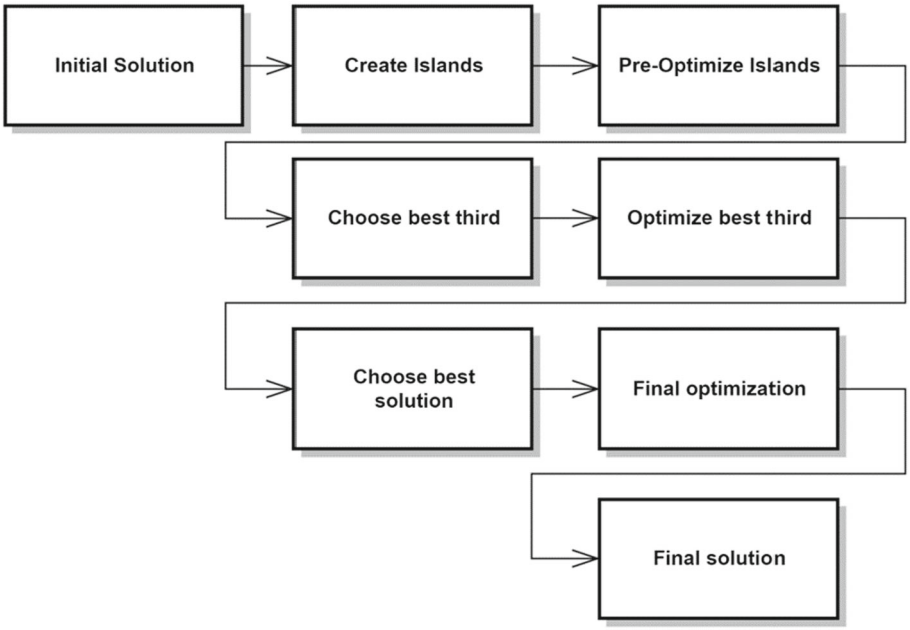

Fig. 11 An evolution strategy model using independent islands. The four levels represent the evolutionary steps done within each island independently from each other. After each level, all the Islands' best solution candidates are compared to each other. Like the normal evolution strategy, only the best $\times$ islands are optimized again in the next level until the best islands' candidates are optimized in the final iterations within the last level

Each of the islands uses an evolution strategy with one parent and five mutants per generation. All initial islands are optimized for two generations. The best third of them undergoes an optimization for an additional five iterations until the final solution is optimized alone for another five generations. The algorithm finishes if the last iteration is reached or if the quality improvement converges.

This island concept also enables high hardware scalability by distributing different islands to separate CPU cores. To improve performance even further, it is also possible to use a graphical processing units (GPUs) for the image processing calculations.

3 Empirical tests

3.1 Test setup

For a detailed evaluation of our algorithm, it is necessary to produce measures which can be compared to a generalized baseline. Two different types of images were used, which were manually segmented to get a baseline, by marking the individual cells or by using manually selected thresholds. These images have very different characteristics such that it should not be possible to use the same preprocessing algorithms and parameters for all sets of images. The manually generated segmentation results contain the information about which part of the image includes cell areas and which are representing background or noise. Because of certain characteristics of the individual images, the manual segmentation does not provide perfectly separated cells. The same images were segmented by workflows generated with the evolution strategy. Pixels are considered true positive if they are detected as cell pixels by the automatically generated workflow as well as the manual segmentation. This leads to the comparison measures listed in Table 2 which are based on the binary image representations of the images.

3.2 Results

3.2.1 16-Bit images of cell particles

These benchmark images provide a variety of different-sized objects to be segmented. The goal is to find a combination of processing algorithms and parameters that simultaneously work well for objects of all sizes, even those of very small size which might be mistaken for background noise. As shown in Fig. 12, there is a measurable difference between the pixels that are part of the cell areas and the ones that are part of the background of the image. The optional parameters for the size constraints were approximately set at 200 pixels for the minimum size and 5000 pixels for the maximum size, even if the actual size of the largest object is more around 1500 pixel. This was done to show that these parameters do not have to set perfectly for the workflow to produce good results. There will only be problems if the parameters eliminate actual cells or particles. For example, if the largest object were 6000 pixels, the parameter would be set to 5000.

Figure 13 shows the result of the cell detection. The optimization process only used the single training image, and the test image was segmented afterward using the generated parameter set. As listed in Table 3, this result used a combination of three algorithms. Due to the high amount of noise in the image, it is no surprise that the Gaussian filter is used for overall noise reduction. Afterward, a bandpass filter is used

Table 2 Comparison measures for the benchmark results

Measure	Description
Accuracy	Relative amount of correctly marked pixel
False positive rate	Wrongly marked cell pixels in relation to the total number of cell pixels
False negative rate	Wrongly marked background pixels in relation to the total number of background pixels

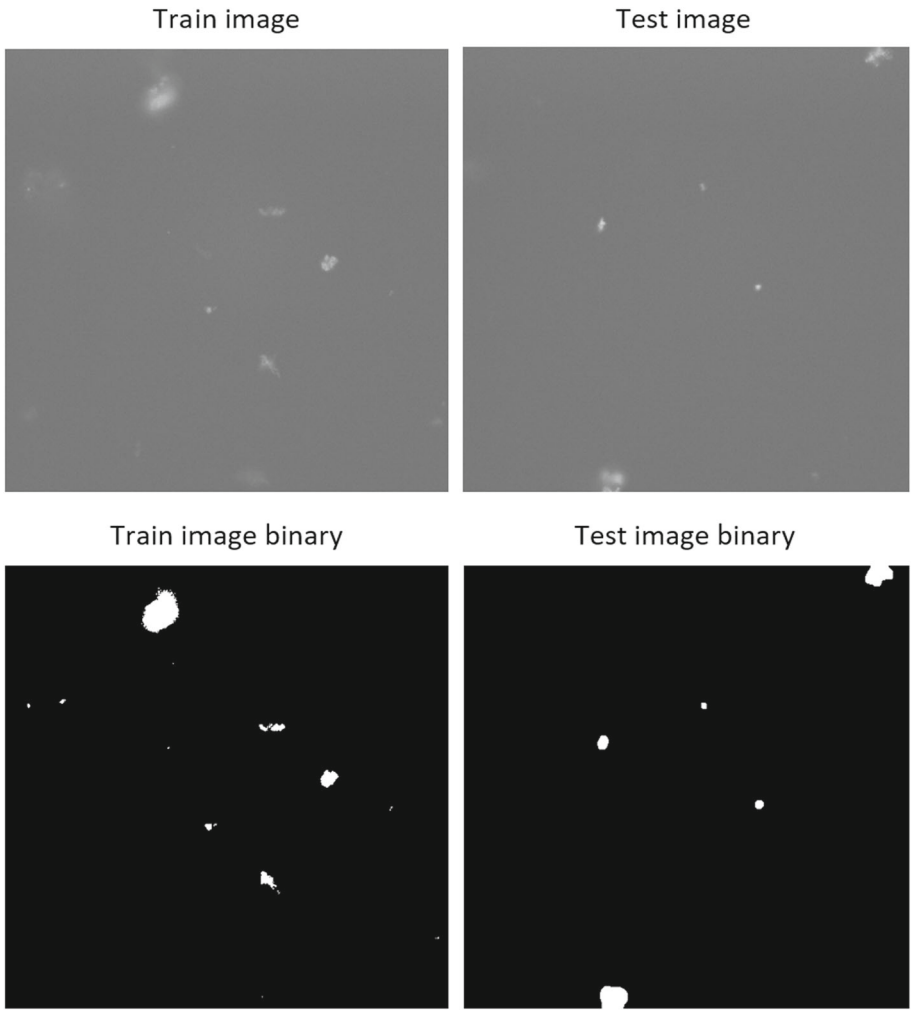

Fig. 12 Microscopy images of cell particles which their corresponding binary representation

to reduce the intensity range, especially the low intensities due to the lower cutoff frequency of 18812. Watershed segmentation acts as the final separation method for the borders between background and cell areas.

Table 4 shows the quality measures for the cell particle images. Both images were segmented with good quality. An overall error below one percent shows the potential of our method.

3.2.2 16-Bit images of DAPI marked cell nuclei

The first benchmark acted as a proof of concept for our automatically generated processing pipeline. To further demon-

Table 3 Evolution strategy workflow processing parameters for the cell particle images

Parameter	Value
Gauss' sigma	2.25
Lower bandpass cutoff frequency	18,812
Higher bandpass cutoff frequency	60,325
Binary threshold	0.43
Switch for Gauss filter	1
Switch for bandpass filter	1
Switch for watershed segmentation	1
Switch for histogram equalization	0

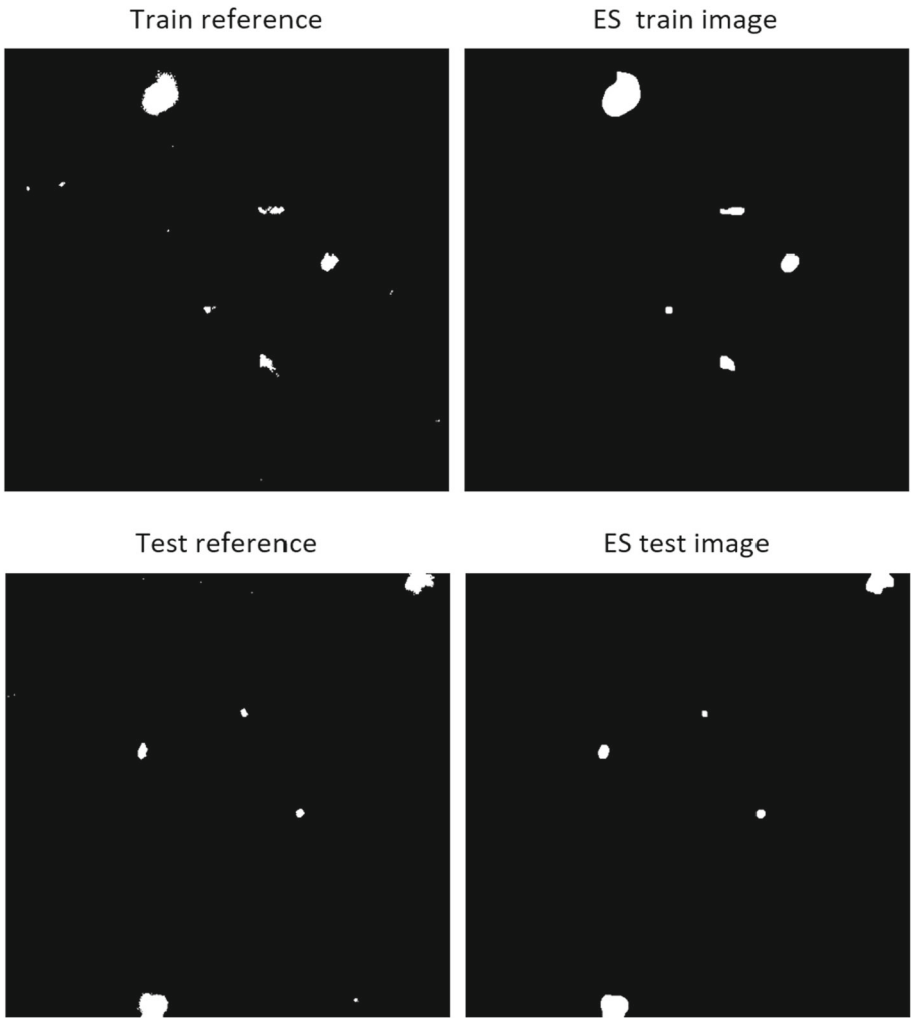

Fig. 13 Microscopy images of cell particles processed by the workflow generated with the evolution strategy in comparison with the manually segmented reference images

Table 4 Evolution strategy quality measures on cell particle images

Measure	Training result (%)	Test result (%)
Accuracy	99.77	99.89
False positive rate	0.18	0.05
False negative rate	5.99	10.07

strate the usefulness of our approach, we compare benchmark results from the generated workflow with the results from two types of state-of-the-art neural networks.

A convolutional neural network (Venkatesan et al. 2018) was built using three convolutional and pooling layer combinations followed by two fully connected layers. Splitting the one training image as shown in Fig. 14 into 20 × 20 pixel windows, this network was trained using a balanced dataset with 807216 samples. Each sample therefore contained the pixel intensities and the information about whether the center pixel with the coordinates (10,10) was part of a cell or not. The dataset was balanced to improve the classification accuracy of the neural network (Kotsiantis 2007) and prevent that the network is biased by one dominating class (Lemnaru and Potolea 2012). Using this dataset, it was possible to achieve an overall accuracy on validation data of 99.65 percent after 50 epochs of training.

For both, for the training of the convolutional neural network, as well as our parameter optimization approach, only one image was used. The difference between the neural network training and the parameter optimization is that our approach does not net a binary representation of the image for this optimization. The presented binary images are only necessary for the testing purpose of this paper. Therefore, it is possible to get a better comparison of these two methodologies in situations where the amount of data is highly limited.

In addition to this sliding window neural network approach, we compared our workflow with the state-of-the-art segmentation methodology called u-net (Sommer et al. 2011). Due to the lack of data, this network could only be trained with 33 images and therefore could only achieve a validation

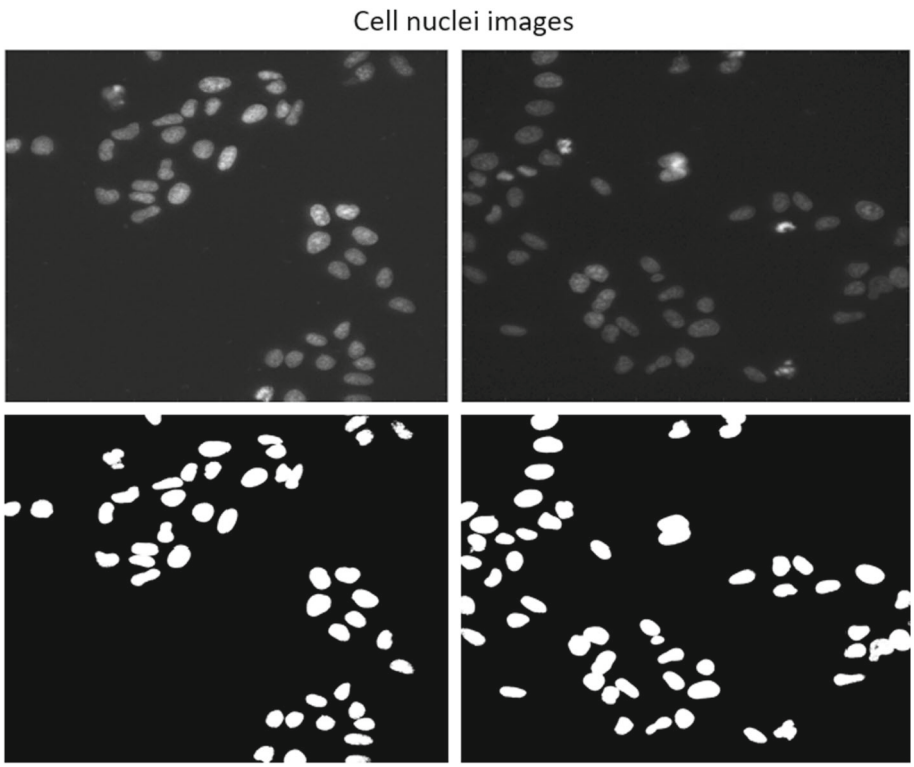

Fig. 14 Microscopy images for testing the algorithms of cell nuclei colored with DAPI and their corresponding binary representation

Table 5 Comparison of quality measures on the cell nuclei images

	ES (%)	Neural Net (%)	U-Net (%)
Accuracy	98.12	99.14	97.28
False positive rate	2.04	0.93	2.98
False negative rate	0.02	0.05	0.0

accuracy of 88.85 percent using different data augmentation strategies (Wong et al. 2016).

This comparison might be somewhat biased due to the low accuracy value of the u-net. However, considering that the main benefit of our approach is its applicability even on very small sets of images, this bias reflects real-world use cases, where lack of data is often an issue.

All three of these methods were tested using the same test images which were not used for the training and optimization process. The optional parameter for the size constraint was again set to a minimum value of 200 and a maximum value of 5000.

The results listed in Table 5 are generated out of the average values achieved by the classification of both test images. These measures show an overall detection accuracy which is quite high. All of the tested approaches achieve an accuracy of above 97 percent looking at the quality measures.

The sliding window neural network was be able to outperform the evolution strategy. However, it takes way more calculations to classify each pixel within such a high-resolution image. The neural network in a sliding window artificially extends the image to be able to also classify the pixels on the borders. This methodology requires 4214784 classifications for one 2028x2048 pixel image. Concerning the extensive amount of calculations needed for each of these classifications due to the number of weights used in the neural network architecture, it is clear that an application of several processing filters is still more efficient, albeit with a slight loss of accuracy. Considering that the overall accuracy of our method on this data was higher than the one achieved using the u-net, it can be said that this method is not recommended for such small amounts of data.

When inspecting the final detection results in Fig. 15, we can see that the only significant difference between the results from the different methods is in the upper right corner where the sliding window delivered a slightly better segmentation than the evolution strategy. This explains the 1 percent increase in accuracy.

Even though the sliding window neural network slightly outperformed the evolution strategy in terms of quality, the high amount of preprocessing needed to get an appropriate model for the classification arguably does not justify this increase in accuracy. In this example, the processing needed to get an appropriate model from the labeled image took 8.5 minutes, not including the time which is necessary to create the labeled windows out of the manually segmented image. The whole evolution strategy process took 7.5 minutes on the same hardware. Each image classification using the trained

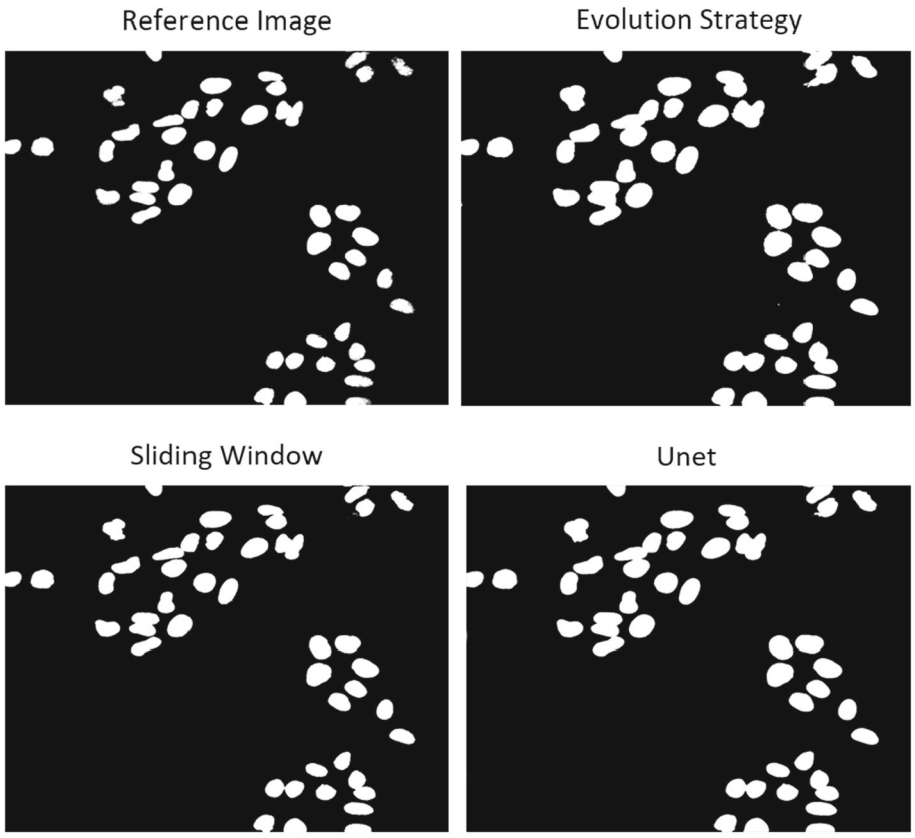

Fig. 15 Comparison of cell nuclei segmentation results on the test image

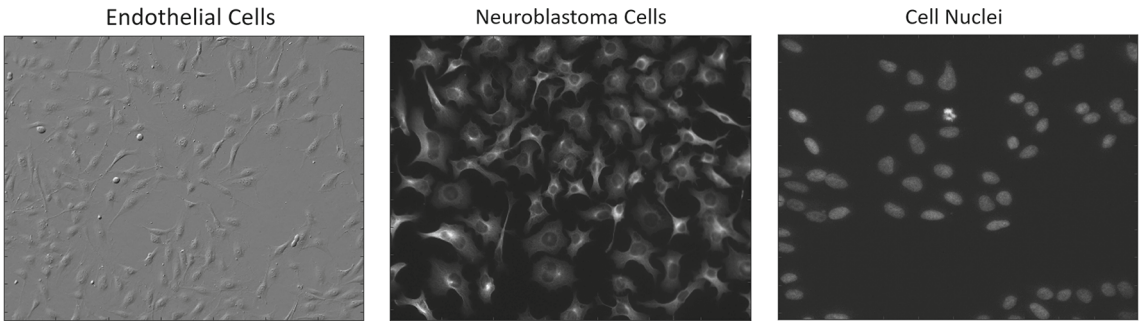

Fig. 16 Selected images for the robustness tests. left: endothelial cells, middle: neuroblastoma cells, right: cell nuclei

sliding window neural network takes nearly a minute due to the high amount of classifications, whereas the application of the generated set of algorithms only took 7 seconds even when all algorithms were applied together which represents the worst case.

3.3 Evolutionary algorithm parameters analysis

Finally, we performed tests with varying parameters for the evolutionary algorithm to test the robustness of our method. The goal of these tests was to measure the impact of different population sizes (i.e., different settings for μ and λ) on the resulting detection quality. For these tests, we used three types of images with differently shaped cell objects and varying light conditions as shown in Fig. 16. The final quality measures are calculated on test images that were not used for parameter optimization during the training phase. For these tests, the optional size constraint (which was parameterized once for each of the three image types) was applied. All algorithm settings were used in 5 independent runs; μ and λ were both varied between 1 and 50; the following result measures represent the mean of these 5 runs per image type.

As shown in Tables 6, 7, 8 and 9, the evolution strategy is able to produce good results if μ and λ are both greater than 1. This is as expected; the more the individuals are in the population, the more the algorithm is able to diversify the individuals within the population, which is necessary for finding the optimal solution.

Table 6 Comparison of quality measures depending on the evolution strategy parameter applied on endothelial cell images

Mu	Lambda	Mean test quality	Test accuracy (%)	False positive rate (%)	False negative rate (%)	Relative runtime (%)
1	1	0.78	99.69	0.08	37.35	100.00
	5	4.46	96.98	3.02	2.13	272.94
5	5	2.51	96.80	3.21	0.66	293.46
	10	3.62	97.00	3.00	2.56	520.67
	20	3.67	96.03	3.99	1.22	909.15
10	10	3.67	96.03	3.99	1.22	527.39
	20	4.46	97.79	2.21	2.27	956.32
	50	6.07	98.89	1.09	3.94	2171.68
20	50	6.88	99.84	0.14	4.85	2301.89

Table 7 Comparison of quality measures depending on the evolution strategy parameter applied on neuroblastoma cell images

Mu	Lambda	Mean test quality	Test accuracy (%)	False positive rate (%)	False negative rate (%)	Relative runtime (%)
1	1	134.44	86.65	6.76	15.02	100.00
	5	131.40	86.03	6.92	15.96	407.08
5	5	142.54	74.71	21.48	14.83	472.07
	10	142.88	86.59	6.71	15.22	843.51
	20	139.59	85.97	6.85	16.19	1630.86
10	10	139.80	74.29	22.60	13.88	930.66
	20	170.87	74.27	22.07	14.75	1689.00
	50	203.11	72.98	24.50	13.40	4062.02
20	50	173.86	73.72	27.64	7.05	4291.07

Table 8 Comparison of quality measures without using the size constraint, depending on the evolution strategy parameter applied on neuroblastoma cell images

Mu	Lambda	Mean test quality	Test accuracy (%)	False positive rate (%)	False negative rate (%)	Relative runtime (%)
1	1	25,945.69	69.28	35.14	3.82	100.00
	5	54,753.36	85.90	6.19	17.36	393.82
5	5	56,258.31	86.02	6.27	17.01	448.97
	10	105,730.37	79.16	16.53	14.07	806.45
	20	64,688.61	77.90	19.56	11.72	1525.91
10	10	62,215.08	86.51	7.53	14.07	863.92
	20	62,133.24	87.10	8.87	10.84	1580.96
	50	71,956.54	86.87	8.01	12.65	3751.95
20	50	97,781.96	76.35	20.75	12.84	3945.14

We can see a quite high mean quality value compared to the runs with the constraint 7. This is the case due to the previously mentioned wrong dividing lines which artificially increase the number of detected cells (which are wrongly detected). The equally high accuracy values show that the real-world quality did not improve compared to the version with the constraint

Even though the population should not be too small, it seems to be optimal (at least for these images) to set μ to 5 and λ to 10, e.g., as these settings lead to very good results without consuming too much runtime. Depending on the available resources, it is also possible to slightly increase the quality in some cases by increasing the population size, even if the needed runtime then rises significantly.

4 Conclusion

The here presented workflow allows for the automatic generation of specific processing pipelines for various types of microscopy images without the need for expert knowledge in the field of image processing. These pipelines are capable of delivering nearly equivalent quality level compared to

Table 9 Comparison of quality measures depending on the evolution strategy parameter applied on cell nuclei images

Mu	Lambda	Mean test quality	Test accuracy (%)	False positive rate (%)	False negative rate (%)	Relative runtime (%)
1	1	9916.56	88.49	12.89	0.03	100.00
	5	10,263.27	88.39	13.01	0.03	396.69
5	5	10,244.81	88.46	12.93	0.03	451.61
	10	10,252.24	88.48	12.91	0.03	817.77
	20	10,265.17	88.41	12.99	0.03	1588.34
10	10	10,257.78	88.42	12.97	0.03	935.84
	20	10,263.48	88.47	12.92	0.03	1658.02
	50	10,269.10	88.42	12.97	0.03	3905.51
20	50	10,268.69	88.39	13.01	0.03	3976.81

specially trained neural networks without the need for costly data preparation.

The flexibility of this approach by adapting the fitness function and the set of used processing algorithms for the specific use case is one of the main advantages over methods like neural networks which only perform well for specially trained use cases. For future implementations, this workflow can easily be scaled depending on the available hardware. Particularly, the optimization of the independent islands scales on every kind of equipment available. Due to the proven robustness of our methodology, it is not necessary to use high population sizes for the evolution strategy which again reduces the overall processing time without a significant impact in the resulting segmentation quality. The use of GPUs is also possible by the fact that optimization of individual islands consists mostly of image processing operations which can be run on the GPU.

Connected components labeling in combination with our workflow will constitute the segmentation base for our future research. The goal of this project is an automated cell tracking pipeline, which allows the user to gather detailed information about the dividing and movement behavior of individual cells.

Acknowledgements Open access funding provided by University of Applied Sciences Upper Austria. The work described in this paper was supported by the Center of Excellence for Technical Innovation in Medicine (TIMED) as well as the Basic Research Programme of the University of Applied Sciences Upper Austria, and by the Austrian Science Fund FWF (project Nr. P 31743-B30).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Affenzeller M, Wagner S (2005) Offspring selection: a new self-adaptive selection scheme for genetic algorithms. In: Adaptive and natural computing algorithms

Affenzeller M, Winkler S, Wagner S, Beham A (2009) Genetic algorithms and genetic programming: modern concepts and practical applications

Al-Kofahi Y, Zaltsman A, Graves R, Marshall W, Rusu M (2018) A deep learning-based algorithm for 2-D cell segmentation in microscopy images. BMC Bioinform. https://doi.org/10.1186/s12859-018-2375-z

Back T, Hoffmeister F, Schwefel H-P (1991) A survey of evolution strategies. In: Proceedings of the fourth international conference on genetic algorithms

Beucher S (1992) The watershed transformation applied to image segmentation. In: Proceedings of the 10th Pfefferkorn conference on signal and image processing in microscopy and microanalysis

Butterworth S (1930) On the theory of filter amplifiers. Exp Wirel Wirel Eng. https://doi.org/citeulike-article-id:5322726

Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL (2018) DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans Pattern Anal Mach Intell. https://doi.org/10.1109/TPAMI.2017.2699184

Deng G, Cahill LW (2005) An adaptive Gaussian filter for noise reduction and edge detection. https://doi.org/10.1109/nssmic.1993.373563

El Aziz MA, Ewees AA, Hassanien AE (2016) Hybrid swarms optimization based image segmentation. In: Hybrid soft computing for image segmentation

Falk T, Mai D, Bensch R, Çiçek Ö, Abdulkadir A, Marrakchi Y et al (2019) U-Net: deep learning for cell counting, detection, and morphometry. Nat Methods. https://doi.org/10.1038/s41592-018-0261-2

Halwa SR, Wójtowicz J, Szuman J, Pawlak B, Adamczyk W, Lorkiewicz Z (2013) A review on Otsu image segmentation algorithm. Kardiologia Polska

Hawkins SF, Thachil J, Hill QA (2014) Leukopenia. Haematol Crit Care A Pract Handb. https://doi.org/10.1002/9781118869147.ch2

Kotsiantis SB (2007) Supervised machine learning: a review of classification techniques. Informatica (Ljubljana). https://doi.org/10.31449/inf.v31i3.148

Lemnaru C, Potolea R (2012) Imbalanced classification problems: systematic study, issues and best practices. Lecture notes in business information processing. https://doi.org/10.1007/978-3-642-29958-2_3

Michalewicz Z (2013) Evolution strategies and other methods. In: Genetic algorithms + data structures = evolution programs

Soille P (1999) Morphological image analysis: principles and applications. Springer, Berlin

Sommer C, Straehle C, Kothe U, Hamprecht FA (2011) Ilastik: Interactive learning and segmentation toolkit. Proc Int Symp Biomed Imaging. https://doi.org/10.1109/ISBI.2011.5872394

Sun G, Zhang A, Wang Z (2016) Grayscale image segmentation using multilevel thresholding and nature-inspired algorithms. https://doi.org/10.1007/978-3-319-47223-2_2

Torbert Shane (2013) Applied computer science. Springer, Berlin

Venkatesan R, Li B, Venkatesan R, Li B (2018) Convolutional neural networks. Convol Neural Netw Vis Comput. https://doi.org/10.4324/9781315154282-4

Whitley D, Rana S, Heckendorn RB (1999) The island model genetic algorithm: on separability, population size and convergence. J Comput Inf Technol

Wong SC, Gatt A, Stamatescu V, McDonnell MD (2016) Understanding data augmentation for classification: when to warp? In: 2016 International conference on digital image computing: techniques and applications, DICTA 2016. https://doi.org/10.1109/DICTA.2016.7797091

Xia X, Kulis B (2017) W-Net: A deep model for fully unsupervised image segmentation. arXiv:1711.08506

Yang J, Chang E, Cherry AM, Bangs CD, Oei Y, Bodnar A et al (1999) Human endothelial cell life extension by telomerase expression. J Biol Chem. https://doi.org/10.1074/jbc.274.37.26141

Yu W, Lee HK, Hariharan S, Bu WY, Ahmed S (2019) CCDB:6843, mus musculus, Neuroblastoma. CIL. Dataset. https://doi.org/10.7295/W9CCDB6843

Zuiderveld K (2013) Contrast limited adaptive histogram equalization. Graph Gems. https://doi.org/10.1016/b978-0-12-336156-1.50061-6