

QuPath Tutorial, Session I: Basics

1 Introduction

QuPath is **open source software for bioimage analysis**. It is often used for **digital pathology** applications because it offers a powerful set of tools for working with **whole slide images** - but it can be applied to lots of other kinds of (2D) image as well.

Features include:

- Powerful annotation & visualization tools
- Built-in algorithms for common tasks, including cell and tissue and anatomical regions detection
- Interactive machine learning, both for object and pixel classification
- Measurements for regions, cells and their relations

We will learn through this session the basic components of QuPath. These components can be assembled into analysis workflows to solve different image-based research analysis questions.

We will use an example of a whole slide Tonsil tissue stained with seven markers, imaged with *Phenolmager*, and further unmixed.

We will use this data to demonstrate how to segment anatomical regions, segment and classify cells, perform different measurements on individual cells and regions and combine region and cells to answer questions like how many cells are in a specific region and in its vicinity. We will explore different ways to visualize the raw and quantified data and how to export the different measurements for further data analysis.

We will discuss and learn how to apply good practices for training classifiers on multiple representative images, and learn how to apply a full workflow to a set of images in a batch.

If you are going to use your own computer make sure to follow the

Get Prepared section (at the end of the document) in order to install the software and download the tutorial data.

Contents

| | | |
|------|---|----|
| 1 | Introduction | 1 |
| 2 | Exercise 1: Create Project, Import and Inspect Images and Metadata, Control Image Display | 4 |
| 3 | Exercise 2: Define Tissue border | 6 |
| 3.1 | Draw tissue border manually using the annotation tools | 6 |
| 3.2 | Use Threshold-based classifier to distinguish tissue from background..... | 6 |
| 4 | Exercise 3: Segment Cells | 7 |
| 4.1 | QuPath Built-in “classical” cell detection..... | 7 |
| 4.2 | Cell detection using StarDist | 7 |
| 5 | Exercise 4: Inspect Measurements..... | 9 |
| 6 | Exercise 5: Classify Cells with single measurement classifier | 9 |
| 6.1 | Set Classes | 9 |
| 6.2 | Classify cells using Single Measurement Classifier | 9 |
| 6.3 | Apply classifiers to the whole field | 10 |
| 6.4 | Inspection of classification | 10 |
| 7 | Exercise 6: Cell classification with Machine learning based classifier | 11 |
| 7.1 | Annotate classes..... | 11 |
| 7.2 | Train cell classifier | 12 |
| 7.3 | Create composite classifier | 12 |
| 7.4 | Test your classifier..... | 13 |
| 8 | Exercise 7: Segment Regions with Pixel Classifier | 13 |
| 9 | Exercise 8: Combine regions and cells analysis | 14 |
| 10 | Exercise 9: Export Results | 15 |
| 10.1 | Export measurements | 15 |

| | | |
|------|--|----|
| 10.2 | Export Annotations and Detections (Advanced, Optional) | 16 |
| 11 | Exercise 10: Apply Analysis workflow to the whole project and inspect results | 16 |
| 12 | Exercise 11: Train classifiers using representative regions from multiple images | 19 |
| 12.1 | Create Training project | 19 |
| 12.2 | Create Training Image | 20 |
| 12.3 | Train a Machine-learning based (single channel) classifier | 22 |
| 13 | Get Prepared | 25 |
| 14 | Acknowledgement | 25 |

2 Exercise 1: Create Project, Import and Inspect Images and Metadata, Control Image Display

QuPath allows you to view and work with single images. You can open an Image in QuPath by either File > Open... or simply drag & drop the file into QuPath. However, if you will be saving and reloading data associated with multiple images it is *highly* recommended to use a **Project**.

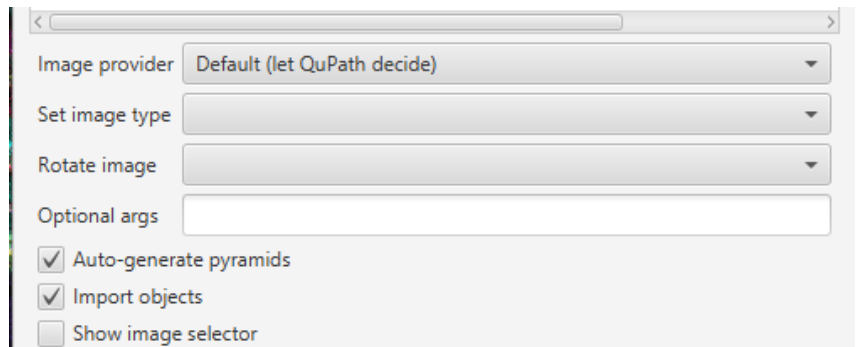
1. Create a Project by either:

- Use **File > Project ... > Create project**, this will prompt you to select an empty folder or
- Create an empty folder (eg under your QuPathTutorial folder) and drag and drop it into QuPath

2. Import images:

- Drag and drop the whole slide + 4 cropped images into QuPath (from *QuPathTutorialData\Tonsil* folder) or
- Click on **Add Images** and select the desired files

Use the following settings:



Projects provide a way to:

- Group together related images
- Easily switch between images (by clicking on thumbnails in the project image list)
- Organize your data files, along with scripts, classifiers and other useful things
- View a larger thumbnail and some file data by hovering the mouse over an image, without needing to open it fully
- Run scripts for a whole project

Note: QuPath does not duplicate the images, but only keeps metadata and extracted information as well as the parameters of the trained models (we'll see it later on).

Inspect the project folder structure to understand how the information is stored within the project

3. Explore the images and their metadata:

- Double click on the image to open it, set the Type to *Fluorescence*
- Switch to the **Image Tab** to inspect the metadata

Questions:

What is the size of the whole slide image?

What is the size of the cropped images?

What is the Pixel Size?

What is the Bit depth?

4. Control Image Display



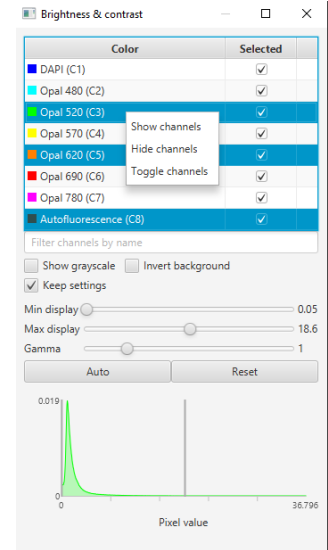
- Click the Brightness & Contrast icon tool
- Show/hide channels using the checkboxes,

Tips:

- You can use the keyboard to toggle on/off specific channel number (1-9)
- To toggle multiple channels at once: highlight the desired channels, right click and show/hide/toggle the channels. To select all channels click Ctrl+A
- Double click on the channel name to **change channel color and name**. Note that (Cn) is not part of the name.

Set DAPI Color to White

- Change Min/Max display values



5. Set up channel names

The channel names are particularly important for fluorescent data analysis, since these typically correspond to the markers of interest. They will also be reused within the names for the cell classifications.

Therefore we usually want them to be short accurate and stripped of any extra text we do not really need.

We can use a script to change all the channel names and colors:

- Open script Editor: *Automate > Show Script Editor* and type

```
setChannelNames('Nuclei', 'CD8', 'PDL1', 'Ki67', 'CD68', 'PanCK', 'CD20', 'AF')
```

- Click on *Run > Run* (Ctrl+R)
- **Apply to all images in the project** using *Run > Run for Project*, select all the images and click *OK*
- Switch to another image to verify that the changes were propagated
- Open the script ***Scripts\ChangeChannelNamesColors-Tonsil.groovy*** and *Run for project* to **change also channel colors and min/max display values**

3 Exercise 2: Define Tissue border

Open the Whole slide image

3.1 Draw tissue border manually using the annotation tools

- Try different tools: rectangle [R], polygon [P], brush [B], magic wand [W]. You can **modify selected annotation** (selected annotation will be yellow) **using brush and** pressing **Alt to decrease and Shift to increase** (Win). You can **avoid overlapping** existing annotation, by deselecting annotations , and drawing using brush while pressing **Ctrl+Shift**
- Annotation in QuPath is zoom dependent, explore the different behavior of the Wand and Brush tools when you zoom-out/in.
- Check the Tools menu for Keyboard shortcuts. They are very useful when annotating, especially getting back to Move mode (no annotation).
- You can **show filled annotations** with **Shift+F** (View > Fill annotations)

See this [Tweeetorial](#) for anything Annotation in QuPath

3.2 Use Threshold-based classifier to distinguish tissue from background

1. Classify > Pixel classification > Create Thresholder

- Use the 'C' (classification) icon to Show/hide the detected region, and the slider next to it to

control transparency.



- Change resolution and see how it effects the results and speed
- Change threshold and sigma
- What is the effect of different Channel settings?
- Create new Class 'WholeTissue' (...>Add/Remove...>Add class from the lower right side of the annotation Tab)
- Set above threshold to be of class 'WholeTissue'

2. Set Classifier name to 'WholeTissue' and Save the classifier

3. Create objects

- Use the default parameters to create annotations, but select "Delete existing objects" and "set new objects to selected". Select "Split objects". Parent object should be the Full image.
- Inspect the Annotation Tab
- Find good values for Minimum object size and Minimum hole size
- To delete annotations you can use the Annotation Tab (Ctrl+A) or the *Objects > Delete...* menu

What happens if you don't split objects?

4 Exercise 3: Segment Cells

Switch to the image C09, Right click on it, Duplicate and add “_ForTraining” to its name. Select the duplicated image and display only the Nuclei channel.

QuPath has a descent built in a cell detection module which is based on classical methods employing object enhancement, threshold based segmentation and watershed setting a threshold to segment nuclei after applying some filtering to enhance the nuclei. It works quite well in many cases, but in our experience recent deep-learning based segmentation usually works better.

4.1 QuPath Built-in “classical” cell detection

1. Create a small annotation encompassing multiple nuclei sizes, shapes and intensities
2. Select the annotation
3. Run Analyze > Cell Detection > Cell Detection with setting similar to these
4. Hit "Run"
5. You can still navigate around QuPath when the Cell detection window is open.
6. Tune Intensity threshold and click ‘Run’ again.

Tip:

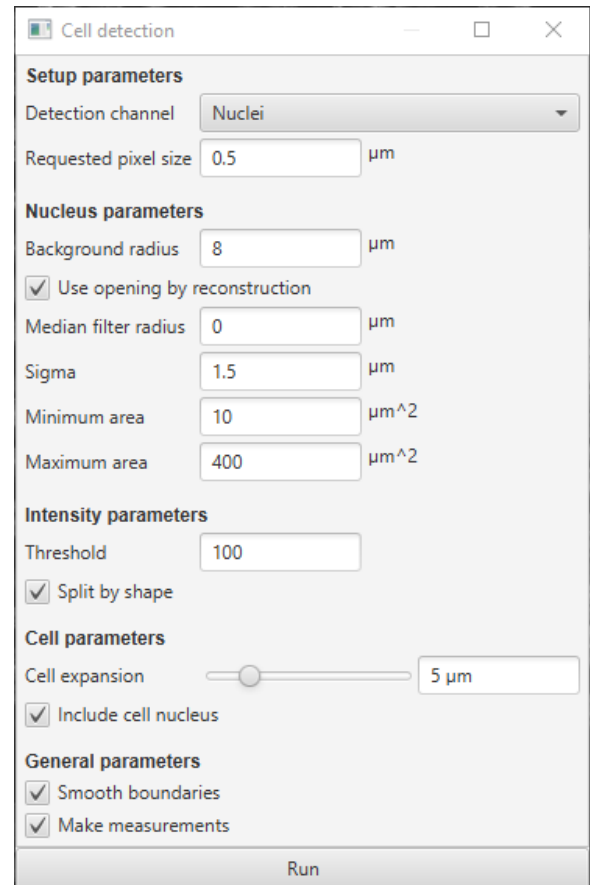
It is useful to **inspect pixel value** on the lower right side

7. Double click on a cell
 - a. Go to the Annotations tab
 - b. Browse through all the measurements on the bottom left panel.

4.2 Cell detection using StarDist

An important feature of QuPath Cell detection algorithm is the ability to (optionally) expand the detected objects (usually nuclei) by predefined value or until they touch a neighboring cell. This is geometrical approximate way to get the whole cell based on nuclei signal only.

This feature is also implemented in the DL based cell segmentation which are based on other external software and implemented as Extensions (this is QuPath term for plugins).



[StarDist](#) is a very good 2D & 3D nuclei segmentation by Martin Weigert and Uwe Schmidt, which is available as Python library and Fiji plugin. It was also implemented into QuPath as an **extension**.

StarDist comes with good 2D pre-trained model and there are good set of tools for retraining when needed.

1. Install StarDist Extension

Find the jar file StarDist in the *Extensions* folder, drag & drop it into QuPath window.

Choose to install in the default folder

Check the Extensions > Installed extensions to verify that StarDist extension (0.4.0) is included

2. Create a small annotation encompassing multiple nuclei sizes, shapes and intensities.

Select the annotation **or create Annotation of the whole image** using
Objects > Annotations > Create full image annotation (Ctrl+Shift+A)

3. Run StarDist

- StarDist cannot be run through QuPath GUI but only using a Script. Drag & drop the script ***RunStarDistOpenCV.groovy*** into QuPath
- Make sure that the values of *NucChannel* and *PixelSize* fit your image settings
- Click *Run*

4. Inspect Nuclei Segmentation

- Browse through different regions and inspect how good the annotation is.
- Use 'D' to show/hide nuclei segmentation

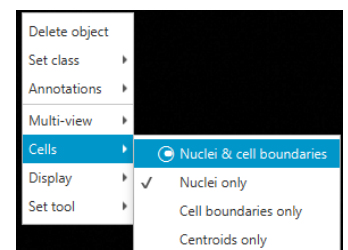
Do you find touching nuclei that are not well separated?

Do you find nucleus that is mistakenly broken into multiple nuclei?

Do you find nuclei that are not miss-detected?

Do you find false-detections?

- Use Right-click > Cells menu to change the appearance of the detected cells
- Turn on different channels , and inspect if the *CellExpansion* cover the cytoplasmic signal (in sparse regions of the image)



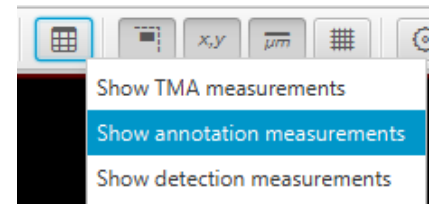
5 Exercise 4: Inspect Measurements

1. Individual cell measurement

- Double click on a cell
- Inspect the different cell measurements in the lower left panel of the Annotation Tab

2. Annotations Table

- Click on the Table icon and open the Annotations table
- **How many cells were detected in your selected annotation?**



3. Detection Table

- Click on the Table icon and open the Detections table
- Sort it by Cell Area, find the largest Cell, is it a well segmented cell?
You may find it useful to show detections as Filled shapes
- Sort by other values
- Show Histograms of intensity values

4. Measurement Map

- *Measure > Show measurement maps* to show the spatial distribution of “Nucleus: Area” and “Nucleus: Circularity”
- Show detections as Filled shapes and set the
- Filter by “Cell: Mean”, show map for different channels



6 Exercise 5: Classify Cells with single measurement classifier

6.1 Set Classes

QuPath come with some pre-defined classes, not all of them relevant for your project, and you may need new ones as well. You can Add/Remove classes with ..>Add/Remove

1. **Populate from Image channels** (don't keep the existing classes)
2. Add class for “WholeTissue”

6.2 Classify cells using Single Measurement Classifier

1. *Classify > Object classification > Create Single measurement classifier*
2. Select CD8 Channel , select the Channel in the GUI
3. QuPath suggest Measurement and Threshold
4. Use *Live preview*
5. Do you need the Nuclei intensity? set cell display to show nuclei and cells, try other options
Measurements and threshold
6. You can double-click on a cell to check its values
7. Save the classifier

8. Set a classifier for Ki67 channel, display the channel, select the channel in the single
9. Select Measurement, set threshold and save the classifier
10. Inspect the Project folder to see where the classifiers are saved

6.3 Apply classifiers to the whole field

1. Close the single measurements classifier GUI
2. If you are still working on small annotation:
 - a. Delete existing objects with *Objects > Delete... > Delete all objects*
 - b. **Create Annotation of the whole image** using *Objects > Annotations > Create full image annotation* (Ctrl+Shift+A)
 - c. **Run StarDist**
3. *Classify > Object classification > Load object classifier*
4. Select both classifiers with Ctrl+left-click, *Apply classifier sequentially*
5. **Note:** You can reset existing classification using *Classify > object classification > reset detection classifications*

6.4 Inspection of classification

1. Turn on the two relevant channels
2. It may be useful in this case to change the color the Ki67 class, so it will not be yellow, as all selected cells are automatically displayed as yellow
3. You can use **View > Show channel viewer** or *View > Show miniviewer* to see the individual channel expression of a small region around the cursor side-by-side
4. You can **hide the double negative cells** by going to the Class Tab, right-click on the *None* class, Show/Hide...> Hide classes in viewer

Are there positive cells that are not identified?

You can double click on such cell and check why it is not identified.

5. **Look for Double positive cells:** Open the Detection table, sort by Class or Name, look for cells that are double positive (Class = CD8:Ki67). When you show the cells Filled you will see them in different color.

Are those cells really double positive?


6. Open the Annotation table: **how many cells there are classified for each class?**
7. You can add the new double-positive class to the list of classes with ...>Populate from existing objects > All classes (including sub-classes).

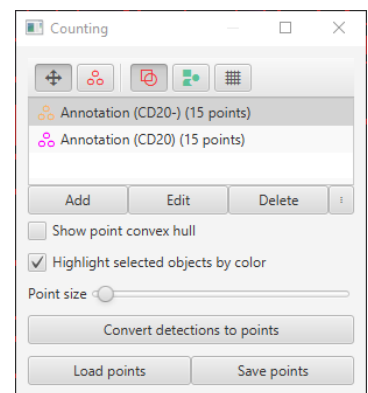
This will enable you to change the color of this class, hide/show these cells or select all them with *Objects > Select... > Select objects by classification*

7 Exercise 6: Cell classification with Machine learning based classifier

You can also classify a cell using multiple measurements by training a machine-learning object classifier. In this case you are annotating few cells that are belonging to that class and few cells that are not belonging to that class, and QuPath automatically extract the relevant combination of measurements to determine whether this cells belong to that class or not.

7.1 Annotate classes

1. Select a channel to work on: eg CD20 and show it
2. Reset classification, show all classes in the viewer
3. Add "CD20-" Class
4. Use the point tool 
5. Use the Add button to add classes for *CD20 (TypeX-Positive)* and *CD20- (TypeX-Negative)*. Use *Set Class* button to set the class types
6. Click on *Auto set*
7. Highlight the class you want to select, and add points for several obvious cases
Do this for the other class as well



Note that when working with points the cursor changes, and each time you click the left-mouse key selects a point. Undo options are limited. Thus it is very useful to switch between Points (.) and navigation (m) modes using keyboard shortcuts.

Useful Keyboard shortcuts

Use keyboard shortcuts: 'm' – Navigation , 'b' – Brush '.' For point annotations

'a' – show/hide annotations, 'd' – show/hide detections, 'f' – toggle filled cells

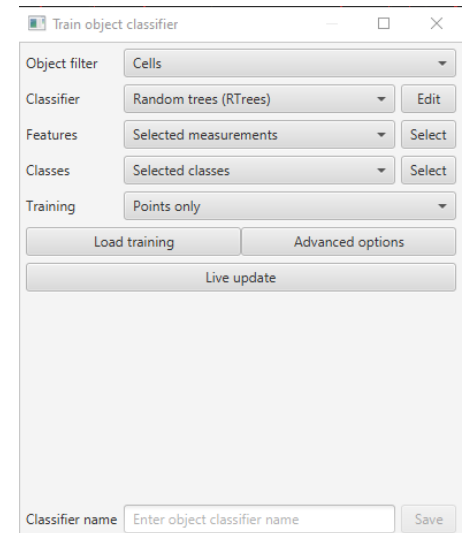
For fluorescent images the 1-9 keyboard buttons toggle the related channel

7.2 Train cell classifier

1. After you selected multiple examples **start an object classifier** with *Classify > Object classification > Train object classifier* (Ctrl+Shift+D)

2. Set Classifier parameters

- Object filter = Cells,
- Classifier = Random Trees,
- Features = selected measurements and select
 - o All shape features,
 - o All Intensity features of the relevant channels for the classifier.You can highlight multiple features and select them at once by right-click > select
- Classes = selected classes, select CD20 (TypeX-Positive) / CD20- (TypeX-Negative)
- Training = points only



3. Click on Live update
4. Inspect results and make Corrections – add points for falsely classified cells,

5. Make sure to save intermediate annotations + classifiers

- a. **Save points** from counting window – all points , name it CD20_points_vN
- b. **save classifier** – from Train object classifier window CD20_ML_vN
 - o **Use meaningful classifiers names.** consider using Date within the classifier name
 - o Keep vN matched between annotations and classifier, as you can load then later on, and undo options are limited for points.
 - o Increase the version vN each time you save

7.3 Create composite classifier

Classify > Object classification > Create Composite classifier

Select all the 3 classifiers (CD8, Ki67, CD20_ML)

Name the composite classifier accordingly: Combined with the versions of the original classifiers eg:
CD8_Ki67_CD20_ML_v1

7.4 Test your classifier

1. Reset classification
2. **Classify > Object classification > Load object Classifier**, this will show you all the classifier available in your project, highlight your preferred one and click *Apply classifier*

8 Exercise 7: Segment Regions with Pixel Classifier

Switch to the image C06, Right click on it, **Duplicate** and add “_ForTraining” to its name.

Select the duplicated image and **display only the Nuclei, Ki67 and PanCK channels**.

We would like to segment two types of regions: 1) Packed Ki67 cells regions (Follicle) and 2) Pan-CK regions (Epithelial).

We can use a single measurement thresholder like we did for the Whole Tissue detection. Here however, we will train a pixel classifier to distinguish between 3 types of regions Follicle / Epithelial / Ignore*

Note: it is important to use the Ignore* when we do not want to create objects for the other class. You can have multiple ignored classes, all indicated by the * indicate this (Other* in our case).

1. Add 2 new classes for Follicle / Epithelial
2. Use the brush tool to annotate parts of tissue that are parts of *Follicle*, *Epithelial*, annotate other regions as *Ignore**.

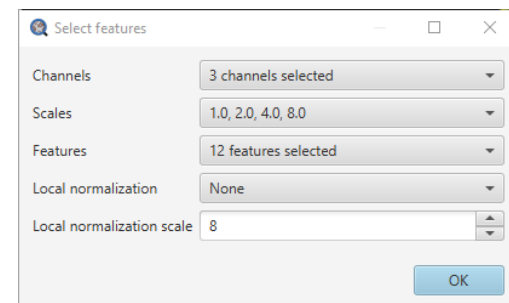
Make sure you highlight the desired class and use *AutoSet* while annotating

3. Create a Pixel classifier : **Classify > Pixel classification > Train pixel classifier**
4. Press Live prediction
5. Add more regions to train your classifier if needed.
6. **Tune Settings:**

- a. Select only relevant channels
- b. Add scales
- c. Add features
- d. Check how Resolution effects your classification

Don't bother small falsely detected regions, we will ignore them by size

7. Save your classifier as "FollicleEpithelialFinder"
8. Move to the original C06 image (Save the changes).
9. Use **Classify > Pixel classification > Load pixel classifier**
 - a. Click "Create objects", "Current selection"
 - b. Select values for minimum object size and minimum hole size
 - c. Keep all options to the defaults



Observe the results on the image

9 Exercise 8: Combine regions and cells analysis

Still with the C06 image.

We will apply the previous cell segmentation and classification and quantify cells within regions and cell distances to regions.

1. Create Full image Annotation.
2. Run StarDist
3. Apply your cell classifier of choice.
4. **How many cells** of your selected type are within Follicle / Epithelial regions?
What is the percentage of your selected cells out of the all cells in these regions?

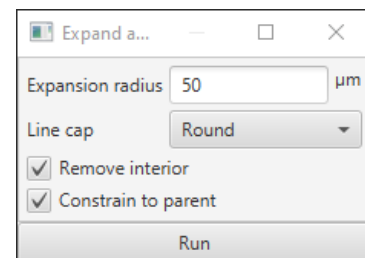
5. How many cells are within 50um from Follicles, and what is their percentage?

Inflate the Follicle regions by 50 um with *Objects > Annotations... > Expand annotations* .

Select *Remove Interior* . Set it to another class (eg *FollicleViscinity*)

6. Create Density map:

- a. *Analyze > Density maps > Create density map*
- b. Hide detections
- c. Show classifications
- d. Show on the Nuclei channel, or without any channel
- e. Change Main type from *Any* to your selected cell type



7. What is the distance of your selected cells from Epithelial regions?

Analyze > Spatial analysis > Signed distance to annotations 2D

Hide the negative cells

Create measurement map for "Signed distance to Epithelial um"

Show histogram of the distances

We want to know the **distance of the each cell to the closest region of a certain type** ...

Analyze > Spatial Analysis > Distance to annotation 2D ... (next step) click Yes

Click on a cell to see the added measurement

10 Exercise 9: Export Results

10.1 Export measurements

There are 3 different ways to export measurements within QuPath, via:

1. The *measurement table* use the *Save* button at the lower right side
2. The *measurement exporter*
3. A *script*: see [QuPath Doc](#)

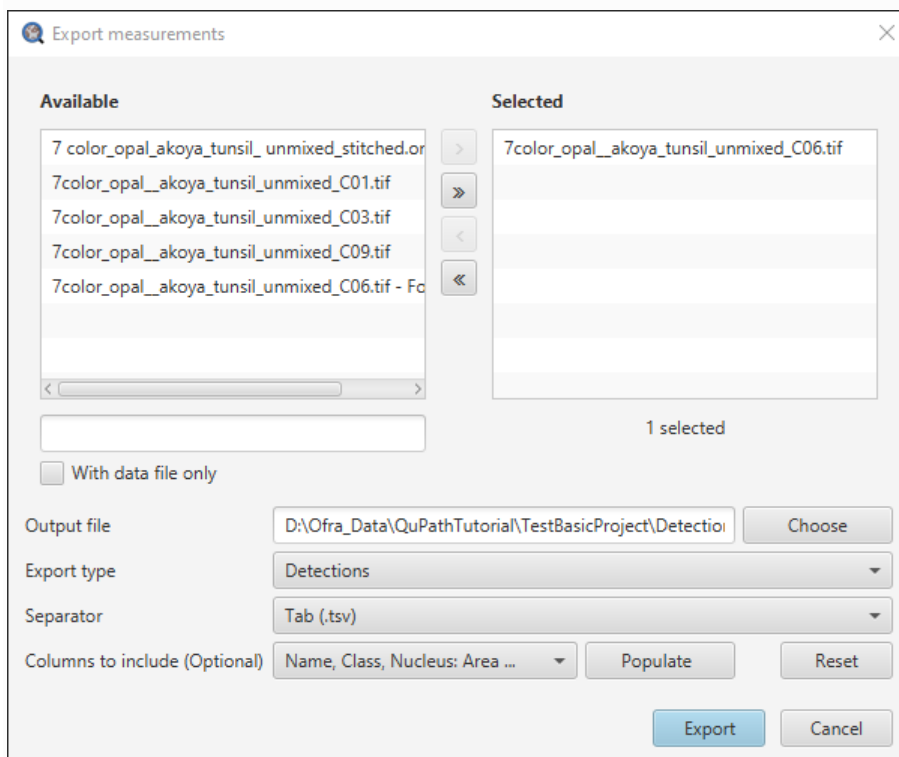
Here we will use the second one

1. Make sure all the images are saved
2. *Measure > Export Measurements*

Select the measurement type to be exported

If you don't want all the data, Use *Populate* to select the columns to be included

3. Inspect the exported files



Label images, masks and more information can be exported as well...

10.2 Export Annotations and Detections (Advanced, Optional)

There are two main ways in which annotations might be exported:

1. Shapes (vertices)

Primarily useful to transfer annotations between different software applications (e.g. a QuPath ROI to an ImageJ ROI).

2. Images (binary or labeled)

Often used to train AI models (e.g. with TensorFlow, PyTorch), but also for further processing

A **Label Image**, is an image in which all the pixels of each individual object are given identical unique value

Here we can will use a script to export detections and annotations as labeled images.

Drag and drop the script *ExportObjectsAsLabelImage.groovy* into QuPath and run it.

It will create new two new images under the `outSubFolder ("image_export")`.

You can open the images in ImageJ to inspect them

You can export annotations as shapes

GeoJSON: *File > Export objects as GeoJSON*, which can be imported to python eg via [Shapely](#)

ImageJ Rois: with *Extensions > ImageJ > send region to ImageJ*, select to include ROI and overlay and then in imageJ you can use *Image > Overlay > To ROI Manager*

11 Exercise 10: Apply Analysis workflow to the whole project and inspect results

You can combine the above steps into a workflow and apply them to a single image or a multiple images using a QuPath script. QuPath scripts are written in groovy or java.

The script ***FullAnalysis_ForProjects.groovy*** implements most of the steps in the workflow.

You can tune multiple aspects by tuning the script parameters values:

- Control which steps to perform
- Which classifiers to use in each step
- Minimal object size
-

1. Drag and drop the script into QuPath.
2. Set the classifiers names and class names to fit your own classifiers:
WholeTissueClassifierName, CellClassifierName, PixelClassifier
3. Open another image (eg C03) and run the script with *Run > Run* from the script editor menu
4. Delete all objects : *Objects > delete... > Delete all objects*
5. Now run the script on multiple images by:
 - o From the script editor menu choose *Run > Run for project*
 - o Select the desired images to apply the script on (C01, C03, C09)

```
// Script Operation Parameters
def UseFullImageAnnotationInsteadOfSelectedClass = 1

def RunWholeTissueClassifier = 1
def deleteExistingObjects = 1
def runStarDist = 1
def filterSomeCells = 1
def classifyCells = 1

def RunPixelClassifier          = 1
def runAnnotationExpansion      = 1
def calcDistanceToAnnotations   = 1
def exportLabelImages           = 1

// Parameters for Whole Tissue Selection
def ClassToSelect               = "WholeTissue"
def WholeTissueClassifierName    = "WholeTissueFinder"
def MinWholeTissueSize          = 200000 // um^2
def MinWholeTissueHolesize      = 10000 // um^2

// StarDist based Cell Segmentation parameters

// Specify the model .pb file (you will need to change this!)
def pathModel =
'A:/shared/QuPathScriptsAndProtocols/QuPath_StarDistModels/dsb2018_heavy_
_augment.pb'

def NucChannel = 'Nuclei'
def ProbabilityThreshold = 0.5
def PixelSize       = 0.2485 // Resolution for detection, make
sure to set it to the Pixel Size of your data
def CellExpansion    = 5.0 // pixels

// Further cell filtering parameters
def MaxNucArea       = 300 // um^2
def MinNucArea       = 10 // um^2
def MinNucIntensity   = 20 // remove any detections with an
intensity less than or equal to this value
```

```
// Cell Classification Parameters
def CellClassifierName = 'CD8_Ki67_CD20_ML_v1'

// Pixel Classifier Parameters
def PixelClassifier = "FollicleEpithelialFinder"
def Minimum_ObjectSize = 10000
def Minimum_LumenSize = 2000

// Annotation expansion parameters
def AnnotationClassToExpand = "Follicle"
def ExpandRadius_um = 50.0

// Results parameters
def ResultsSubFolder = 'export'
def downsample = 1 // 10
def labelsSubFolder = "image_export"
def outputSuffixCellLabels = "_CellLabels.tif"
def outputSuffixAnnotationsLabels = "_AnnotationLabels.tif"
```

12 Exercise 11: Train classifiers using representative regions from multiple images

A typical QuPath Workflow depends on multiple classifiers that need to be tuned and trained for the specific application and image dataset:

1. Whole Tissue detection
2. Cell segmentation algorithm
3. Cell Classifiers for individual dyes
4. Pixel classifier for Identification of specific regions of interest

QuPath utilizes the concept of **Projects** to group images belonging to an experiment, for ease of navigation on the one hand, but also for batch processing.

Good Practice for project organization and training of classifiers and segmentation models

To keep things organized we highly recommend to have separate QuPath projects for training and for final analysis. We will refer them as *training-project* and *analysis-project*.

Note that object classifiers are saved within the project subfolders structure, thus once you trained the cell classifier, you need to **copy it from *training-project\classifiers\object_classifiers* to *analysis-project\classifiers\object_classifiers***.

When training classifiers, it is important to use multiple representative regions from multiple images and conditions, **to cover all the different anatomical regions, variation of backgrounds, variation of cell types and densities**.

Doing the training on multiple full images may be possible (depending on the images size) but it will require much resources. Instead of this we **select representative regions** from different images and **combine** them together **into** a single mosaic **training image**.

If we need **to train multiple classifiers** we will **duplicate the training image for each of them** to avoid confusion between ground truth training annotations.

In this exercise we will follow the steps needed for creating a training project and train classifiers from multiple images.

12.1 Create Training project

1. **Create a Project** by either:
 - Use **File > Project ... > Create project**, this will prompt you to select an empty folder or
 - Create an empty folder (eg "*QuPath_DoublePositive-TrainProj*") and drag and drop it into QuPath. We will refer to it as ***training-project***

2. Import images:

We will use the images BC01-BC03 from the Tonsil data folder (“7 color_opal_akoya_tunsil_unmixed_**BC01**.ome.tif”)

- Drag and drop your images into QuPath or
- Click on **Import**

In your own project make sure to include representative images that cover the different conditions in your experiment your images

3. Set up channel names using by Running for Project the script **“ChangeChannelNamesColors-Tonsil.groovy”**

12.2 Create Training Image

Usually when working with big slides we recommend selecting multiple (10-15) regions of ~ 500*500 um from each slide as the protocol below suggest. If your images are small (eg 1000*1000 pixels) you can consider combining multiple full images.

Select representative regions from multiple images, to cover the different anatomical regions, variation of cell types and densities, variation of backgrounds and artifacts. Make sure the regions are not too small so you can understand the context from the cropped region.

1. **Classify > Training Images > Create Region annotations:** Go through all your images and for each of them select 7-15 regions of your selected size (1024*1024 pixels / 512*512 um / 1000*1000 um), *unclassified* for each image.
Note that when you create a region it is created in the middle of the displayed region, you can click on it (it becomes yellow) and move it around to another location.
Make sure you save all the images, including the last one you change
2. **Classify > Training Images > Create Training Image** – This will create combined mosaic image composed of all the representative regions.
Use *Unclassified* regions, Rectangles only
3. On the training image, **draw rectangle annotations to cover the whole area**, but not to cover empty space. Set them to Class *Region**
You can use the same representative image for cell segmentation training (StarDist/CellPose) which is covered in another protocol.
However make sure to duplicate it for each classifier / DL model training (step 5 below)

4. Run Cell segmentation

This protocol assumes that membrane stain is not available for cell segmentation and use StarDist for cell segmentation. You can use Cellpose segmentation instead especially if you do have membrane stain.

Open the provided script *RunStarDistOpenCV.groovy* adapt it to match the name of your Nuclei channel. Make sure that the PixelSize match your data. Select the annotations and Run the script

```
def NucChannel          = 'Nuclei'
def ProbabilityThreshold = 0.5
def PixelSize           = 0.2485
def CellExpansion       = 5.0
```

Note that for Cell classification we usually run StarDist (which segment Nuclei) with cell expansion to include cytoplasmic marker expression. Make sure that cytoplasmic markers are mostly within this borders, otherwise you may need to tune this parameter. The same expansion parameter value should be used for training and later on in the Final Analysis step.

5. Quality Control – Check Cell Segmentation

The first thing you need to check is how good the nuclei segmentation is. For this use the *Nuclei Only* view. Especially try to check:

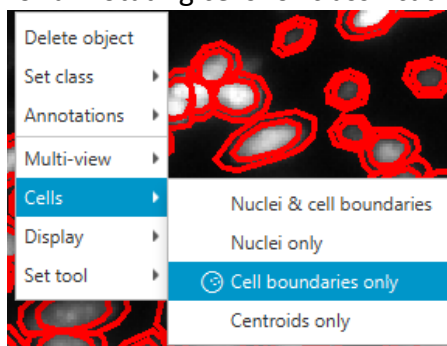
- Are nuclei boundaries accurate ?
- Are touching nuclei separated well or being segmented as single cell
- Are there nuclei that mistakenly broken into multiple nuclei? Especially look at big, elongated and dim nuclei.

If the segmentation is not good enough you should consider another segmentation method or retraining StarDist, please consult with an Image Analyst.

If segmentation is good enough move forward to train cell classifier

Tips

You can **control the appearance of the cell boundaries** by Right-Click in the image panel and using the *Cells* menu . To check how good is the Nuclei segmentation *Nuclei only* may be the best choice, while for annotating cells for classification *Cell boundaries only* may be better.



6. To avoid multiple runs of cell segmentation, **Save the image and then duplicate the image** to create independent copy for training and evaluating each cell classifier. Make sure to duplicate the data files as well. To duplicate the image: Select the image in the Project panel, right click and select *Duplicate Image*. **Switch to the duplicated image**.

12.3 Train a Machine-learning based (single channel) classifier

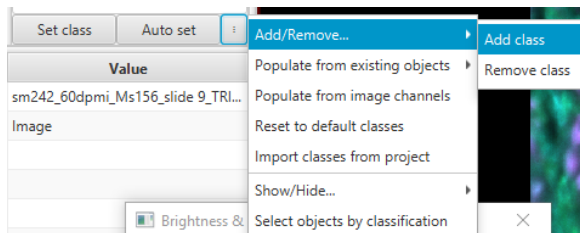
Repeat this instructions for each individual cell expression type that you want to classify.

1. Choose and Add the correct classes to classify

The instructions here assumes that each a cell can be positive for each of two markers independently, and you can have any combination of TypeX/TypeY. In this case you will train two independent classifiers. One for TypeX-Positive/TypeX-Negative and another one for TypeY-Positive/TypeY-Negative. After training the individual classifiers for TypeX and TypeY, you will create a combined classifier that will give you all 4 options.

However, if this is not the case, than you need to use other train strategies. For example if you have cells that can be either TypeX or TypeY or no one of them, but cannot be double positive for both X and Y , you will want to train the a classifier for 3 classes: TypeX / TypeY / Other


Add Classes for (the classes of choice **TypeX-Positive** and **TypeX-Negative** using ... > Add/Remove...>Add Class from the lower right side of the Annotations Tab.



Now you will interactively annotate and train the classifier, inspect the results, correct them by adding more annotations and retraining, until you are satisfied with the results.

So you will switch between steps 2 and 3

2. Annotate classes

- Use the point tool 
- Use the Add button to add classes for *TypeX-Positive* and *TypeX-Negative*. Use *Set Class* button to set the class types
- Click on *Auto set*
- Add points for several obvious cases

3. Train classifier:

After you selected multiple examples **start an object classifier** with *Classify > Object classification > Train object classifier* (Ctrl+Shift+D)

Set Classifier parameters

- Object filter = Cells,
- Classifier = Random Trees,
- Features = selected measurements and select
 - o All shape features,
 - o All Intensity features of the relevant channels for the classifier. Usually it is just all features related to TypeX channel
- You can highlight multiple features and select them at once by right-click > select
- Classes = selected classes, select TypeX-Positive / TypeX-Negative
- Training = points only

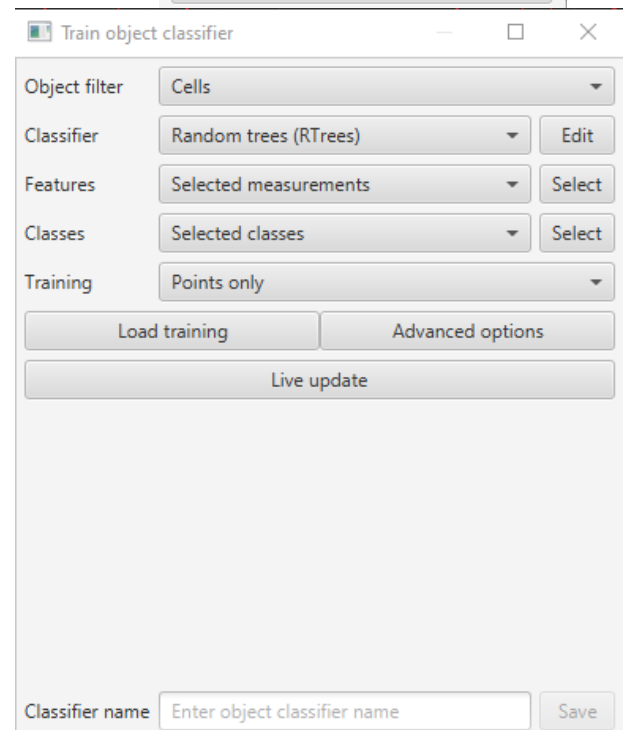
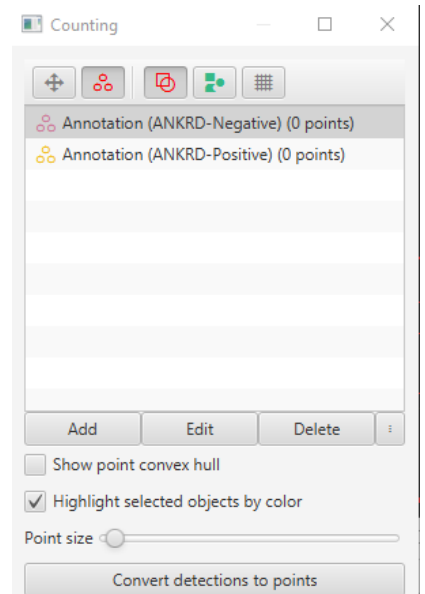
Click on *Live update*

4. Inspect results and make Corrections – add points for falsely classified cells,

Make sure to move to other regions of the sparse image,

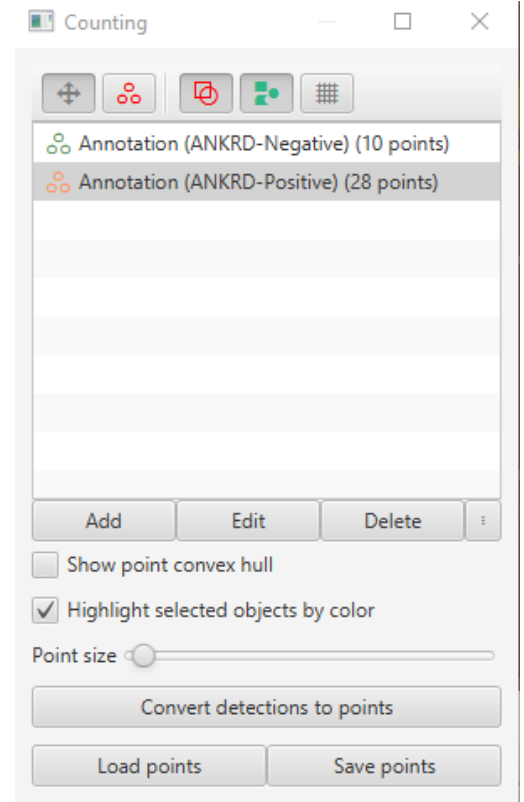
Make use of keyboard shortcuts for Move(M) / Point(.) / Fill (F) and of showing hiding different channels

Don't select points for cells on the border of tiles



5. Make sure to save intermediate annotations + classifiers

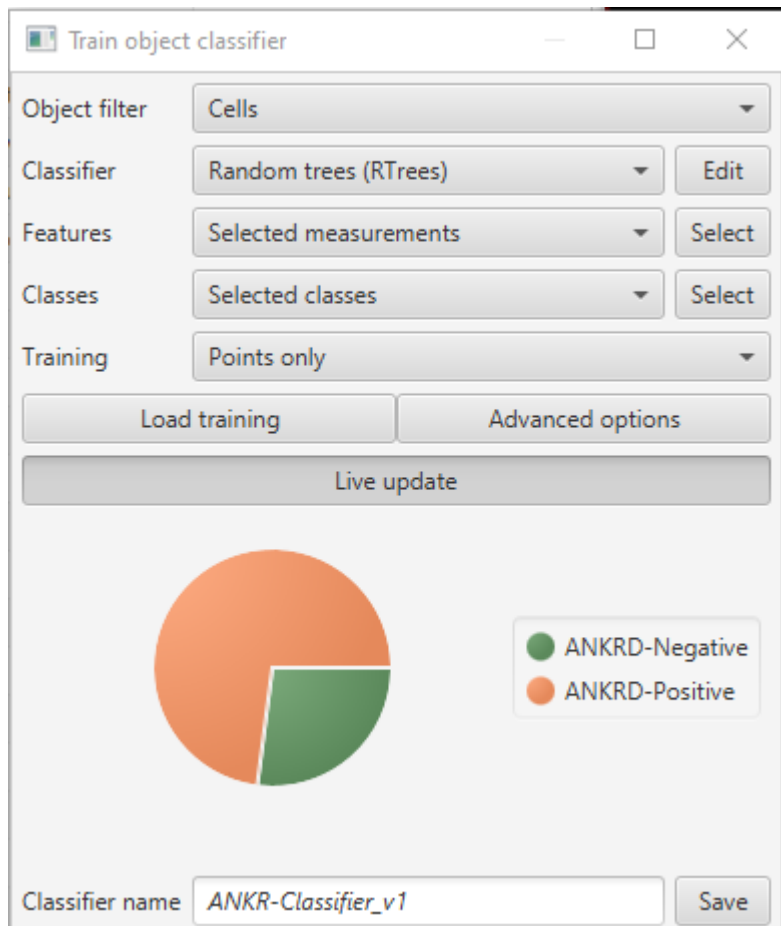
- a. **Save points from counting window** – all points , name it
XXX_points_vN



b. save classifier – from Train object classifier window

- **Use meaningful classifiers names.** consider using Date within the classifier name
- Keep vN matched between annotations and classifier, as you can load then later on, and undo options are limited for points.
- Increase the version vN each time you save,

Note that when working with points the cursor changes and each time you click the left-mouse key, selects a point. Undo options are limited. Thus it is very useful to switch between Points (.) and navigation (m) modes using keyboard shortcuts.



13 Get Prepared

Installation requirement for QuPath Tutorial on own computers

1. [QuPath 0.4.2](#)
2. [QuPath StarDist Extension](#)
3. [QuPath BIOP extension](#)
4. [QuPath Cellpose extension](#).

Cellpose is needed only for the advanced training

Cellpose extension requires also local Cellpose installation via conda,

I suggest to use the installation described in the above link

14 Acknowledgement

If you use QuPath, please [cite it](#):

Bankhead, P. et al. **QuPath: Open source software for digital pathology image analysis**. *Scientific Reports* (2017).

<https://doi.org/10.1038/s41598-017-17204-5>

Big Thanks goes to Pete Bankhead Author of QuPath

Olivier Burry, Nicolas Chiaruttini, Romain Guet of the [EPFL BIOP group](#)

The QuPath Community on image.sc forum, especially:

Zbigniew Mikulski, Core Director for Microscopy and Histology Facility at LJI, La Jolla Institute for Immunology

Mike Nelson

Mark Zaidi

Anonymous Research Associate