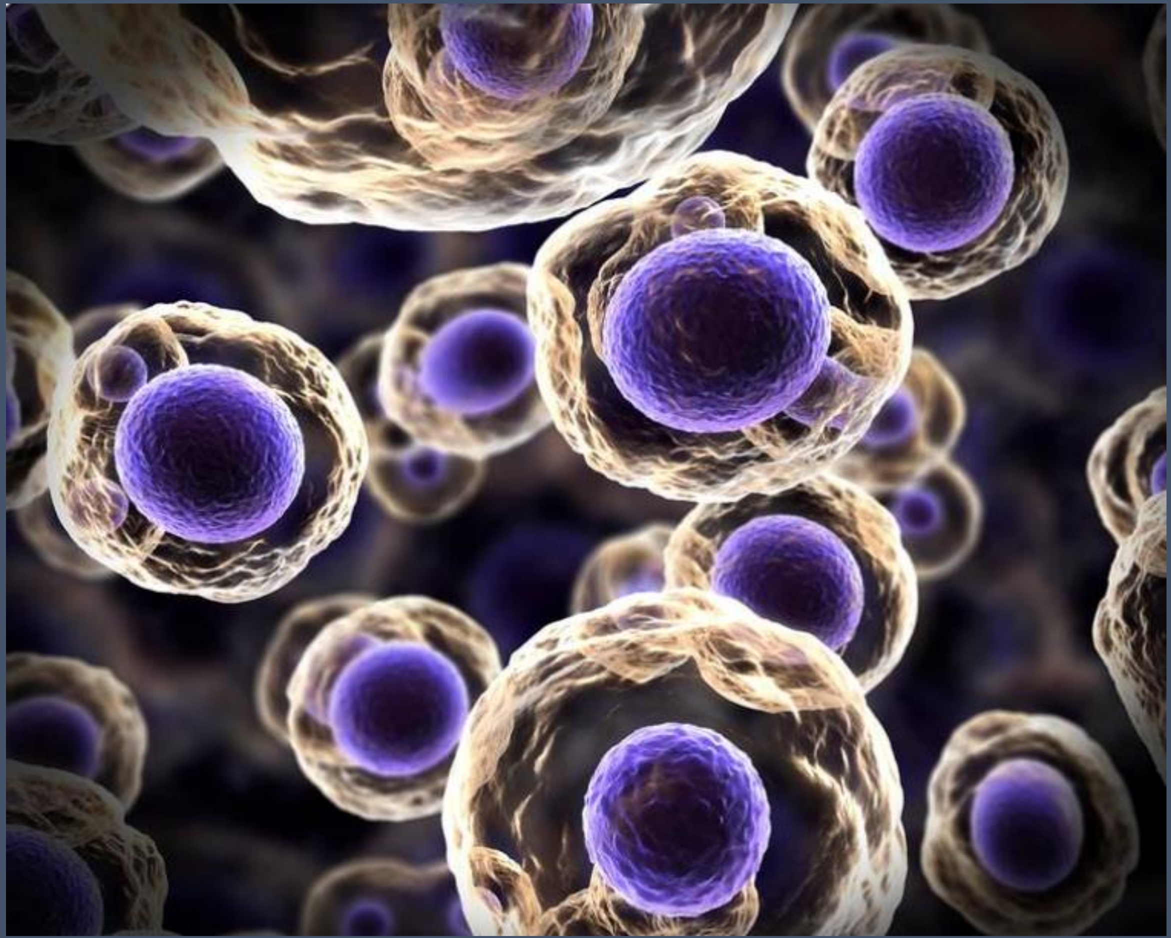
Implementation and evaluation of Otsu's thresholding

Final presentation

Elizaveta Chernova, Veronika Schuler, Laura Wächter, Hannah L. Winter

21.07.2021



Cell nuclei segmentation

https://bio.libretexts.org/Bookshelves/Introductory_and_General_Biology/Book%3A_Introductory_Biology_(CK-12)/02%3A_Cell_Biology/2.08%3A_Cell_Nucleus

Timeline

12.05 – Project Proposal
Research Otsu Tresholding and Dice Score
Team

19.05 – Explore Data
Histograms and Data Analysis
Team

26.05 – Code algorithms
Otsu: E.C., H.W.
Dice: L.W., V.S.

02.06 – Implement Preprocessing
Histogram stretching
Gauss filter
Median filter
Team

09.06 – Implemented
MSD, Hausdorff: L.W., V.S.
Cell nuclei counting: E.C., H.W.

30.06 – Testing of preprocessing combinations
Team
Optimized and tidied up the code
Team

14.07 – Cooperation
With Team 4.5
Team

19.07 – Report
Jupyter Notebook
Team

21.07 – Final presentation
Team

Workflow

01 Otsu‘s Thresholding

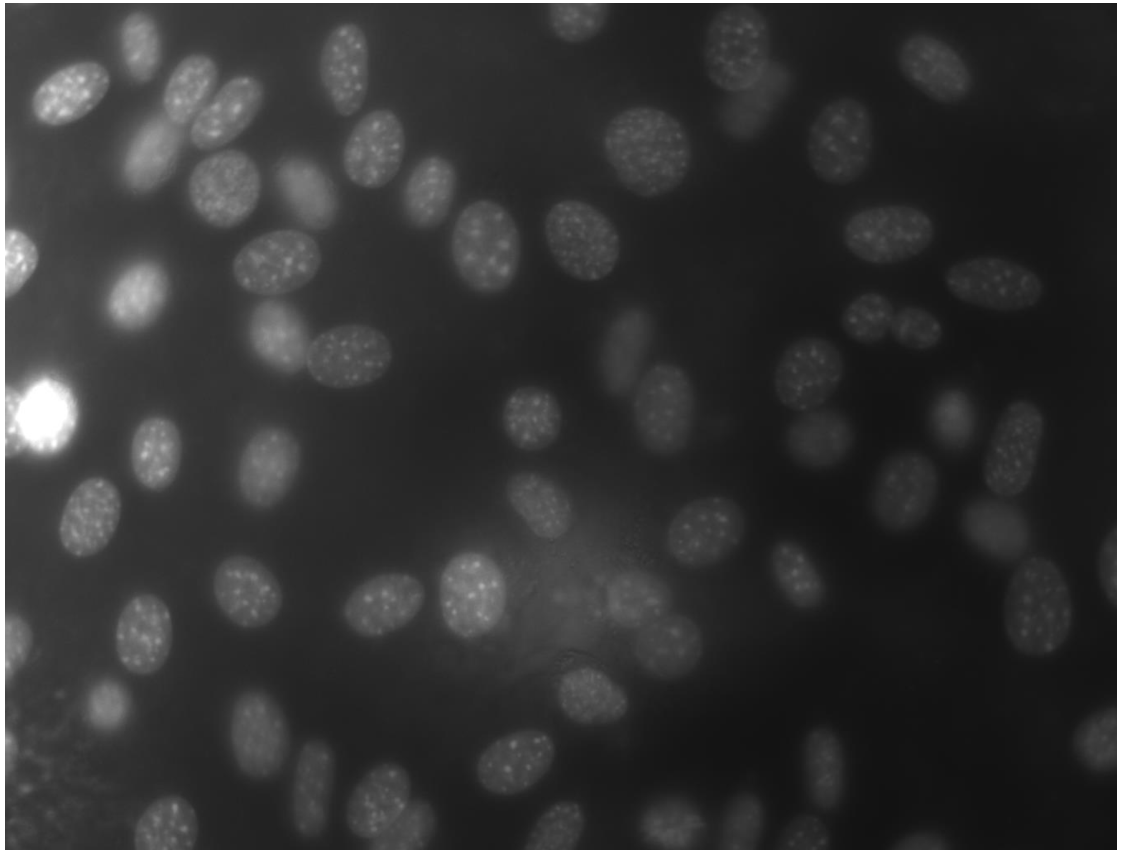
02 Evaluation methods

03 Preprocessing

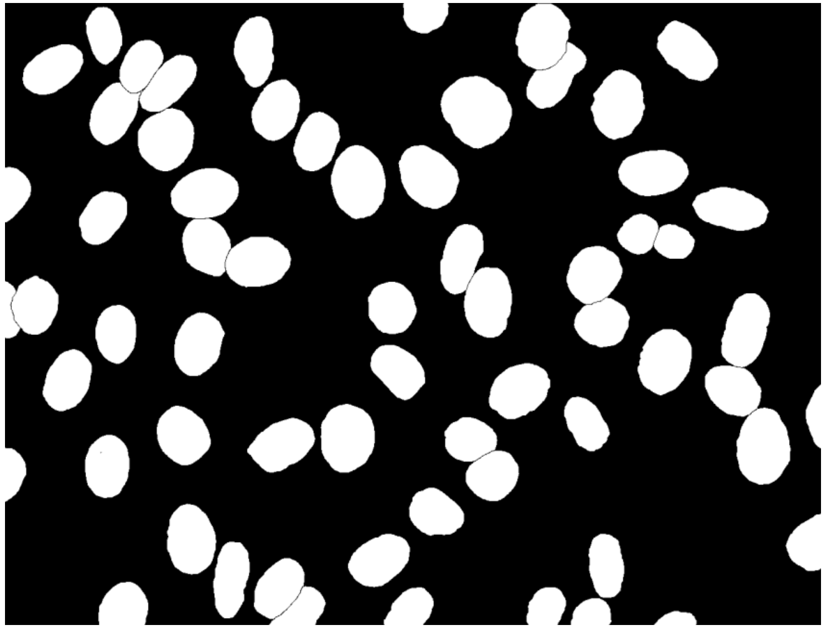
04 Cell nuclei counting

05 Evaluation

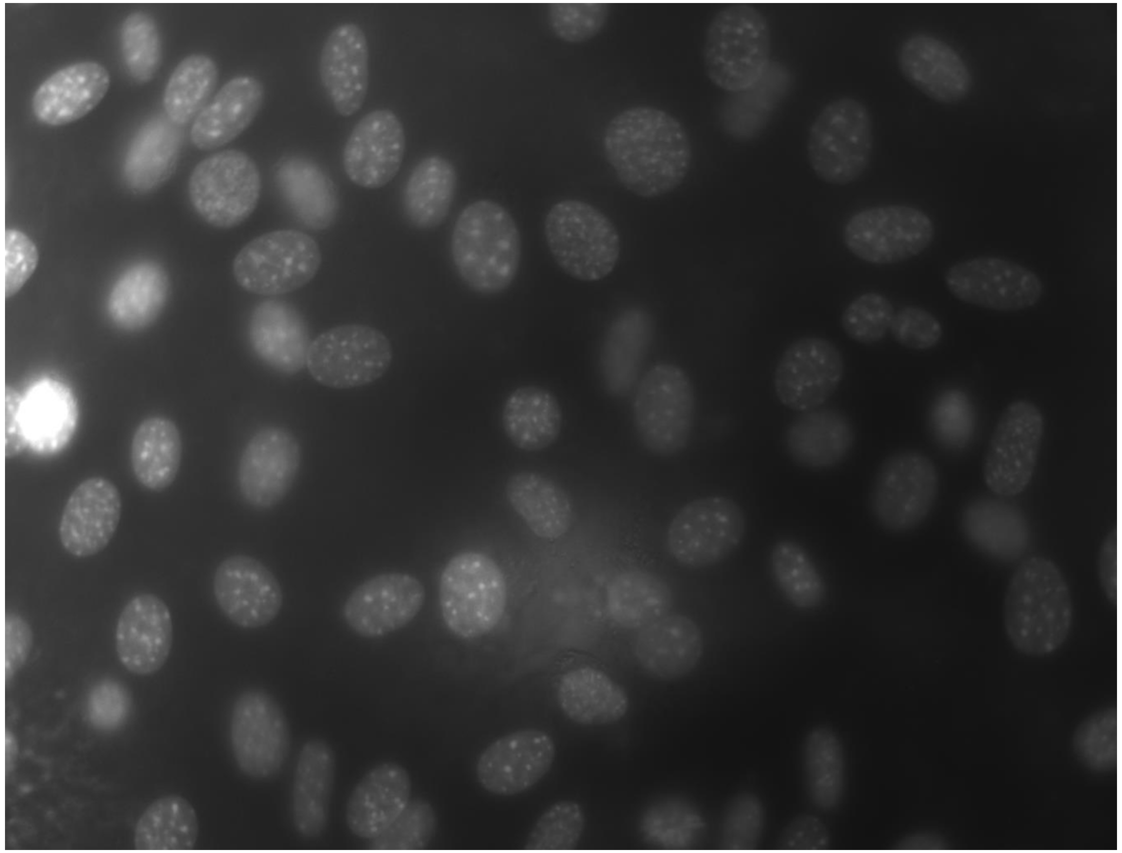
Otsu's Thresholding

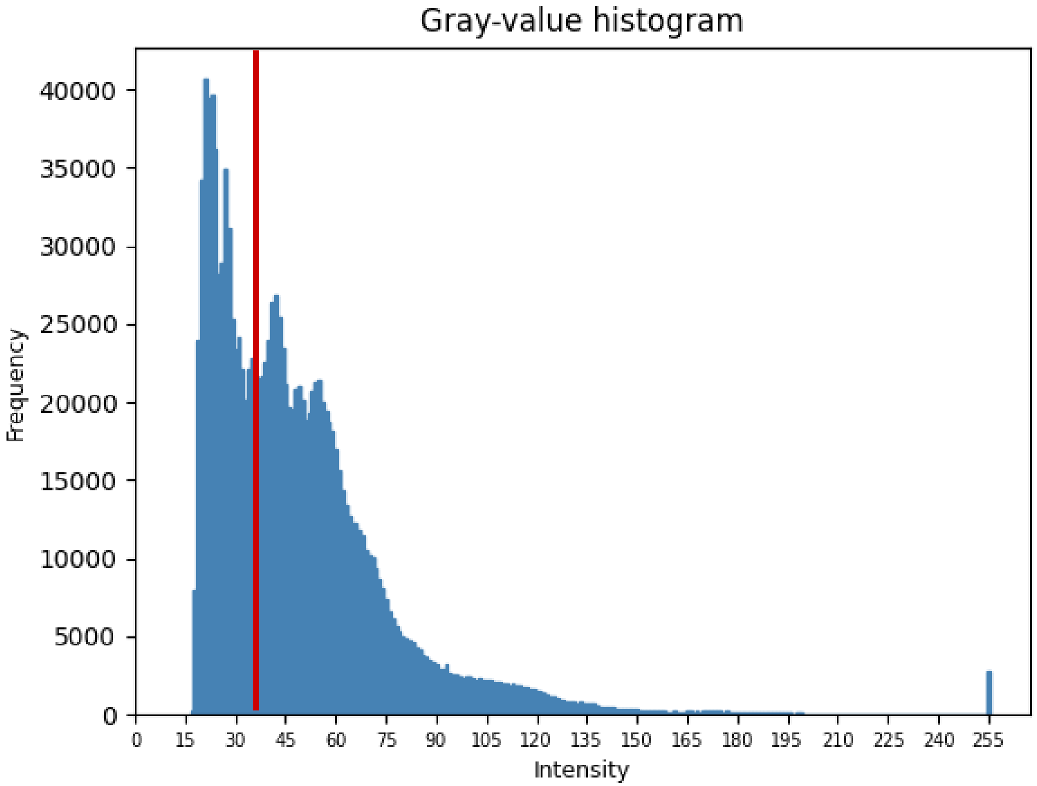


?



Otsu's Thresholding





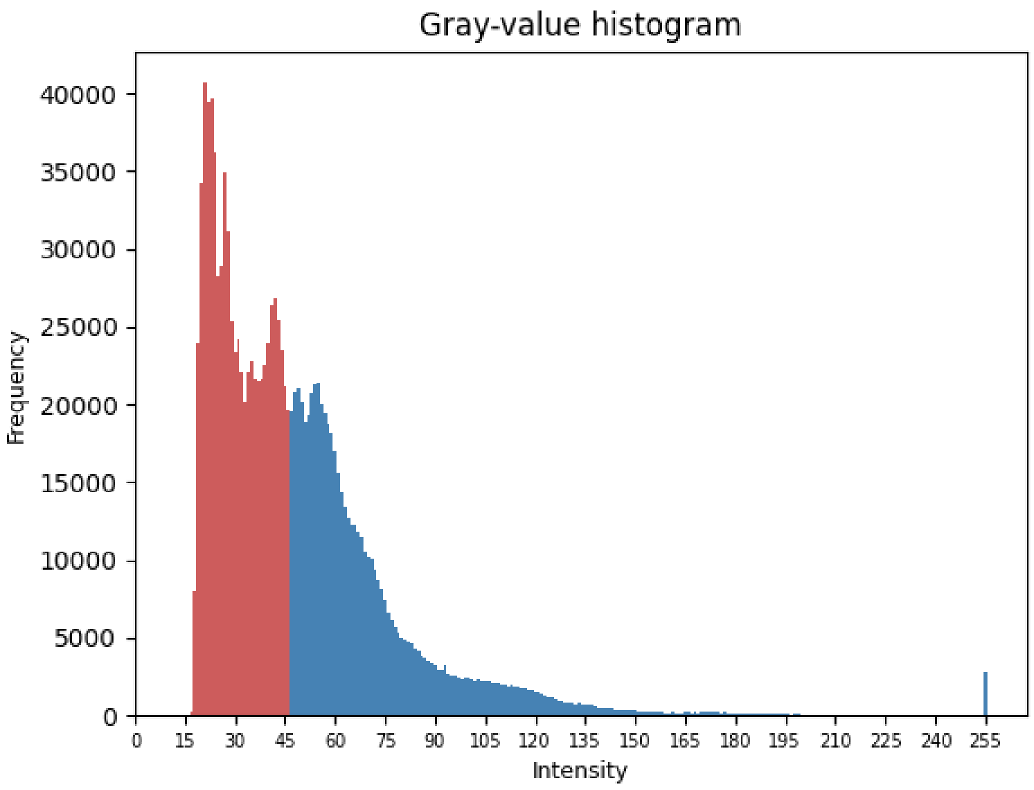
Threshold value k $\in [0,255]$

Otsu's Thresholding

Between-class variance

$$\sigma_B = \omega_0\omega_1(\mu_1 - \mu_0)^2$$

$\omega_{0,1}$ = probability of class occurrence
$\mu_{0,1}$ = mean intensity values



Threshold value $k \in [0,255]$

Otsu's Thresholding

```
histogram = np.histogram(image, bins=np.arange(intensity_lvls + 1), density=True)

class_probability = np.cumsum(histogram[0])
class_mean = np.cumsum(histogram[0] * np.arange(intensity_lvls))
total_mean = np.mean(image)

with np.errstate(divide='ignore'):
    inbetween_variance = (total_mean * class_probability - class_mean) ** 2 / (
            class_probability * (1 - class_probability))

# Inf values are invalid
inbetween_variance[inbetween_variance == np.inf] = np.nan
optimal_threshold = np.nanargmax(inbetween_variance)

return optimal_threshold
```

$$\frac{n_i}{N}$$

$$\omega(k) = \sum_{i=1}^{k} \frac{n_i}{N}$$

$$\mu(k) = \sum_{i=1}^{k} \frac{n_i}{N} i$$

$$\mu_T$$

Otsu's Thresholding

```
histogram = np.histogram(image, bins=np.arange(intensity_lvls + 1), density=True)

class_probability = np.cumsum(histogram[0])
class_mean = np.cumsum(histogram[0] * np.arange(intensity_lvls))
total_mean = np.mean(image)

with np.errstate(divide='ignore'):
    inbetween_variance = (total_mean * class_probability - class_mean) ** 2 / (
            class_probability * (1 - class_probability))

# Inf values are invalid
inbetween_variance[inbetween_variance == np.inf] = np.nan
optimal_threshold = np.nanargmax(inbetween_variance)

return optimal_threshold
```

$$\sigma_B^2 = \frac{(\mu_T\omega(k) - \mu(k))^2}{\omega(k)(1-\omega(k))}$$

$$\max(\sigma_B^2)$$

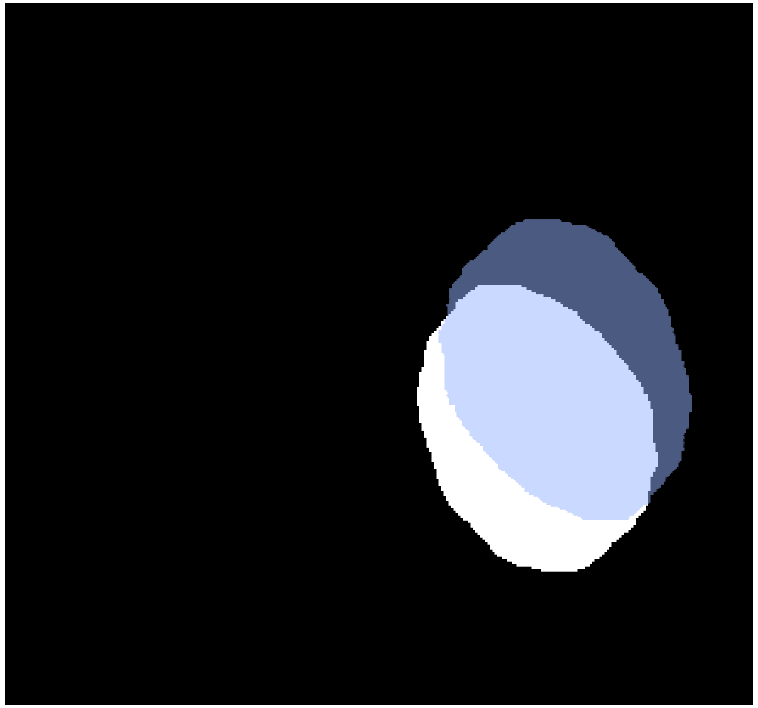
Implementation of the Dice Score

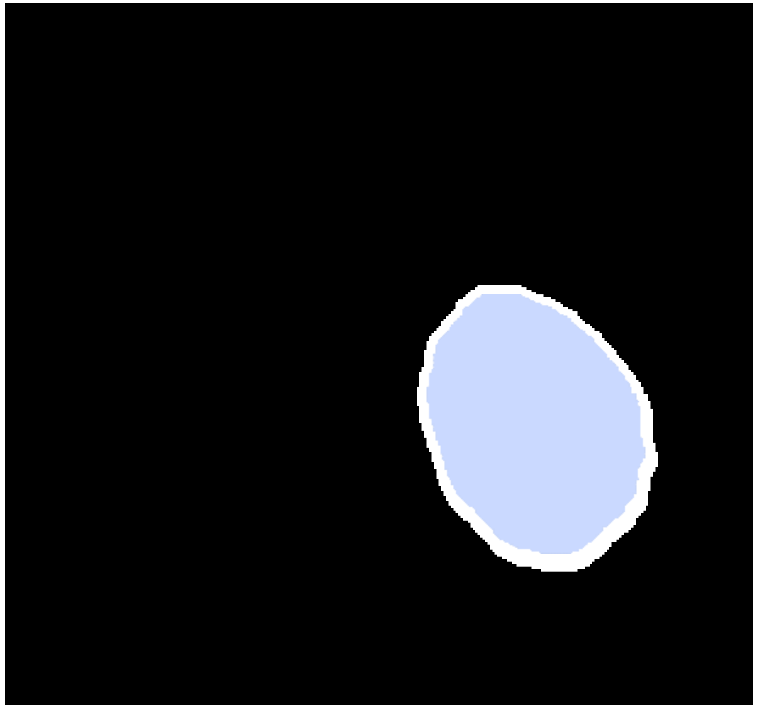
```python
def dice(clipped_image, ground_truth):

    # Assign 1 to all pixels, that have a non-zero intensity
    work_gt[ground_truth!= 0] = 1
    work_clipped[clipped_image != 0] = 1

    intersection = np.sum(work_clipped * work_gt)
    sum_all = np.sum(work_clipped) + np.sum(work_gt)
    dice_score = (2 * intersection) / sum_all

    return dice_score
```

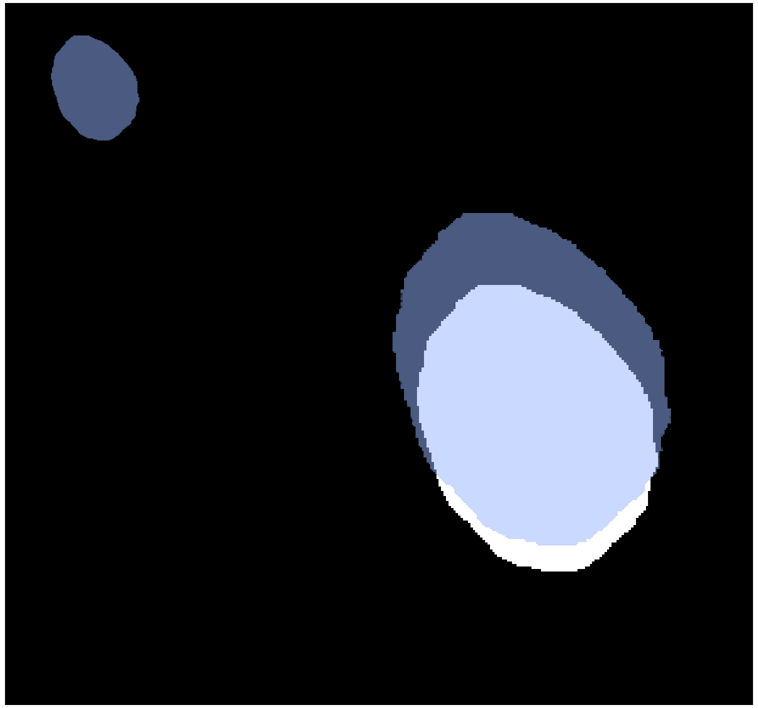
to make the images binary

$$DSC = \frac{2 \times |A \cap B|}{|A| + |B|}$$

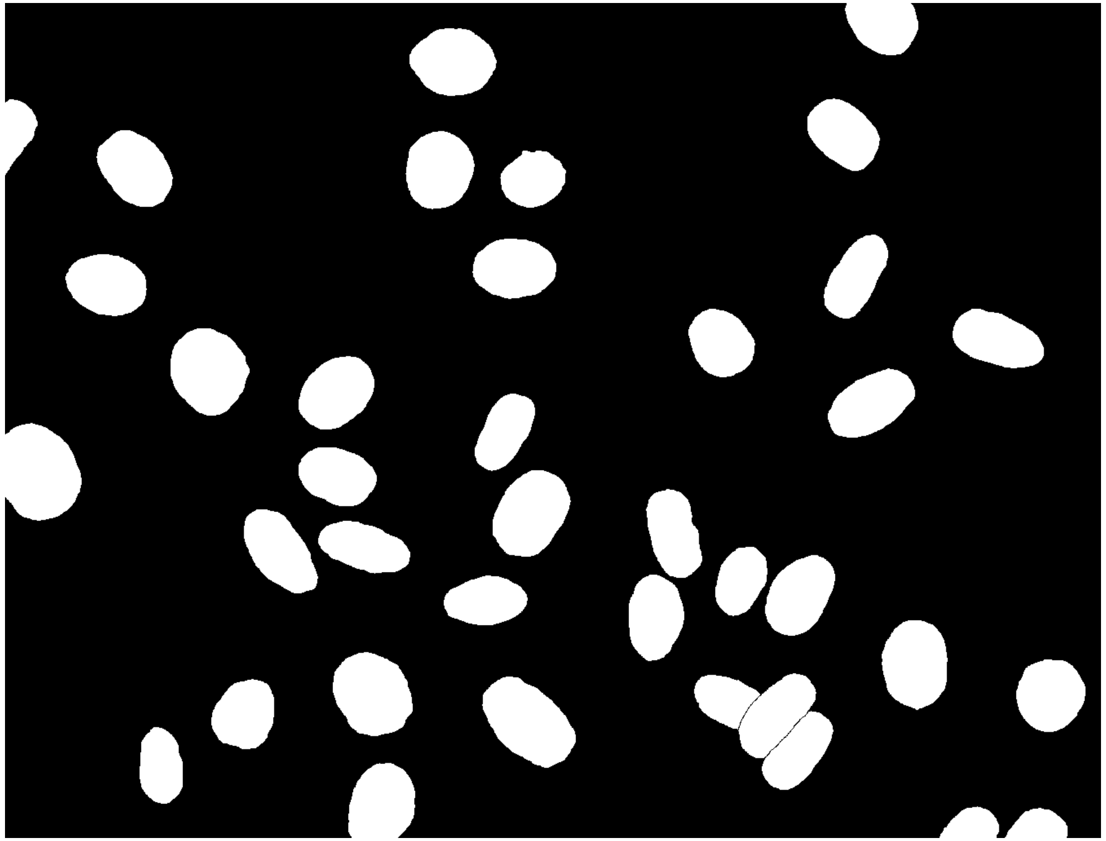
DSC vs. MSD vs. HD

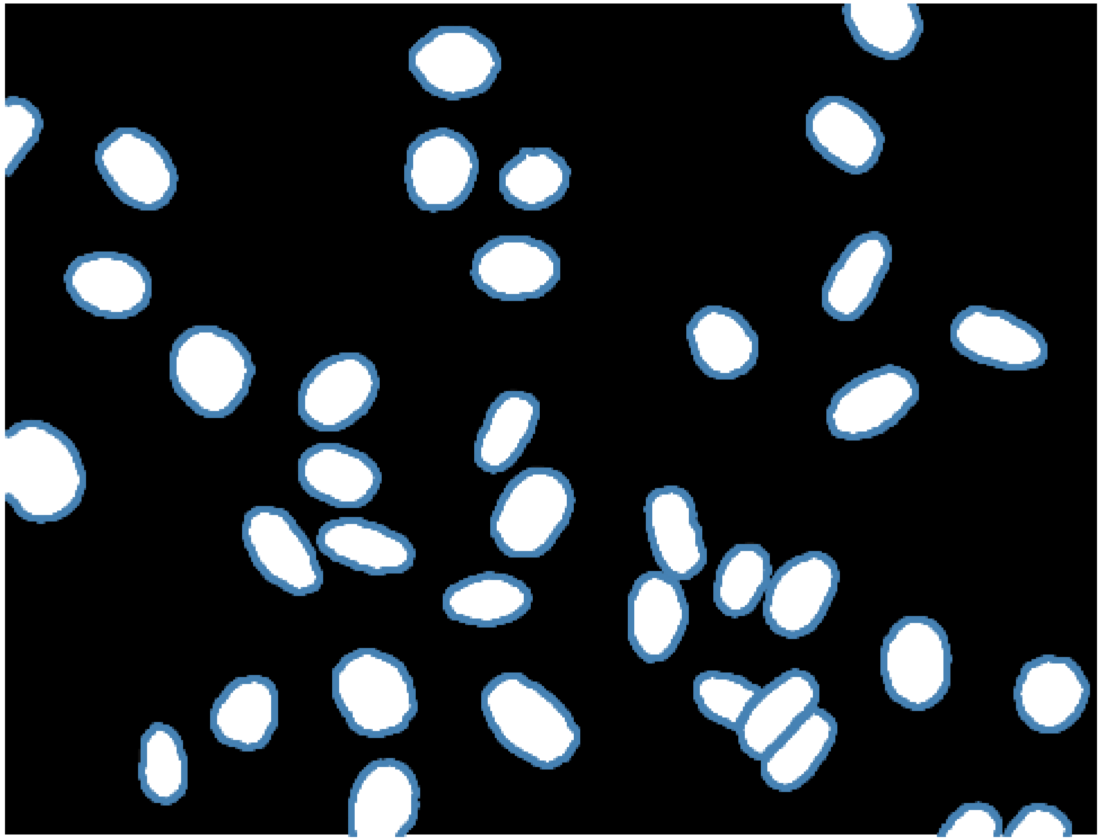




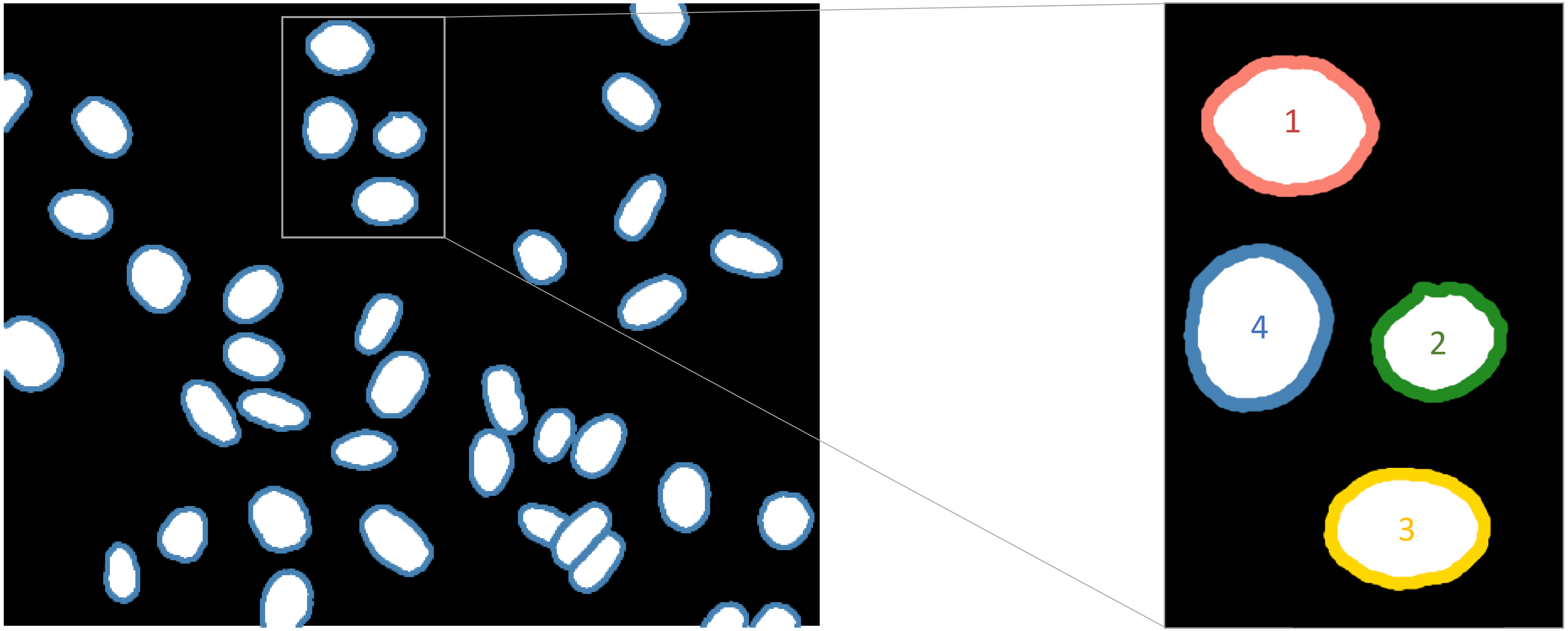


Cell Counting

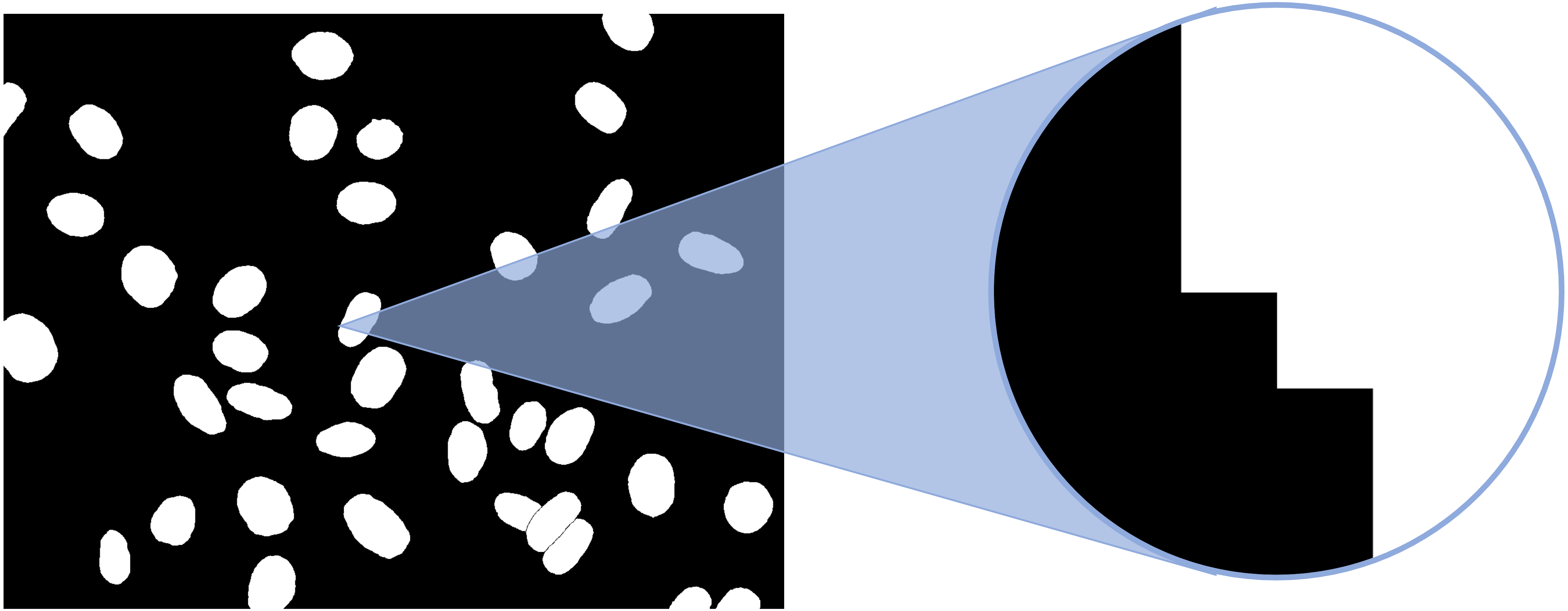




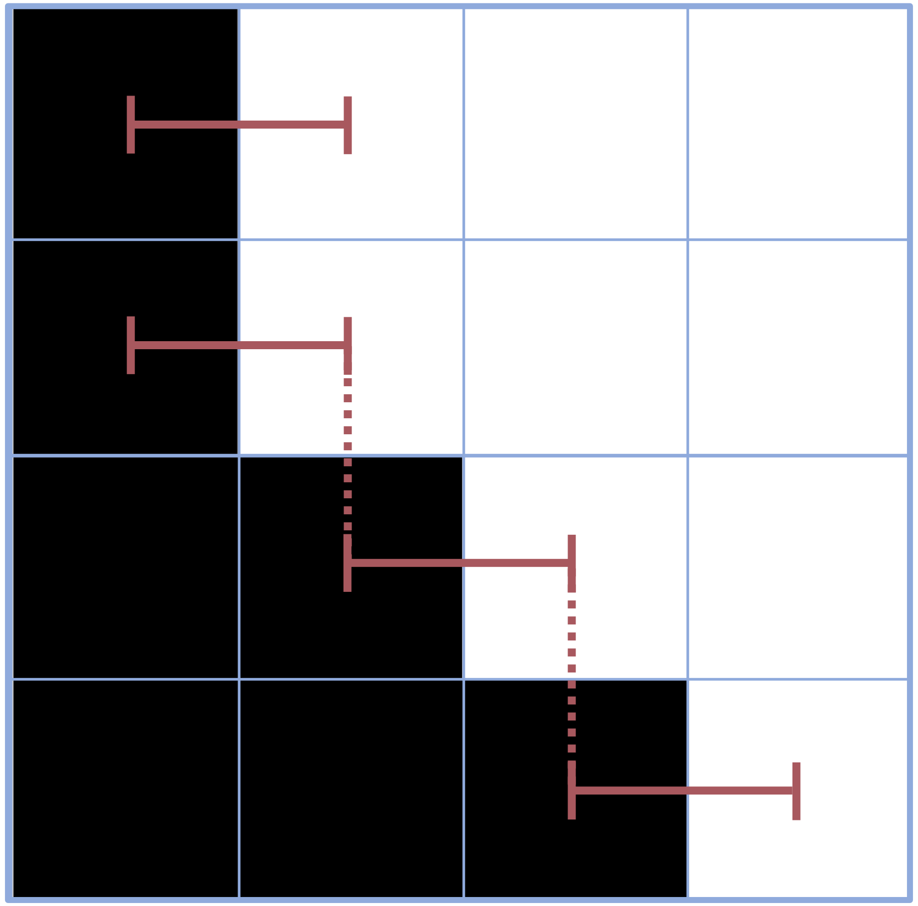
Cell Counting



Cell Counting



Cell Counting



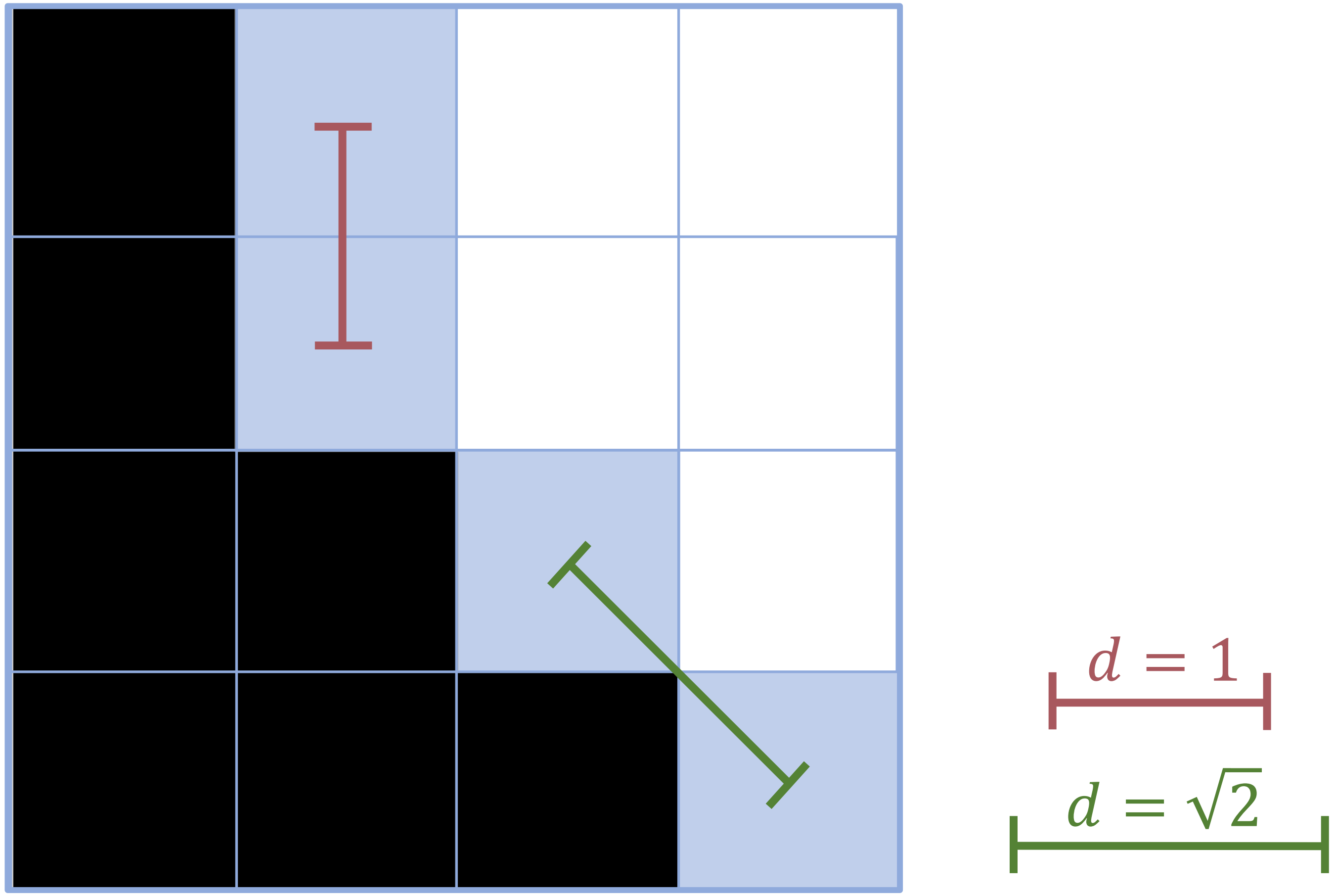
$d = 1$

Cell Counting

```
edge_pixels = []

for index in np.ndindex(img.shape):
    if workimg[index[0]][index[1]] == 1:
        if 0 in workimg[(index[0] - 1):(index[0] + 2), (index[1] - 1):(index[1] + 2)]:
            edge_pixels.append(index)

return edge_pixels
```

$d = 1$

Cell Counting

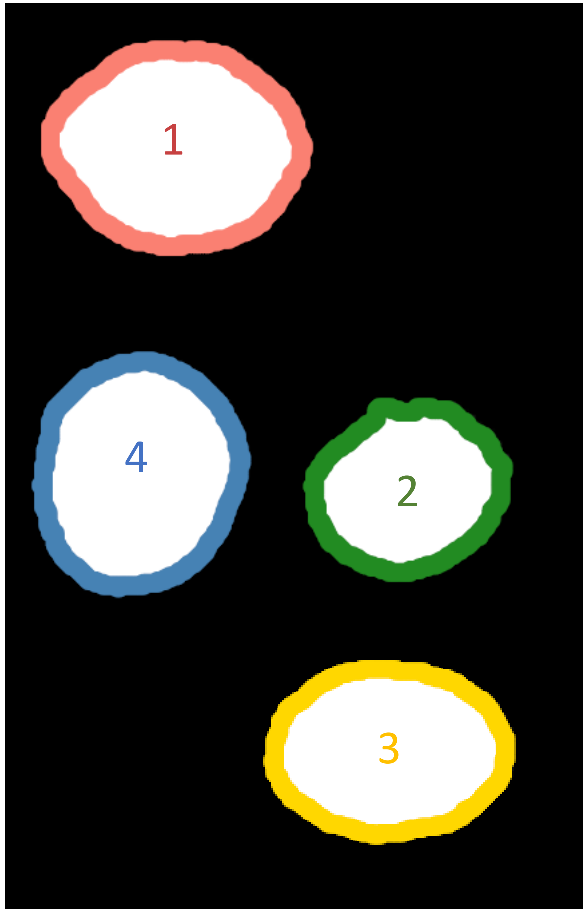


Cell Counting

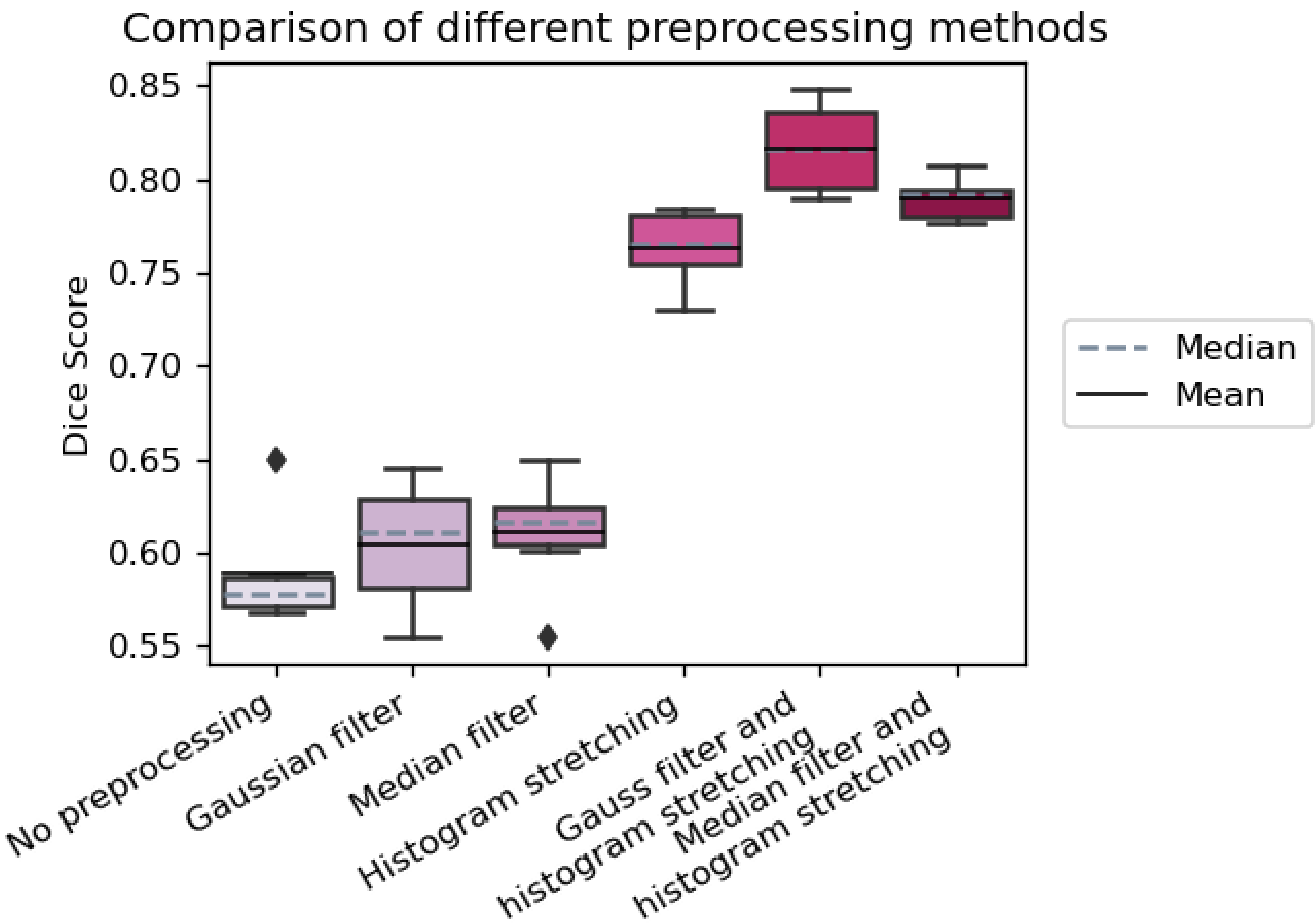
```
for start_pixel in border_pixels:
    old_group = [start_pixel]
    new_group = []
    first_run = True
    while old_group != new_group:
        if not first_run:
            old_group = new_group
        first_run = False
        for pixel in old_group:
            for other_pixel in border_pixels:
                if math.dist(pixel, other_pixel) < 2:
                    new_group.append(other_pixel)
                    border_pixels.remove(other_pixel)
            if pixel in border_pixels:
                border_pixels.remove(pixel)
    all_groups.append(new_group)
```

Cell Counting

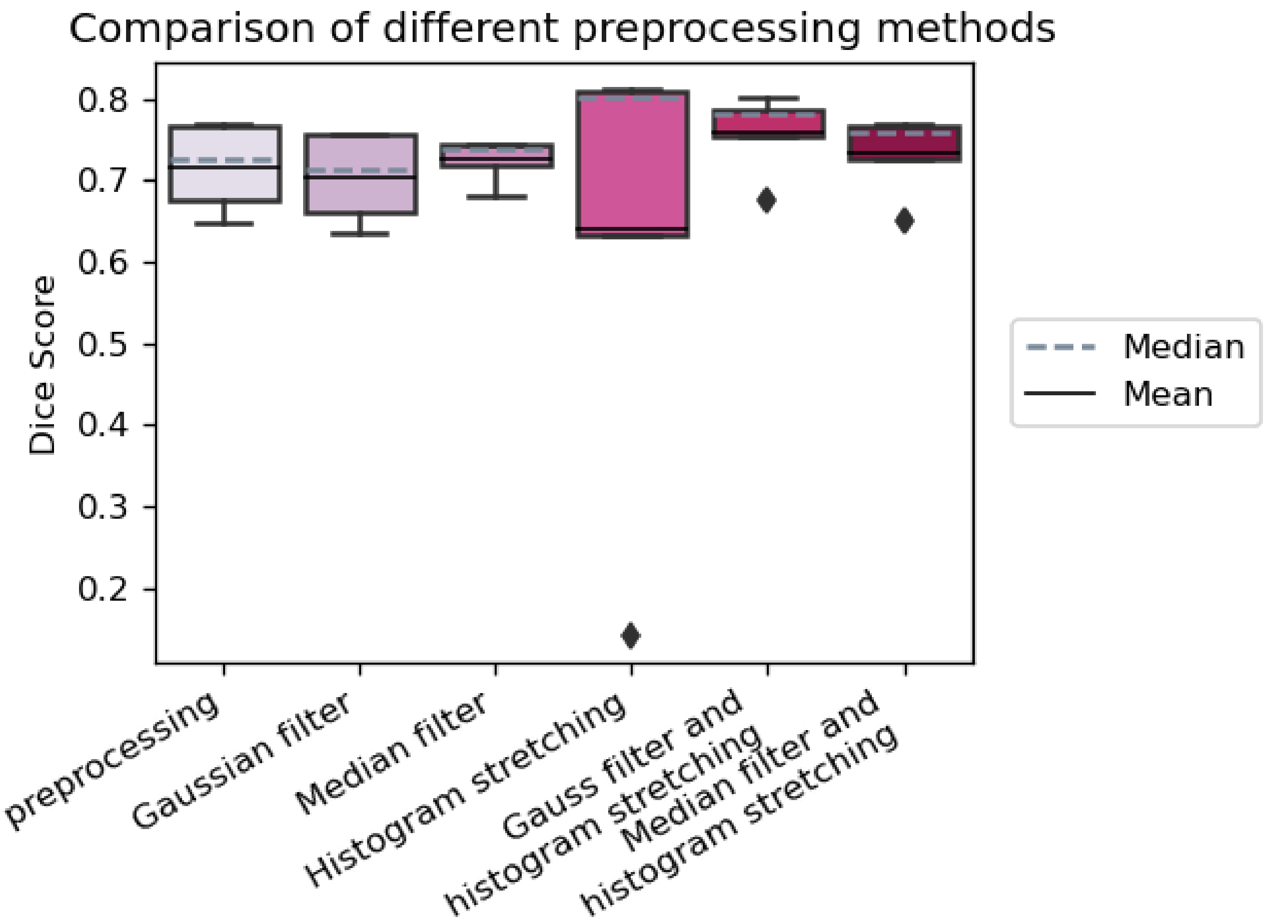
```
for start_pixel in border_pixels:
    old_group = [start_pixel]
    new_group = []
    first_run = True
    while old_group != new_group:
        if not first_run:
            old_group = new_group
        first_run = False
        for pixel in old_group:
            for other_pixel in border_pixels:
                if math.dist(pixel, other_pixel) < 2:
                    new_group.append(other_pixel)
                    border_pixels.remove(other_pixel)
            if pixel in border_pixels:
                border_pixels.remove(pixel)
    all_groups.append(new_group)
```



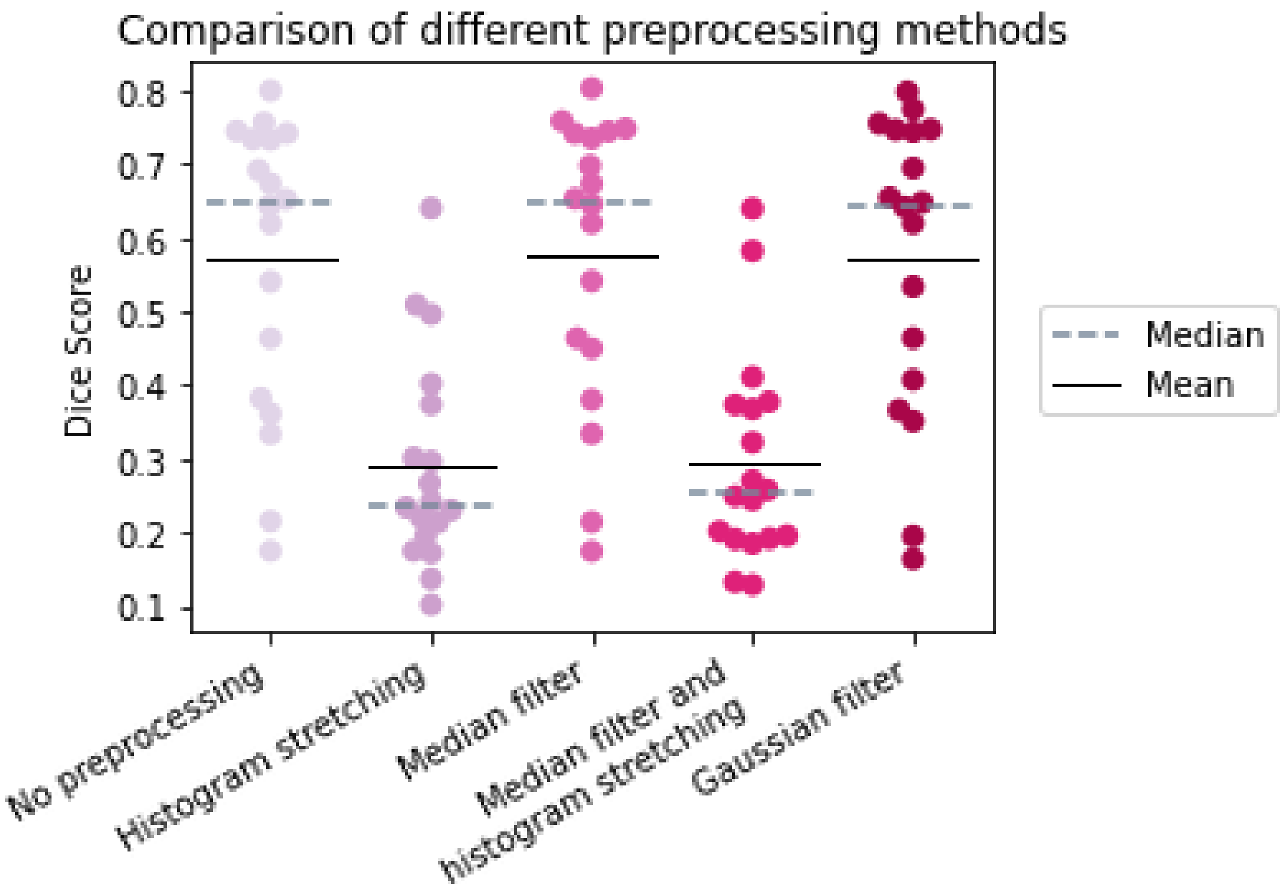
Evaluation N2DH-GOWT1



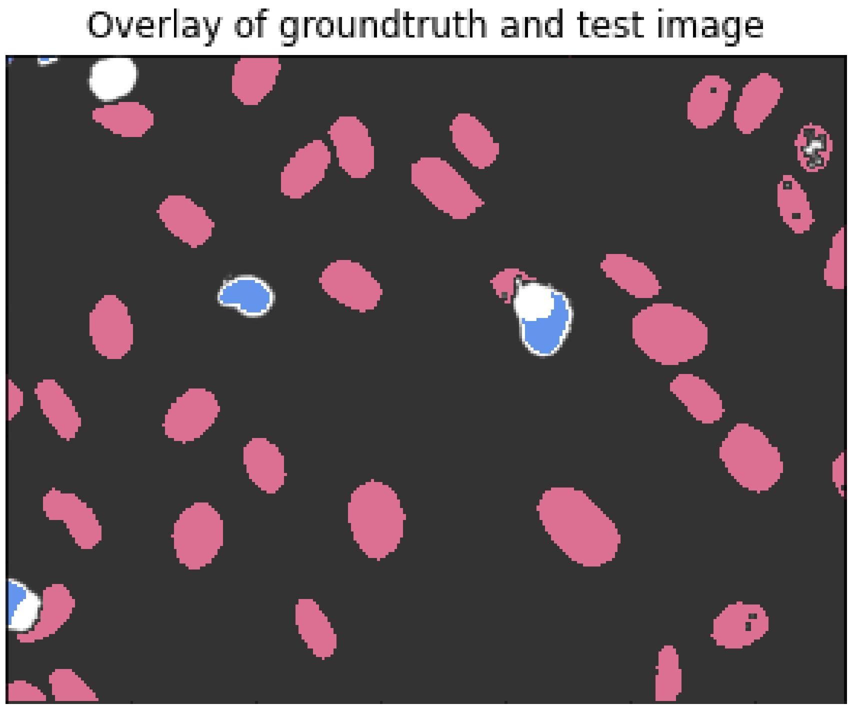
Evaluation N2DL-HeLa

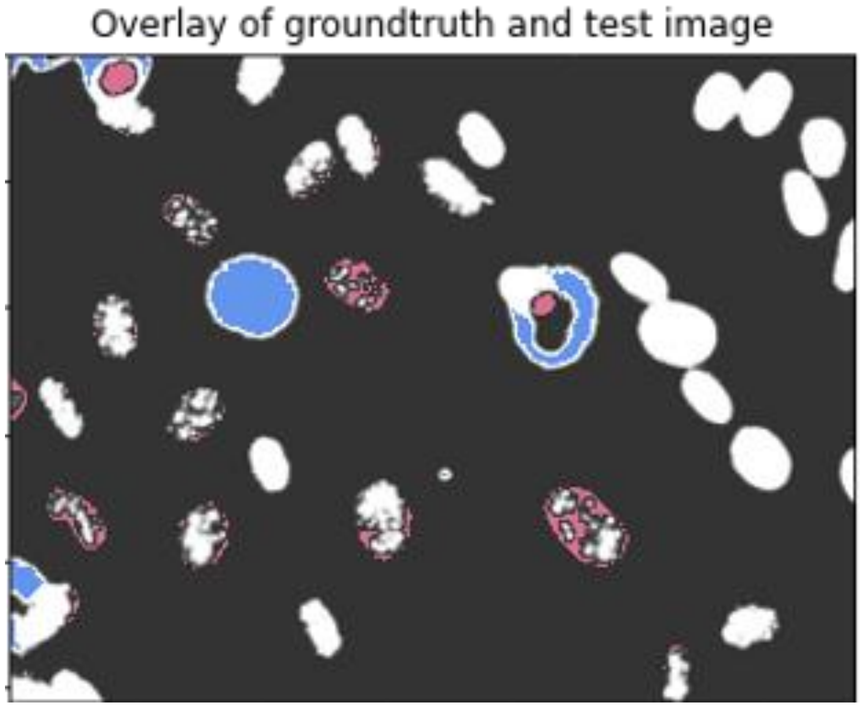


Evaluation NIH3T3



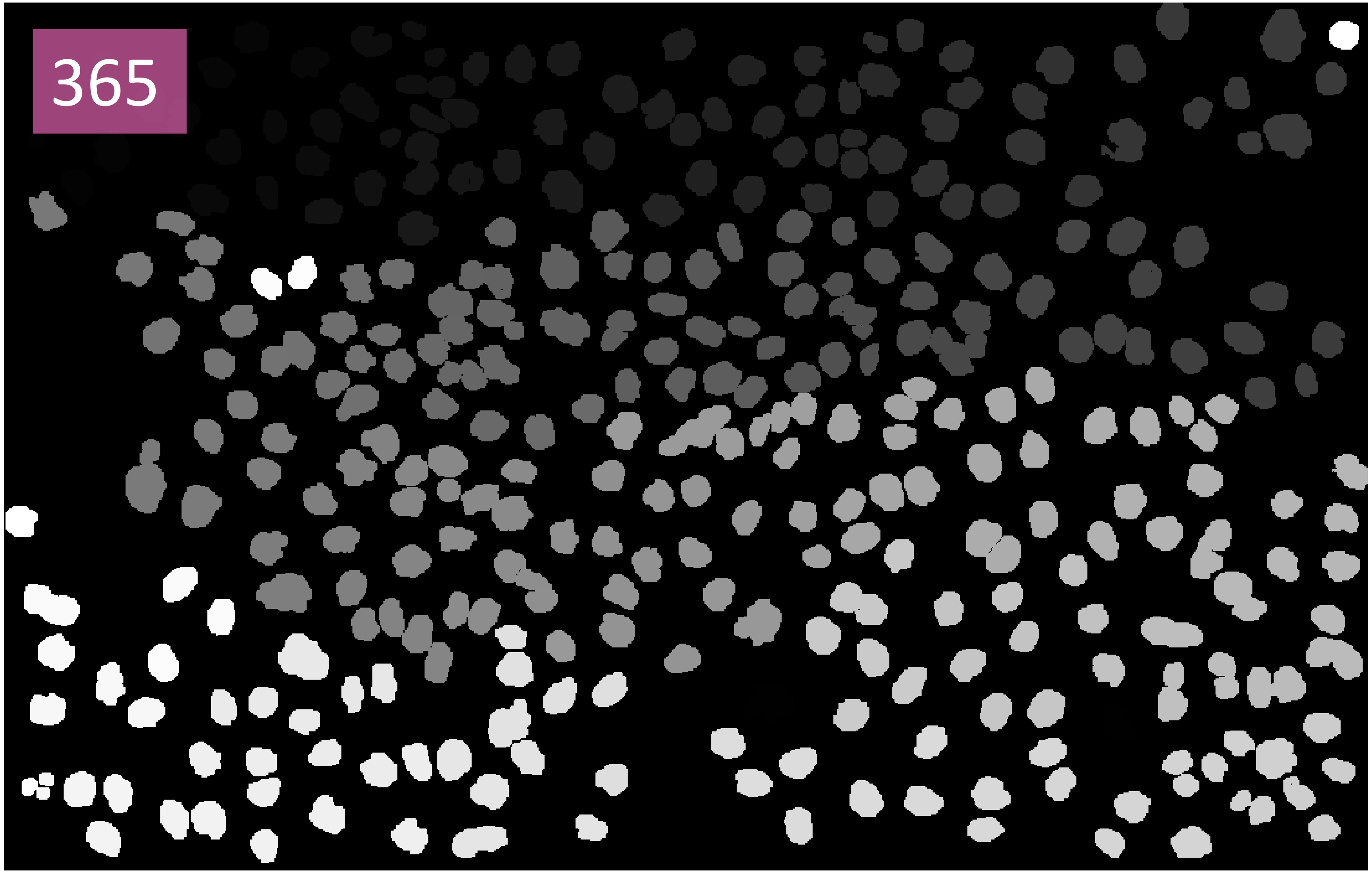
Evaluation NIH3T3





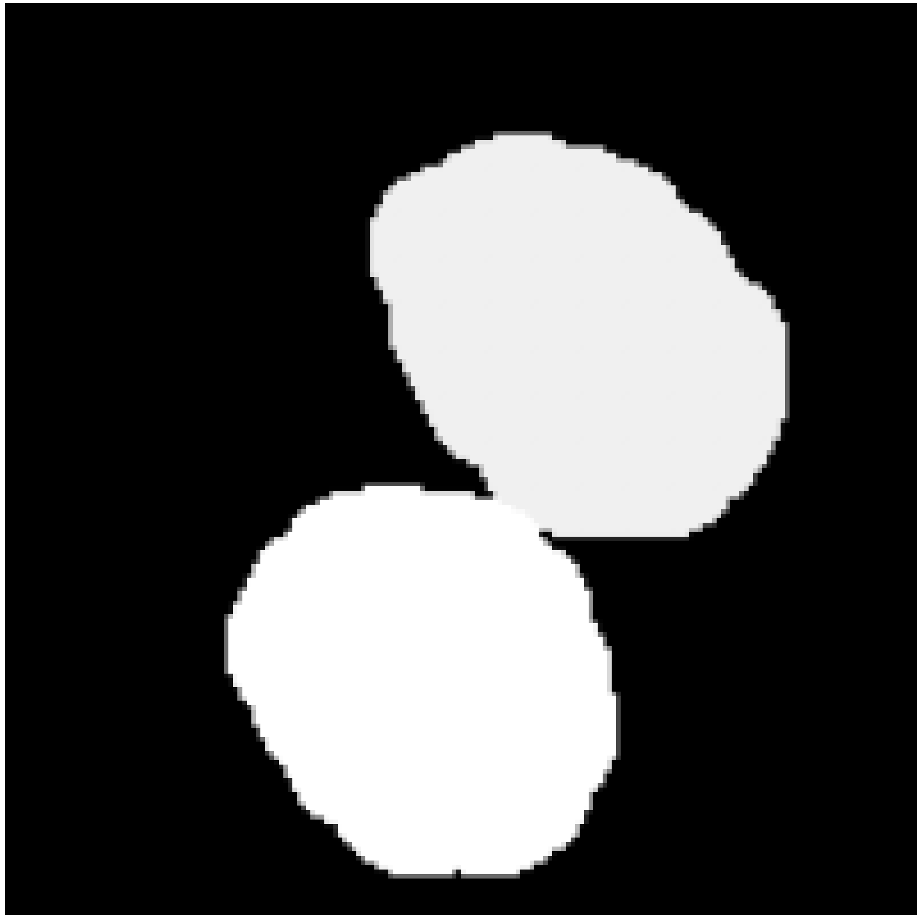
False negatives
False positives

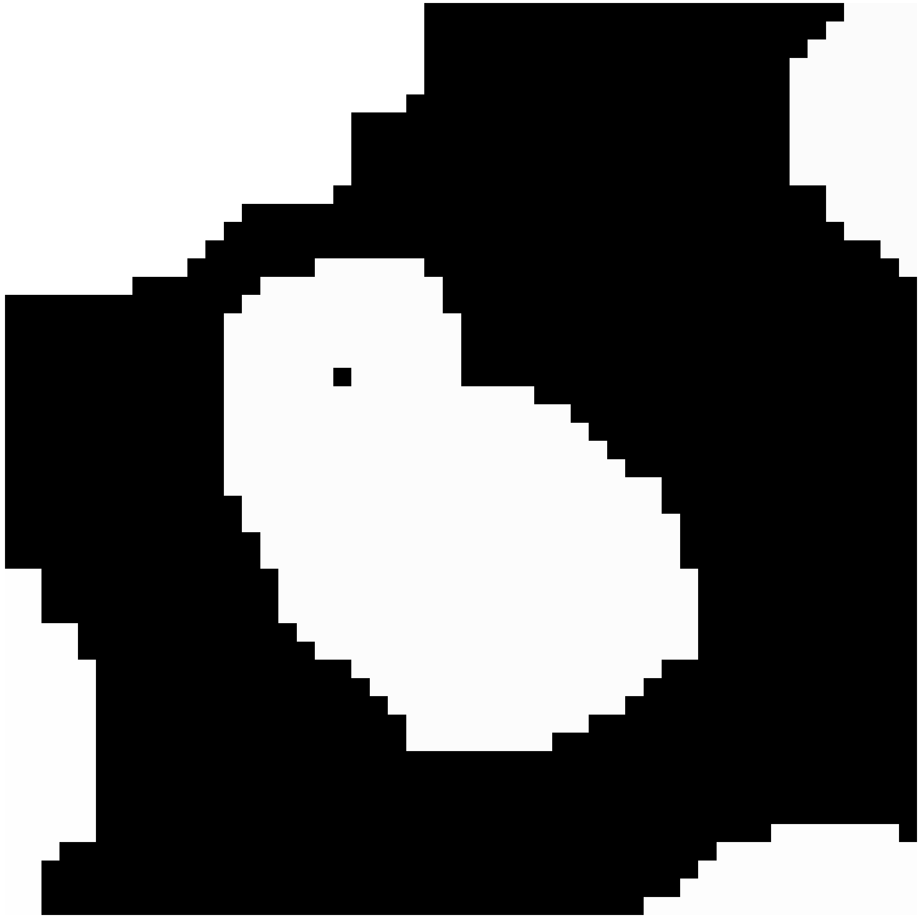
Evaluation cell nuclei count



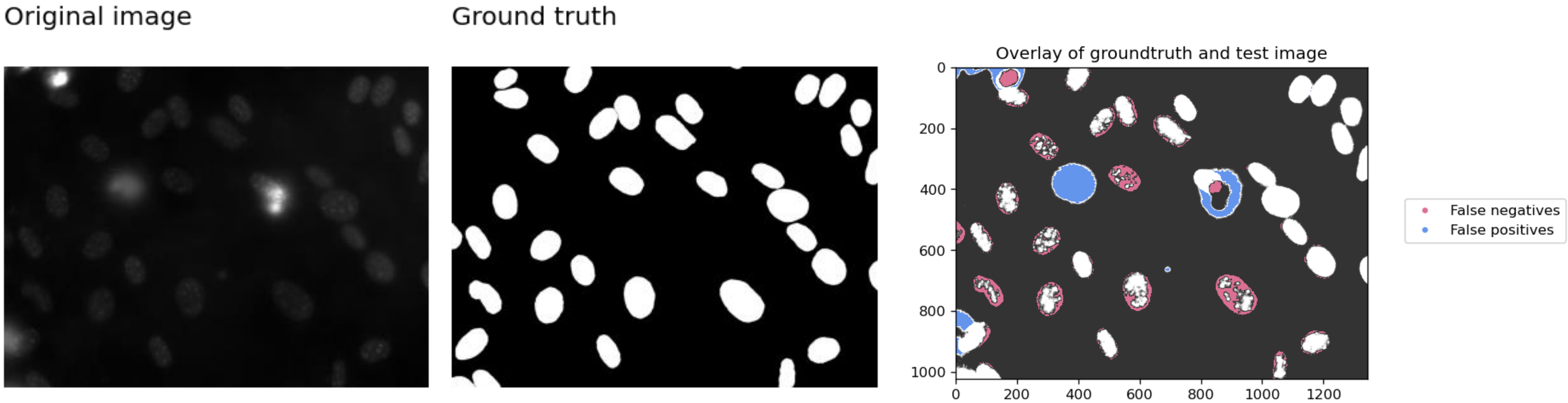
349

Evaluation cell nuclei count





Conclusion



Successful implementation → Evaluation of methods → Future improvements possible

Laura Wächter, Veronika Schuler, Elizaveta Chernova, Hannah L. Winter

Thank you for your attention!

Additional slide – Histogram stretching

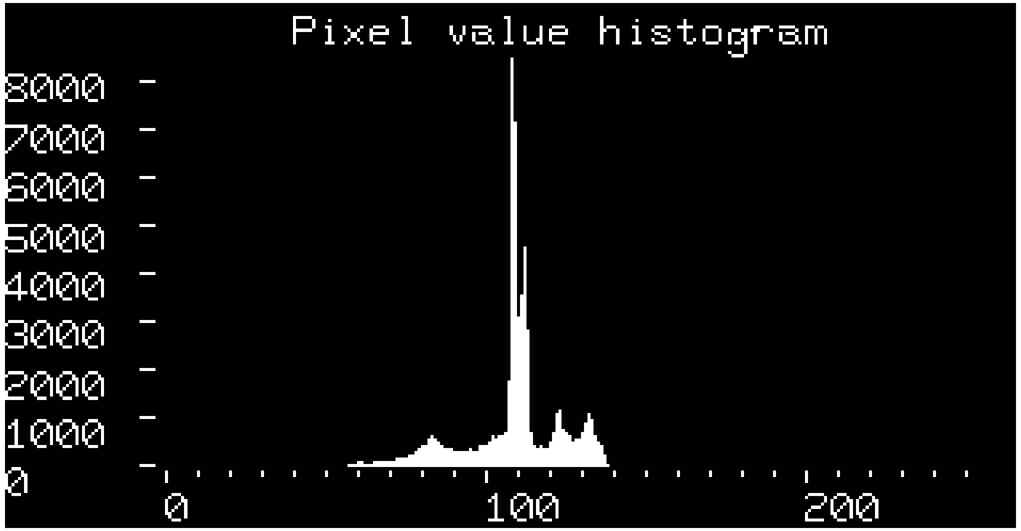
a = 0, b =255

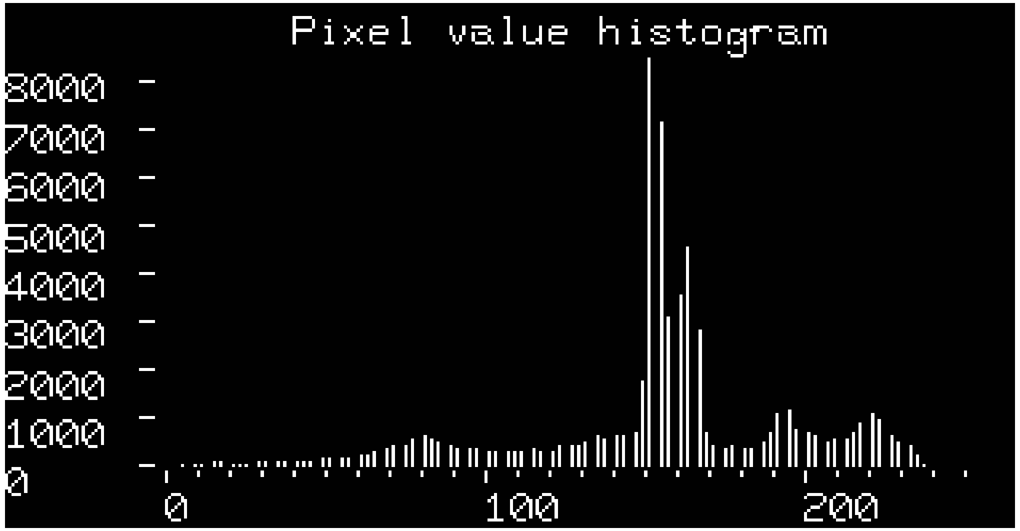
c – lowest pixel intensity in the image

d – highest pixel intensity in the image

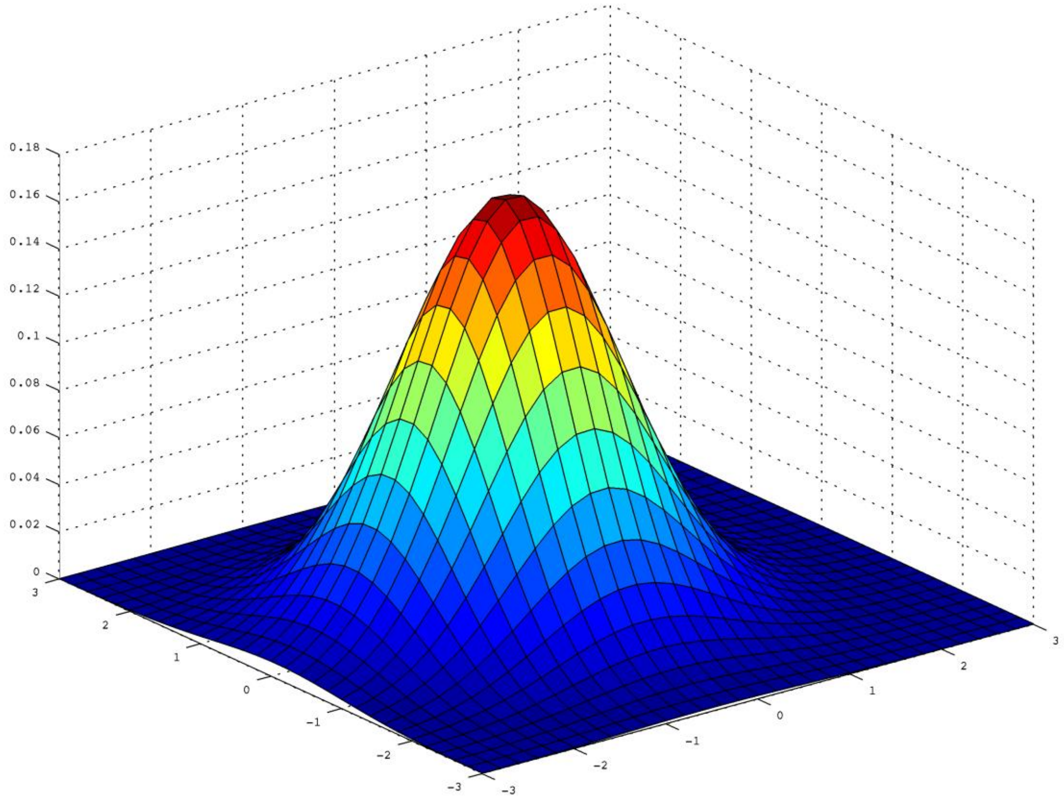
$$P_{out} = (P_{in} - c)\left(\frac{b - a}{d - c}\right) + a$$

Additional slide – Histogram stretching





Additional slide – Gaussian filter



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

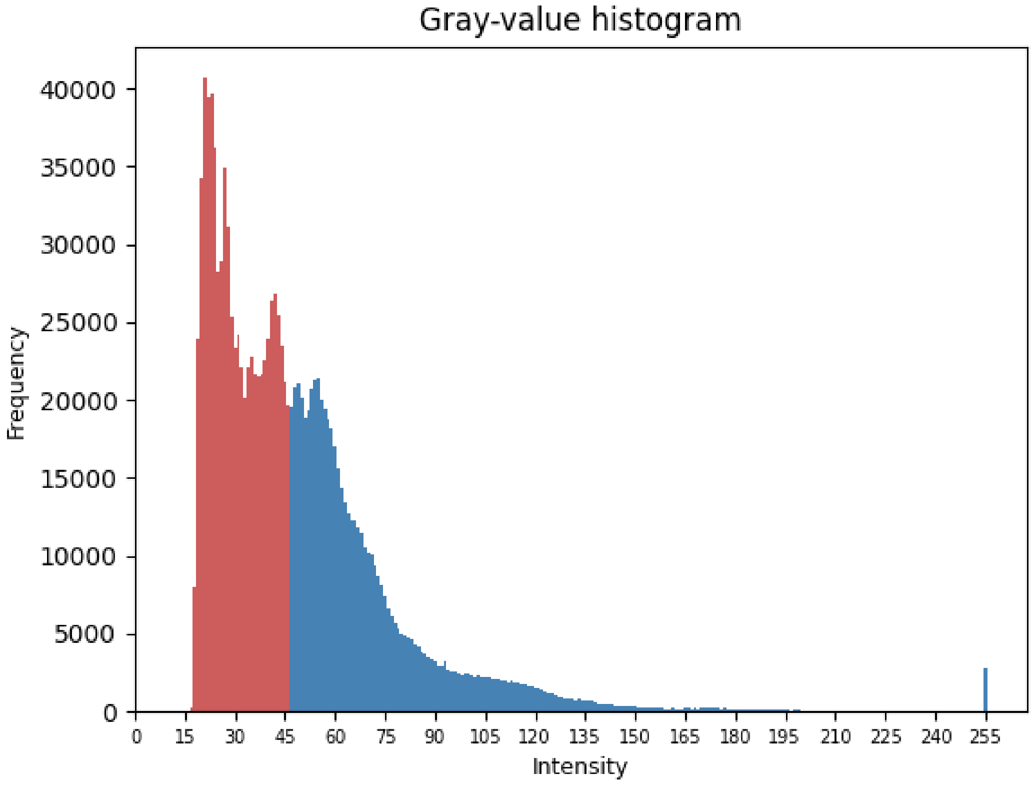
https://stats.stackexchange.com/questions/102632/plot-two-dimensional-gaussian-density-function-in-matlab

Additional slide – Criterion measure

Criterion measure

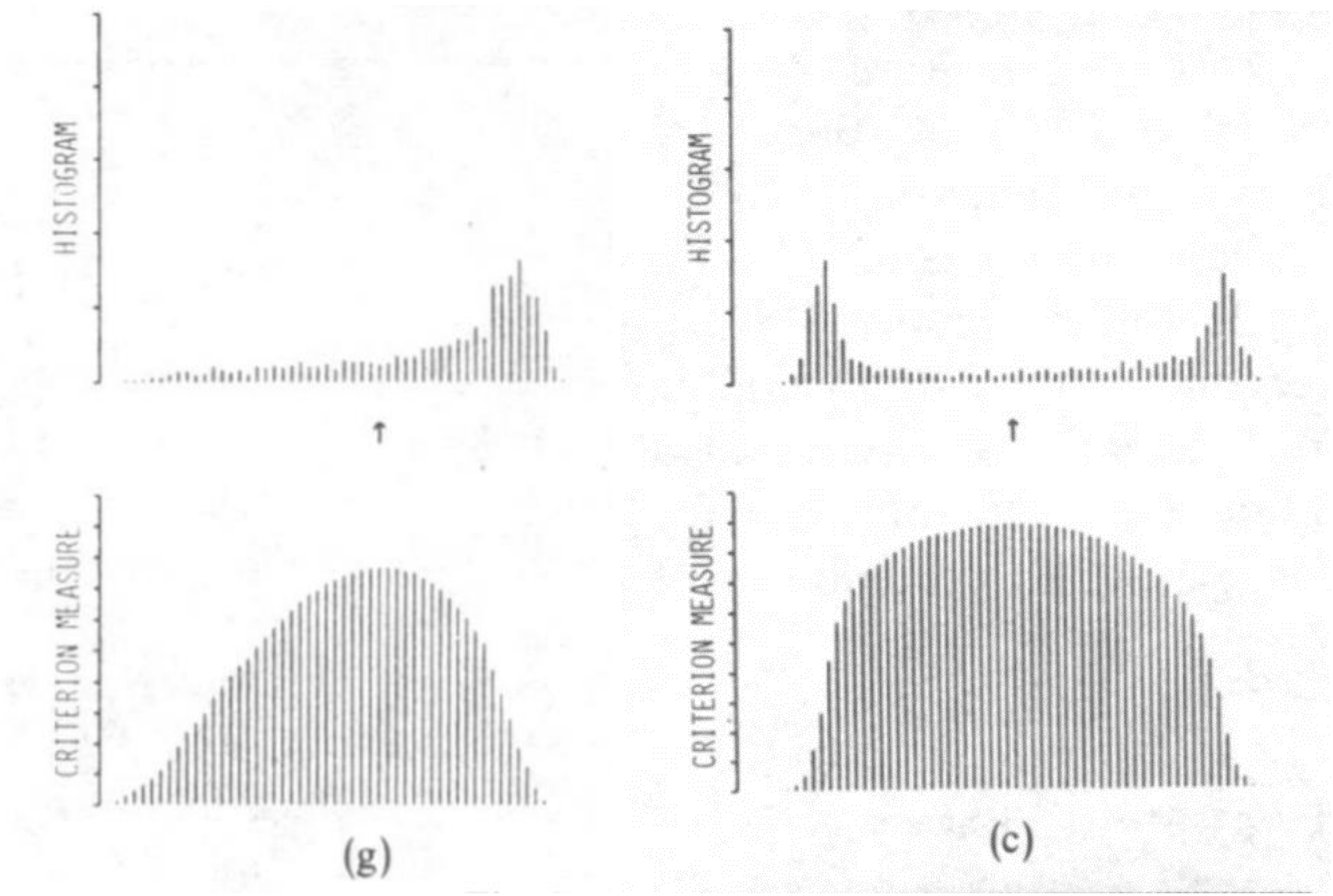
$$\eta(k) = \frac{{\sigma_B}^2(k)}{{\sigma_T}^2}$$

σ_B = between-class variance
σ_T = total variance
$\eta(k) \in [0,1]$



Threshold value $k \in [0,255]$

Additional slide – Criterion measure



Otsu, 1979

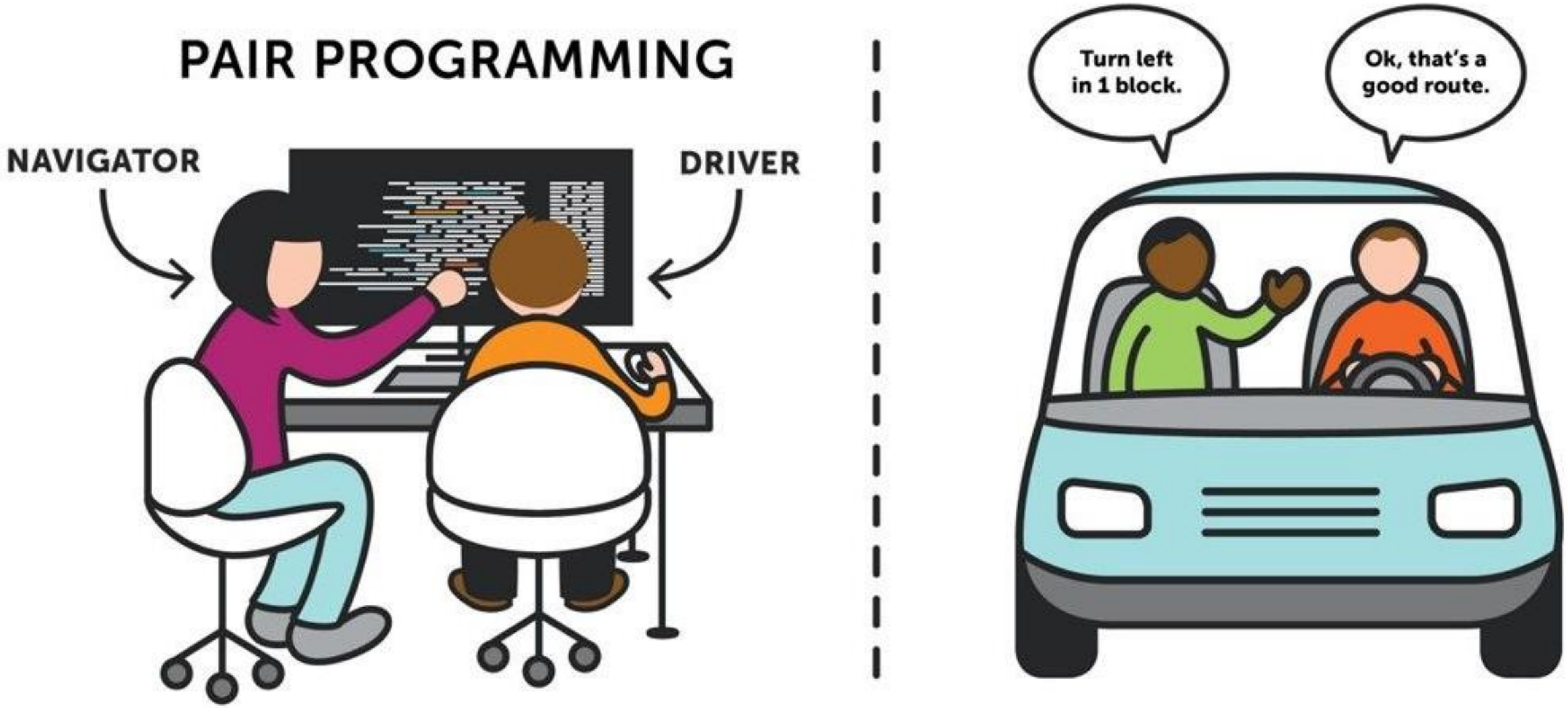
Additional slide – Otsu disadvantages

(a) (b) (c)

(a) (b) (c)

Not suitable for images with high complexity

Additional slide – Pair Programming



Additional slide – 2D Otsu

Intensity level of pixel is compared with immediate neighborhood pixels

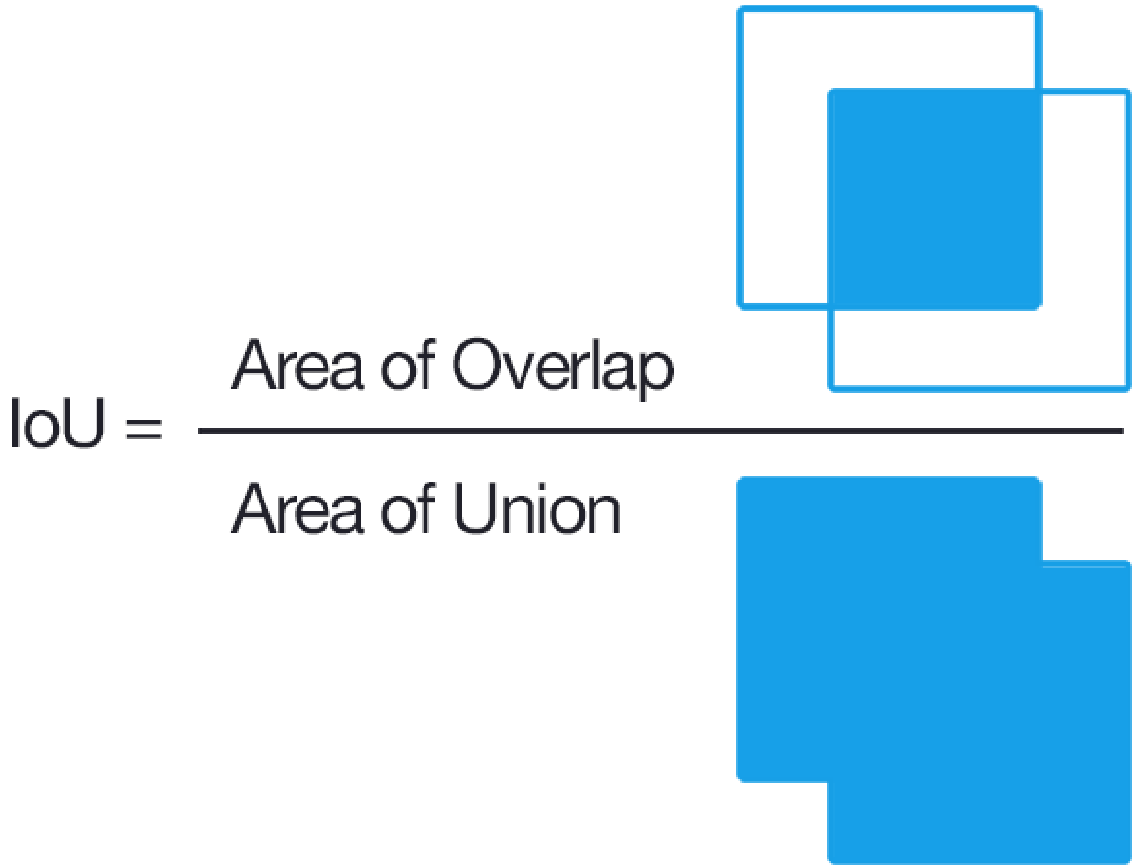
Algorithm:

- For each pixel calculate average gray-level of neighborhood
- Gray level of pixel and average gray levels are divided in L discrete values
- Form pairs: pixel gray level i and neighborhood average j
- There are $L \times L$ possible pairs
- Frequency $f_{i,j}$ of a pair (i, j) divided by the total pixel number N defines probability mass function in a 2D histogram:

$$P_{i,j} = \frac{f_{i,j}}{N} \qquad \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} P_{i,j} = 1$$

Additional slide – IoU

IoU = Intersection-Over-Union

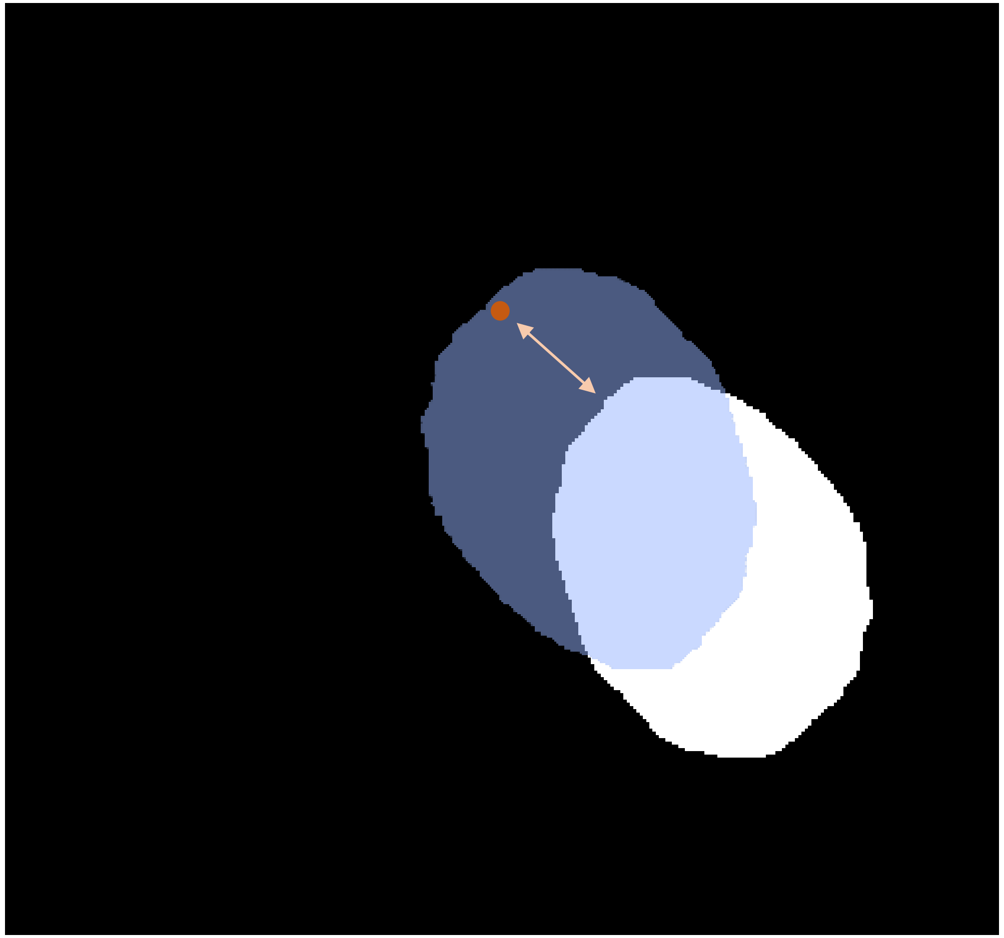


Additional slide - MSD

MSD = mean surface distance

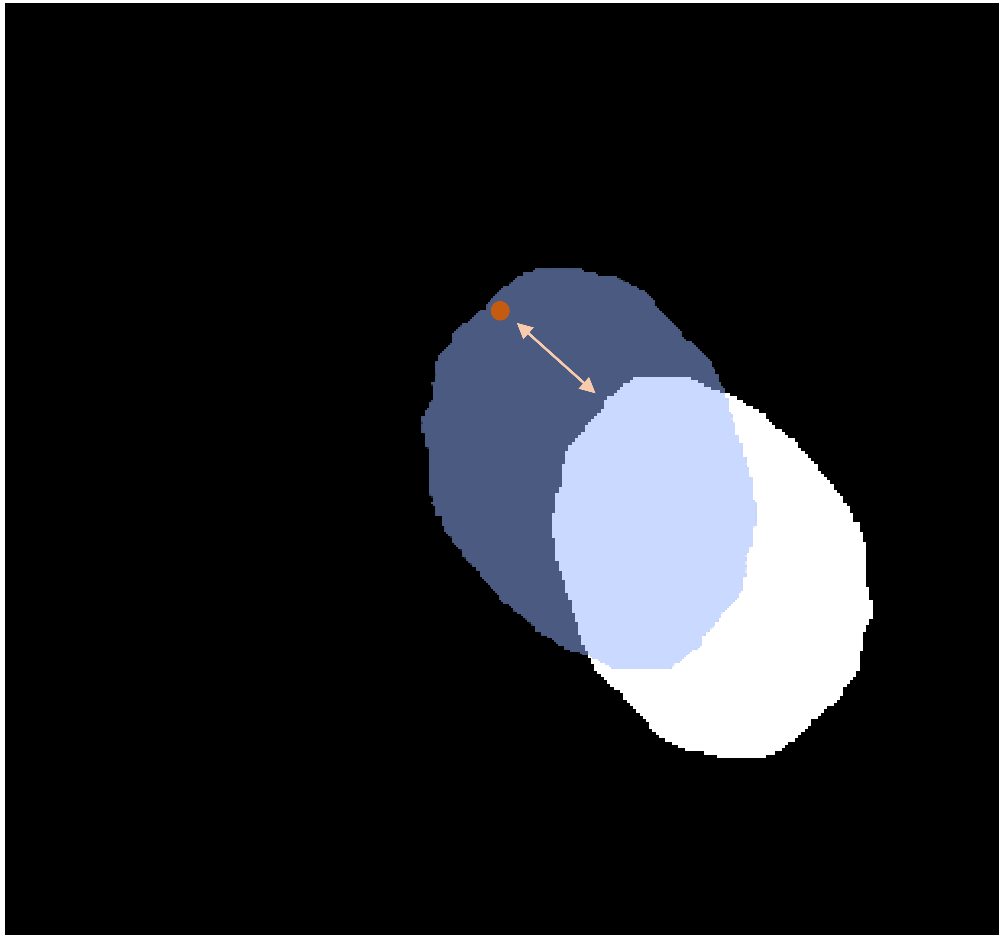
$$d(p, S') = \min_{p' \in S'} ||p - p'||_2$$

$$\text{MSD} = \frac{1}{n_S + n_{S'}} \left(\sum_{p=1}^{n_S} d(p, S') + \sum_{p'=1}^{n_{S'}} d(p', S) \right)$$

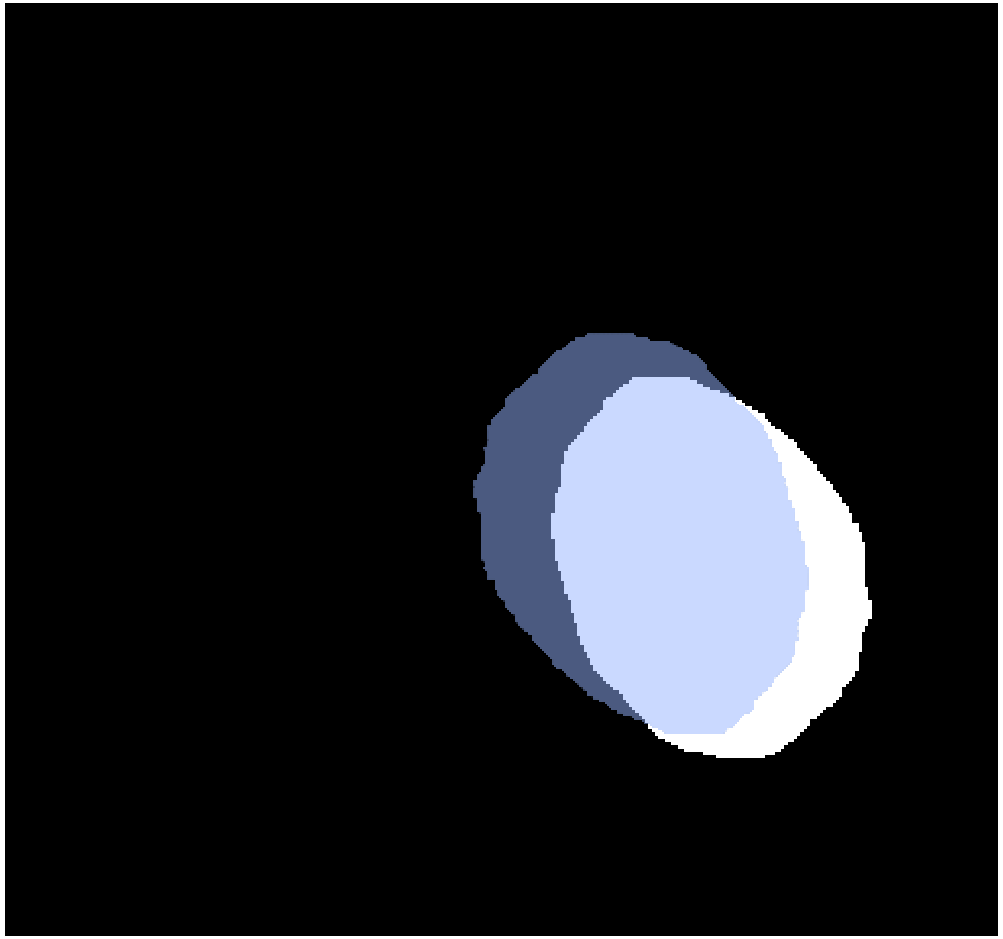


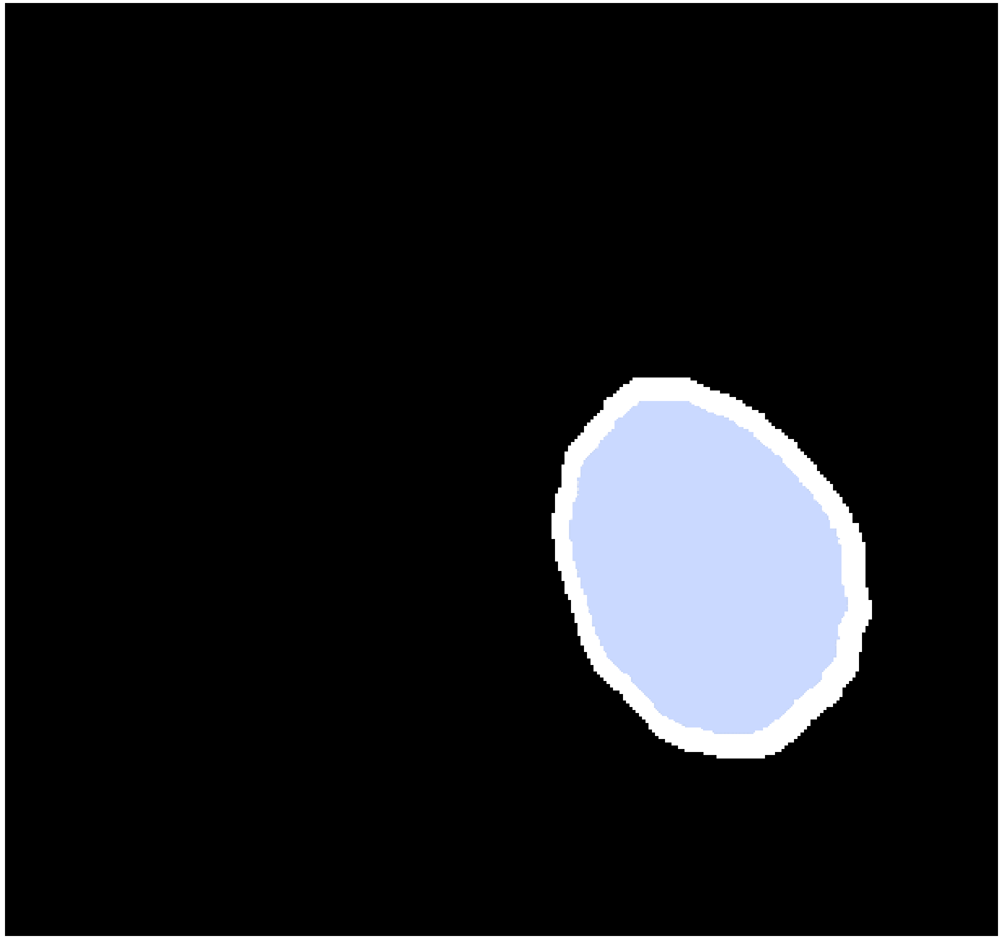
Additional slide – Hausdorff

HD=max[d(S,S'),d(S',S)]



Additional slide – Hausdorff





Additional slide – MSD code

```
# calculate minimum distances for each point in seg to the sets of points in gt
tree_seg_gt = spatial.cKDTree(gt_array)
mindist_seg_gt, minid_seg_gt = tree_seg_gt.query(seg_array)

# calculate sum and length of arrays with minimal distances
sum_seg_gt = np.sum(mindist_seg_gt)
size_seg_gt = len(mindist_seg_gt)
```

```
mean_surface_distance = (1/(size_gt_seg+size_seg_gt))*(sum_gt_seg + sum_seg_gt)

return mean_surface_distance
```

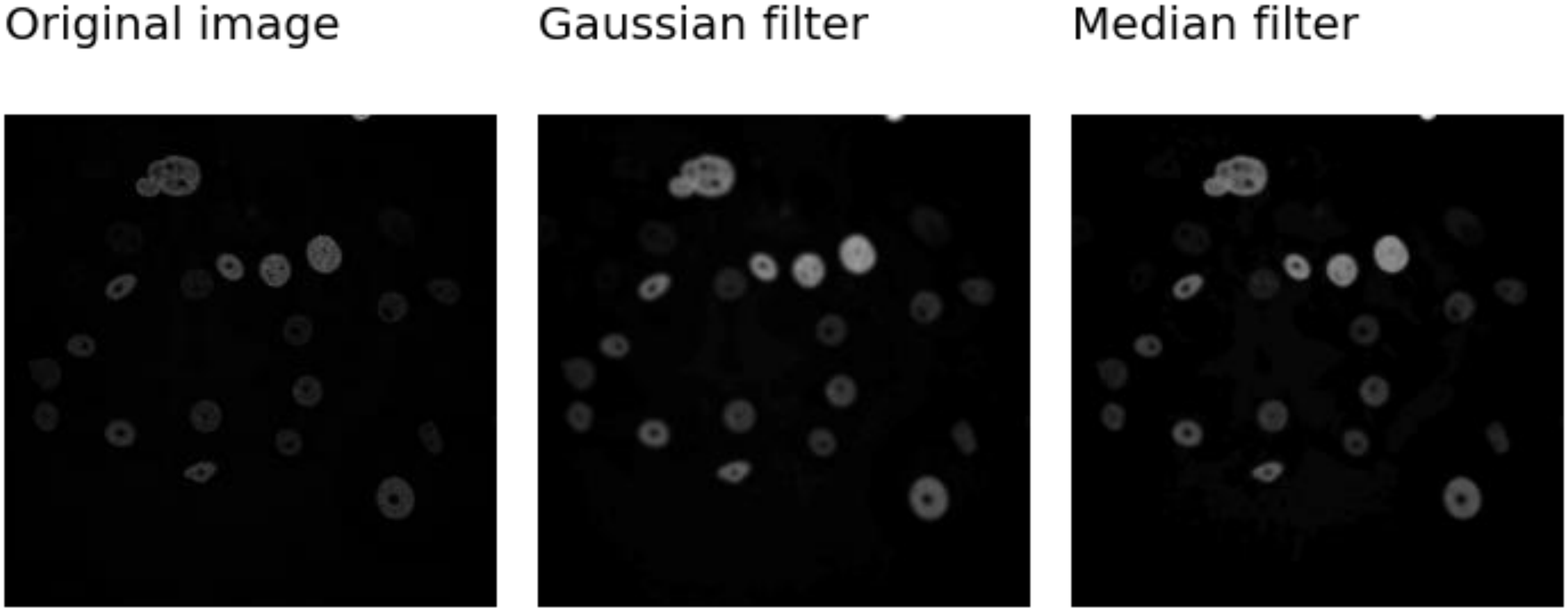
Additional slide – HD code

```
# calculate minimum distances for each point in seg to the sets of points in gt
tree_seg_gt = spatial.cKDTree(gt_array)
mindist_seg_gt, minid_seg_gt = tree_seg_gt.query(seg_array)

# calculate sum and length of arrays with minimal distances
sum_seg_gt = np.sum(mindist_seg_gt)
size_seg_gt = len(mindist_seg_gt)
```

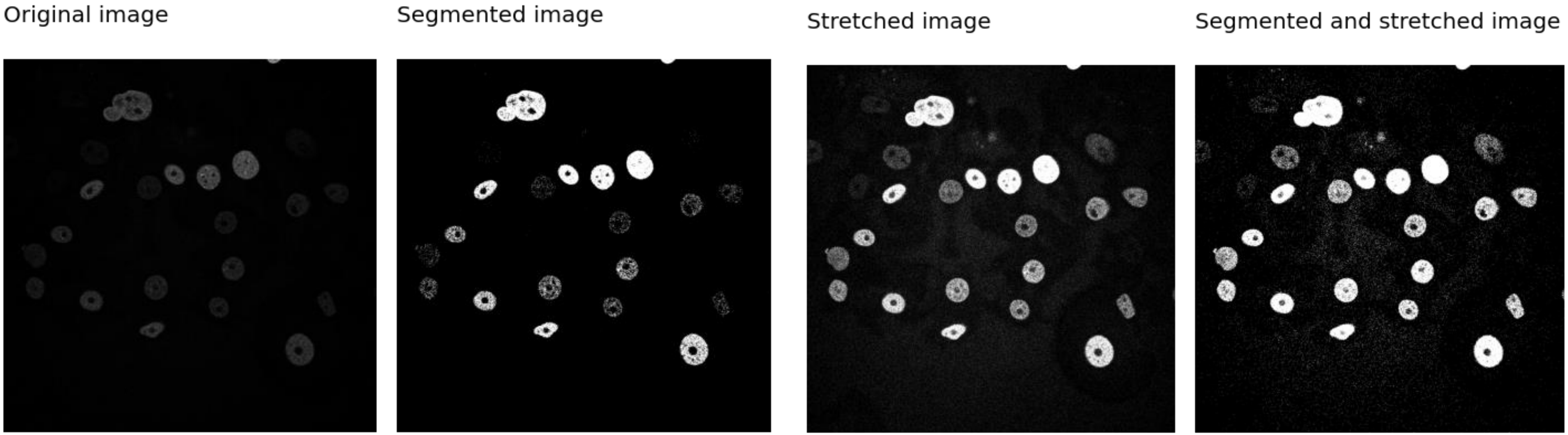
```
hausdorff_distance = max(max_gt_seg,max_seg_gt)

return hausdorff_distance
```

https://mlnotebook.github.io/post/surface-distance-function/#:~:text=Surface%20Distance%20measures%20are%20a,the%20ground%20truth%20(GT).&text=It'll%20help%20us%20to,the%20Hausdorff%20Distance%20(HD)

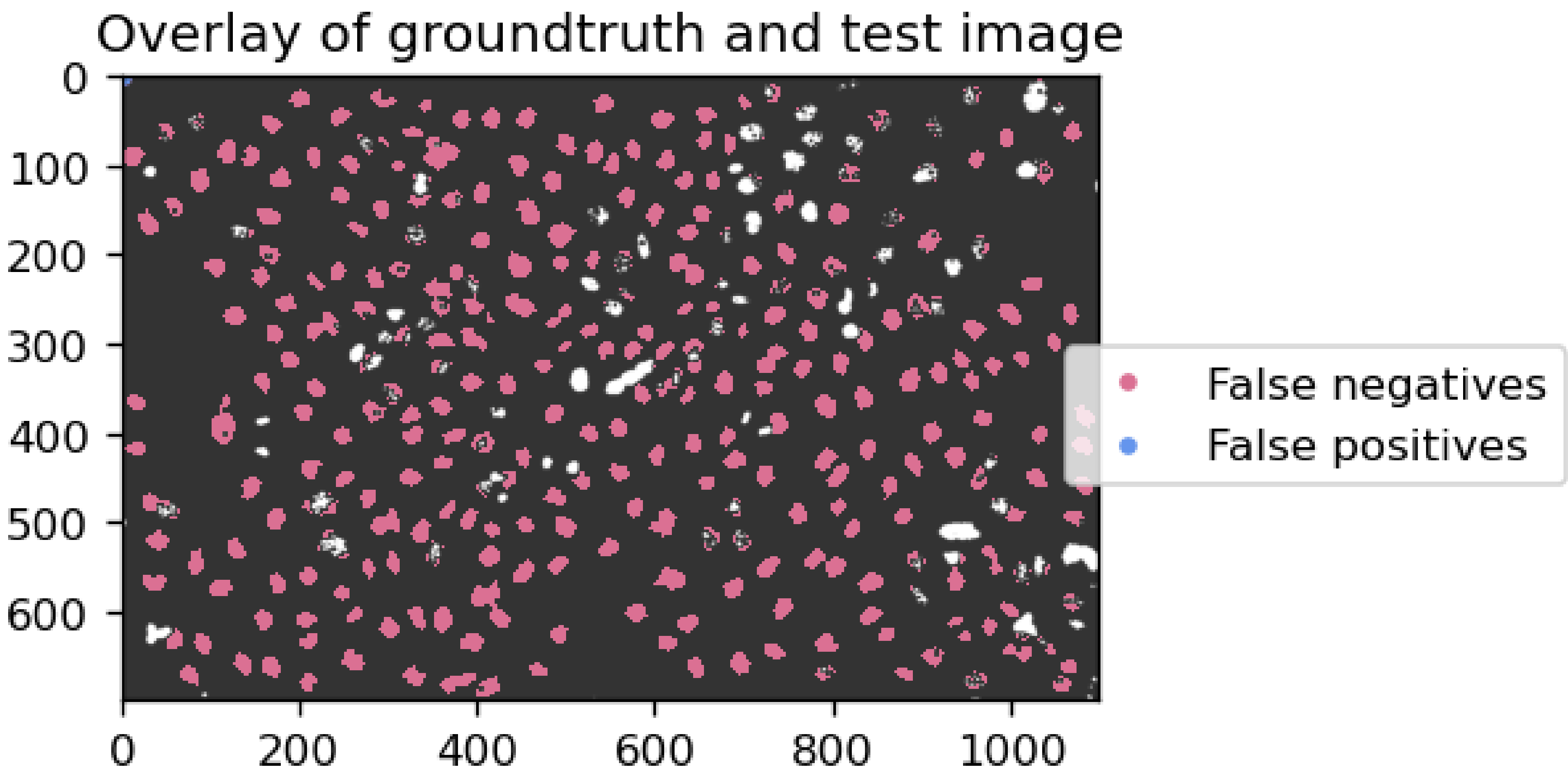
Additional slide – filters on N2DH-GOWT1



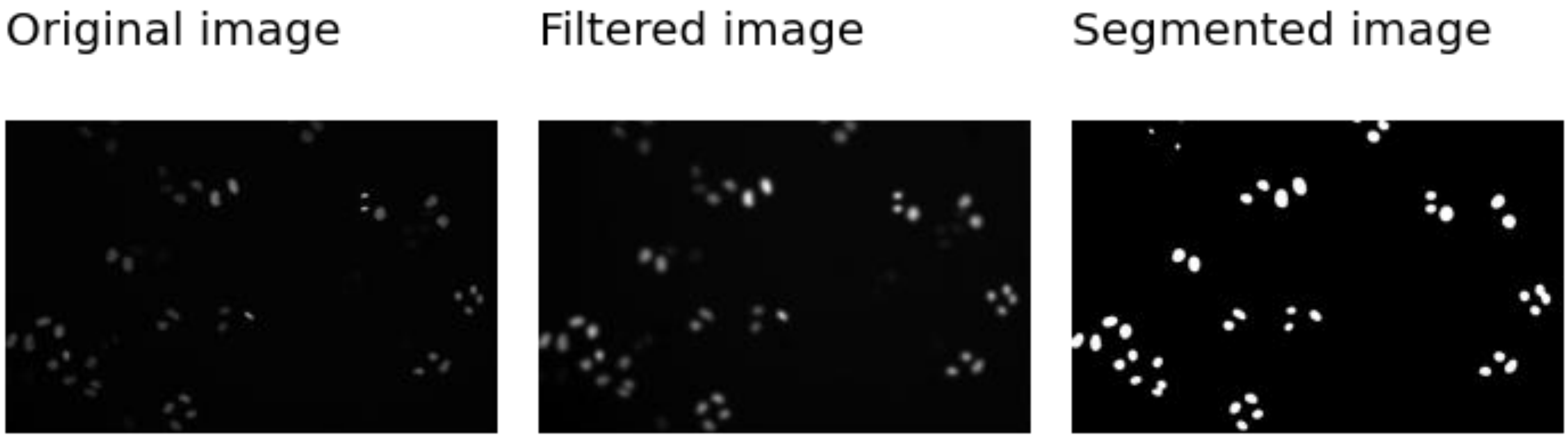
Additional slide – N2DH-GOWT1



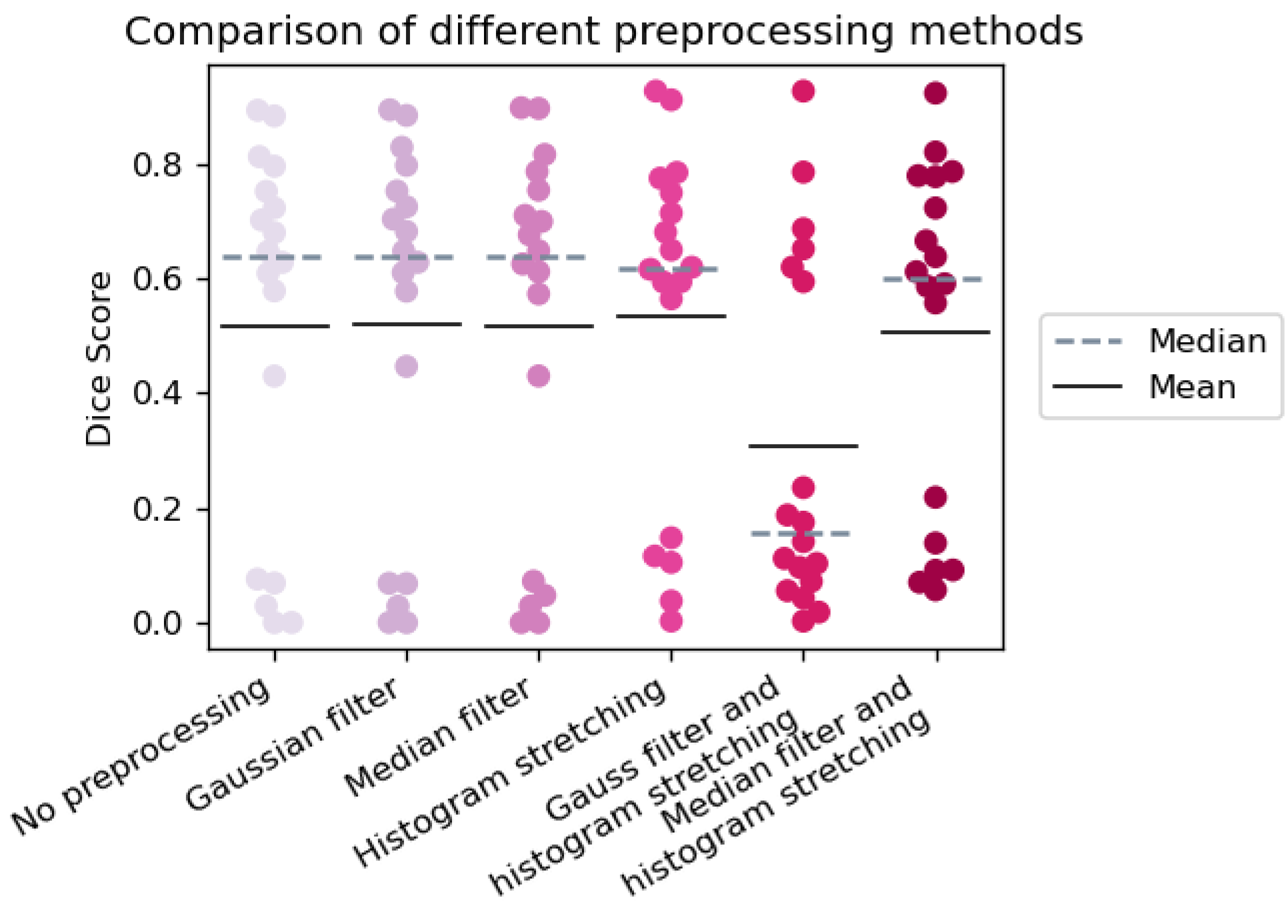
Additional slide – N2DL-HeLa



Additional slide – N2DL-HeLa

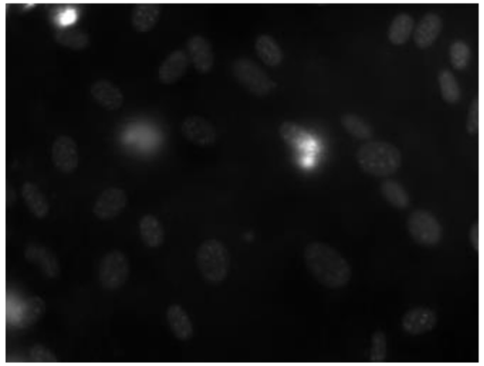


Additional slide – NIH3T3 – one-level

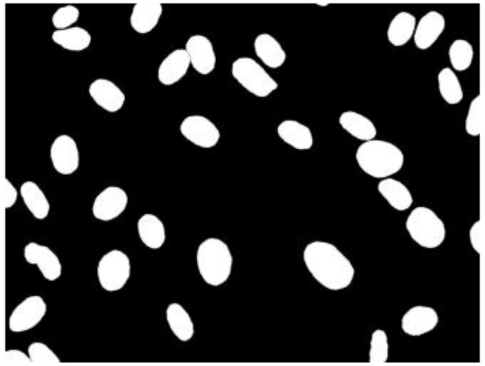


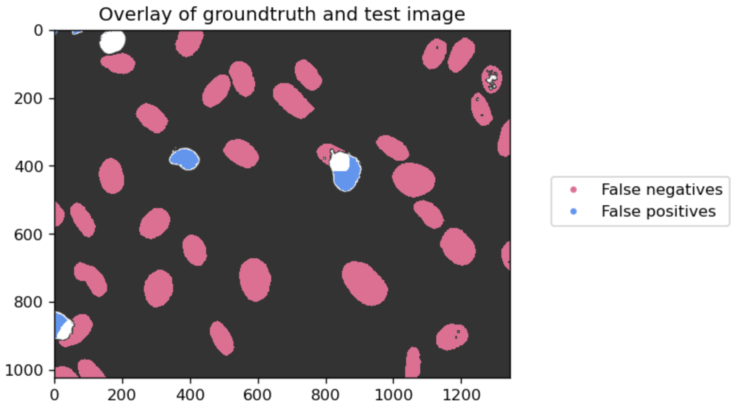
Additional slide - NIH3T3

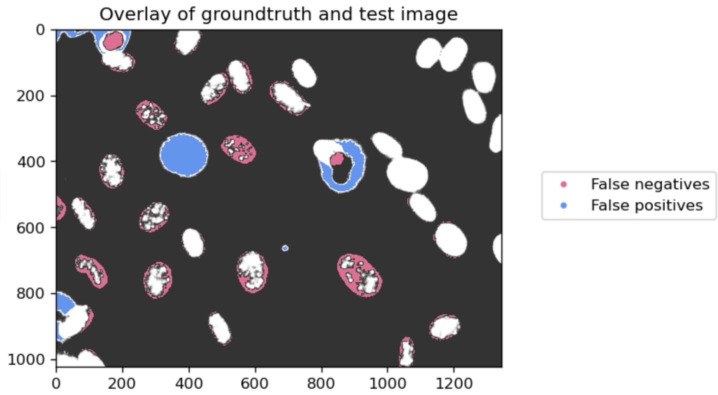
Original image



Ground truth







Additional slide - cell counting dataset 1

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

Additional slide – cell counting dataset 2

	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

Two-level Otsu's thresholding code