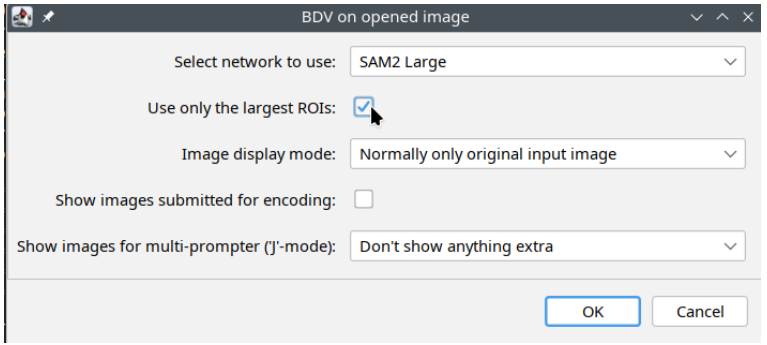
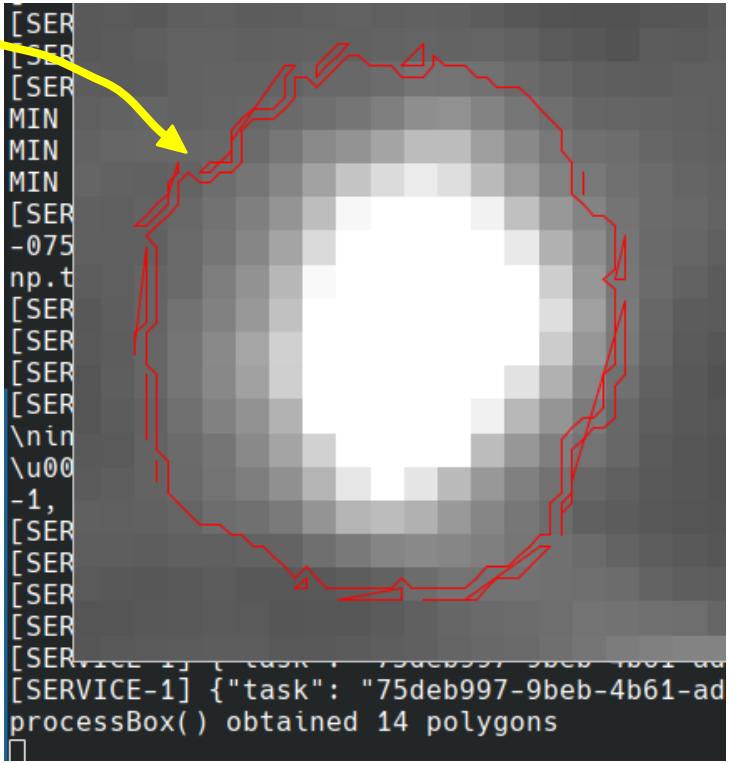
SAM2 is great, but:

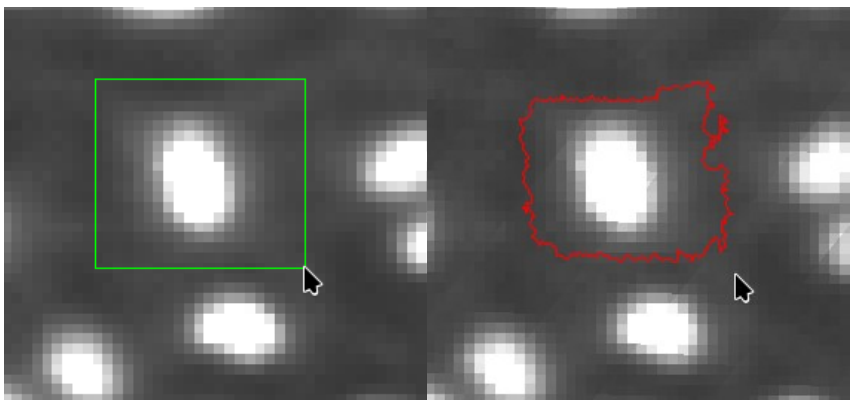
- SAM2 networks over-segment (produce too many polygons/segments for one object)
  - At least this is the case for SAM2 Tiny and Large
  - Sooo:



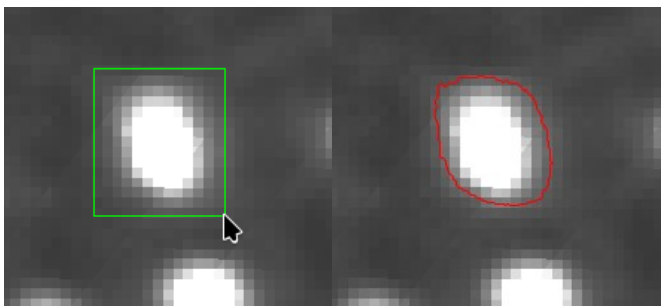


  - Taking largests polygon works well, but ultimately means:
    (at most) one obj is annotated per one prompt

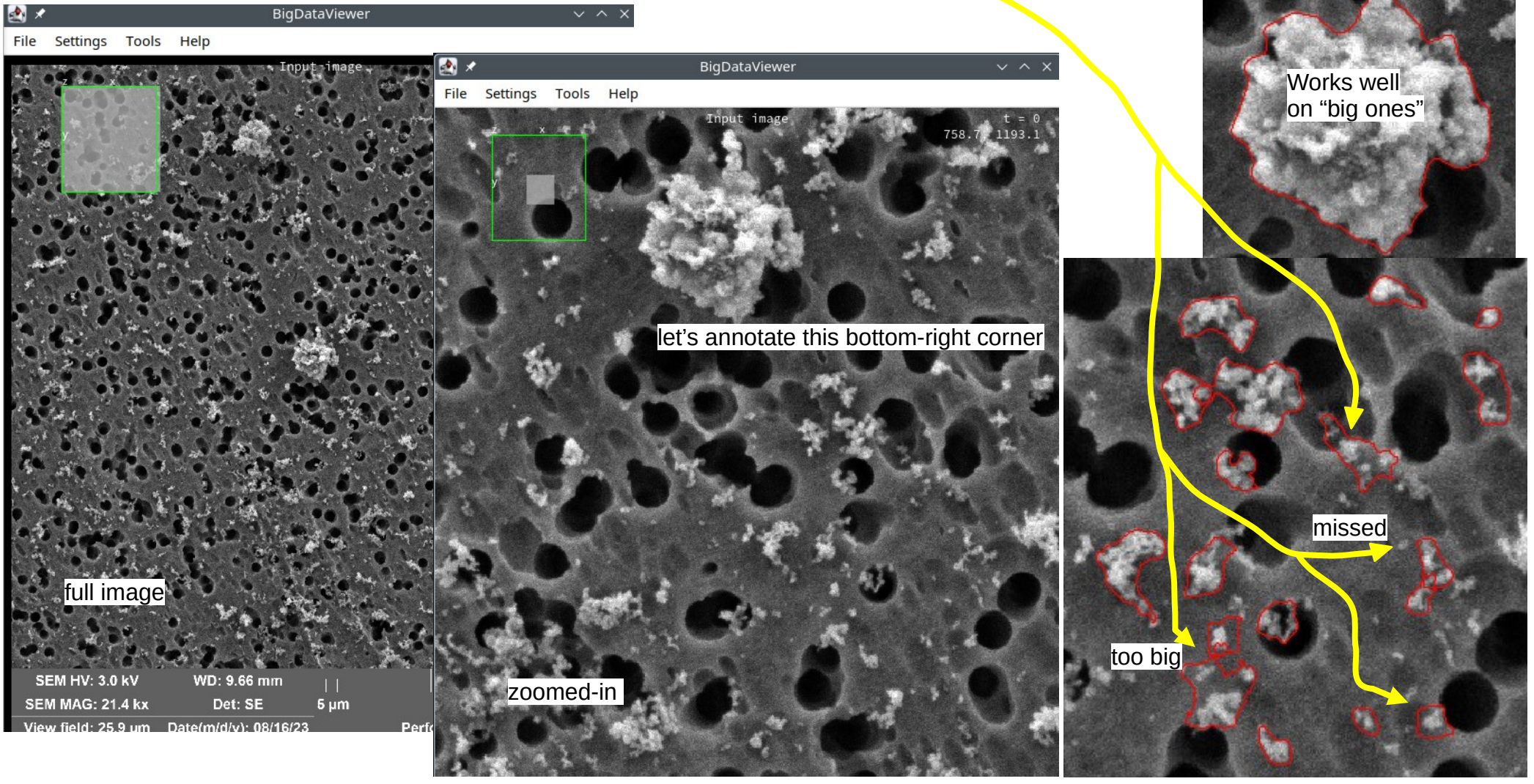
- There exist data when one has to be carefull with prompts
  ...not to make them too large, too “vague” around the obj of interest



VS.



Dust-particles electron microscopy example where one wants to avoid “careless prompting”:



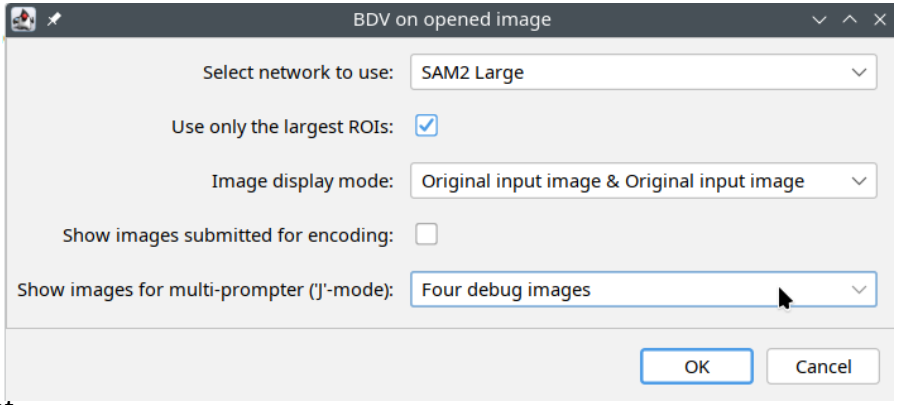
What to do about it?

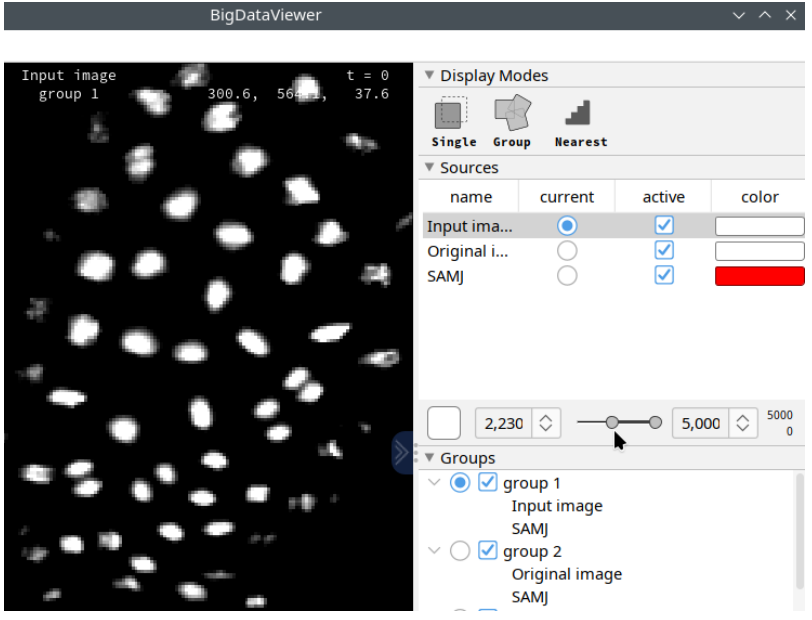
- Annotator has to be careful (and patient) → takes very long

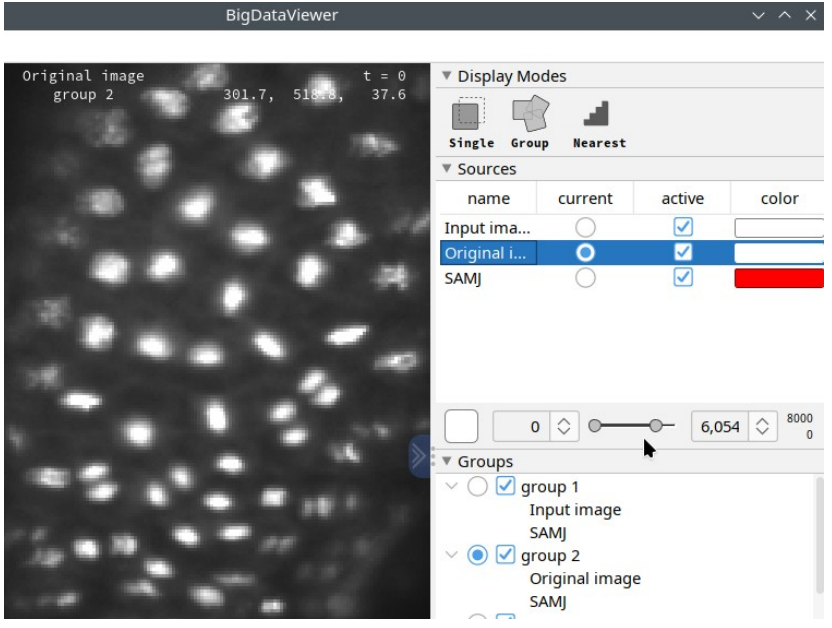
- Or, have system that **fine-tunes annotator's prompt**:
  ...by creating seeds where user originally prompted,
  and runs SAM iteratively per each seeds separately

  - Here, **IMPLICIT/internal** fine-tuner is used, for which:

    - It is better to show the same original image twice
      - Once (*left fig.*) as contrast-adjusted so that "seeds" are apparent
      - Once (*right fig.*) with "normal view" which is here _only_ for visual check if the annotation went well
      - Note: BDV automatically sets up viewing groups, display mode etc.; user "only" adjusts contrast settings







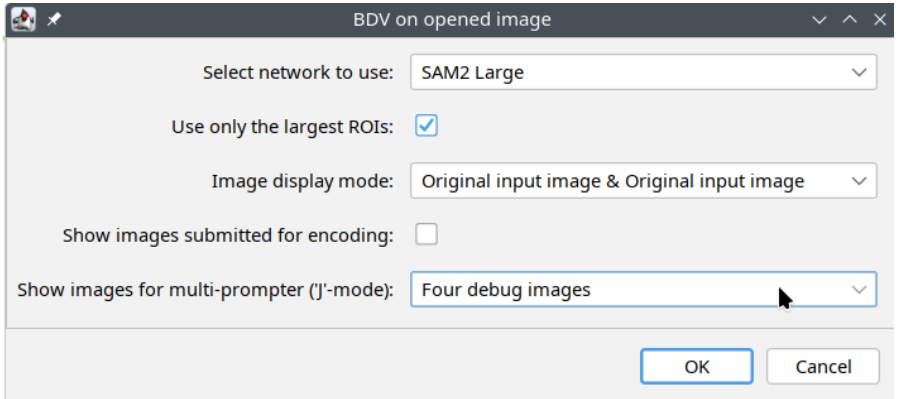
What to do about it?

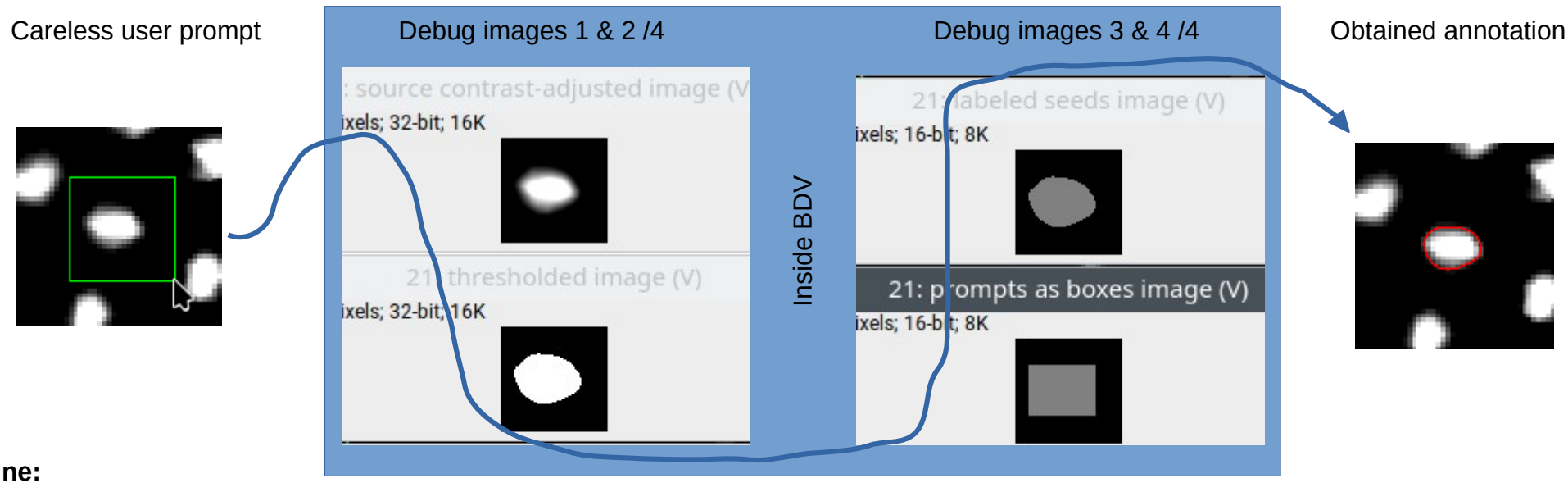
- Annotator has to be careful (and patient) → takes very long

- Or, have system that **fine-tunes annotator's prompt**:
  ...by creating seeds where user originally prompted,
  and runs SAM iteratively per each seeds separately

  - Here, **IMPLICIT/internal** fine-tuner is used, for which:

    - It is better to show the same original image twice
      - Once (*left fig.*) as contrast-adjusted so that "seeds" are apparent
      - Once (*right fig.*) with "normal view" which is here _only_ for visual check if the annotation went well
      - **Note**: Obviously, you don't have to have debug images displayed at all….

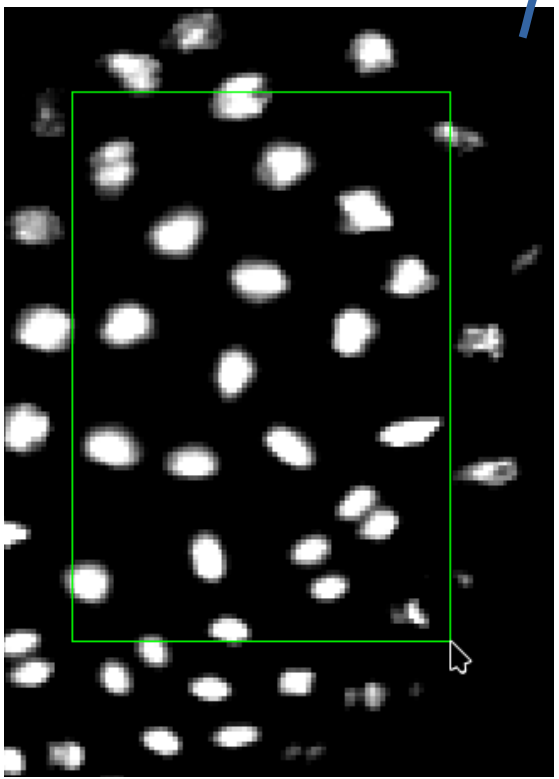


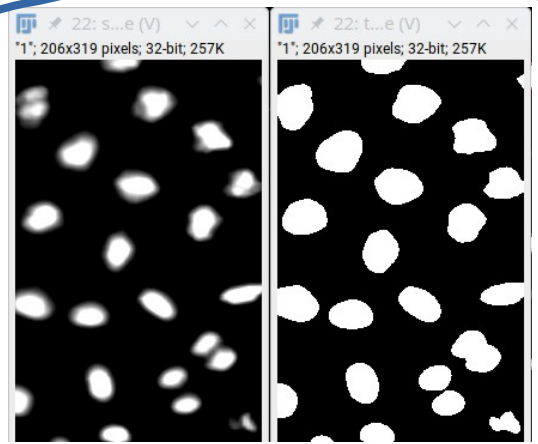


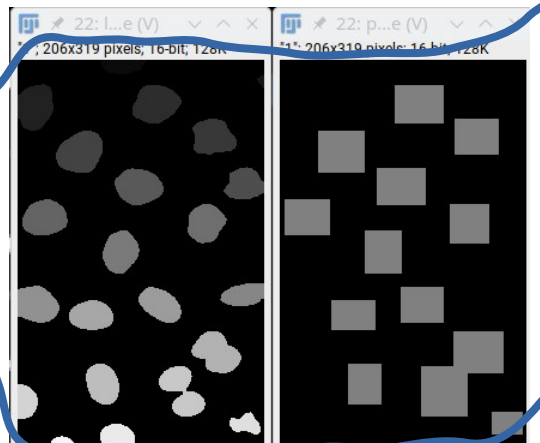
Pipeline:
User-given GreenBox cropped out the by-user-contrast-adjusted image, threshold >0, CCA, bounding box → SAM on **orig image** → yay! :-)

Multi-prompting as a Side-effect (it's exactly the same pipeline as for the **fine-tuning**)

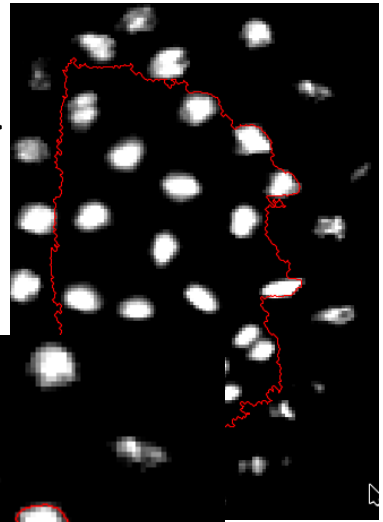
The same INTERNAL image processing
to create seeds to obtain prompts.



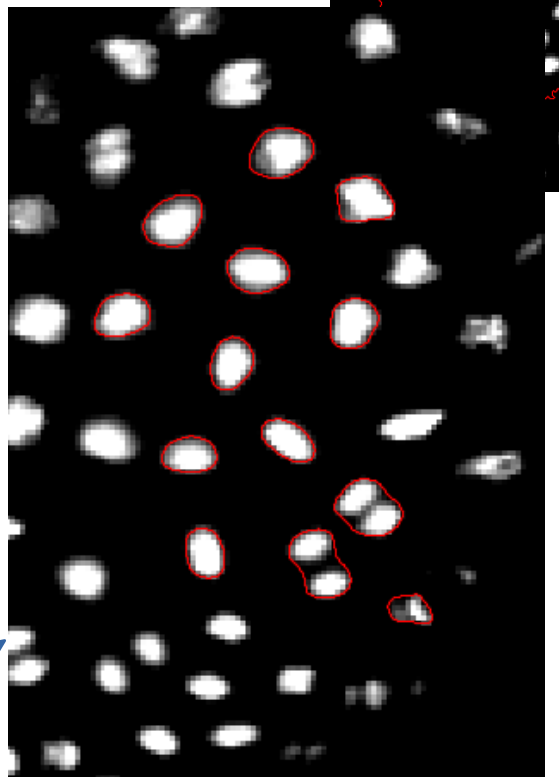




Standard SAMJ:
---------->



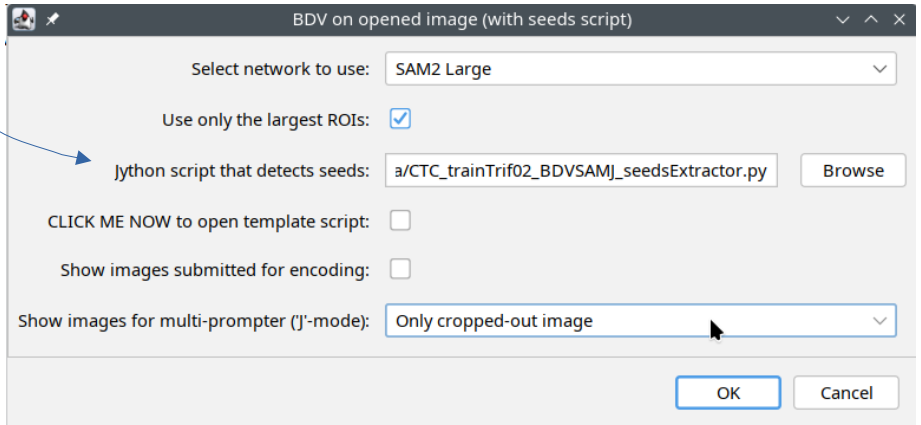
Multi-prompting:



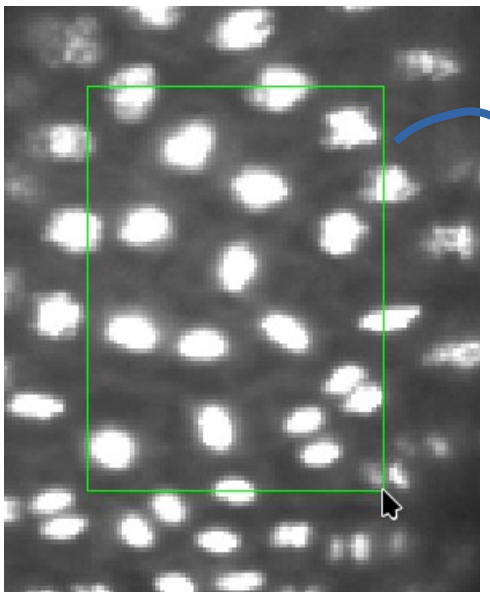
Small and boundary-touching prompts/boxes are ignored.

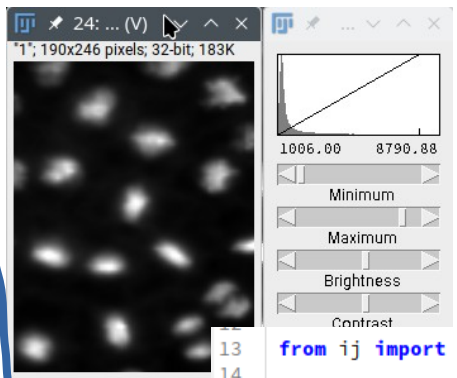
What if I don’t wanna adjust contrast and/or don’t wanna use thresholding to create seeds?

→ Just point BDV on a script that converts crop-out of the original image to an image with seed(s):



- Here, **EXPLICIT/external** fine-tuner is used, for which:
  - The mechanism works exactly the same
  - *Except that instead of internal thresholding, an external script is executed to obtain seed image*
  - *Except that original, not-contrast-adjusted crop is used*
  - Seeds CCA and SAM prompting happens again in BDV





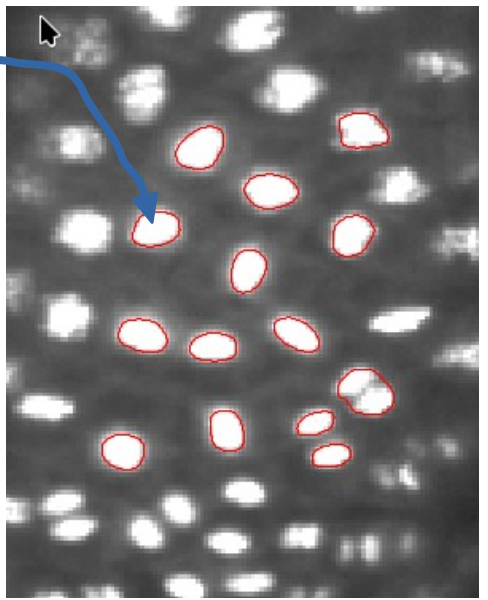
Inside BDV:

SAM Prompting

Bounding boxes

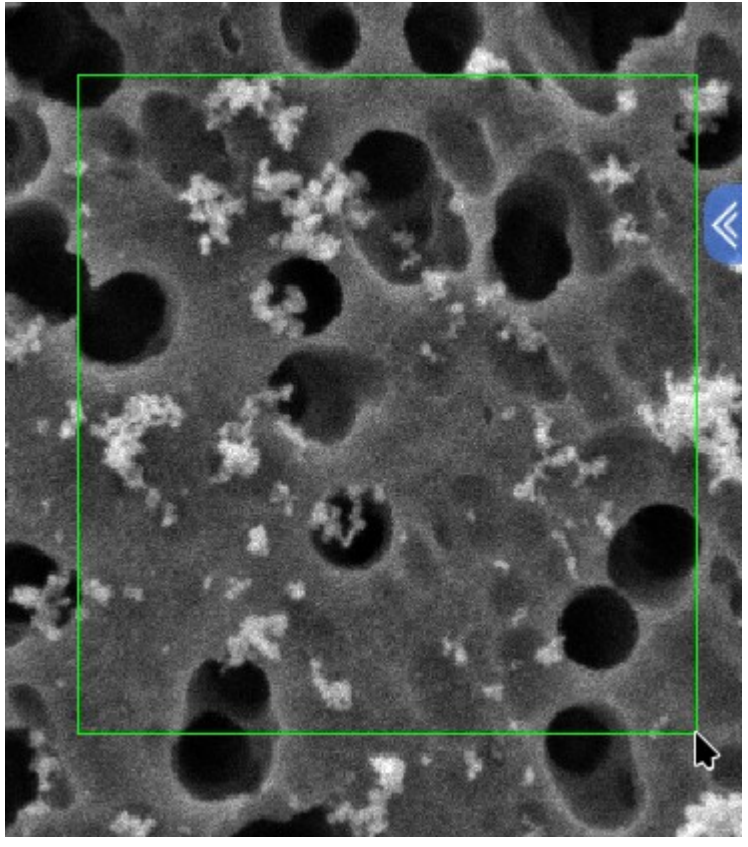
CCA’ing seeds

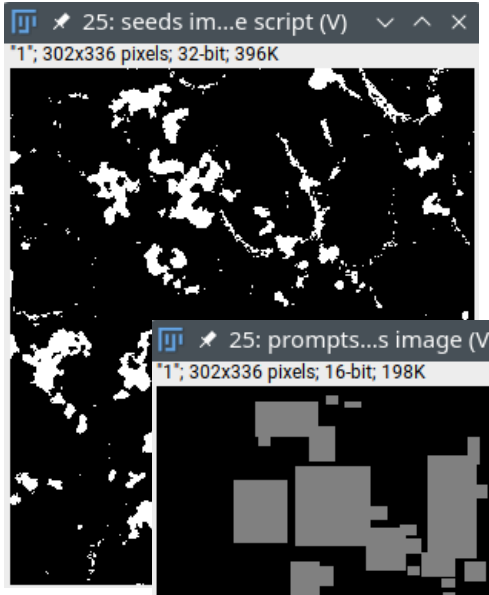
```
from ij import IJ

def threshold(thres_val):
    pxs = imp.getProcessor().getPixels()
    for i in range(len(pxs)):
        pxs[i] = 1 if pxs[i] > thres_val else 0

IJ.run(imp, "Maximum...", "radius=3")
threshold(4000)
```

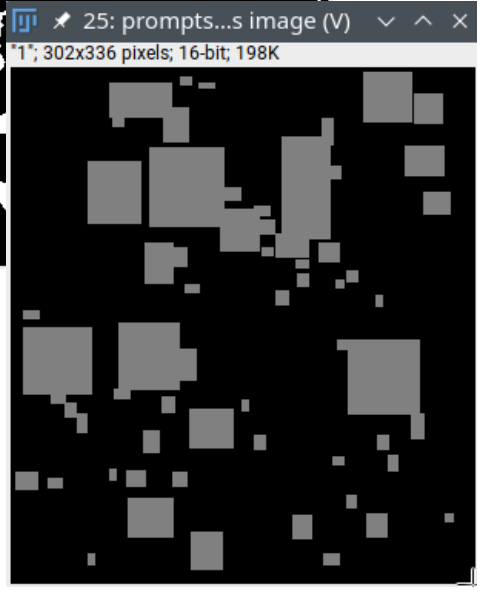


Dust-particles electron microscopy example again
where **externally-obtained seeds guided the multi-prompting**:

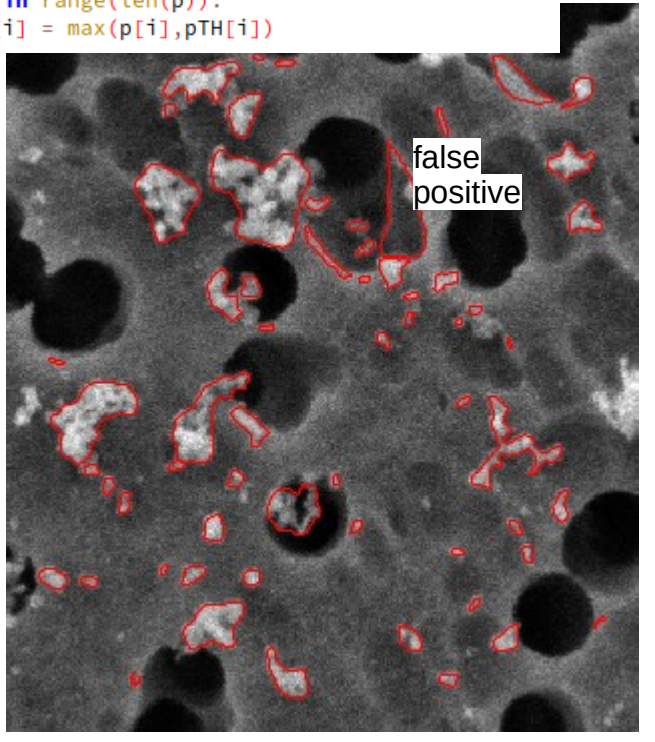
Result is more detailed (compare with Slide 2),
but not perfect either (has false positives – a lousy script I prepared).








```
from ij import IJ

def threshold(imp, thres_val):
    pxs = imp.getProcessor().getPixels()
    for i in range(len(pxs)):
        pxs[i] = 1 if pxs[i] > thres_val else 0

impTH = imp.duplicate();
IJ.run(impTH, "Top Hat...", "radius=5");
threshold(impTH, 10000)

threshold(imp,35000)

p = imp.getProcessor().getPixels()
pTH = impTH.getProcessor().getPixels()
for i in range(len(p)):
    p[i] = max(p[i],pTH[i])
```



Is there a template script to start with?

Yes.

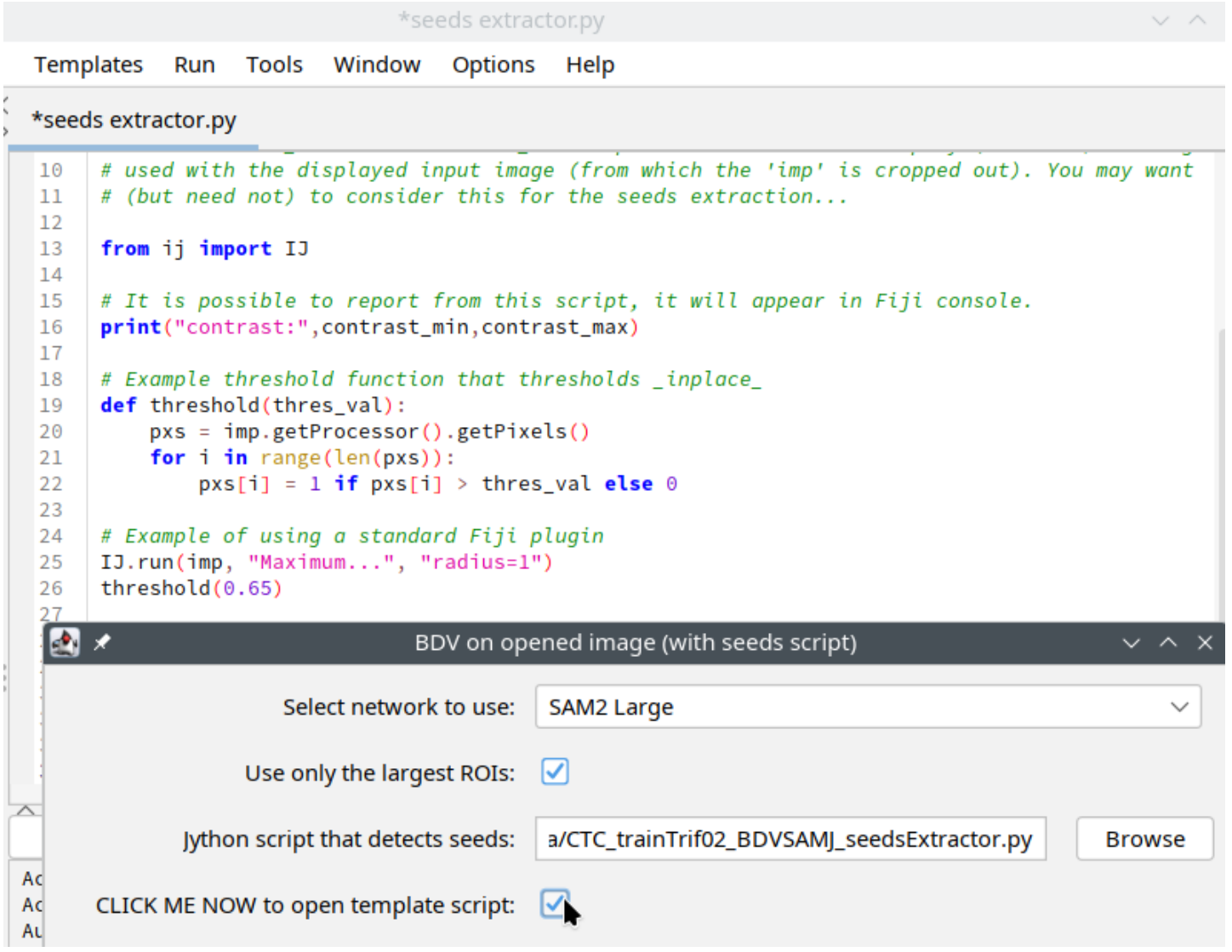
(
Just don't know how to create button in IJ2 auto-generated dialogs, so one has to toggle the "CLICK ME NOW" checkbox….

And the script editor comes out.
)

- One cancels/closes the "BDV on opened…" dialog
- Edit the script
- Save-as the script somewhere

- Re-open the "BDV on opened…"
- Point on the edited script

The script is re-read from disk prior every use of it, so one can adjust the seeds-extracting-pipeline during the same annotation session. This is useful, e.g., when different thresholding values work in different corners of the data.



The full initial/template script:

Note:

One can excercise and fine-tune the script outside BDV. It's a normal Jython script afterall...

*seeds extractor.py

```
# RESAVE THIS SEEDS SCRIPT AND POINT THE BDV_WITH_SEEDS DIALOG ON IT

#@ ImagePlus imp
#@ float contrast_min
#@ float contrast_max

# It is important that seeds (any non-zero pixels) are stored directly into the input 'imp' image!

# The 'contrast_min' and 'contrast_max' report the current BDV display (contrast) setting
# used with the displayed input image (from which the 'imp' is cropped out). You may want
# (but need not) to consider this for the seeds extraction...

from ij import IJ

# It is possible to report from this script, it will appear in Fiji console.
print("contrast:",contrast_min,contrast_max)

# Example threshold function that thresholds _inplace_
def threshold(thres_val):
    pxs = imp.getProcessor().getPixels()
    for i in range(len(pxs)):
        pxs[i] = 1 if pxs[i] > thres_val else 0

# Example of using a standard Fiji plugin
IJ.run(imp, "Maximum...", "radius=1")
threshold(0.65)

# Don't use the updateAndRepaintWindow() in conjunction with BDV+SAMJ,
# but it is useful when running (debugging) this script directly from Fiji
# (e.g. on some of the debug crop-out that came from BDV+SAMJ).
#
# imp.updateAndRepaintWindow()

```