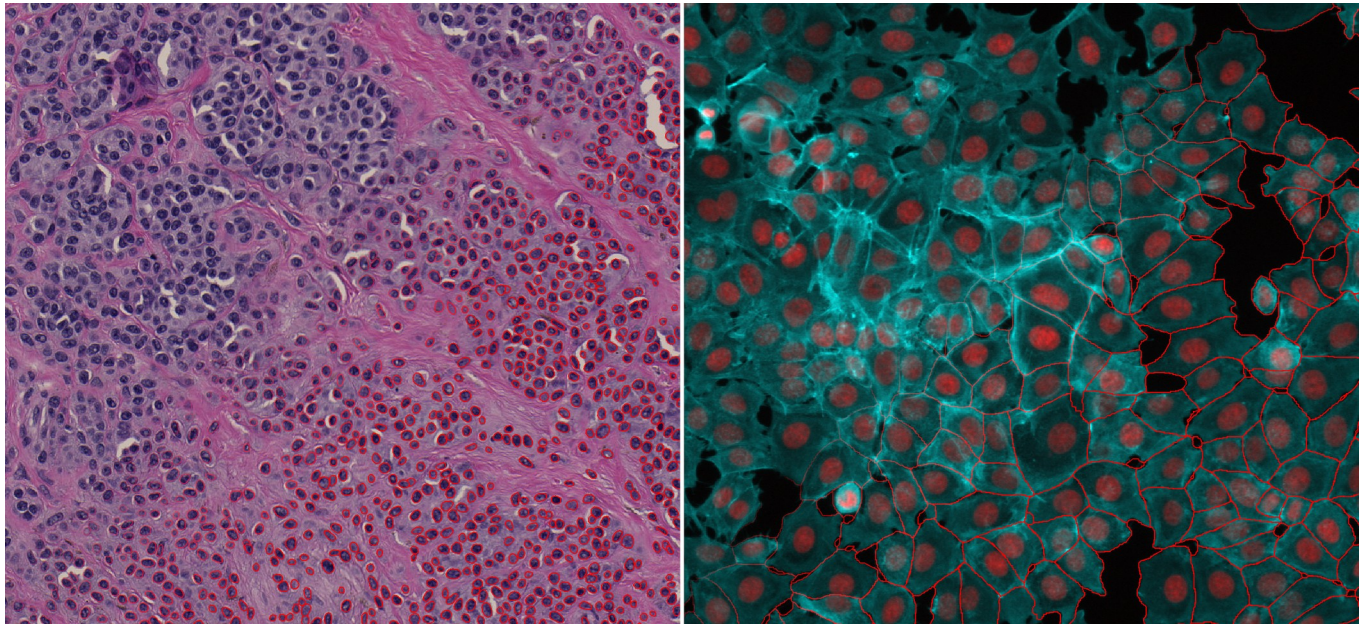
Chapter 5

Cells detection

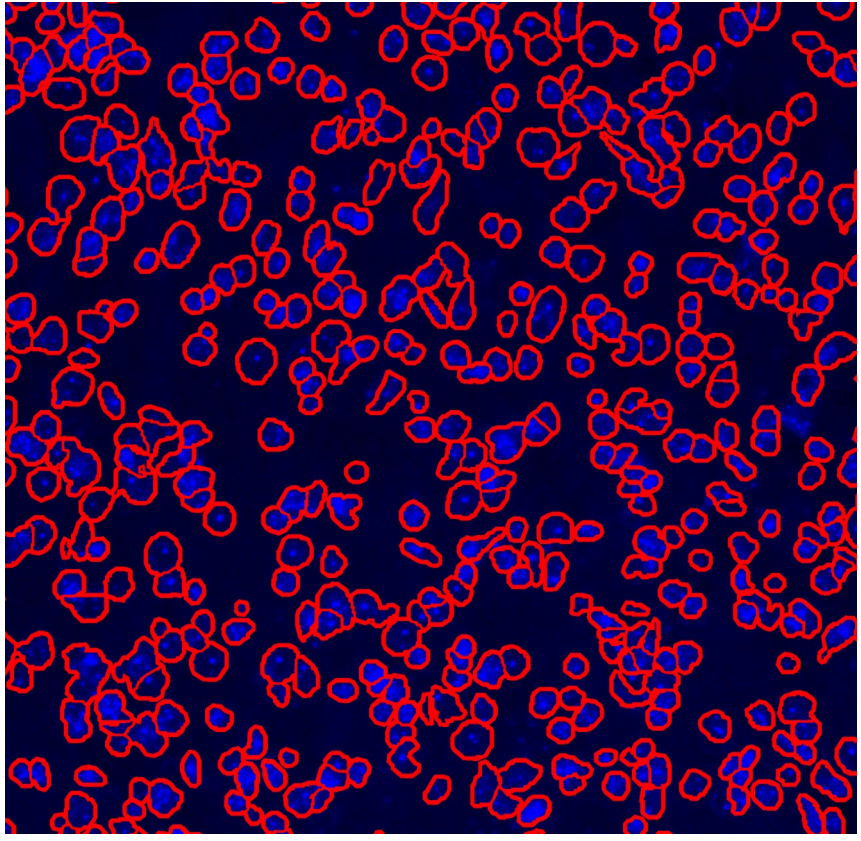


BAECKER Volker
BENEDETTI Clément
HASSEN-KHODJA Cédric

Introduction

Goals:

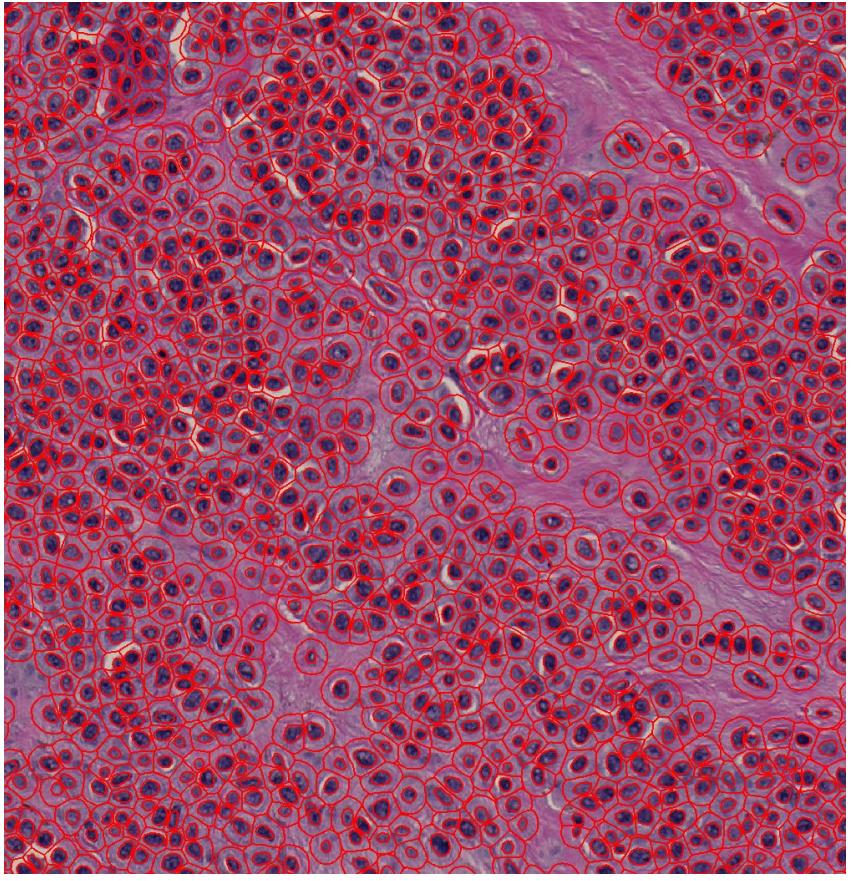
- Segment **individual** nuclei.



Introduction

Goals:

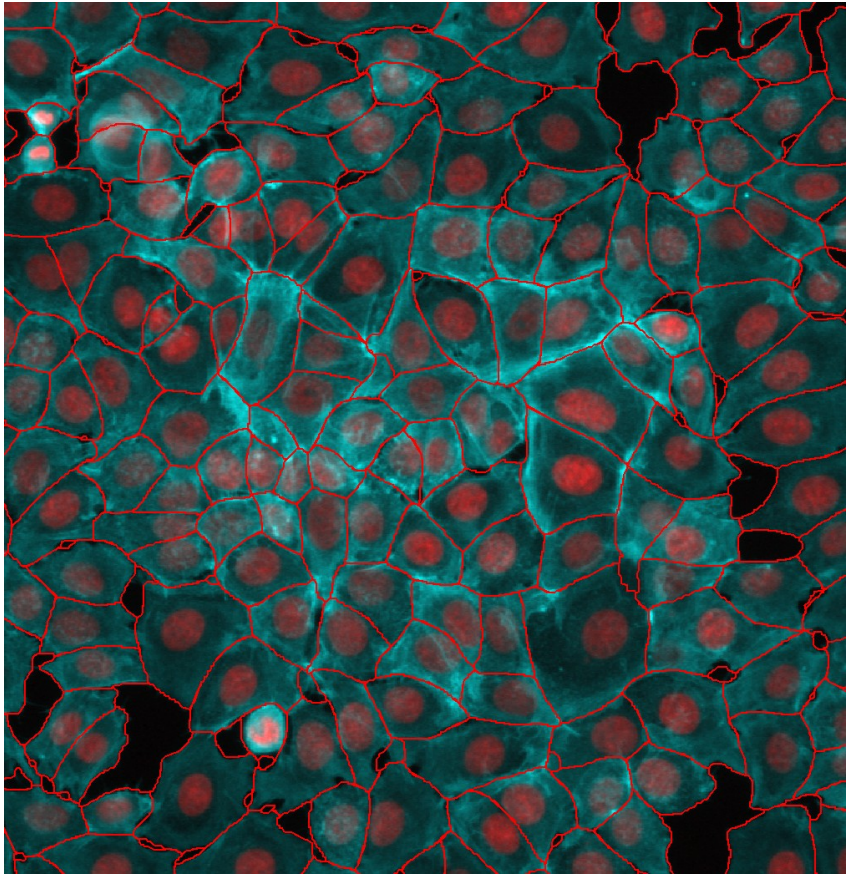
- Segment **individual** nuclei.
- Segment cells:
  - Deduce from nuclei.



Introduction

Goals:

- Segment **individual** nuclei.
- Segment cells:
  - Deduce from nuclei.
  - Cyto/membrane/actin/… staining.



Introduction

Goals:

- Segment **individual** nuclei.
- Segment cells:
  - Deduce from nuclei.
  - Cyto/membrane/actin/… staining.
- Extract statistics:
  - Locally (per cell/nucleus)

Area µm^2	8157
Length µm	402.3282
Circularity	0.6333
Solidity	0.8806
Max diameter µm	128.5342
Min diameter µm	106.6958
Channel 1: Mean	799.2405
Channel 1: Median	560
Channel 1: Min	320
Channel 1: Max	2944
Channel 1: Std.Dev.	567.6478
Channel 2: Mean	4963.4067
Channel 2: Median	4656
Channel 2: Min	1200
Channel 2: Max	12016
Channel 2: Std.Dev.	2041.4315
Channel 3: Mean	2389.7266
Channel 3: Median	2256
Channel 3: Min	896
Channel 3: Max	5344
Channel 3: Std.Dev.	537.9851
Channel 4: Mean	544.2646
Channel 4: Median	472
Channel 4: Min	254
Channel 4: Max	8769
Channel 4: Std.Dev.	490.7437

Introduction

Goals:

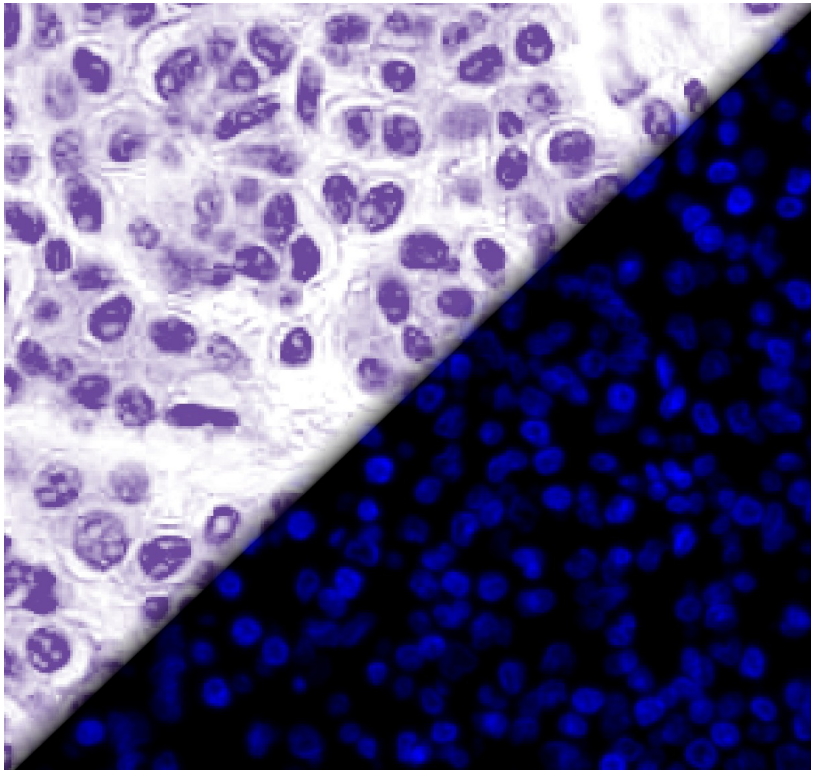
- Segment **individual** nuclei.
- Segment cells:
  - Deduce from nuclei.
  - Cyto/membrane/actin/… staining.
- Extract statistics:
  - Locally (per cell/nucleus)
  - Globally (per slide/organ/region)

Area µm^2	1310720
Perimeter µm	4608
Num Detections	284

1. QuPath's built-in method

Workflow:

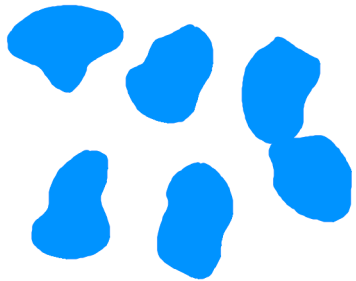
- Based on:
  - **Fluo**: Nuclei
  - **IHC**: Deconvolued hematoxylin



1. QuPath's built-in method

Workflow:

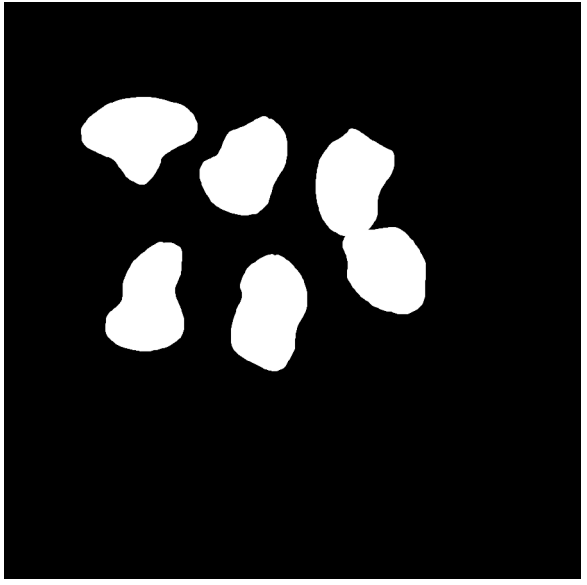
- Based on:
  - **Fluo**: Nuclei
  - **IHC**: Deconvolued hematoxylin
- Start from the intensities



1. QuPath's built-in method

Workflow:

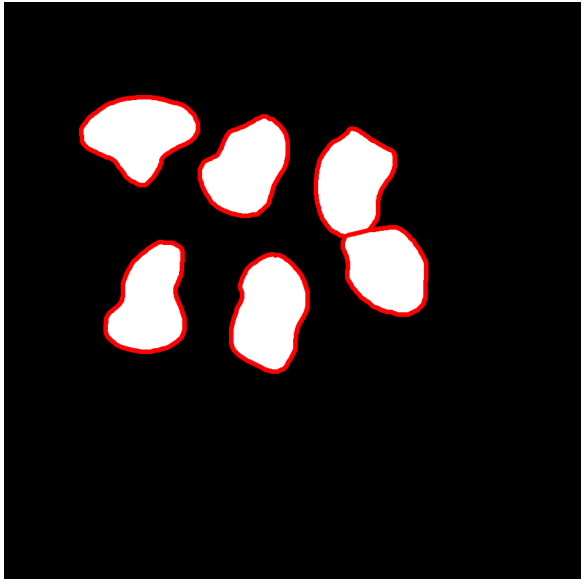
- Based on:
  - **Fluo**: Nuclei
  - **IHC**: Deconvolued hematoxylin
- Start from the intensities
- Make a mask by thresholding



1. QuPath's built-in method

Workflow:

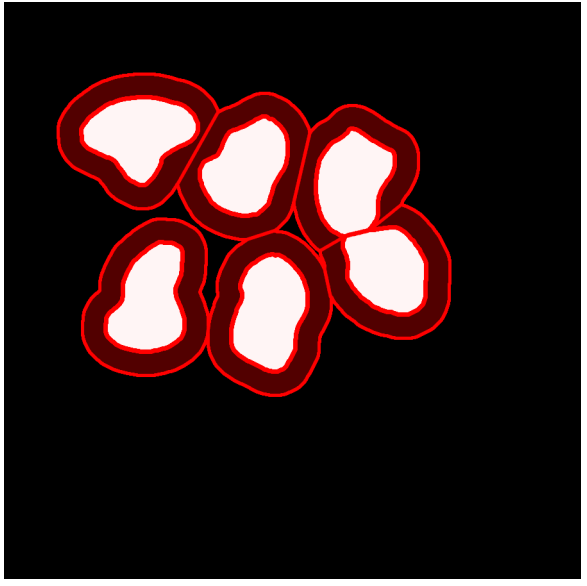
- Based on:
  - **Fluo**: Nuclei
  - **IHC**: Deconvolued hematoxylin
- Start from the intensities
- Make a mask by thresholding
- Break aggregates with watershed



1. QuPath's built-in method

Workflow:

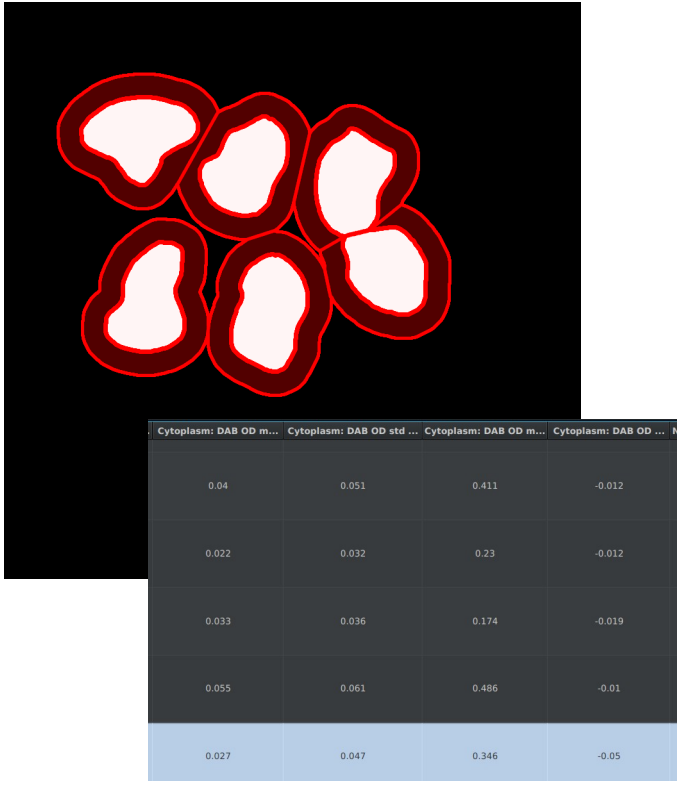
- Based on:
  - **Fluo**: Nuclei
  - **IHC**: Deconvolued hematoxylin
- Start from the intensities
- Make a mask by thresholding
- Break aggregates with watershed
- Dilate polygons (with collisions)



1. QuPath's built-in method

Workflow:

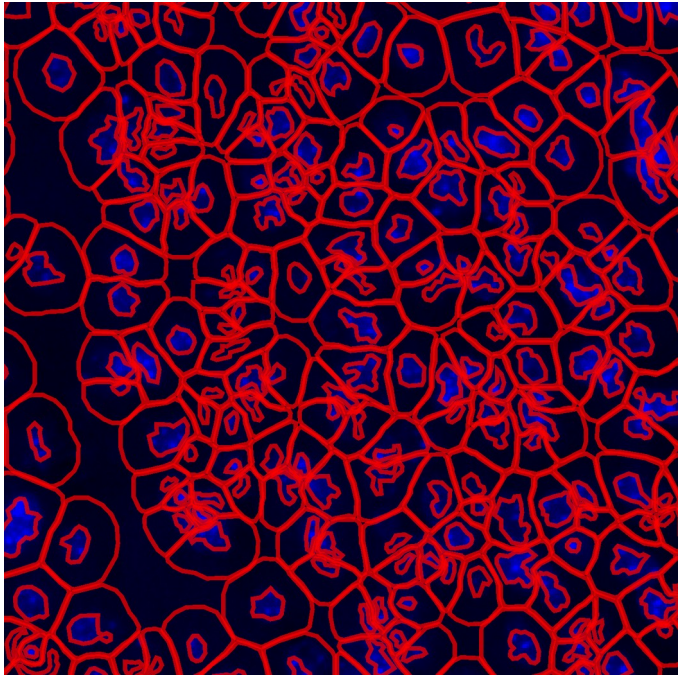
- Based on:
  - **Fluo**: Nuclei
  - **IHC**: Deconvolued hematoxylin
- Start from the intensities
- Make a mask by thresholding
- Break aggregates with watershed
- Dilate polygons (with collisions)
- Measure (shape + intensities)



1. QuPath's built-in method

Workflow:

- Based on:
  - **Fluo**: Nuclei
  - **IHC**: Deconvolued hematoxylin
- Start from the intensities
- Make a mask by thresholding
- Break aggregates with watershed
- Dilate polygons (with collisions)
- Measure (shape + intensities)

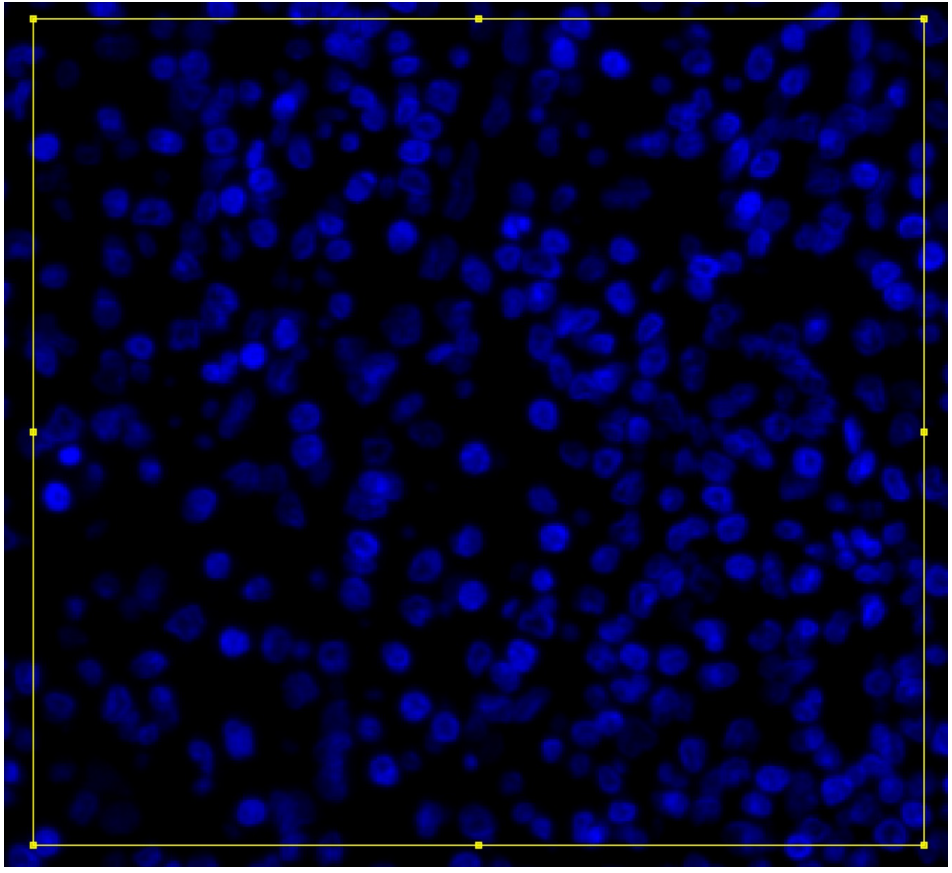


⚠️ Prone to errors!

1. QuPath’s built-in method

In QuPath:

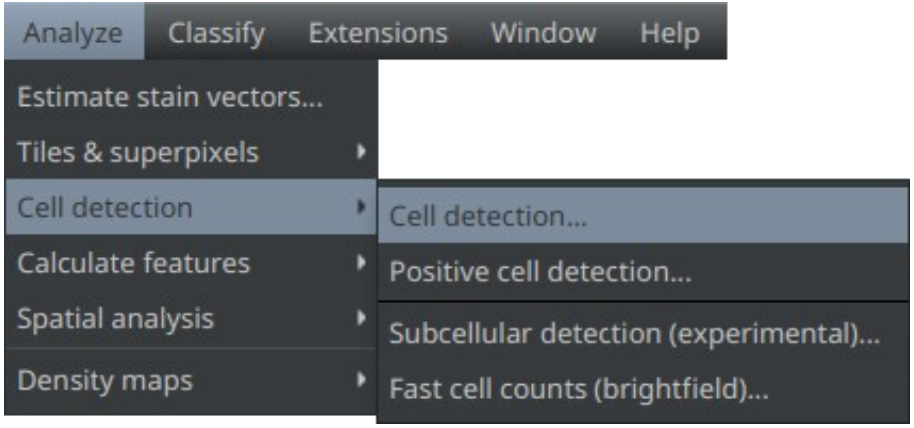
- Works only in a parent annotation.



1. QuPath's built-in method

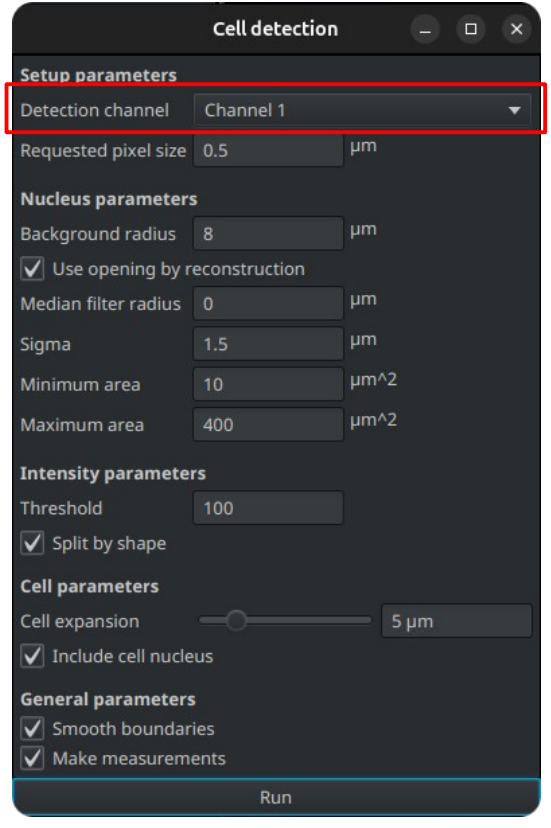
In QuPath:

- Works only in a parent annotation.
- Open the "Cell detection" tool.



1. QuPath's built-in method

In QuPath:

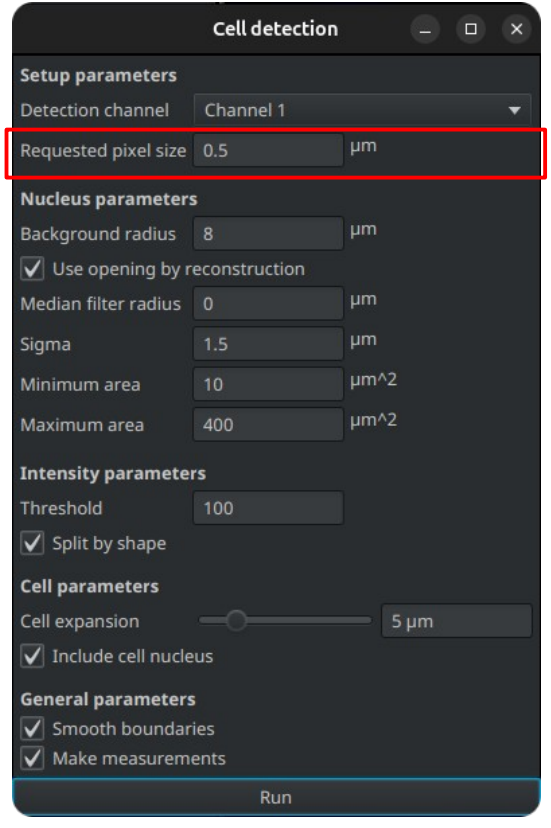


Detection channel:

- Channel for segmentation.

1. QuPath's built-in method

In QuPath:

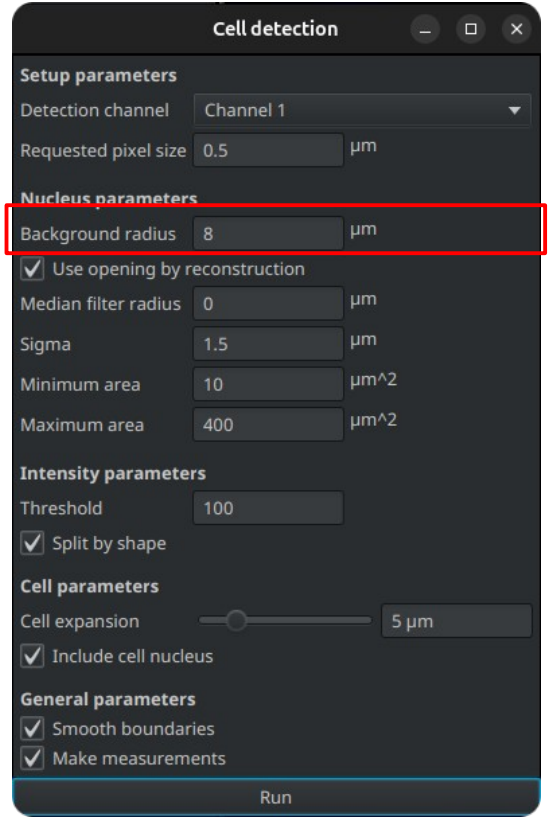


Requested pixel size:

- Resolution to use for segmentation.
- Higher number == less details.

1. QuPath's built-in method

In QuPath:

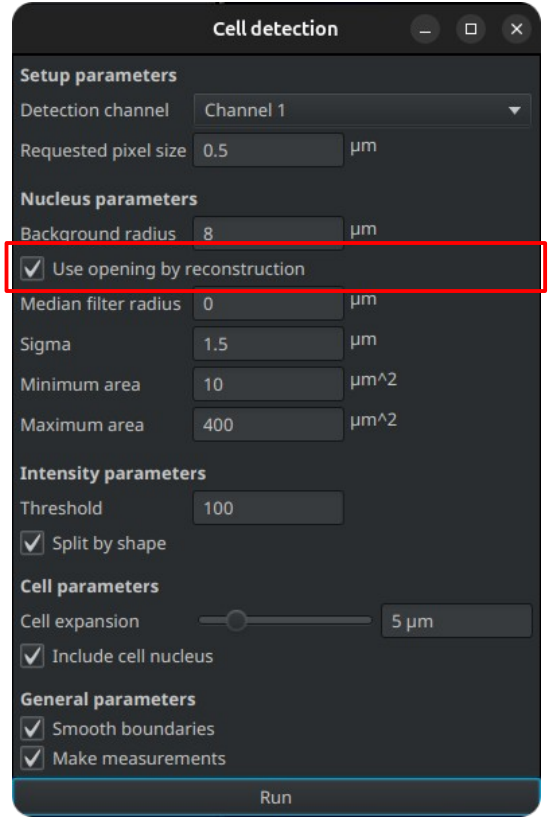


Background radius:

- Radius used for:
  - Rolling ball.
  - Opening by reconstruction.

1. QuPath's built-in method

In QuPath:

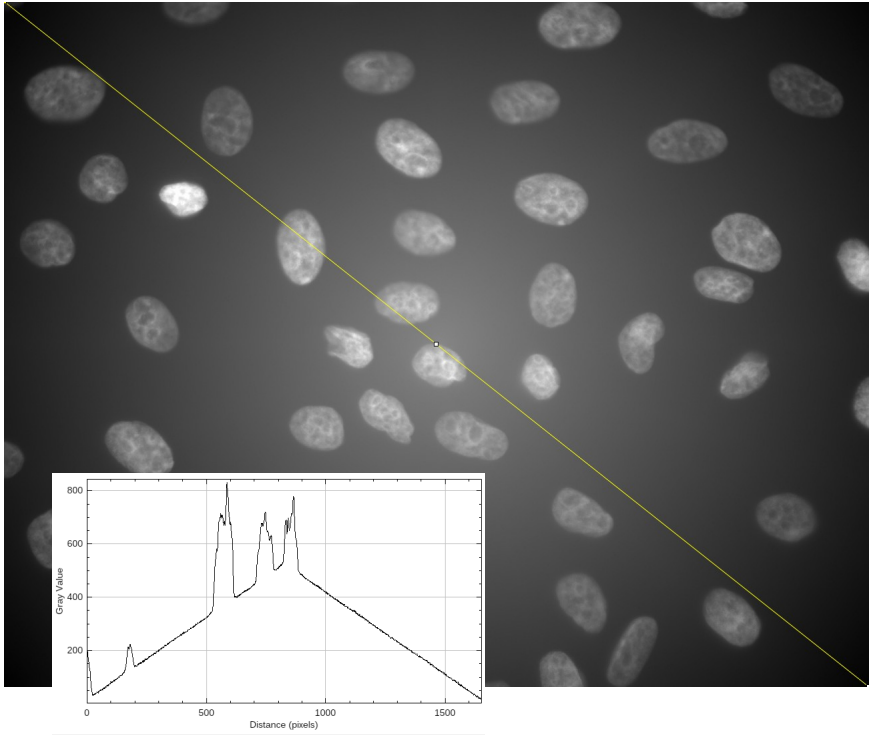


Use opening by reconstruction:

- ☐: Rolling ball
- ☑: Opening by reconstruction

1. QuPath's built-in method

In QuPath:

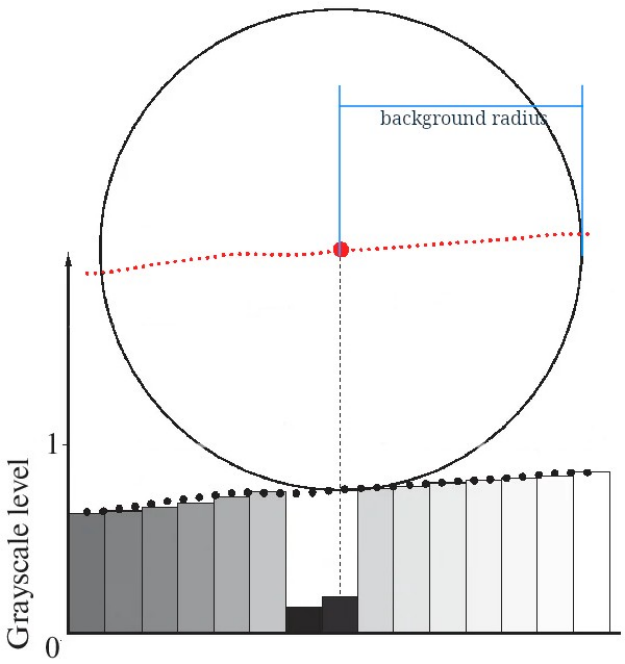


📝 **Rolling ball:**

- Useful is **gradient** in **background**.

1. QuPath's built-in method

In QuPath:



📝 **Rolling ball:**

- Useful is **gradient** in **background**.
- Ball rolls: we record the **center's height**.
- **Ball radius** == "background radius".

1. QuPath's built-in method

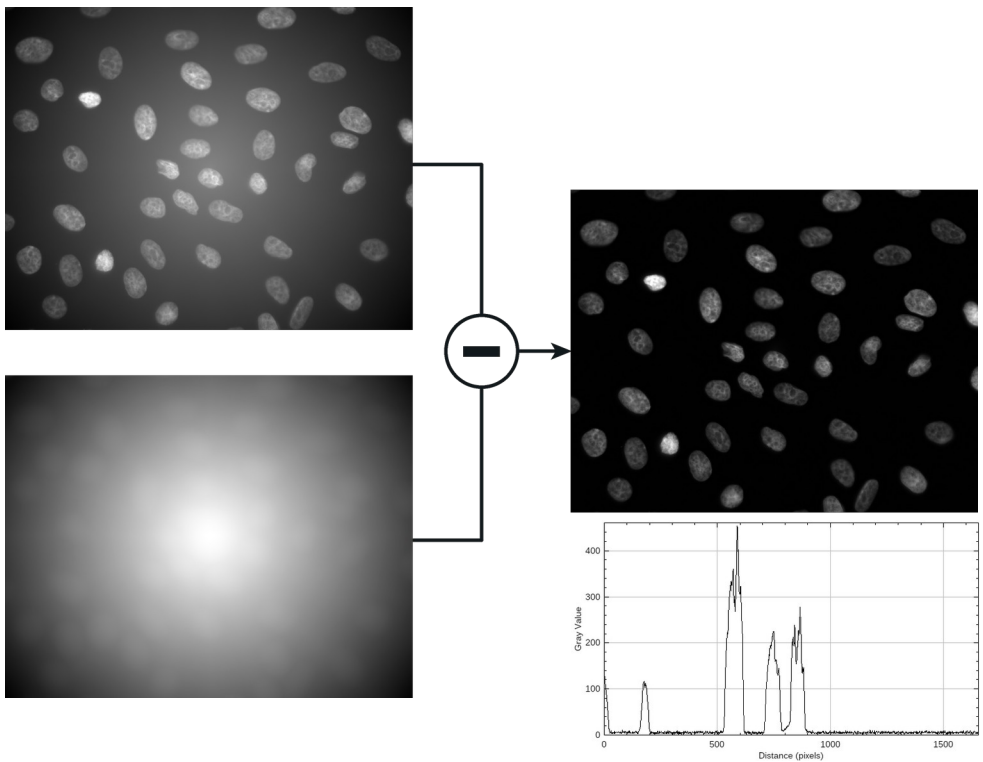
In QuPath:

📝 **Rolling ball:**

- Useful is **gradient** in **background**.
- Ball rolls: we record the **center's height**.
- **Ball radius** == "background radius".
- Produce an image of the background.

1. QuPath's built-in method

In QuPath:

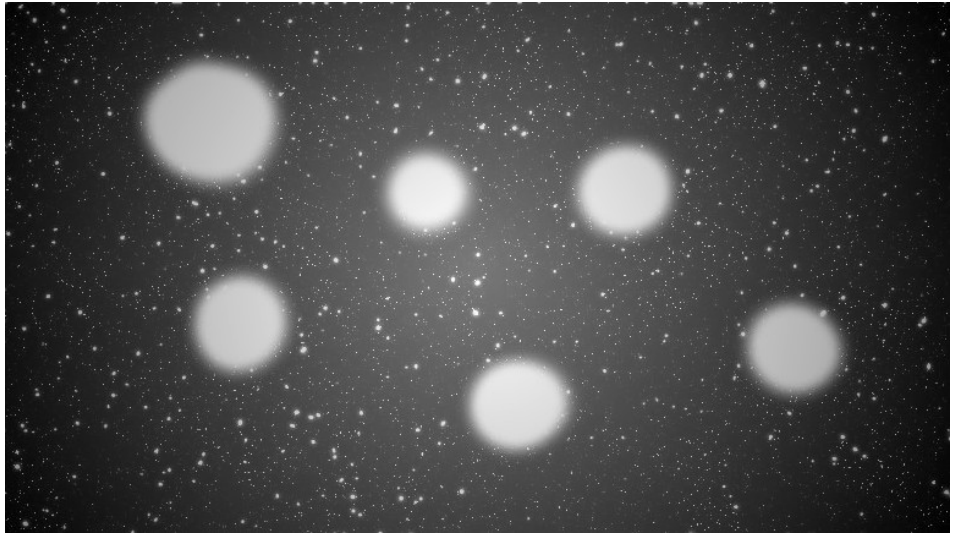


📝 **Rolling ball:**

- Useful is **gradient** in **background**.
- Ball rolls: we record the **center's height**.
- **Ball radius** == "background radius".
- Produce an image of the background.
- **Subtract** the **background** from the image.

1. QuPath's built-in method

In QuPath:

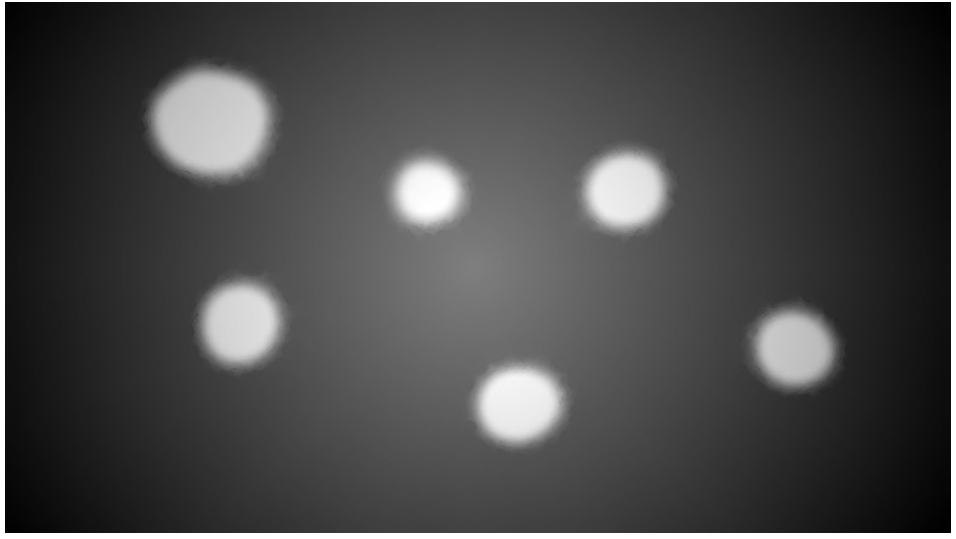


📝 **Opening by reconstruction:**

- Removes **gradient** and **big structures**.
- Steps:

1. QuPath's built-in method

In QuPath:

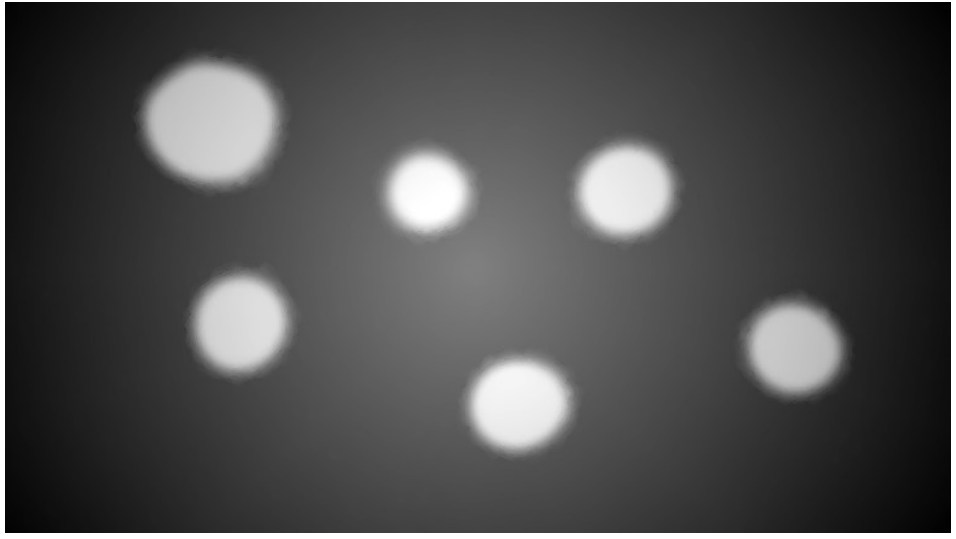


📝 **Opening by reconstruction:**

- Removes **gradient** and **big structures**.
- Steps:
  - **Erosion**: Removes structures of interest.

1. QuPath's built-in method

In QuPath:

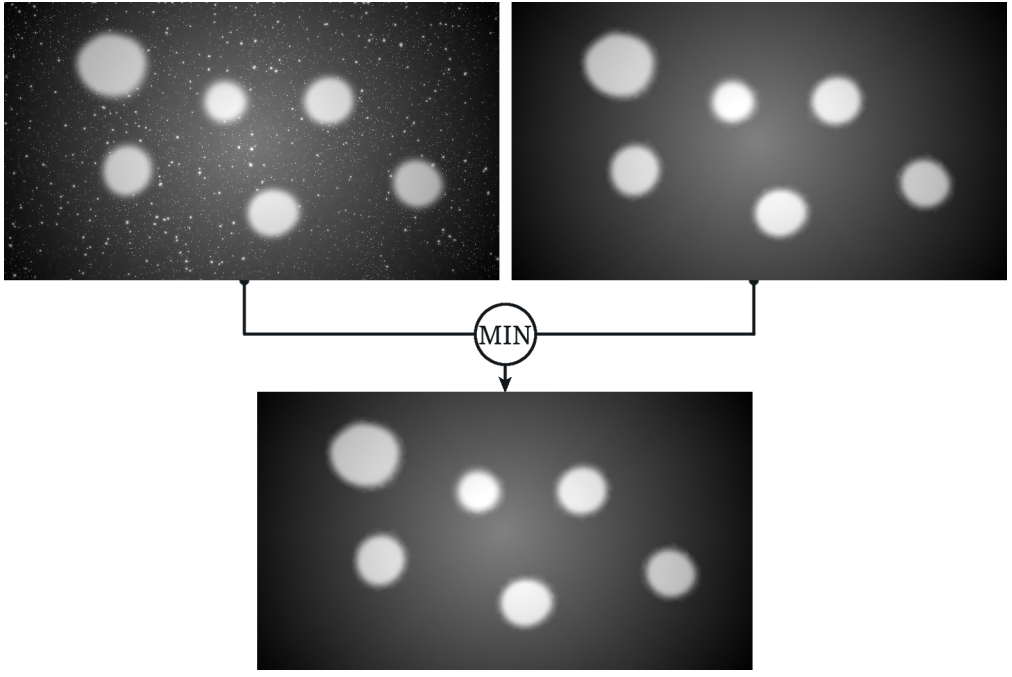


📝 **Opening by reconstruction:**

- Removes **gradient** and **big structures**.
- Steps:
  - **Erosion**: Removes structures of interest.
  - **Dilation**: Restores big blobs original size.

1. QuPath's built-in method

In QuPath:

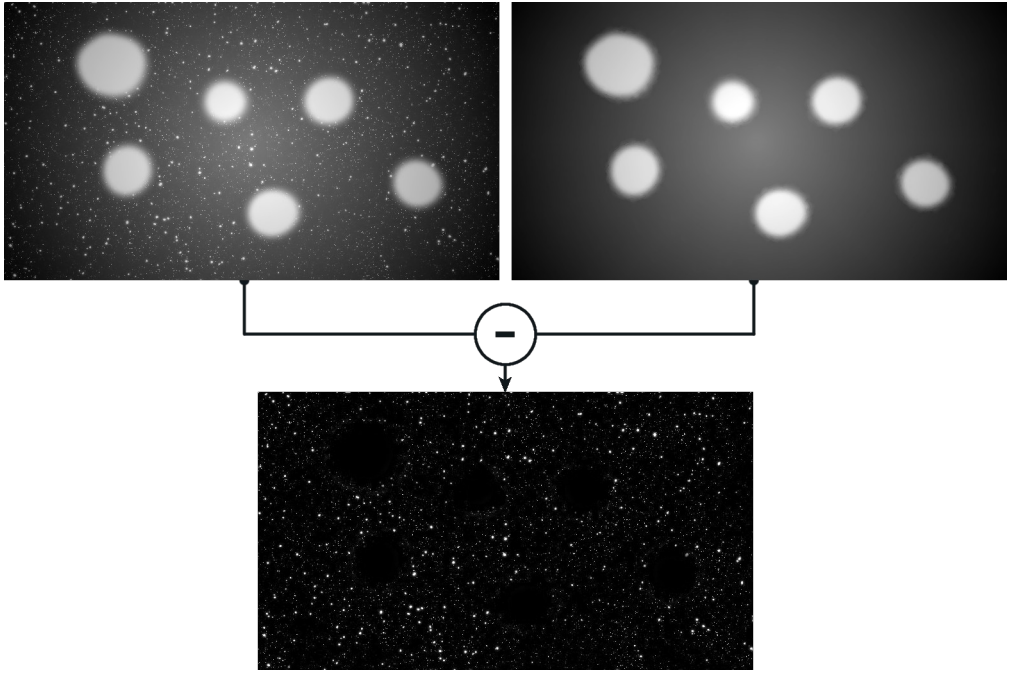


📝 **Opening by reconstruction:**

- Removes **gradient** and **big structures**.
- Steps:
  - **Erosion**: Removes structures of interest.
  - **Dilation**: Restores big blobs original size.
  - **min(ori, res)**: Avoid intensity bleeding.

1. QuPath's built-in method

In QuPath:

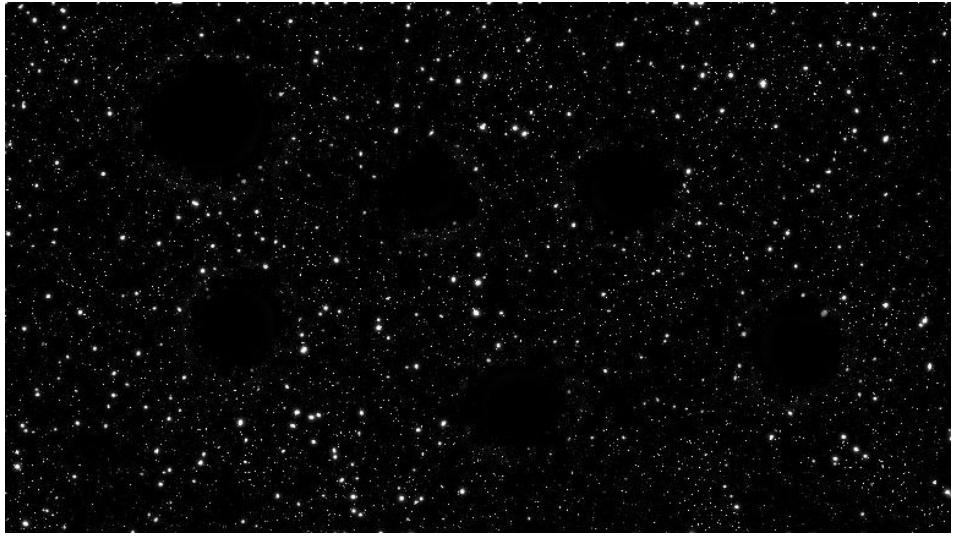


📝 **Opening by reconstruction:**

- Removes **gradient** and **big structures**.
- Steps:
  - **Erosion**: Removes structures of interest.
  - **Dilation**: Restores big blobs original size.
  - **min(ori, res)**: Avoid intensity bleeding.
  - **Result = ori – min**.

1. QuPath's built-in method

In QuPath:

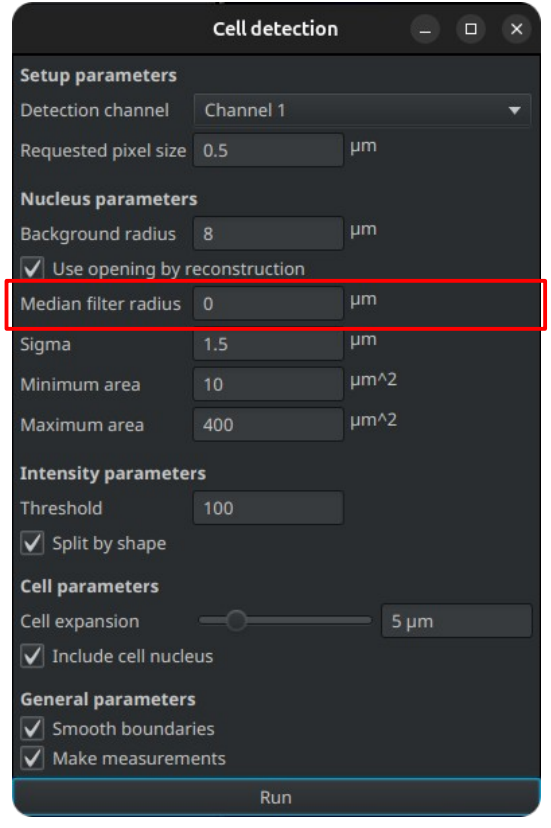


📝 **Opening by reconstruction:**

- Removes **gradient** and **big structures**.
- Steps:
  - **Erosion**: Removes structures of interest.
  - **Dilation**: Restores big blobs original size.
  - **min(ori, res)**: Avoid intensity bleeding.
  - **Result = ori – min**

1. QuPath's built-in method

In QuPath:

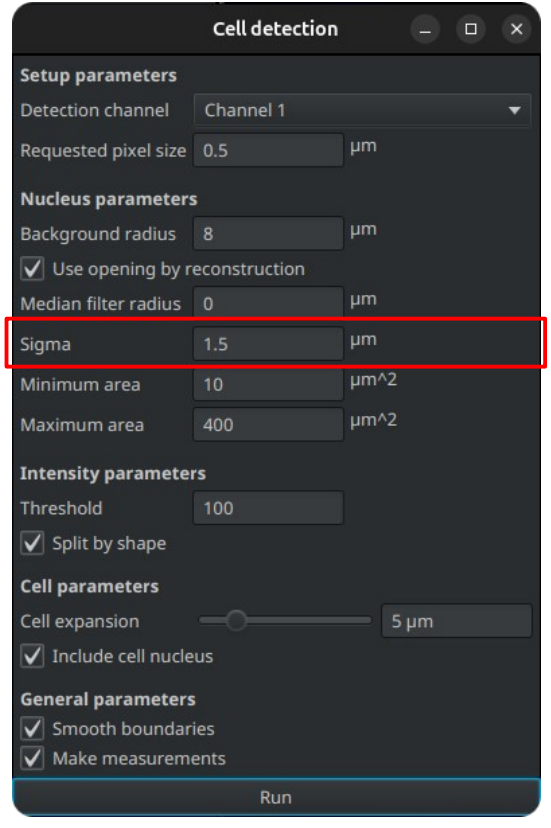


Median filter as processing:

- Useful to denoise images.
- 0 → not used.

1. QuPath's built-in method

In QuPath:

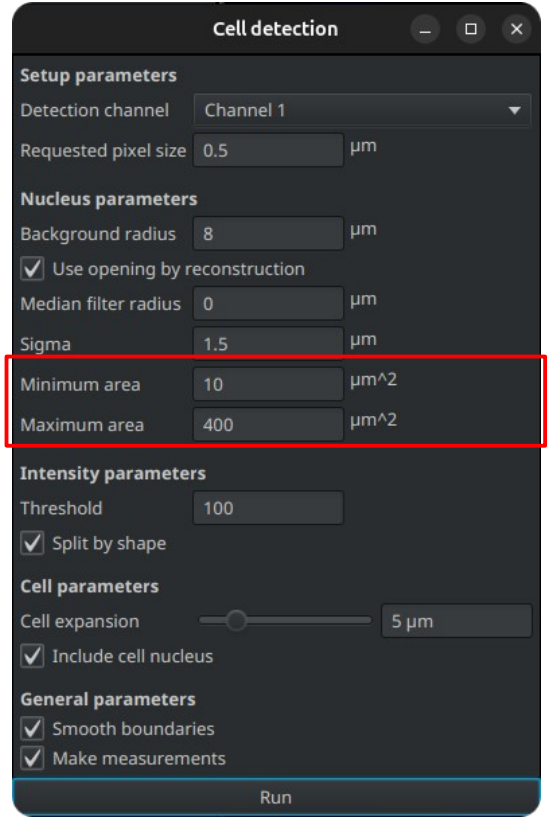


Sigma:

- Sigma used for a Gaussian filter.
- Used to blur inner structures in nuclei.

1. QuPath's built-in method

In QuPath:

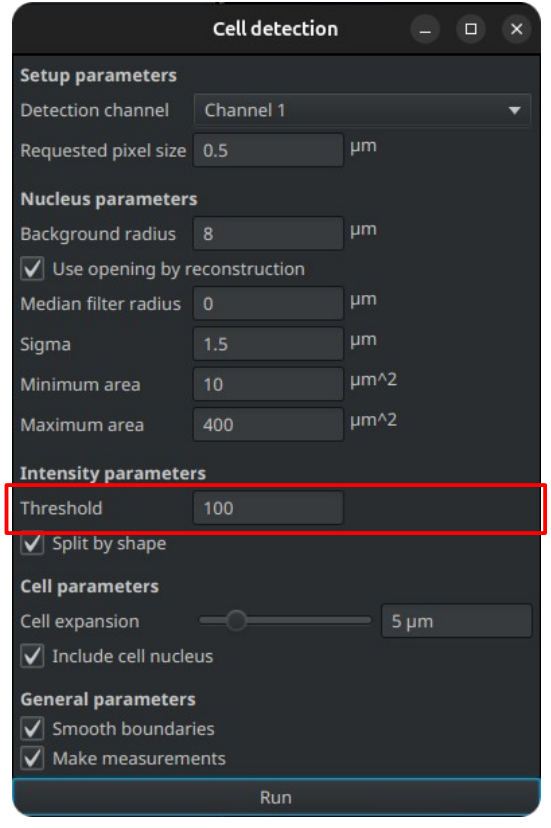


Area threshold:

- Objects smaller than min area: removed.
- Objects bigger than max area: removed.

1. QuPath's built-in method

In QuPath:

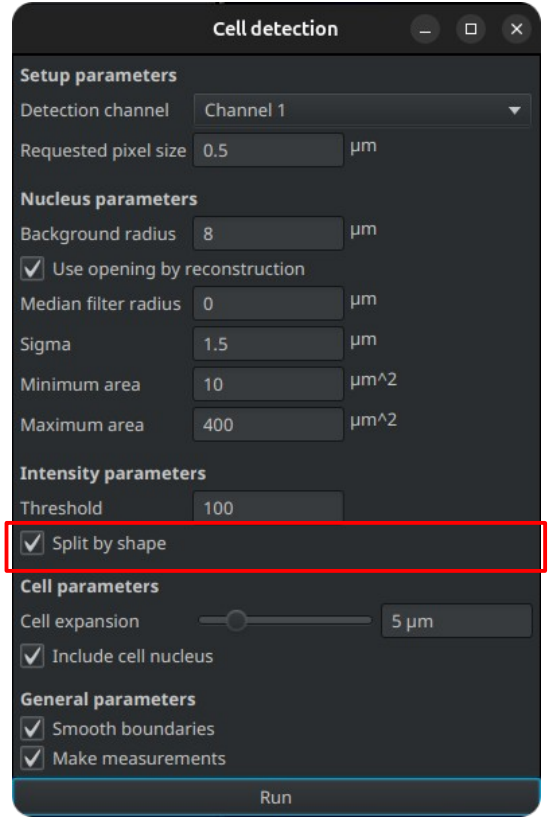


Threshold:

- Intensity threshold
- Separate nuclei from background

1. QuPath's built-in method

In QuPath:

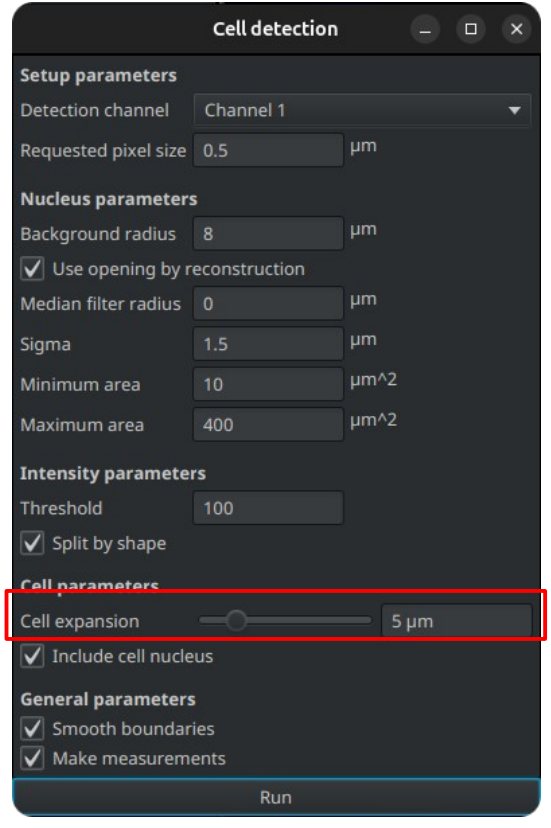


Split by shape:

- Use the watershed algorithm or not.
- If unchecked → touching nuclei not split.

1. QuPath's built-in method

In QuPath:

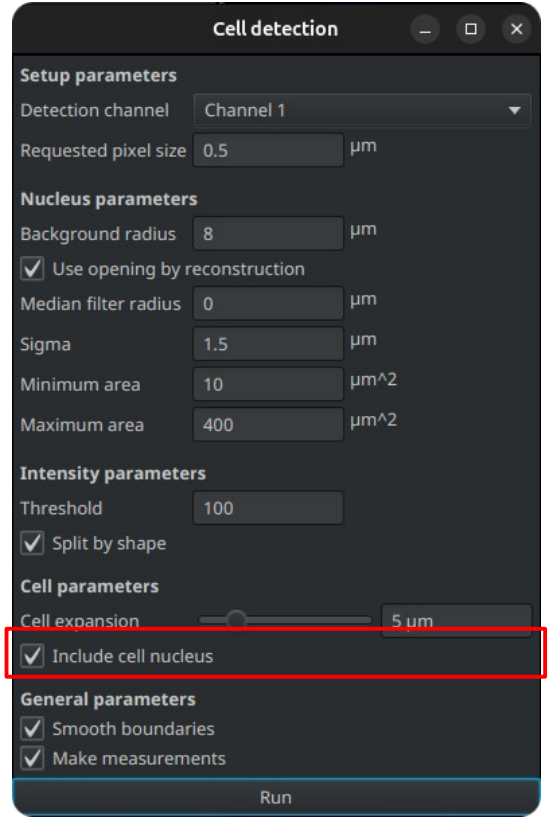


Cell expansion:

- Growing distance of nuclei polygons.
- Used to simulate cells without cyto staining.
- Collisions are handled automatically.
- 0µm == deactivated.

1. QuPath's built-in method

In QuPath:

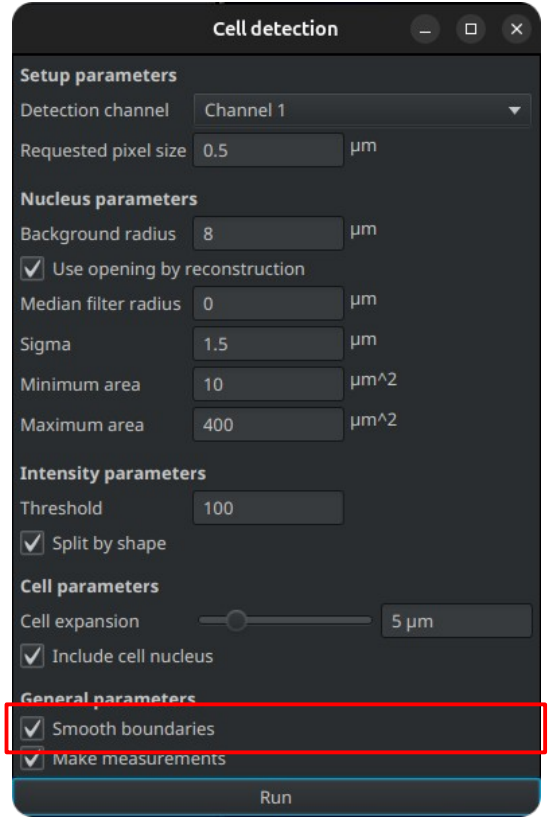


Include cell nucleus:

- Remove the nuclei after expansion?

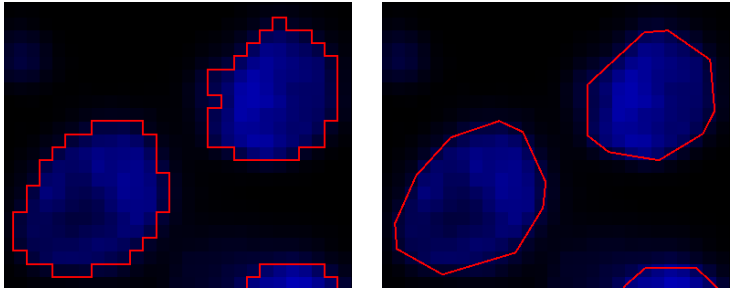
1. QuPath's built-in method

In QuPath:



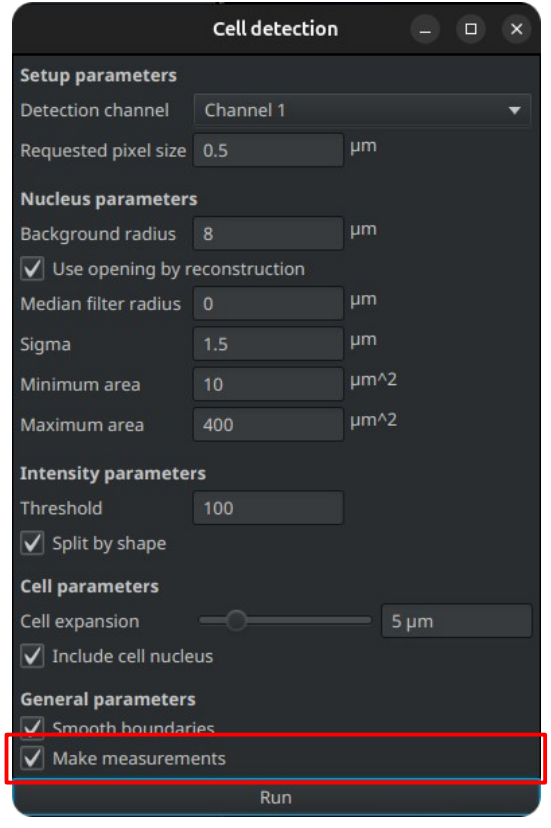
Smooth boundaries:

- Create polygons:
  - Using sub-pixel resolution?
  - Following pixels closely?



1. QuPath's built-in method

In QuPath:



Make measurements:

- Make measurements per object on the fly.
- **Shape** measurements:
  - Area
  - Perimeter
  - Sphericity
  - …
- **Intensity** measurements:
  - Mean intensity in each channel
  - StdDev in each channel
  - …

→ Exercise 5.1: QuPath built-in method

2. Use StarDist from QuPath

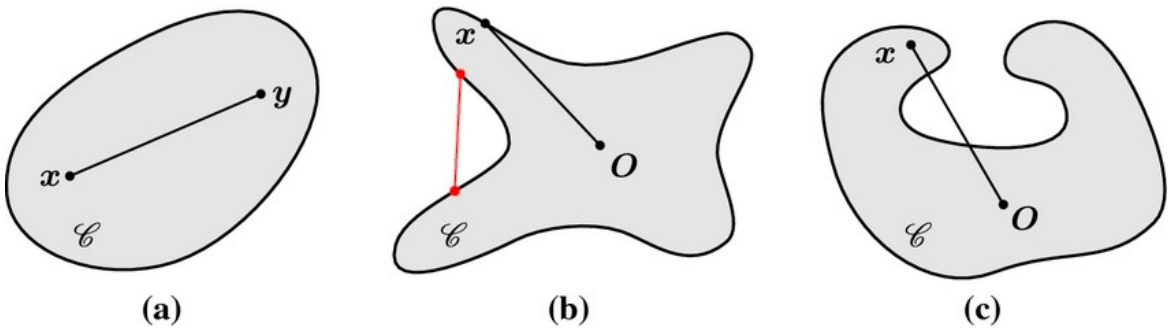
Principle:

- **Deep-learning** architecture.

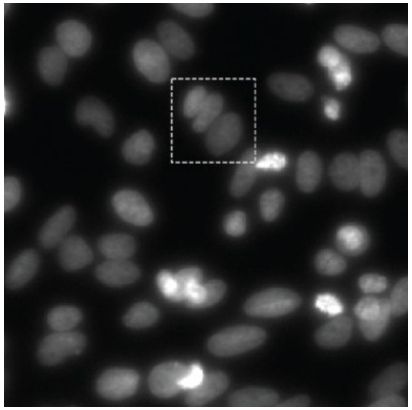
2. Use StarDist from QuPath

Principle:

- **Deep-learning** architecture.
- Tries to find **star-convex** polygons.

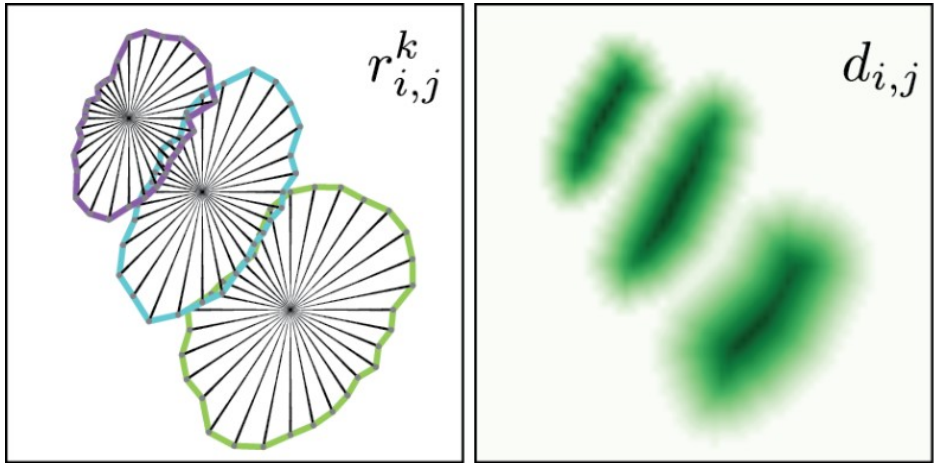


2. Use StarDist from QuPath

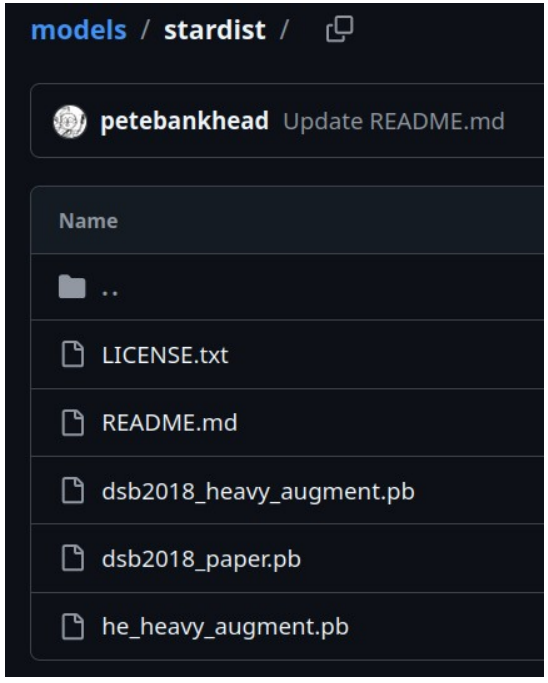


Principle:

- **Deep-learning** architecture.
- Tries to find **star-convex** polygons.
- Predicts:
  - **Distance map**
  - **Ray maps** representing polygons



2. Use StarDist from QuPath



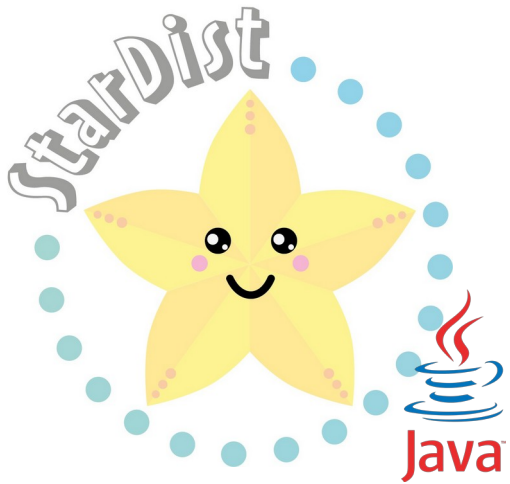
github.com/qupath/models/tree/main/stardist

Principle:

- **Deep-learning** architecture.
- Tries to find **star-convex** polygons.
- Predicts:
  - **Distance map**.
  - **Ray maps** representing polygons.
- **Pre-trained** models available:
  - "*dsb2018_heavy_augment.pb*": **fluo (DAPI, Hoechst, ...)**
  - "*he_heavy_augment.pb*": **IHC (HE, HDAB, HES, ...)**

2. Use StarDist from QuPath

In QuPath:



StarDist
Plugin for QuPath
(re-implemented in Java)
→ not a bridge

QuPath
Stand-alone software
(Java)

2. Use StarDist from QuPath

InQuPath:

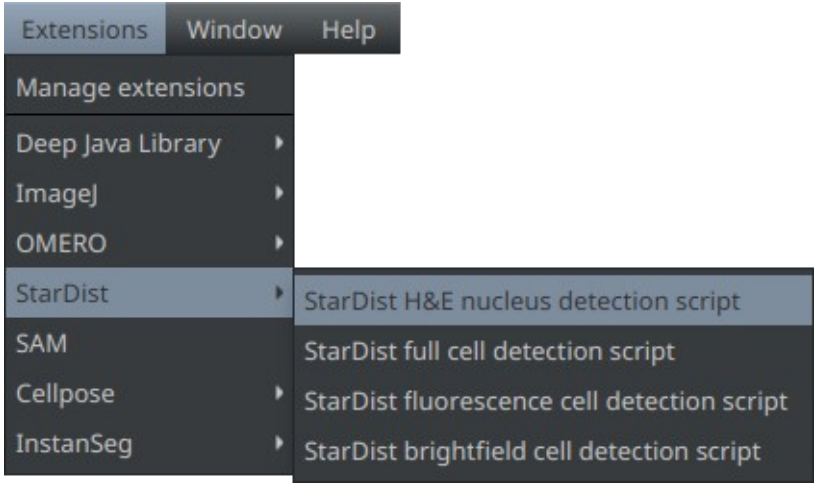
- Only exposed via **scripting**.

```
// Customize how the StarDist detection should b
// Here some reasonable default options are spec
def stardist = StarDist2D
    .builder(modelPath)
    .normalizePercentiles(1, 99) // Percentile n
    .threshold(0.5)              // Probability
    .pixelSize(0.4551)              // Resolutio
    .cellExpansion(5)
    .measureShape()              // Add shape me
    .measureIntensity()          // Add nucleus
    .build()

// Define which objects will be used as the 'par
// Use QP.getAnnotationObjects() if you want to
def pathObjects = QP.getSelectedObjects()

// Run detection for the selected objects
def imageData = QP.getCurrentImageData()
if (pathObjects.isEmpty()) {
    QP.getLogger().error("No parent objects are
    return
}
stardist.detectObjects(imageData, pathObjects)
stardist.close() // This can help clean up & reg
println('Done!')
```

2. Use StarDist from QuPath

InQuPath:

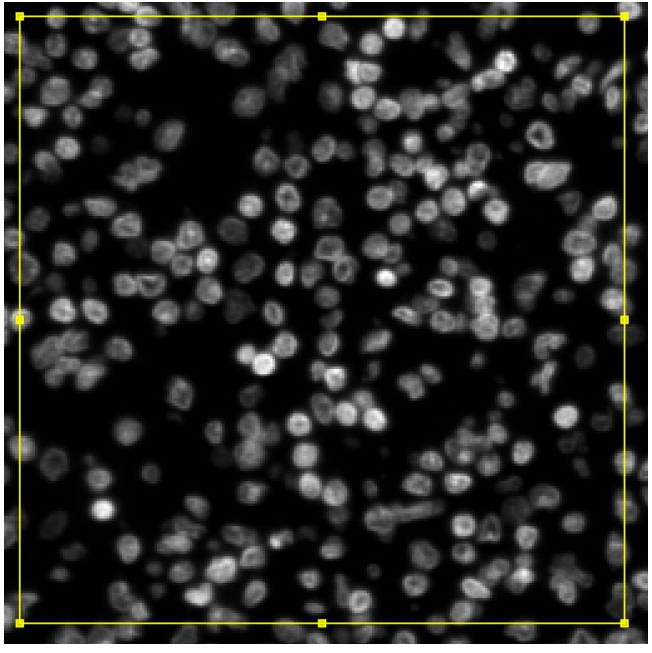
- Only exposed via **scripting**.
- Comes with **scripts templates**!
  1) For nuclei on IHC images (RGB)
  2) Example using all possible settings
  3) For nuclei on fluo image (DAPI, …)
  4) For nuclei on mono-channel BF (BF, DIC, ...)



2. Use StarDist from QuPath

InQuPath: 1. Create a parent annotation

- Needs **one or more** active annotations.



2. Use StarDist from QuPath

InQuPath: 2. Prepare the script

- The different blocks:

  1) Imports required to work.

```
import qupath.ext.stardist.StarDist2D
import qupath.lib.scripting.QP

// IMPORTANT! Replace this with the path to your StarDist model
// that takes a single channel as input (e.g. dsb2018_heavy_augment.pb)
// You can find some at https://github.com/qupath/models
// (Check credit & reuse info before downloading)
def modelPath = "/path/to/model.pb"

// Customize how the StarDist detection should be applied
// Here some reasonable default options are specified
def stardist = StarDist2D
    .builder(modelPath)
    .channels('DAPI')            // Extract channel called 'DAPI'
    .normalizePercentiles(1, 99) // Percentile normalization
    .threshold(0.5)              // Probability (detection) threshold
    .pixelSize(0.5)              // Resolution for detection
    .cellExpansion(5)            // Expand nuclei to approximate cell bound
    .measureShape()              // Add shape measurements
    .measureIntensity()          // Add cell measurements (in all compartme
    .build()

// Define which objects will be used as the 'parents' for detection
// Use QP.getAnnotationObjects() if you want to use all annotations, rather
def pathObjects = QP.getSelectedObjects()

// Run detection for the selected objects
def imageData = QP.getCurrentImageData()
if (pathObjects.isEmpty()) {
    QP.getLogger().error("No parent objects are selected!")
    return
}
stardist.detectObjects(imageData, pathObjects)
stardist.close() // This can help clean up & regain memory
println('Done!')
```

2. Use StarDist from QuPath

InQuPath: 2. Prepare the script

- The different blocks:

  1) Imports required to work.

  2) Model's location (path) on the disk.

```
import qupath.ext.stardist.StarDist2D
import qupath.lib.scripting.QP

// IMPORTANT! Replace this with the path to your StarDist model
// that takes a single channel as input (e.g. dsb2018_heavy_augment.pb)
// You can find some at https://github.com/qupath/models
// (Check credit & reuse info before downloading)
def modelPath = "/path/to/model.pb"

// Customize how the StarDist detection should be applied
// Here some reasonable default options are specified
def stardist = StarDist2D
    .builder(modelPath)
    .channels('DAPI')            // Extract channel called 'DAPI'
    .normalizePercentiles(1, 99) // Percentile normalization
    .threshold(0.5)              // Probability (detection) threshold
    .pixelSize(0.5)              // Resolution for detection
    .cellExpansion(5)            // Expand nuclei to approximate cell bound
    .measureShape()              // Add shape measurements
    .measureIntensity()          // Add cell measurements (in all compartme
    .build()

// Define which objects will be used as the 'parents' for detection
// Use QP.getAnnotationObjects() if you want to use all annotations, rather
def pathObjects = QP.getSelectedObjects()

// Run detection for the selected objects
def imageData = QP.getCurrentImageData()
if (pathObjects.isEmpty()) {
    QP.getLogger().error("No parent objects are selected!")
    return
}
stardist.detectObjects(imageData, pathObjects)
stardist.close() // This can help clean up & regain memory
println('Done!')
```

2. Use StarDist from QuPath

InQuPath: 2. Prepare the script

- The different blocks:

  1) Imports required to work.

  2) Model's location (path) on the disk.

  3) StarDist settings.

```
import qupath.ext.stardist.StarDist2D
import qupath.lib.scripting.QP

// IMPORTANT! Replace this with the path to your StarDist model
// that takes a single channel as input (e.g. dsb2018_heavy_augment.pb)
// You can find some at https://github.com/qupath/models
// (Check credit & reuse info before downloading)
def modelPath = "/path/to/model.pb"

// Customize how the StarDist detection should be applied
// Here some reasonable default options are specified
def stardist = StarDist2D
    .builder(modelPath)
    .channels('DAPI')            // Extract channel called 'DAPI'
    .normalizePercentiles(1, 99) // Percentile normalization
    .threshold(0.5)              // Probability (detection) threshold
    .pixelSize(0.5)              // Resolution for detection
    .cellExpansion(5)            // Expand nuclei to approximate cell bound
    .measureShape()              // Add shape measurements
    .measureIntensity()          // Add cell measurements (in all compartme
    .build()

// Define which objects will be used as the 'parents' for detection
// Use QP.getAnnotationObjects() if you want to use all annotations, rather
def pathObjects = QP.getSelectedObjects()

// Run detection for the selected objects
def imageData = QP.getCurrentImageData()
if (pathObjects.isEmpty()) {
    QP.getLogger().error("No parent objects are selected!")
    return
}
stardist.detectObjects(imageData, pathObjects)
stardist.close() // This can help clean up & regain memory
println('Done!')
```

2. Use StarDist from QuPath

InQuPath: 2. Prepare the script

- The different blocks:

  1) Imports required to work.

  2) Model's location (path) on the disk.

  3) StarDist settings.

  4) Working annotations gathering.

```
import qupath.ext.stardist.StarDist2D
import qupath.lib.scripting.QP

// IMPORTANT! Replace this with the path to your StarDist model
// that takes a single channel as input (e.g. dsb2018_heavy_augment.pb)
// You can find some at https://github.com/qupath/models
// (Check credit & reuse info before downloading)
def modelPath = "/path/to/model.pb"

// Customize how the StarDist detection should be applied
// Here some reasonable default options are specified
def stardist = StarDist2D
    .builder(modelPath)
    .channels('DAPI')            // Extract channel called 'DAPI'
    .normalizePercentiles(1, 99) // Percentile normalization
    .threshold(0.5)              // Probability (detection) threshold
    .pixelSize(0.5)              // Resolution for detection
    .cellExpansion(5)            // Expand nuclei to approximate cell bound
    .measureShape()              // Add shape measurements
    .measureIntensity()          // Add cell measurements (in all compartme
    .build()

// Define which objects will be used as the 'parents' for detection
// Use QP.getAnnotationObjects() if you want to use all annotations, rather
def pathObjects = QP.getSelectedObjects()

// Run detection for the selected objects
def imageData = QP.getCurrentImageData()
if (pathObjects.isEmpty()) {
    QP.getLogger().error("No parent objects are selected!")
    return
}
stardist.detectObjects(imageData, pathObjects)
stardist.close() // This can help clean up & regain memory
println('Done!')
```

2. Use StarDist from QuPath

InQuPath: 2. Prepare the script

- The different blocks:
  1) Imports required to work.
  2) Model's location (path) on the disk.
  3) StarDist settings.
  4) Working annotations gathering.
  5) Launch.

```
import qupath.ext.stardist.StarDist2D
import qupath.lib.scripting.QP

// IMPORTANT! Replace this with the path to your StarDist model
// that takes a single channel as input (e.g. dsb2018_heavy_augment.pb)
// You can find some at https://github.com/qupath/models
// (Check credit & reuse info before downloading)
def modelPath = "/path/to/model.pb"

// Customize how the StarDist detection should be applied
// Here some reasonable default options are specified
def stardist = StarDist2D
    .builder(modelPath)
    .channels('DAPI')            // Extract channel called 'DAPI'
    .normalizePercentiles(1, 99) // Percentile normalization
    .threshold(0.5)              // Probability (detection) threshold
    .pixelSize(0.5)              // Resolution for detection
    .cellExpansion(5)            // Expand nuclei to approximate cell bound
    .measureShape()              // Add shape measurements
    .measureIntensity()          // Add cell measurements (in all compartme
    .build()

// Define which objects will be used as the 'parents' for detection
// Use QP.getAnnotationObjects() if you want to use all annotations, rather
def pathObjects = QP.getSelectedObjects()

// Run detection for the selected objects
def imageData = QP.getCurrentImageData()
if (pathObjects.isEmpty()) {
    QP.getLogger().error("No parent objects are selected!")
    return
}
stardist.detectObjects(imageData, pathObjects)
stardist.close() // This can help clean up & regain memory
println('Done!')
```

2. Use StarDist from QuPath

InQuPath: 2. Prepare the script

- Imports required to work
  - Don’t modify them.

```
13
14 import qupath.ext.stardist.StarDist2D
15 import qupath.lib.scripting.QP
16
```

2. Use StarDist from QuPath

InQuPath: 2. Prepare the script


```
21 def modelPath = "/path/to/model.pb"
```

- Model's location (path) on the disk
  - **Absolute** path of the model.
  - On **Windows**, either:
    - Replace \ by / (ex: `"C:/Users/Me/…"`)
    - Double every \ (ex: `"C:\\Users\\Me\\…"`)

2. Use StarDist from QuPath

InQuPath: 2. Prepare the script

- StarDist settings
  - Each line is a setting:
    - **channels**: on which perform segmentation.
    - **normalizePercentile**: for global normalization.
    - **threshold**: To turn distance map in mask.
    - **pixelSize**: Requested pixel size.
    - **cellExpansion**: Growing distance to simulate cells.
    - **measureShape**: process per-cell area, perimeter, circularity, …
    - **measureIntensities**: process per-cell mean, stddev, … for each channel.

```
def stardist = StarDist2D
    .builder(modelPath)
    .channels('DAPI')
    .normalizePercentiles(1, 99)
    .threshold(0.5)
    .pixelSize(0.5)
    .cellExpansion(5)
    .measureShape()
    .measureIntensity()
    .build()
```

- All existing settings: qupath.readthedocs.io/en/stable/docs/deep/stardist.html.
- Activate/deactivate setting with: "//"

```
31 //      .cellExpansion(5)
```

2. Use StarDist from QuPath

InQuPath: 2. Prepare the script

- Working annotations gathering
  - Works either on:
    - All selected annotations.
    - All annotations.
  - Default: all selected annotations

```
// Use QP.getAnnotationObjects() if you want to use all annot
def pathObjects = QP.getSelectedObjects()

// Run detection for the selected objects
def imageData = QP.getCurrentImageData()
if (pathObjects.isEmpty()) {
    QP.getLogger().error("No parent objects are selected!")
    return
}
```

2. Use StarDist from QuPath

InQuPath: 3. Run the script

- Launch
  - Nothing to change.
  - Run for image or for project.
  - “Done!” in terminal when over.

```
stardist.detectObjects(imageData, pathObjects)
stardist.close() // This can help clean up & regain memory
println('Done!')
```

→ Exercise 5.2: Use StarDist in QuPath

3. Use CellPose from QuPath

Principle:

- **Deep-learning** architecture.

3. Use CellPose from QuPath

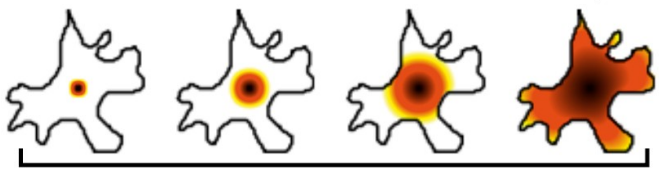
Principle:

- **Deep-learning** architecture.
- Based of heat propagation in metal.

3. Use CellPose from QuPath

Principle:

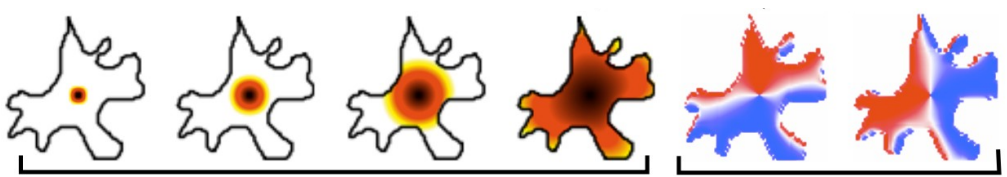
- **Deep-learning** architecture.
- Based of heat propagation in metal.
  - Diffuse heat from object's center



3. Use CellPose from QuPath

Principle:

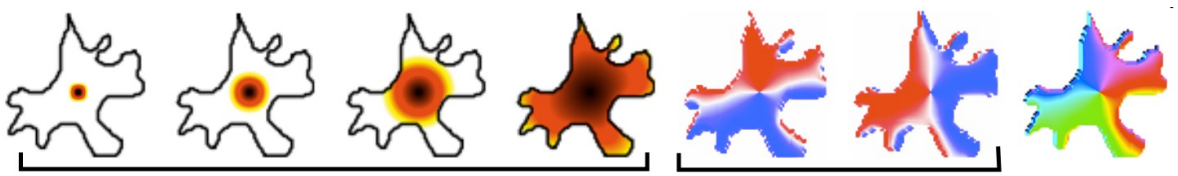
- **Deep-learning** architecture.
- Based of heat propagation in metal.
  - Diffuse heat from object's center
  - Extract X and Y gradients



3. Use CellPose from QuPath

Principle:

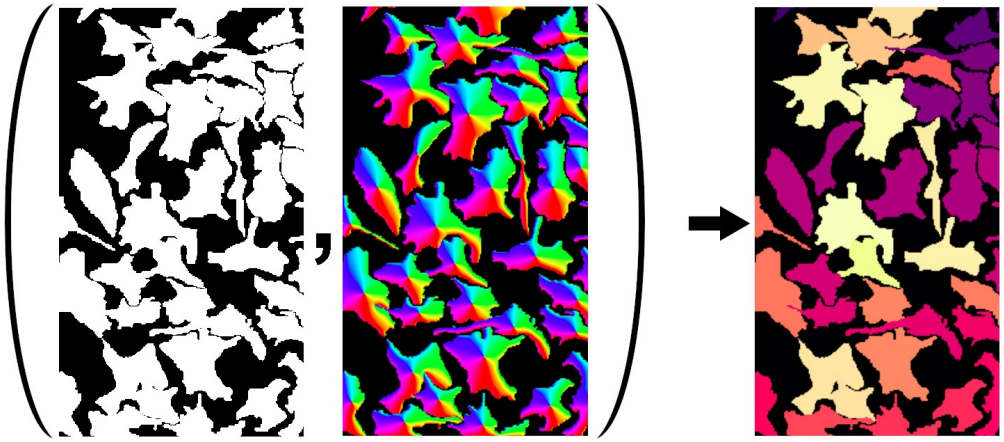
- **Deep-learning** architecture.
- Based of heat propagation in metal.
  - Diffuse heat from object's center
  - Extract X and Y gradients
  - Turn them in flow (2D vectors)



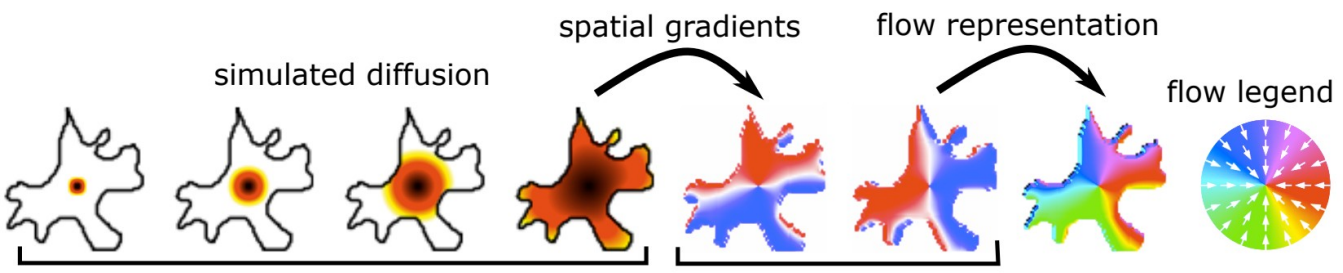
3. Use CellPose from QuPath

Principle:

- **Deep-learning** architecture.
- Based of heat propagation in metal.
  - Diffuse heat from object's center
  - Extract X and Y gradients
  - Turn them in flow (2D vectors)



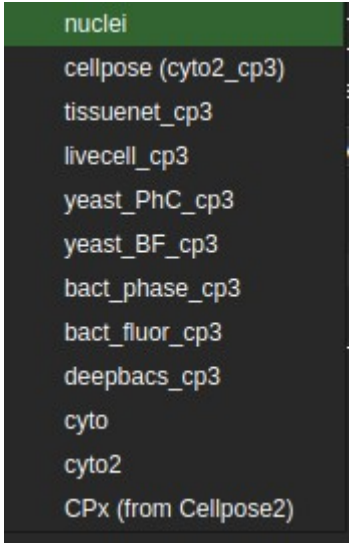
- Gradient descent → split touching objects.



3. Use CellPose from QuPath

Principle:

- **Deep-learning** architecture.
- Based of heat propagation in metal.
  - Diffuse heat from object's center
  - Extract X and Y gradients
  - Turn them in flow (2D vectors)
- Gradient descent → split touching objects.
- Comes with pre-trained models.
  - Most polyvalent: **cyto3**



3. Use CellPose from QuPath

Extra:

- For **fluo** (single-channel images) only.
- If task = segmenting cells: CellPose uses:
  - Cyto/membrane/actin/tubulin/… as **main** channel
  - Nuclei as **secondary** channel (optional)

  ⇨ Better result

3. Use CellPose from QuPath

In QuPath:

CellPose
Stand-alone / module
(Python)

BIOP-qupath-cellpose
Extension for communication
(Java)

QuPath
Stand-alone software
(Java)

3. Use CellPose from QuPath

InQuPath:

- Only exposed via **scripting**.

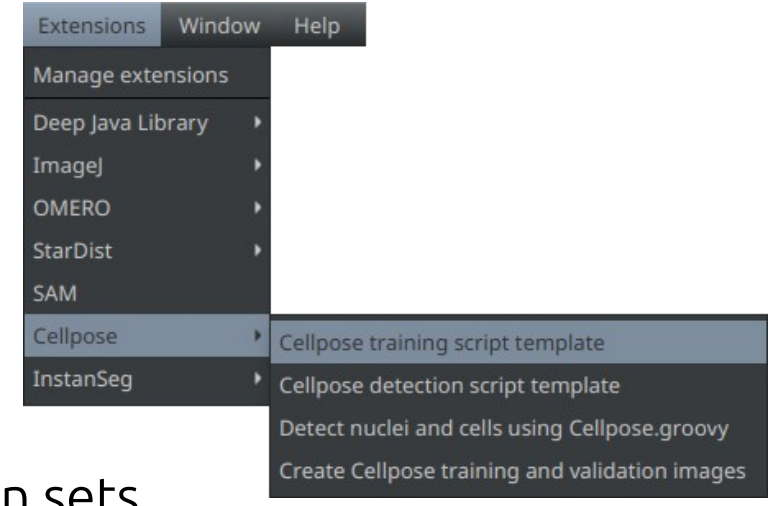
```
// Specify the model name (cyto, nuclei, cyto2, ... or a path to your custom
// Other models for Cellpose https://cellpose.readthedocs.io/en/latest/models
// And for Omnipose: https://omnipose.readthedocs.io/models.html
def pathModel = 'cyto3'
def cellpose = Cellpose2D.builder( pathModel )
        .pixelSize( 0.5 )                  // Resolution for detection in
        .channels( 'DAPI' )                // Select detection channel(s)
//         .tempDirectory( new File( '/tmp' ) )         // Temporary directory
//         .preprocess( ImageOps.Filters.median( 1 ) )  // List of preprocessi
//         .normalizePercentilesGlobal( 0.1, 99.8, 10 ) // Convenience global
//         .tileSize( 1024 )                 // If your GPU can take it, make
//         .cellposeChannels( 1,2 )          // Overwrites the logic of this
//         .cellprobThreshold( 0.0 )         // Threshold for the mask detect
//         .flowThreshold( 0.4 )             // Threshold for the flows, defa
//         .diameter( 15 )                   // Median object diameter. Set t
//         .useOmnipose()                    // Use omnipose instead
//         .useCellposeSAM()                 // Use cellposeSAM (i.e. cellpos
//         .addParameter( "cluster" )        // Any parameter from cellpose d
//         .addParameter( "save_flows" )     // Any parameter from cellpose d
//         .addParameter( "anisotropy", "3" ) // Any parameter from cellpose d
//         .cellExpansion( 5.0 )             // Approximate cells based upon
//         .cellConstrainScale( 1.5 )        // Constrain cell expansion usin
//         .classify( "My Detections" )      // PathClass to give newly creat
//         .measureShape()                   // Add shape measurements
//         .measureIntensity()               // Add cell measurements (in all
//         .createAnnotations()              // Make annotations instead of d
//         .simplify( 0 )                    // Simplification 1.6 by default
//         .useGPU(false)                    // Force using CPU. Default useG
        .build()

// Run detection for the selected objects
def imageData = getCurrentImageData()
def pathObjects = getSelectedObjects() // To process only selected annotation
// def pathObjects = getAnnotationObjects() // To process all annotations. Fo
if (pathObjects.isEmpty()) {
    Dialogs.showErrorMessage( "Cellpose", "Please select a parent object!" )
    return
}

cellpose.detectObjects( imageData, pathObjects )

// You could do some post-processing here, e.g. to remove objects that are to
// do this in a separate script so you can see the results before deleting an
```

3. Use CellPose from QuPath

InQuPath:

- Only exposed via **scripting**.
- Comes with **scripts templates**!

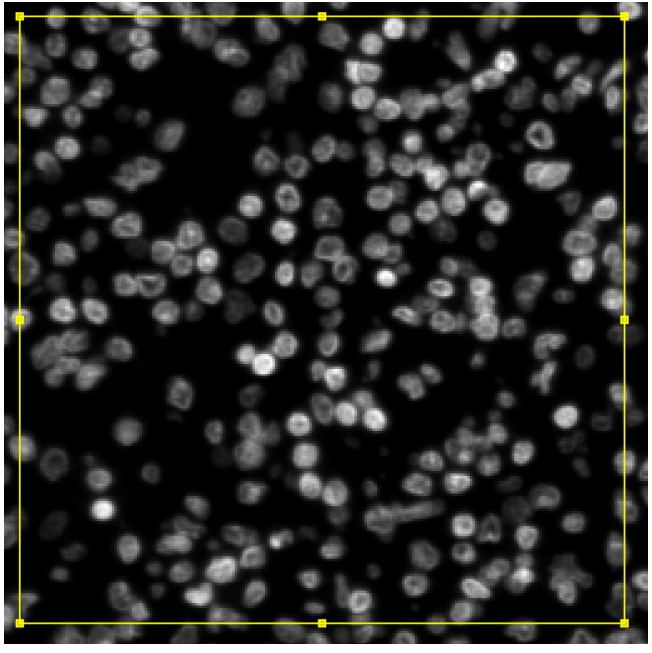
  1) Train a new model from manual annotations.
  2) Segment cells/nuclei from IHC/fluo images.
  3) Segment both cells and nuclei from fluo only.
  4) Splits project's images in training and validation sets.



3. Use CellPose from QuPath

InQuPath: 1. Create a parent annotation

- Needs **one or more** active annotations.



3. Use CellPose from QuPath

InQuPath: 2. Prepare the script

- The different blocks:

  1) Choose the pretrained model

```
// And for Omnipose: https://omnipose.readthedocs.io/models.html
def pathModel = 'cyto3'
def cellpose = Cellpose2D.builder( pathModel )
        .pixelSize( 0.5 )                       // Resolution for detection in u
        .channels( 'DAPI' )                     // Select detection channel(s)
//        .tempDirectory( new File( '/tmp' ) )         // Temporary directory t
//        .preprocess( ImageOps.Filters.median( 1 ) )  // List of preprocessing
//        .normalizePercentilesGlobal( 0.1, 99.8, 10 ) // Convenience global pe
//        .tileSize( 1024 )                   // If your GPU can take it, make l
//        .cellposeChannels( 1,2 )            // Overwrites the logic of this pl
//        .cellprobThreshold( 0.0 )           // Threshold for the mask detectio
//        .flowThreshold( 0.4 )               // Threshold for the flows, defaul
//        .diameter( 15 )                     // Median object diameter. Set to
//        .useOmnipose()                      // Use omnipose instead
//        .useCellposeSAM()                   // Use cellposeSAM (i.e. cellpose
//        .addParameter( "cluster" )          // Any parameter from cellpose or
//        .addParameter( "save_flows" )       // Any parameter from cellpose or
//        .addParameter( "anisotropy", "3" ) // Any parameter from cellpose or
//        .cellExpansion( 5.0 )               // Approximate cells based upon nu
//        .cellConstrainScale( 1.5 )          // Constrain cell expansion using
//        .classify( "My Detections" )        // PathClass to give newly created
//        .measureShape()                     // Add shape measurements
//        .measureIntensity()                 // Add cell measurements (in all c
//        .createAnnotations()                // Make annotations instead of det
//        .simplify( 0 )                      // Simplification 1.6 by default,
//        .useGPU(false)                      // Force using CPU. Default useGPU
        .build()

// Run detection for the selected objects
def imageData = getCurrentImageData()
def pathObjects = getSelectedObjects() // To process only selected annotations,
// def pathObjects = getAnnotationObjects() // To process all annotations. For
if (pathObjects.isEmpty()) {
    Dialogs.showErrorMessage( "Cellpose", "Please select a parent object!" )
    return
}

cellpose.detectObjects( imageData, pathObjects )

// You could do some post-processing here, e.g. to remove objects that are too
// do this in a separate script so you can see the results before deleting anyt

println 'Cellpose detection script done'

import qupath.ext.biop.cellpose.Cellpose2D
```

3. Use StarDist from QuPath

InQuPath: 2. Prepare the script

- The different blocks:

  1) Choose the pretrained model

  2) CellPose's settings

```
// And for Omnipose: https://omnipose.readthedocs.io/models.html
def pathModel = 'cyto3'
def cellpose = Cellpose2D.builder( pathModel )
        .pixelSize( 0.5 )                       // Resolution for detection in u
        .channels( 'DAPI' )                     // Select detection channel(s)
//        .tempDirectory( new File( '/tmp' ) )         // Temporary directory t
//        .preprocess( ImageOps.Filters.median( 1 ) )  // List of preprocessing
//        .normalizePercentilesGlobal( 0.1, 99.8, 10 ) // Convenience global pe
//        .tileSize( 1024 )                   // If your GPU can take it, make l
//        .cellposeChannels( 1,2 )            // Overwrites the logic of this pl
//        .cellprobThreshold( 0.0 )           // Threshold for the mask detectio
//        .flowThreshold( 0.4 )               // Threshold for the flows, defaul
//        .diameter( 15 )                     // Median object diameter. Set to
//        .useOmnipose()                      // Use omnipose instead
//        .useCellposeSAM()                   // Use cellposeSAM (i.e. cellpose
//        .addParameter( "cluster" )          // Any parameter from cellpose or
//        .addParameter( "save_flows" )       // Any parameter from cellpose or
//        .addParameter( "anisotropy", "3" ) // Any parameter from cellpose or
//        .cellExpansion( 5.0 )               // Approximate cells based upon nu
//        .cellConstrainScale( 1.5 )          // Constrain cell expansion using
//        .classify( "My Detections" )        // PathClass to give newly created
//        .measureShape()                     // Add shape measurements
//        .measureIntensity()                 // Add cell measurements (in all c
//        .createAnnotations()                // Make annotations instead of det
//        .simplify( 0 )                      // Simplification 1.6 by default,
//        .useGPU(false)                      // Force using CPU. Default useGPU
        .build()

// Run detection for the selected objects
def imageData = getCurrentImageData()
def pathObjects = getSelectedObjects() // To process only selected annotations,
// def pathObjects = getAnnotationObjects() // To process all annotations. For
if (pathObjects.isEmpty()) {
    Dialogs.showErrorMessage( "Cellpose", "Please select a parent object!" )
    return
}

cellpose.detectObjects( imageData, pathObjects )

// You could do some post-processing here, e.g. to remove objects that are too
// do this in a separate script so you can see the results before deleting anyt

println 'Cellpose detection script done'

import qupath.ext.biop.cellpose.Cellpose2D
```

3. Use StarDist from QuPath

InQuPath: 2. Prepare the script

- The different blocks:

  1) Choose the pretrained model

  2) CellPose's settings

  3) Gathering working annotations

```
// And for Omnipose: https://omnipose.readthedocs.io/models.html
def pathModel = 'cyto3'
def cellpose = Cellpose2D.builder( pathModel )
        .pixelSize( 0.5 )                       // Resolution for detection in u
        .channels( 'DAPI' )                     // Select detection channel(s)
//        .tempDirectory( new File( '/tmp' ) )         // Temporary directory t
//        .preprocess( ImageOps.Filters.median( 1 ) )  // List of preprocessing
//        .normalizePercentilesGlobal( 0.1, 99.8, 10 ) // Convenience global pe
//        .tileSize( 1024 )                   // If your GPU can take it, make l
//        .cellposeChannels( 1,2 )            // Overwrites the logic of this pl
//        .cellprobThreshold( 0.0 )           // Threshold for the mask detectio
//        .flowThreshold( 0.4 )               // Threshold for the flows, defaul
//        .diameter( 15 )                     // Median object diameter. Set to
//        .useOmnipose()                      // Use omnipose instead
//        .useCellposeSAM()                   // Use cellposeSAM (i.e. cellpose
//        .addParameter( "cluster" )          // Any parameter from cellpose or
//        .addParameter( "save_flows" )       // Any parameter from cellpose or
//        .addParameter( "anisotropy", "3" ) // Any parameter from cellpose or
//        .cellExpansion( 5.0 )               // Approximate cells based upon nu
//        .cellConstrainScale( 1.5 )          // Constrain cell expansion using
//        .classify( "My Detections" )        // PathClass to give newly created
//        .measureShape()                     // Add shape measurements
//        .measureIntensity()                 // Add cell measurements (in all c
//        .createAnnotations()                // Make annotations instead of det
//        .simplify( 0 )                      // Simplification 1.6 by default,
//        .useGPU(false)                      // Force using CPU. Default useGPU
        .build()

// Run detection for the selected objects
def imageData = getCurrentImageData()
def pathObjects = getSelectedObjects() // To process only selected annotations,
// def pathObjects = getAnnotationObjects() // To process all annotations. For
if (pathObjects.isEmpty()) {
    Dialogs.showErrorMessage( "Cellpose", "Please select a parent object!" )
    return
}

cellpose.detectObjects( imageData, pathObjects )

// You could do some post-processing here, e.g. to remove objects that are too
// do this in a separate script so you can see the results before deleting anyt

println 'Cellpose detection script done'

import qupath.ext.biop.cellpose.Cellpose2D
```

3. Use StarDist from QuPath

InQuPath: 2. Prepare the script

- The different blocks:
  1) Choose the pretrained model
  2) CellPose's settings
  3) Gathering working annotations
  4) Run segmentation

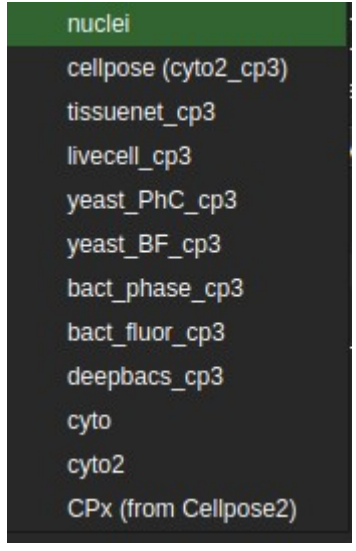
```
// And for Omnipose: https://omnipose.readthedocs.io/models.html
def pathModel = 'cyto3'
def cellpose = Cellpose2D.builder( pathModel )
        .pixelSize( 0.5 )                       // Resolution for detection in u
        .channels( 'DAPI' )                     // Select detection channel(s)
//        .tempDirectory( new File( '/tmp' ) )         // Temporary directory t
//        .preprocess( ImageOps.Filters.median( 1 ) )  // List of preprocessing
//        .normalizePercentilesGlobal( 0.1, 99.8, 10 ) // Convenience global pe
//        .tileSize( 1024 )                   // If your GPU can take it, make l
//        .cellposeChannels( 1,2 )            // Overwrites the logic of this pl
//        .cellprobThreshold( 0.0 )           // Threshold for the mask detectio
//        .flowThreshold( 0.4 )               // Threshold for the flows, defaul
//        .diameter( 15 )                     // Median object diameter. Set to
//        .useOmnipose()                      // Use omnipose instead
//        .useCellposeSAM()                   // Use cellposeSAM (i.e. cellpose
//        .addParameter( "cluster" )          // Any parameter from cellpose or
//        .addParameter( "save_flows" )       // Any parameter from cellpose or
//        .addParameter( "anisotropy", "3" ) // Any parameter from cellpose or
//        .cellExpansion( 5.0 )               // Approximate cells based upon nu
//        .cellConstrainScale( 1.5 )          // Constrain cell expansion using
//        .classify( "My Detections" )        // PathClass to give newly created
//        .measureShape()                     // Add shape measurements
//        .measureIntensity()                 // Add cell measurements (in all c
//        .createAnnotations()                // Make annotations instead of det
//        .simplify( 0 )                      // Simplification 1.6 by default,
//        .useGPU(false)                      // Force using CPU. Default useGPU
        .build()

// Run detection for the selected objects
def imageData = getCurrentImageData()
def pathObjects = getSelectedObjects() // To process only selected annotations,
// def pathObjects = getAnnotationObjects() // To process all annotations. For
if (pathObjects.isEmpty()) {
    Dialogs.showErrorMessage( "Cellpose", "Please select a parent object!" )
    return
}

cellpose.detectObjects( imageData, pathObjects )

// You could do some post-processing here, e.g. to remove objects that are too
// do this in a separate script so you can see the results before deleting anyt

println 'Cellpose detection script done'

import qupath.ext.biop.cellpose.Cellpose2D
```

3. Use StarDist from QuPath

InQuPath: 2. Prepare the script

- Choose a model:
  - **Pretrained** model: Give it's name (ex: "cyto3", "nuclei", …)
  - **Custom** model: Absolute path to the ".pt" file.

```
23 def pathModel = 'cyto3'
```



3. Use StarDist from QuPath

InQuPath: 2. Prepare the script

- CellPose's settings:
  - All settings are disable at start.
  - Most settings are similar to StarDist's.

```
def cellpose = Cellpose2D.builder( pathModel )
        .pixelSize( 0.5 )                          // Reso
        .channels( 'DAPI' )                        // Sele
//        .tempDirectory( new File( '/tmp' ) )
//        .preprocess( ImageOps.Filters.median( 1 ) )
//        .normalizePercentilesGlobal( 0.1, 99.8, 10 )
//        .tileSize( 1024 )                        // If you
//        .cellposeChannels( 1,2 )                 // Overwr
//        .cellprobThreshold( 0.0 )                // Thresh
//        .flowThreshold( 0.4 )                    // Thresh
//        .diameter( 15 )                          // Median
//        .useOmnipose()                           // Use om
//        .useCellposeSAM()                        // Use ce
//        .addParameter( "cluster" )               // Any pa
//        .addParameter( "save_flows" )            // Any pa
//        .addParameter( "anisotropy", "3" )       // Any pa
//        .cellExpansion( 5.0 )                    // Approx
//        .cellConstrainScale( 1.5 )               // Constr
//        .classify( "My Detections" )             // PathCl
//        .measureShape()                          // Add sh
//        .measureIntensity()                      // Add ce
//        .createAnnotations()                     // Make a
//        .simplify( 0 )                           // Simpli
//        .useGPU(false)                           // Force
        .build()
```

3. Use StarDist from QuPath

InQuPath: 2. Prepare the script

- CellPose's settings:
  - **`channels`**:
    - Nuclei staining only: `channels("nuclei")`
    - Cells staining only: `channels("cells")`
    - Both: `channels("cells", "nuclei")`
  - **`diameter`**: Mean radius of objects in pixels.
  - **`normalizePercentileGlobal`**: Histogram ends skipping.

```
def cellpose = Cellpose2D.builder( pathModel )
        .pixelSize( 0.5 )                          // Reso
        .channels( 'DAPI' )                        // Sele
//          .tempDirectory( new File( '/tmp' ) )
//          .preprocess( ImageOps.Filters.median( 1 ) )
//          .normalizePercentilesGlobal( 0.1, 99.8, 10 )
//          .tileSize( 1024 )                      // If you
//          .cellposeChannels( 1,2 )               // Overwr
//          .cellprobThreshold( 0.0 )              // Thresh
//          .flowThreshold( 0.4 )                  // Thresh
//          .diameter( 15 )                        // Median
//          .useOmnipose()                         // Use om
//          .useCellposeSAM()                      // Use ce
//          .addParameter( "cluster" )             // Any pa
//          .addParameter( "save_flows" )          // Any pa
//          .addParameter( "anisotropy", "3" )     // Any pa
//          .cellExpansion( 5.0 )                  // Approx
//          .cellConstrainScale( 1.5 )             // Constr
//          .classify( "My Detections" )           // PathCl
//          .measureShape()                        // Add sh
//          .measureIntensity()                    // Add ce
//          .createAnnotations()                   // Make a
//          .simplify( 0 )                         // Simpli
//          .useGPU(false)                         // Force
        .build()
```

biop.github.io/qupath-extension-cellpose/qupath/ext/biop/cellpose/CellposeBuilder.html

3. Use StarDist from QuPath

InQuPath: 2. Prepare the script

- Gathering working annotations:
  - Default: work on active annotations.
  - Otherwise: work on all annotations.

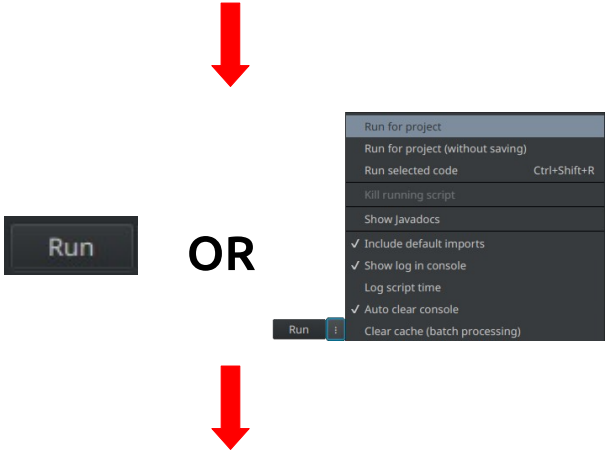
```
// Run detection for the selected objects
def imageData = getCurrentImageData()
def pathObjects = getSelectedObjects() // To process only selected annotati
// def pathObjects = getAnnotationObjects() // To process all annotations.
if (pathObjects.isEmpty()) {
    Dialogs.showErrorMessage( "Cellpose", "Please select a parent object!"
    return
}
```

3. Use StarDist from QuPath

InQuPath: 3. Run the script

- Launch segmentation:
  - Don’t edit it.
  - Wait for the “Cellpose detection script done” message

```
59 cellpose.detectObjects( imageData, pathObjects )
60
61 // You could do some post-processing here, e.g. t
62 // do this in a separate script so you can see th
63
64 println 'Cellpose detection script done'
65
66 import qupath.ext.biop.cellpose.Cellpose2D
```




```
INFO: Cellpose2D: 2026-03-14 13:22:35,668 [INFO] 100%|##########| 1/1 [00:01<00:00,  1.27s/it]
INFO: Cellpose2D: 2026-03-14 13:22:35,668 [INFO] >>>> completed in 1.432 sec
INFO: Reading Temp_71_37_z0_t0_cp_masks.tif
INFO: Cellpose detection script done
```

→ Exercise 5.4: Use CellPose in QuPath

4. Use InstanSeg from QuPath

Principle:

- **Deep-learning** architecture.

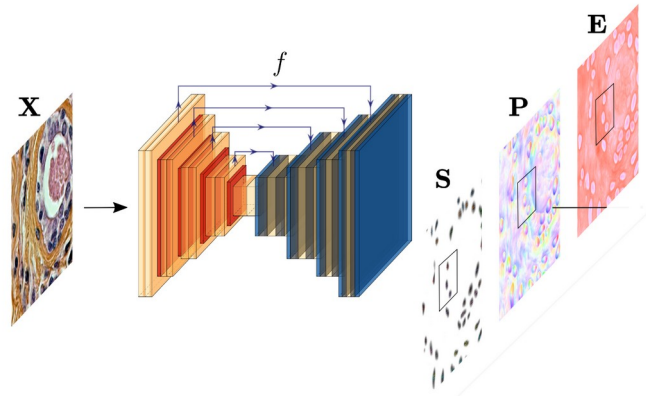
4. Use InstanSeg from QuPath

Principle:

- **Deep-learning** architecture.
- “Two-strokes” architecture.

1) Predicts three maps:

- Distance map
- Positional embeddings
- Conditional embeddings



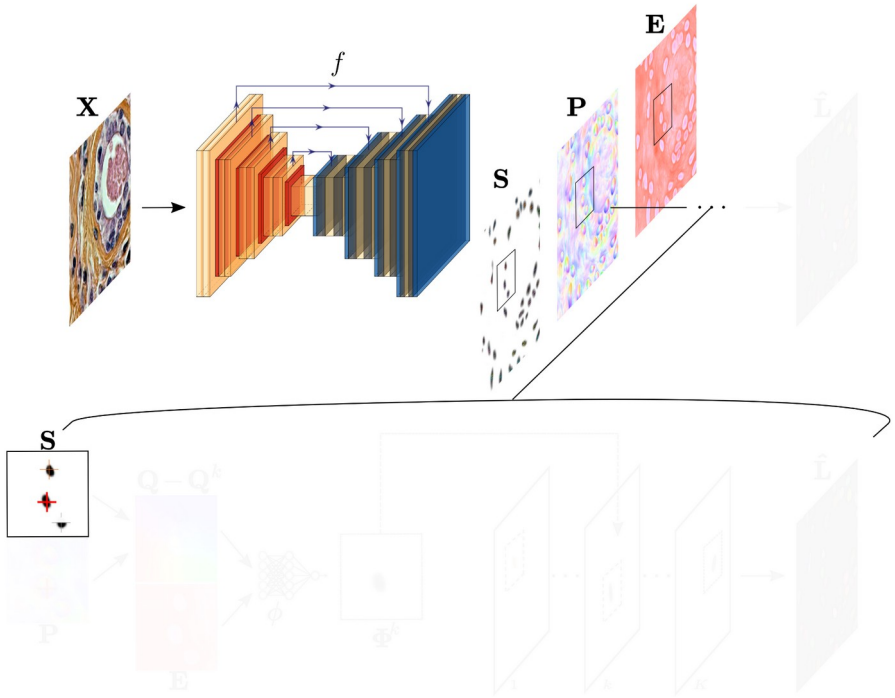
4. Use InstanSeg from QuPath

Principle:

- **Deep-learning** architecture.
- “Two-strokes” architecture.

1) Predicts three maps:
   - Distance map
   - Positional embeddings
   - Conditional embeddings

2) Seeds from distance map (local minima)



4. Use InstanSeg from QuPath

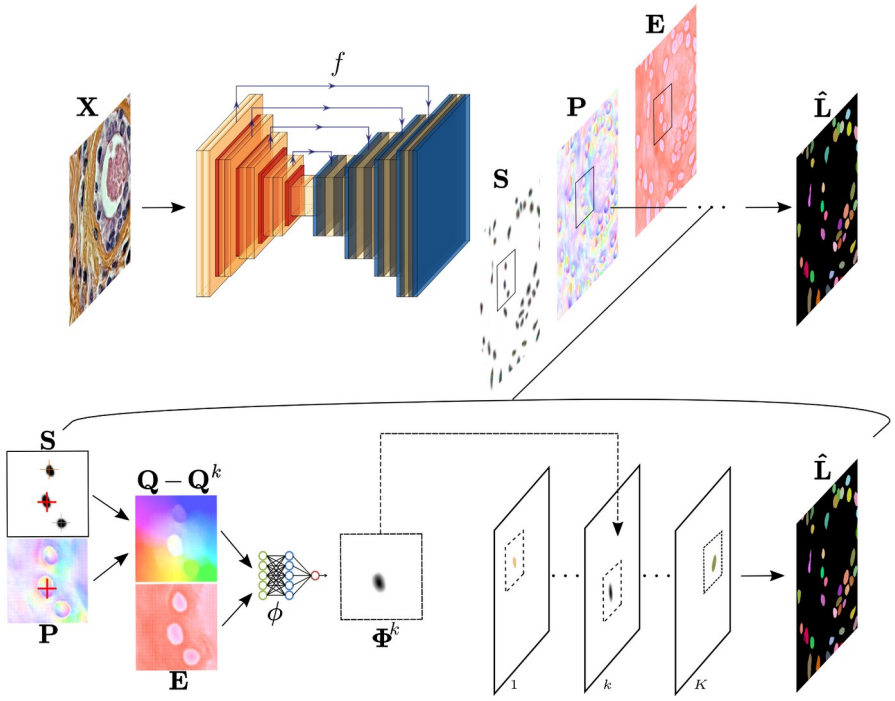
Principle:

- **Deep-learning** architecture.
- “Two-strokes” architecture.

1) Predicts three maps:
   - Distance map
   - Positional embeddings
   - Conditional embeddings

2) Seeds from distance map (local minima)

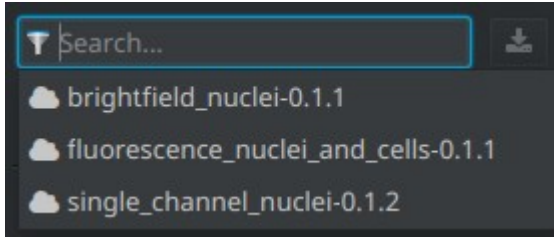
3) Relative offset + conditional embeddings → labels



4. Use InstanSeg from QuPath

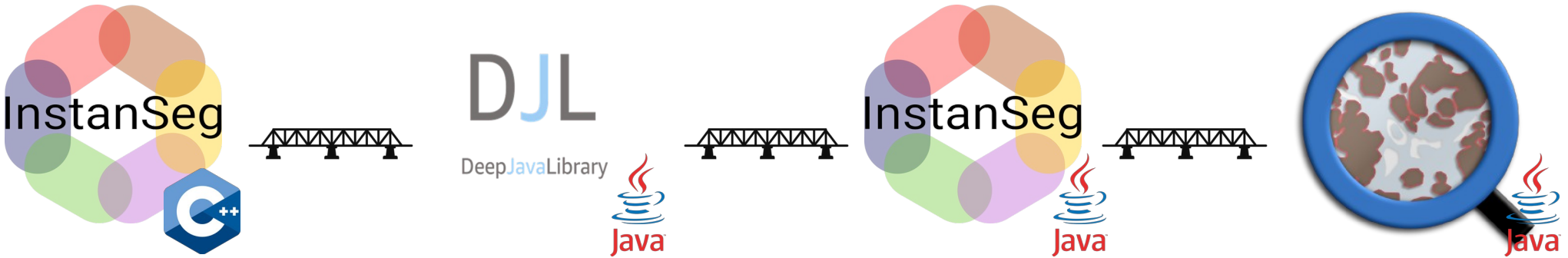
Principle:

- Pre-trained models available.
  - IHC (RGB) nuclei
  - Fluo stained nuclei and cells
  - Transmitted light (ex: BF from fluo images)



4. Use InstanSeg from QuPath

In QuPath:



InstanSeg
External library
(TorchScript → C++)

DeepJavaLibrary
Extension for DL inference
(Java)

InstanSeg Extension
Extension for interface
(Java)

QuPath
Stand-alone software
(Java)

4. Use InstanSeg from QuPath

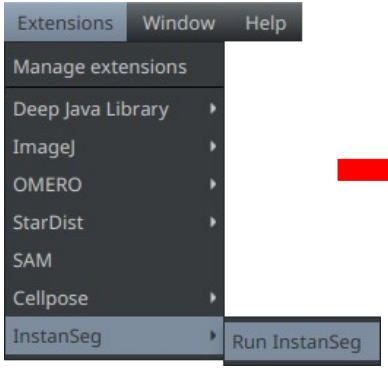
InQuPath:

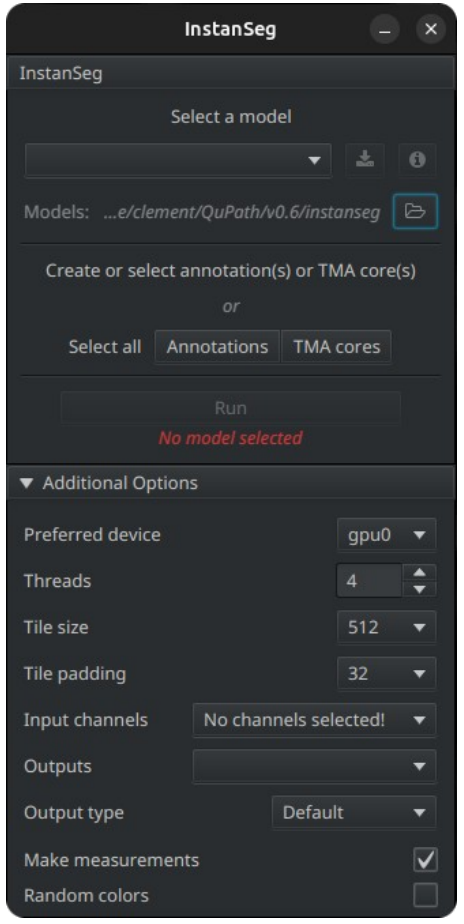
- Only exposed via **scripting**.

4. Use InstanSeg from QuPath

InQuPath:

- Only exposed via **scripting**.
- Comes with a **GUI**!!!

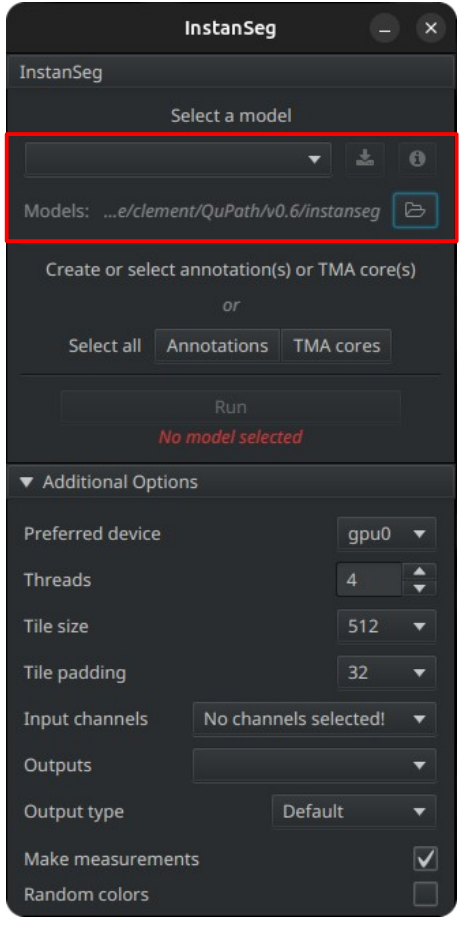




4. Use InstanSeg from QuPath

InQuPath:

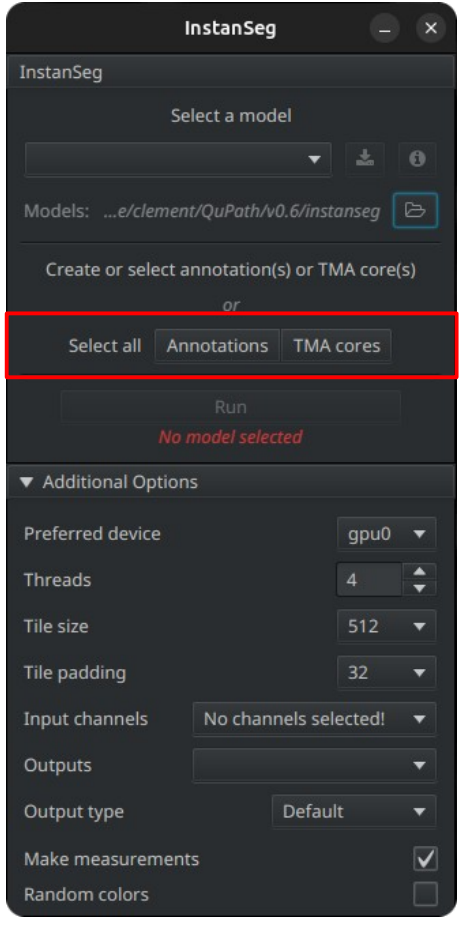
- Choose a pre-trained model.



4. Use InstanSeg from QuPath

InQuPath:

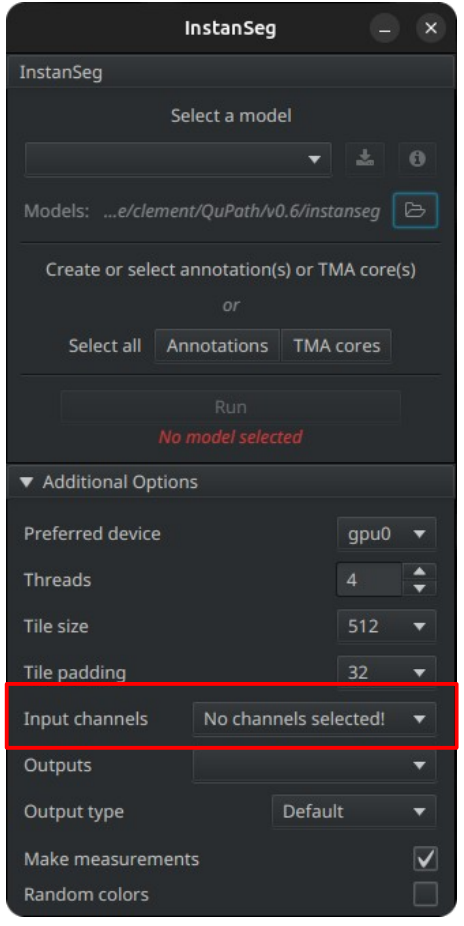
- Choose a pre-trained model.
- In which annotations launch inference?



4. Use InstanSeg from QuPath

InQuPath:

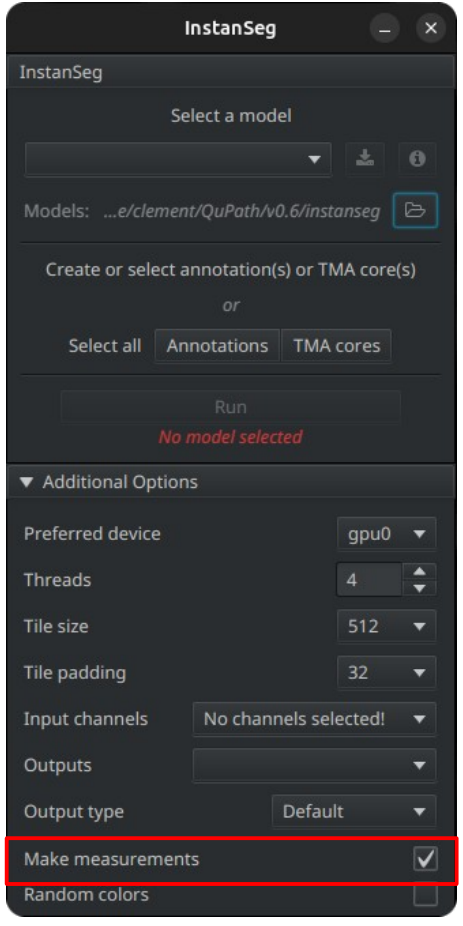
- Choose a pre-trained model.
- In which annotations launch inference?
- On which channel (fluo) run the inference?



4. Use InstanSeg from QuPath

InQuPath:

- Choose a pre-trained model.
- In which annotations launch inference?
- On which channel (fluo) run the inference?
- Make all measurements (shape + intensity).

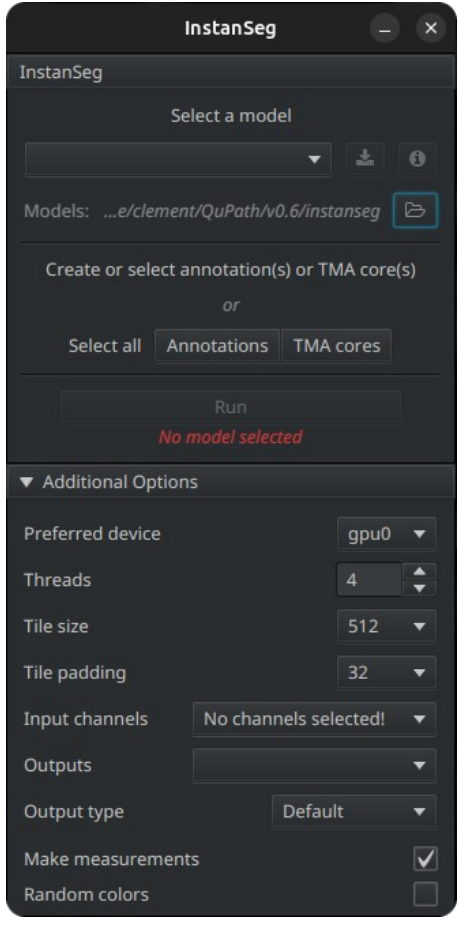


4. Use InstanSeg from QuPath

InQuPath:

- Choose a pre-trained model.
- In which annotations launch inference?
- On which channel (fluo) run the inference?
- Make all measurements (shape + intensity).

⚠️ No requested resolution: **pixel size** must be set.



→ Exercise 5.5: Use InstanSeg in QuPath