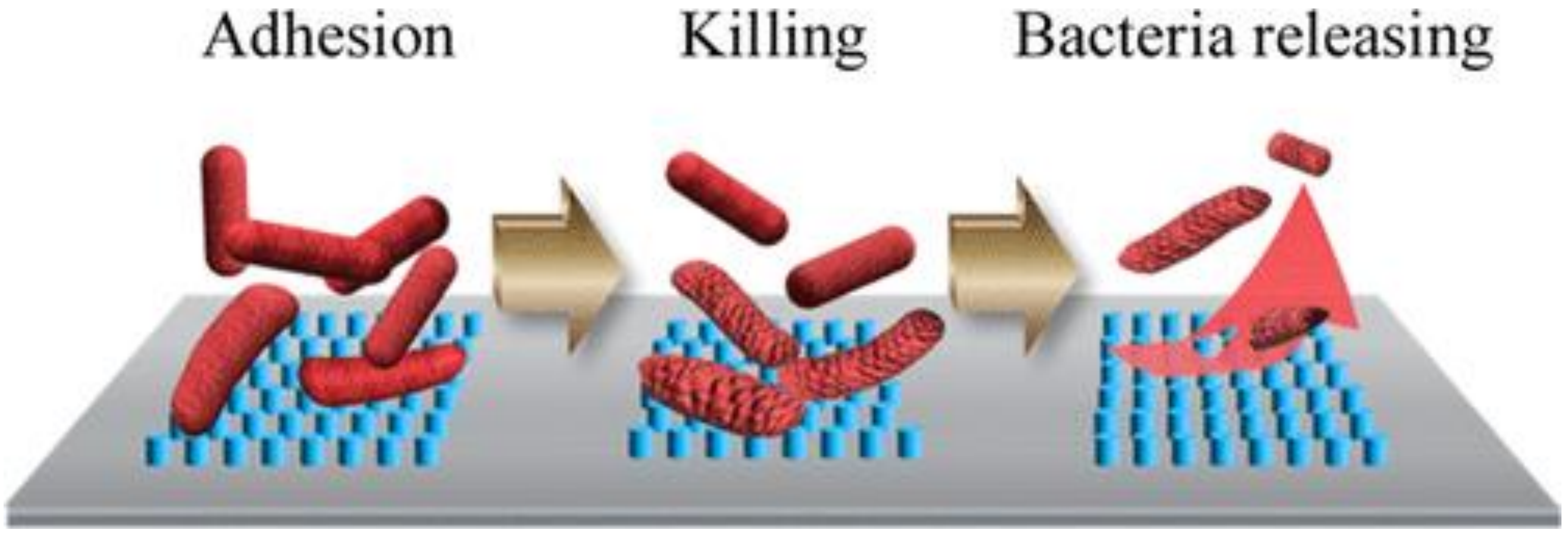
ACS Appl. Mater. Interfaces 2023, 15, 2, 3420-3432



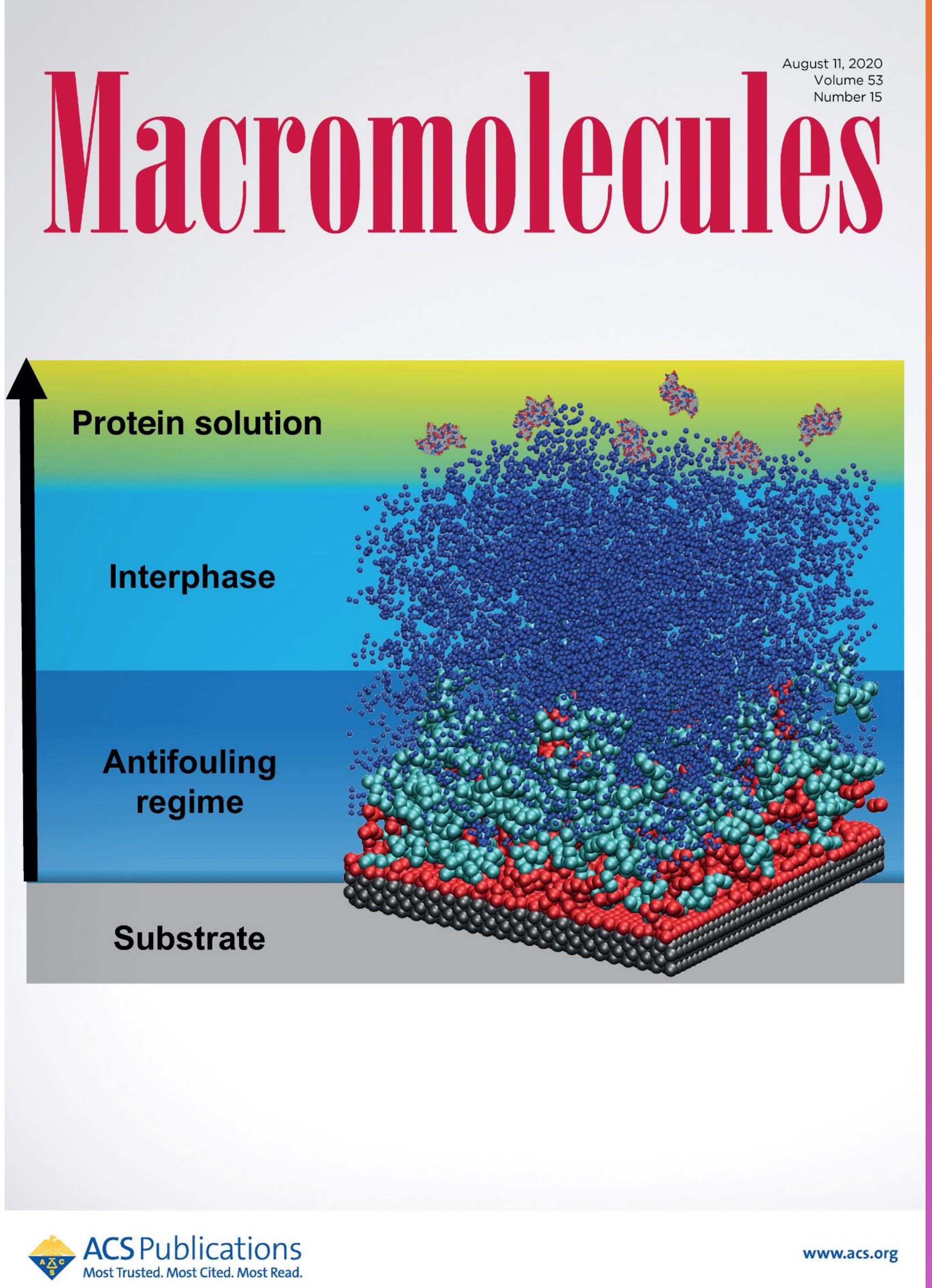
Deep learning-based analysis pipeline for studies of biological contaminants

By: Avi Bajaj

Macromolecules 2020, 53, 15, 6547-6554

Scope

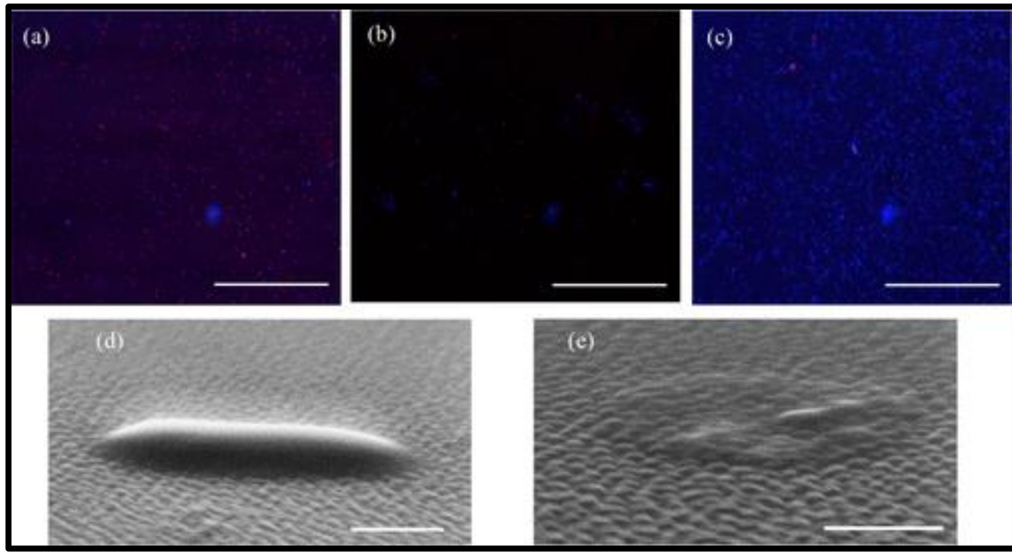
- Bacterial/viral adhesion and resilience to surfaces is a global healthcare problem that was exacerbated by the (ongoing) COVID-19 pandemic.
- My group is exploring the possibility of surface topology modification as a physical mitigation strategy.

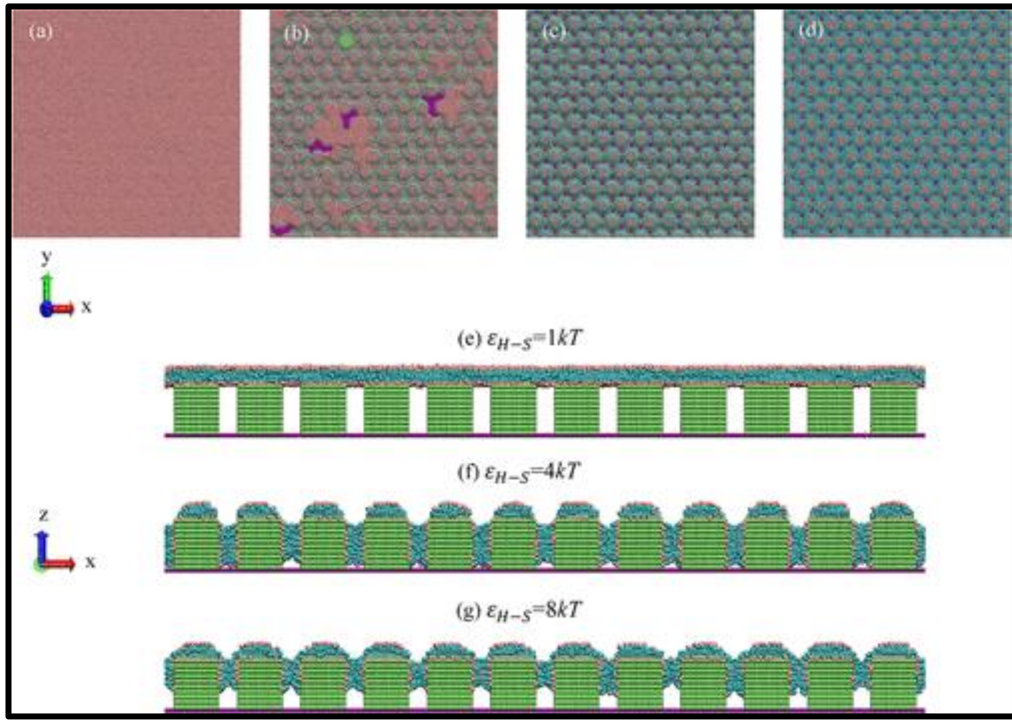


ACS Appl. Mater. Interfaces 2023, 15, 2, 3420-3432

Scope – cont.

- We are currently evaluating polymer systems and topologies based on their adhesion resistance to biological contaminants.
- We are integrating a suite of experimental (microscopy-based) and computational (simulation-based) techniques.
- We need an efficient way to capture the trends from microscopy data without compromising on accuracy!




(a)
(b)
(c)
(d)
y
x
(e) ε_{H-S}=1kT
(f) ε_{H-S}=4kT
z
x
(g) ε_{H-S}=8kT

About me

- 2nd year PhD student
- Work regularly with Python and C/C++ to develop pre/postprocessing routines for various kinds of data
- Avid tennis player and lifter of weights

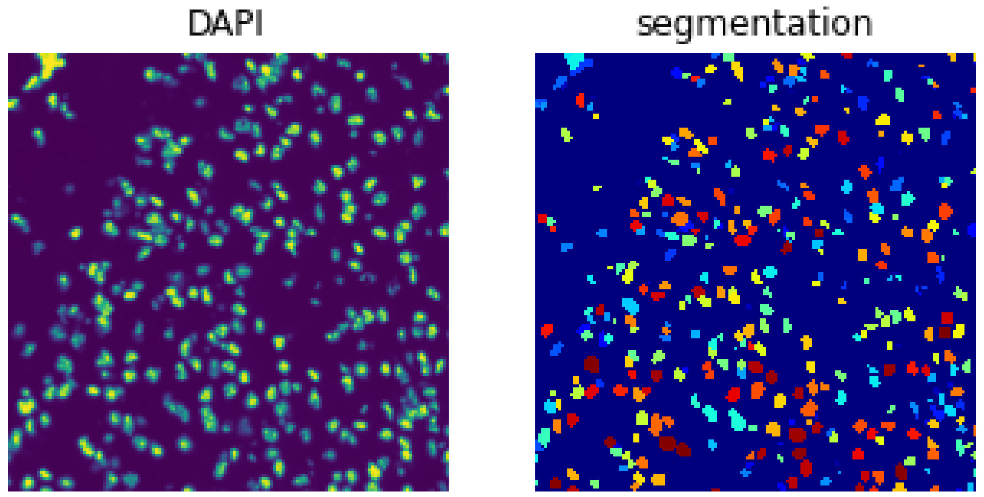
Key Learnings:

- Deep learning integration is a big step forward for scientific imaging, where image quality can be questionable
- Software can be GUI-reliant without being completely GUI-dependent
- It's up to the user to read the documentation and find a creative solution that suits their need(s)

Tools and Techniques

Software/packages:

- Fiji (ImageJ) v1.54f: https://downloads.imagej.net/fiji/archive/20230710-2317/
  - StarDist: https://imagej.net/plugins/stardist
- CSBDeep: https://pypi.org/project/csbdeep/
- Imageio: https://pypi.org/project/imageio/
- Tifffile: https://pypi.org/project/tifffile/
- NumPy: https://pypi.org/project/numpy/

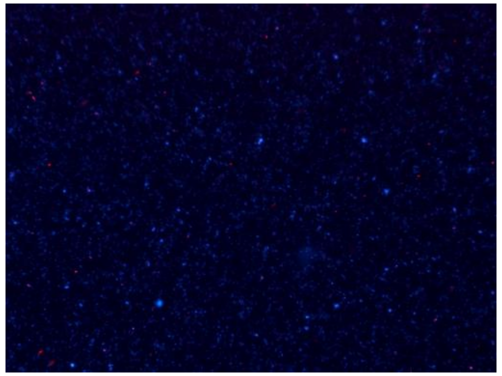


Methods:

- Deep learning (cell/nuclei detection)
- Fluorescence microscopy

https://squidpy.readthedocs.io/en/stable/notebooks/tutorials/tutorial_stardist.html

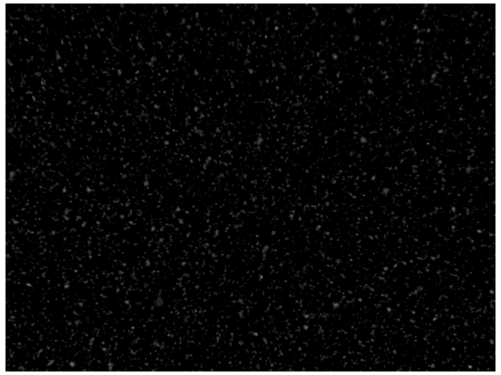
Source sample images (*E. coli*)



Apply preprocessing commands by running ImageJ headlessly, i.e.,

- subtract background
- normalize brightness

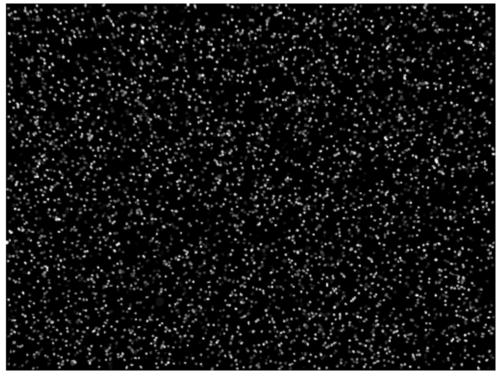
Convert preprocessed images to 8-bit (grayscale)



Run StarDist2D with pretrained model/parameters, i.e.,

- 2D versatile (fluorescent nuclei)

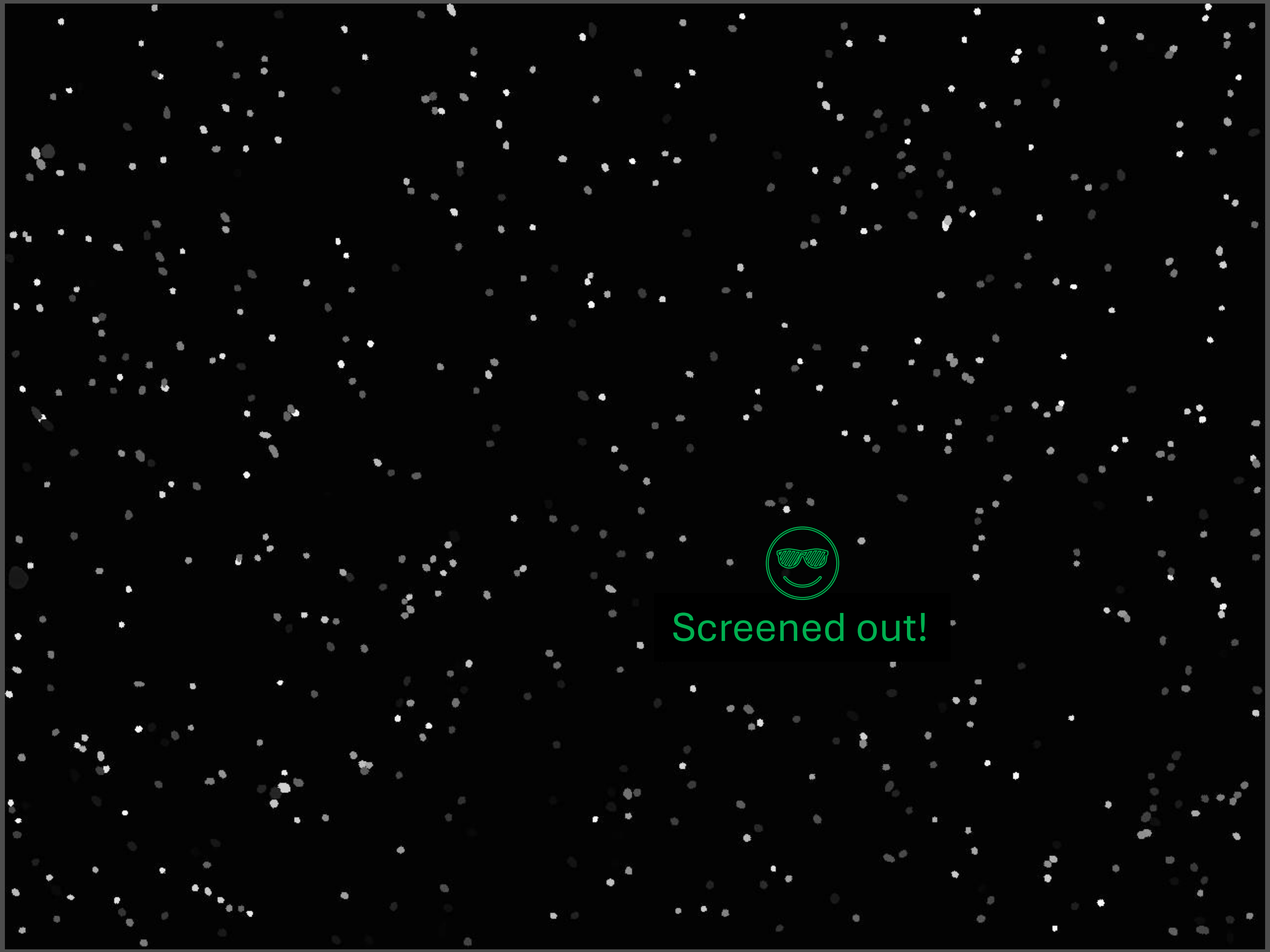
Export number of regions of interest (ROIs) per image to an external file



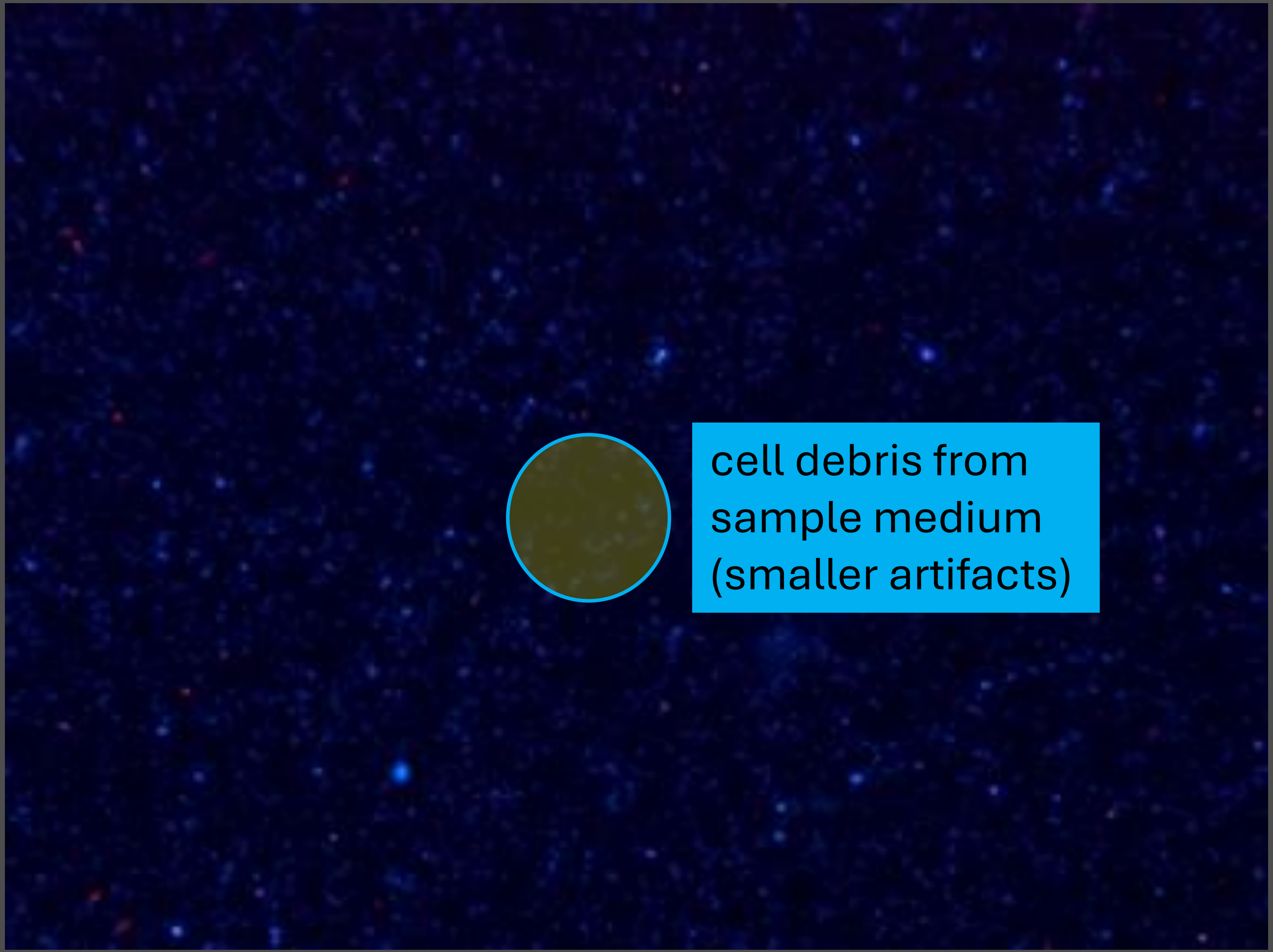
Convert postprocessed images to 8-bit (grayscale)

Results – cont.

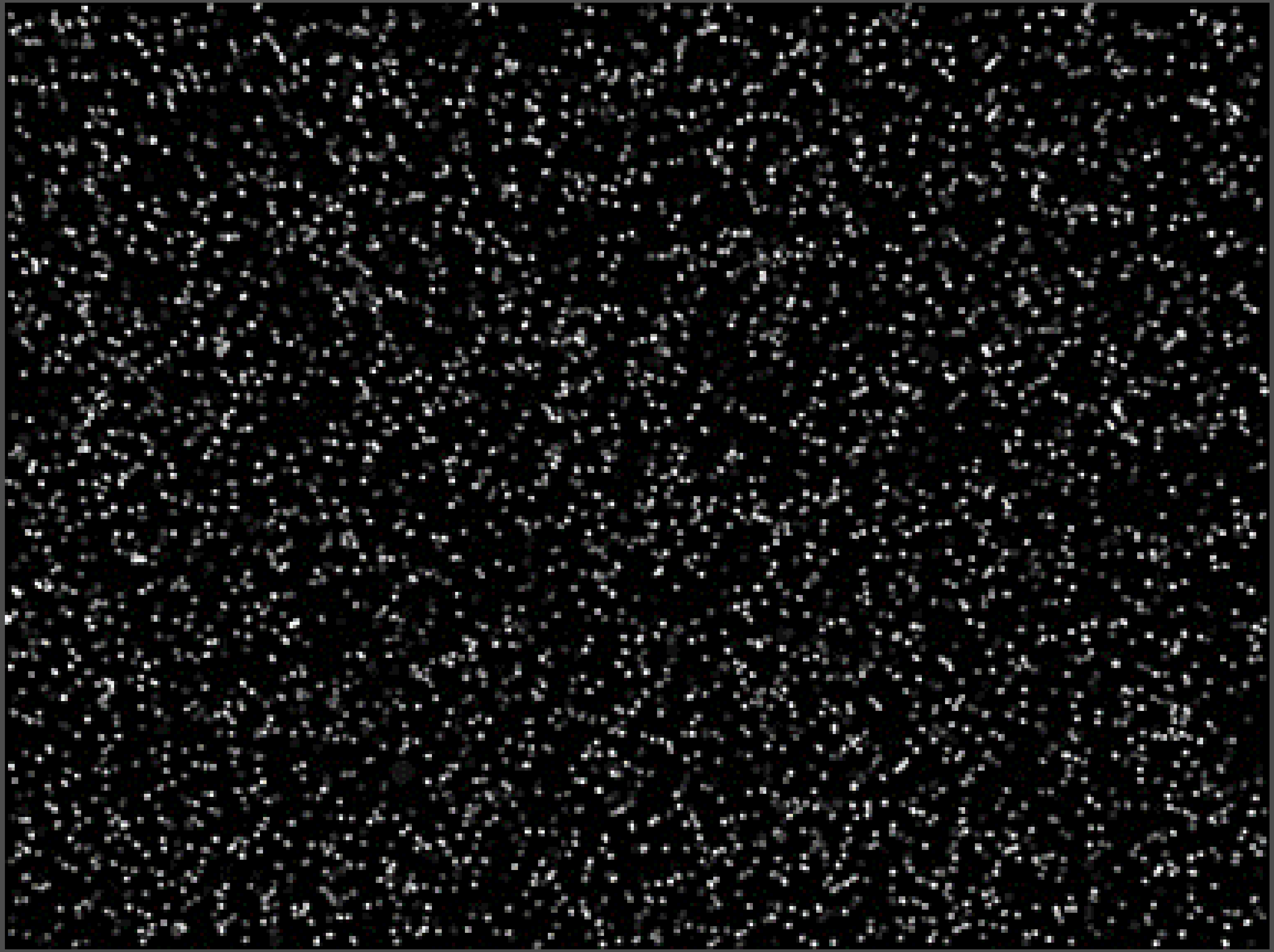
Results – cont.



Results – cont.



Results – cont.

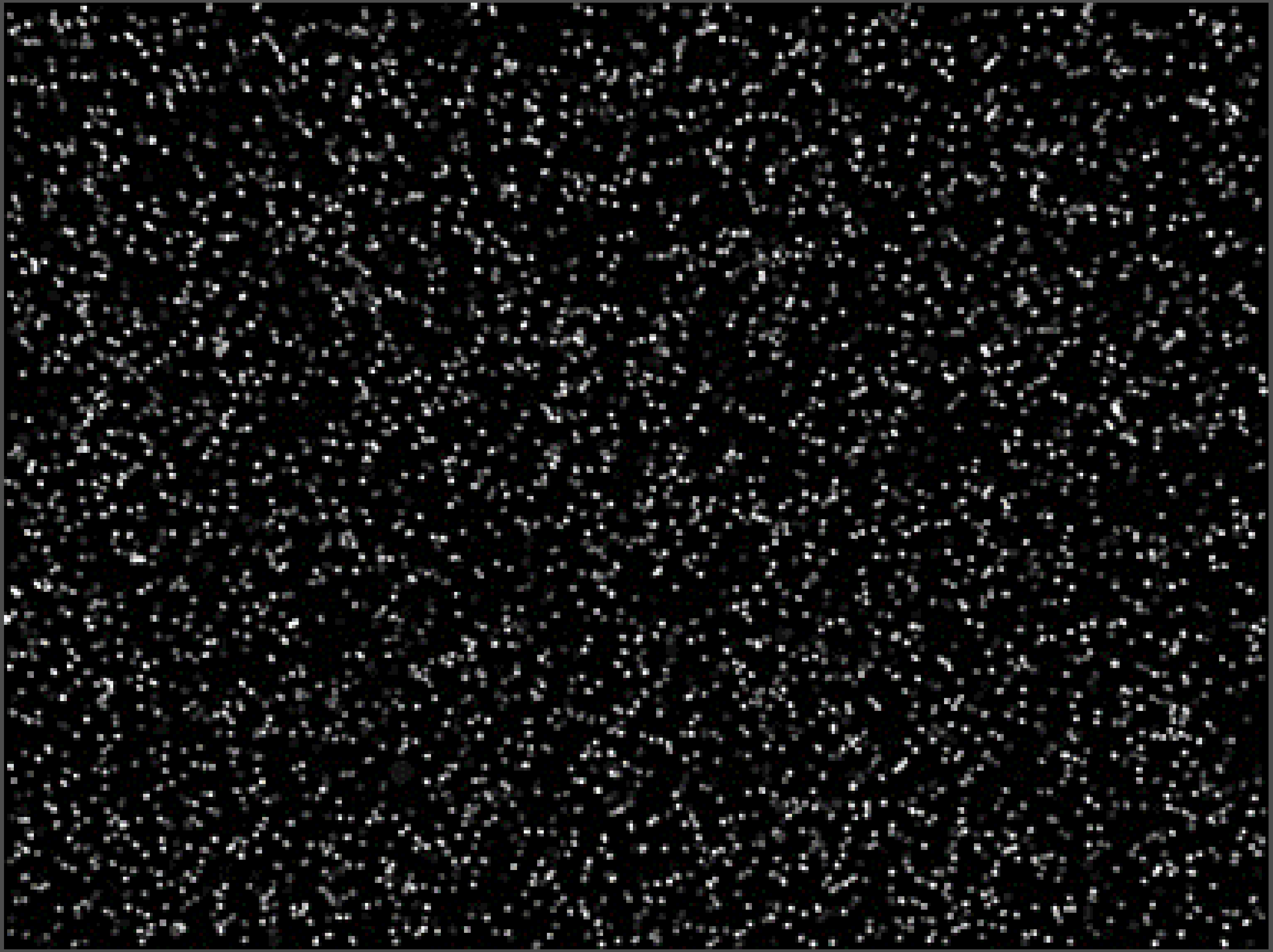


Not so lucky…

We can train a model with our own images or refine the segmentation parameters.

We can also try changing the image brightness during preprocessing.

Performance

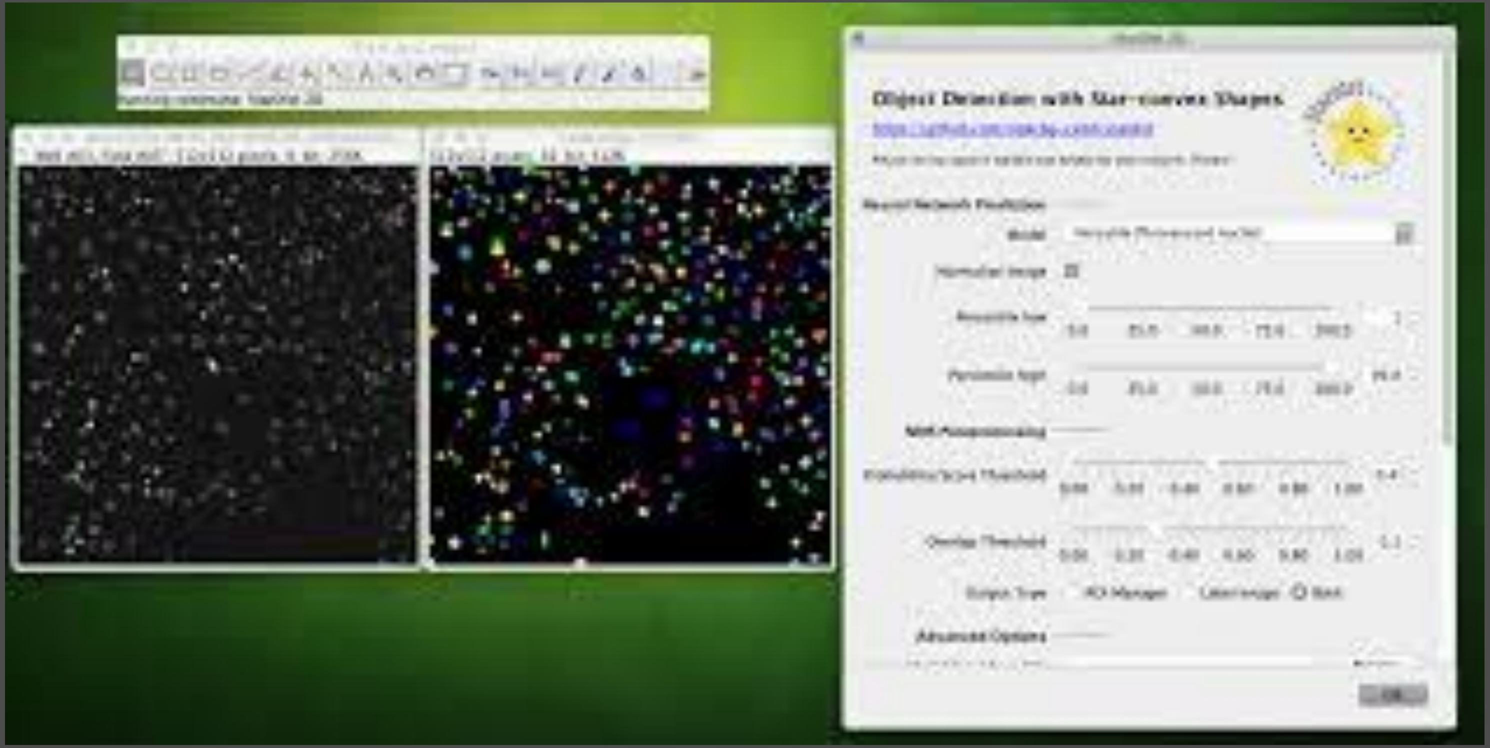


Start-to-finish processing time of **~7.5 s/image**

This is **~97.5% decrease in elapsed time** with the manual workflow that required us to launch ImageJ **(5+ min/image)**

Outcomes

- Developed an efficient cell counting pipeline that can be run entirely from cmd
- Large image artifacts are screened out
- Pre/postprocessed images are saved for qualitative comparisons and sanity checks
- Counts still vary with brightness settings – no “one size fits all”
- Smaller image artifacts are still being counted, so we may need to play with the segmentation algorithm parameters
- 8-bit conversion can be clunky



https://imagej.net/plugins/stardist