SLDC: an open-source workflow for object detection in multi-gigapixel images

Romain Mormont, Jean-Michel Begon, Renaud Hoyoux, Raphaël Marée

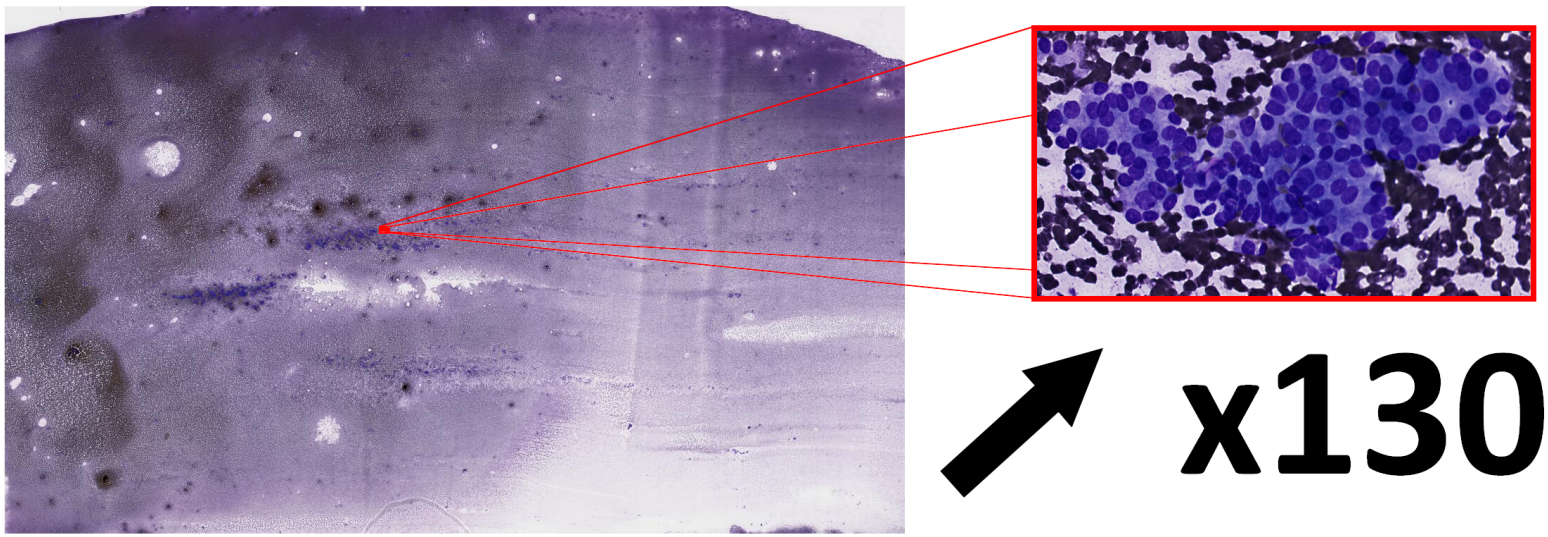
Montefiore Institute, University of Liège, Belgium

6th September 2016

Outline

1. Context
2. SLDC
   - Framework
   - How it works
   - Features
   - Toy example
3. SLDC at work: thyroid nodule malignancy
   - Cytomine
   - Thyroid nodule malignancy diagnosis
   - Workflow
   - Results

Context



Microscope slide smeared with cell samples (15 gigapixels).

- Huge slides usually **analysed manually** !
- Computer programs could be used to assist humans

SLDC: framework

SLDC is an **open-source Python framework** created for accelerating development of large image analysis workflows.

How ?

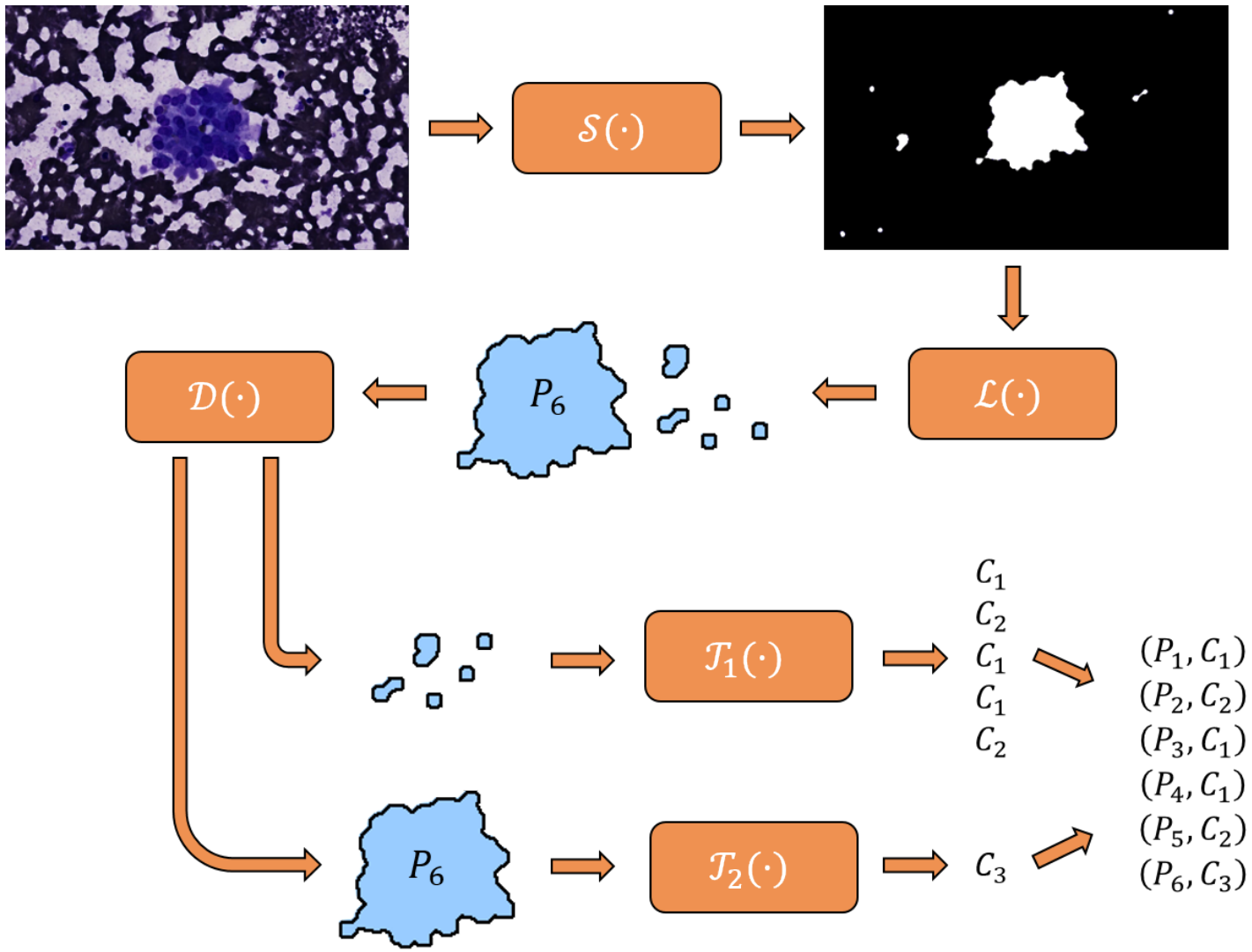
- It encapsulates problem-independent logic (parallelism, memory limitation due to large images handling,. . .)
- It provides a concise way of declaring problem dependant components (segmentation, object classification,. . .)

Where ? On GitHub at https://github.com/waliens/sldc

SLDC: features

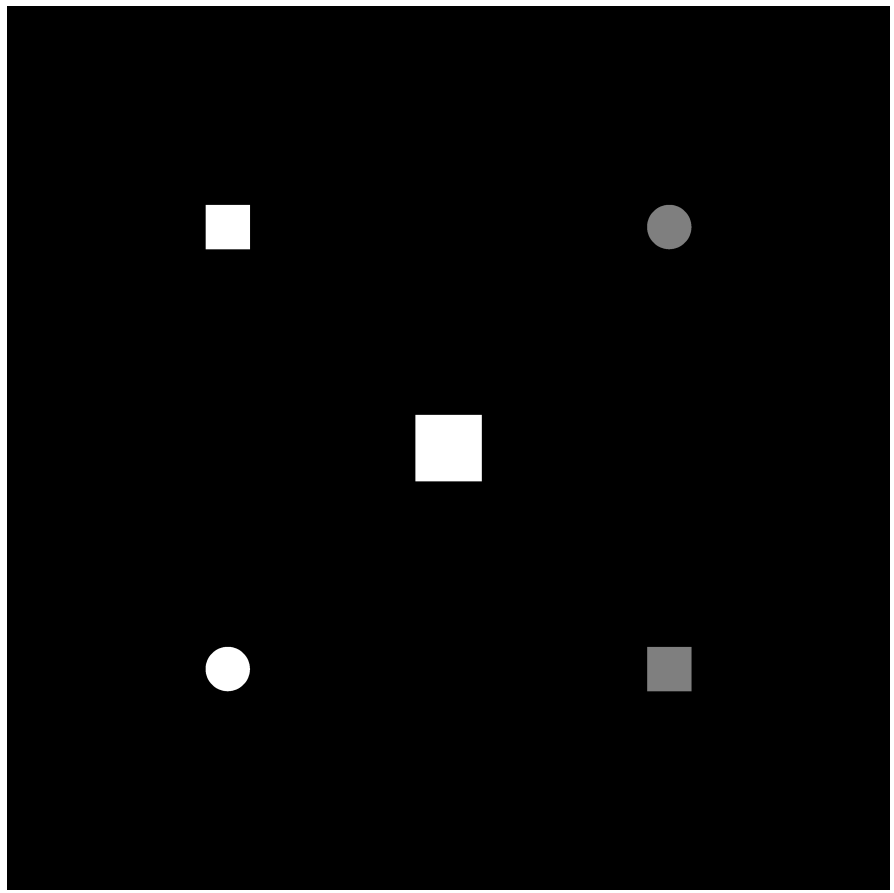
- **Tile-based processing** to avoid loading a full image into memory
- Several level of **parallelism**: tiles, objects, images,...
- A **customizable logging system** providing a rich feedback about the execution
- **Effortless integration** with other Python libraries: scikit-learn (ML), open-cv (CV), PyCuda (GPU),...
- **Builder components** providing an easy way of constructing complex workflows

SLDC: how it works



SLDC: toy example

The aim is to detect circles in the following image. As a bonus, we want to know their center color.



SLDC: toy example (cont'd)

```python
# Defining a segmenter
class CustomSegementer(Segmenter):
    """All non-black pixels are in an object of interest"""
    def segment(self, image):
        return (image > 0).astype(np.uint8)

# Defining a dispatching rule
class CircleRule(DispatchingRule):
    """A rule which matches circle polygons"""
    def evaluate_batch(self, image, polygons):
        return [circularity(p) > 0.85 for p in polygons]

# Defining a polygon classifier
class ColorClassifier(PolygonClassifier):
    """
    A classifier which returns the color (greyscale)
    of the center pixel of the object
    """
    def predict_batch(self, image, polygons):
        classes = [center_pxl_color(image, p) for p in polygons]
        probas = [1.0] * len(polygons)
        return classes, probas
```

SLDC: toy example (cont'd)

```
# Build the workflow
builder = WorkflowBuilder()
builder.set_n_jobs(100)
builder.set_segmenter(CustomSegementer())
builder.add_classifier(CircleRule(), ColorClassifier(), disp_label="circle")
workflow = builder.get()

# Process an image
results = workflow.process(image)

# Go through the detected objects
for polygon, dispatch, label, proba in results:
  print "Detected polygon {}".format(polygon)
  print "Dispatched by '{}'".format(dispatch)
  print "Predicted class {}".format(label)
  print "Probability {}".format(proba)
  print ""
```

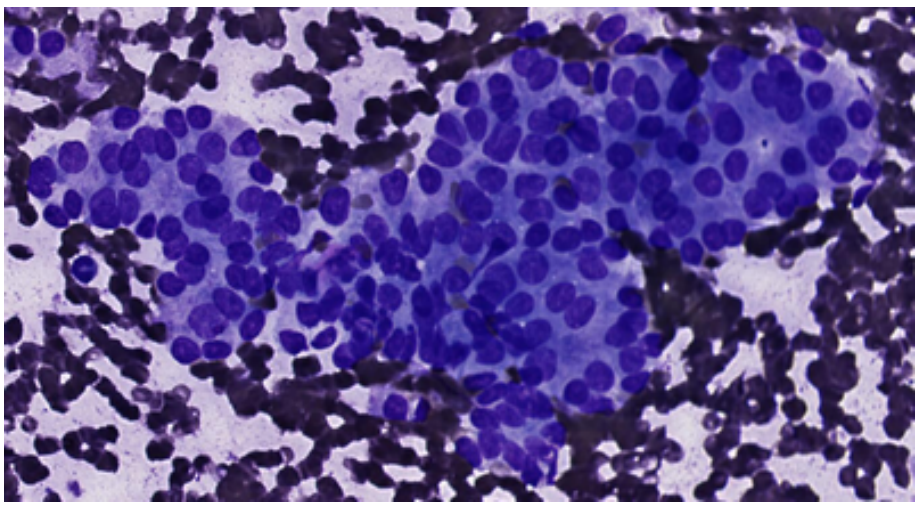
SLDC: toy example (cont'd)

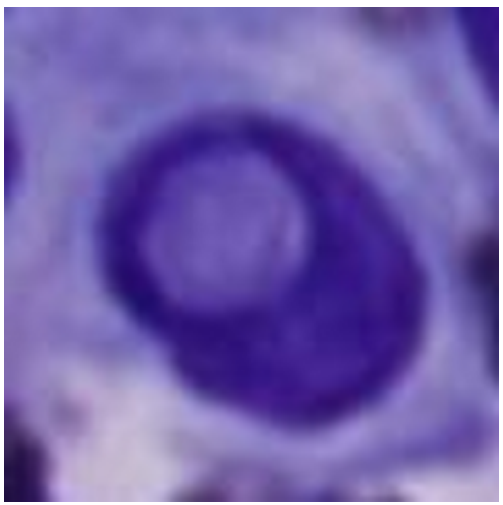
```
Detected polygon POLYGON ((...))
Dispatched by 'circle'
Predicted class 128
Probability 1.0

Detected polygon POLYGON ((...))
Dispatched by 'circle'
Predicted class 255
Probability 1.0
```

SLDC at work

Aim: detect **cells with inclusion** and **proliferative architectural patterns**





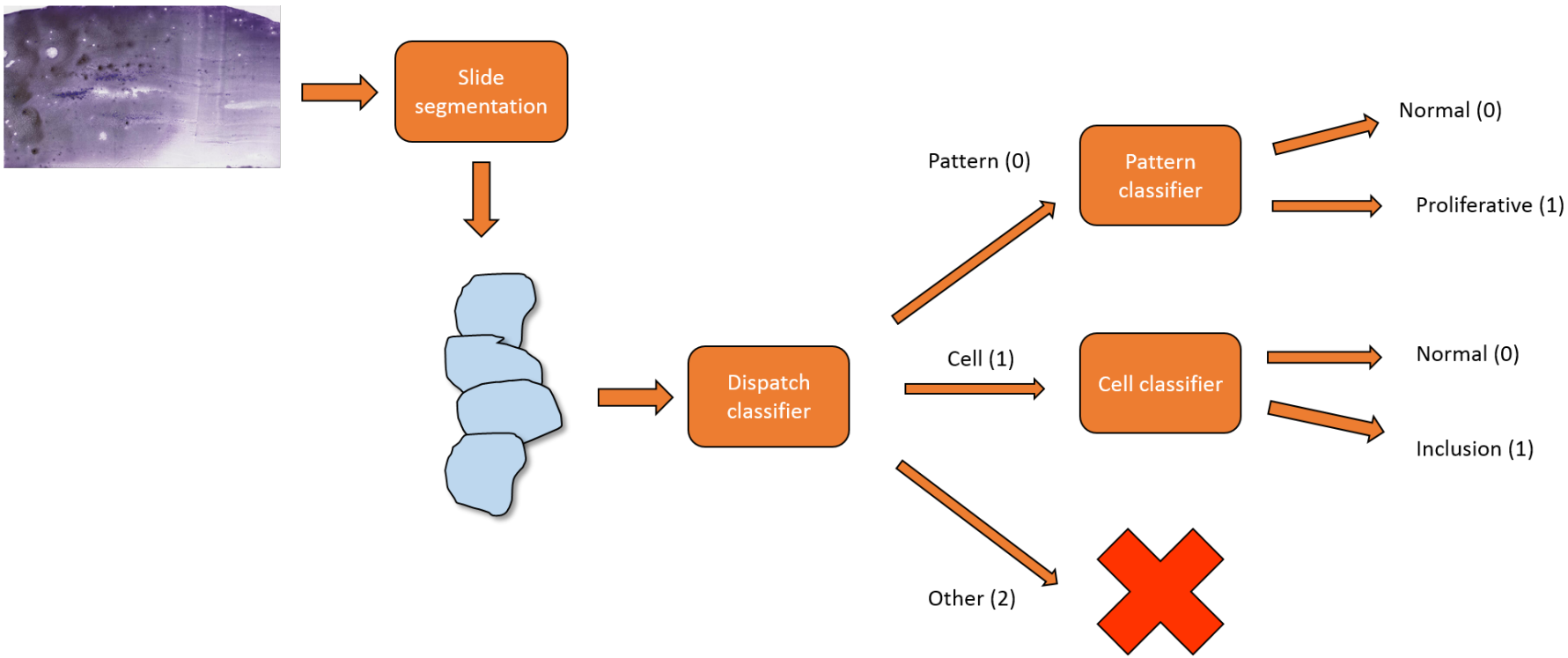
SLDC at work: Cytomine

TODO

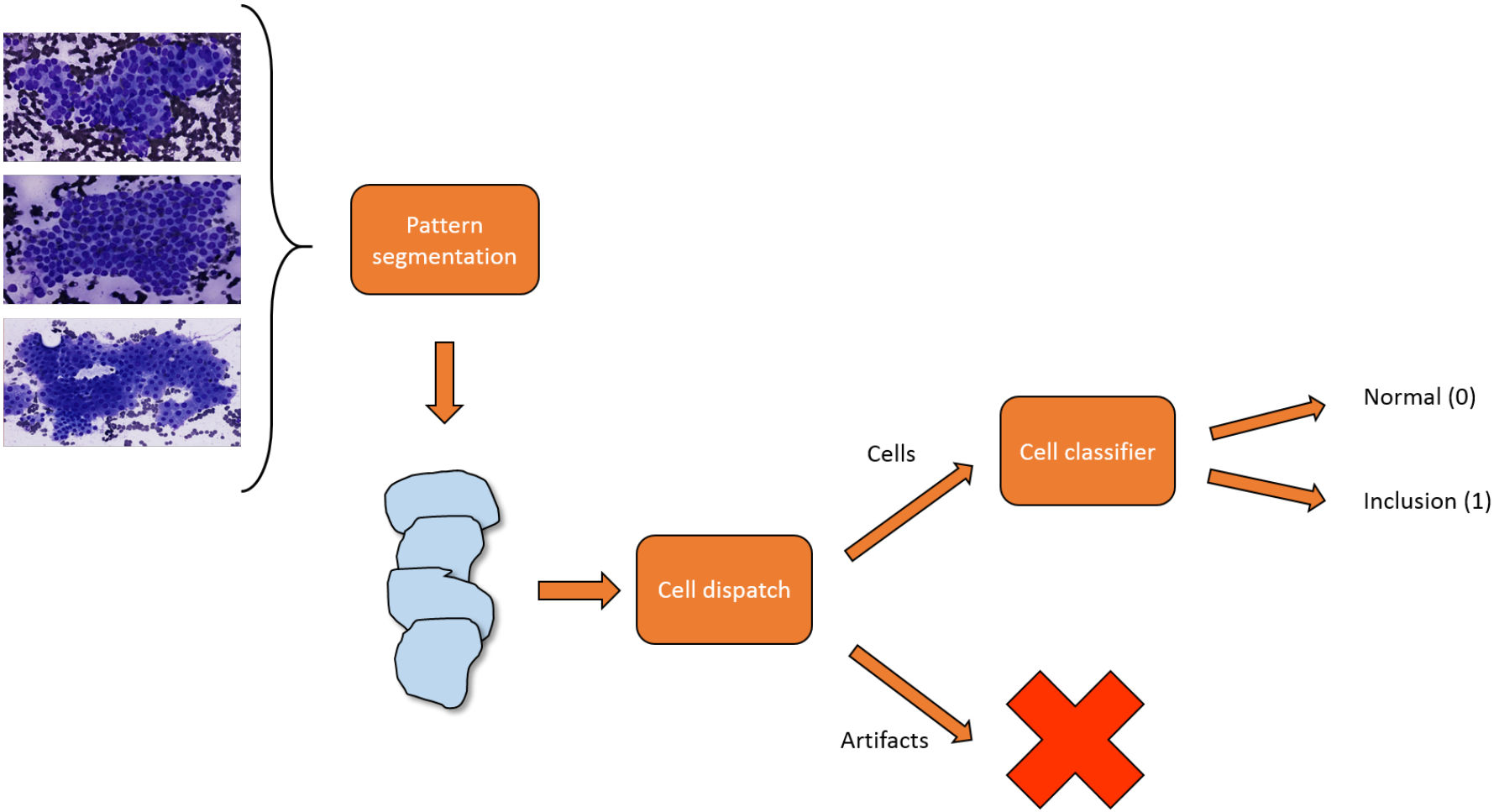
SLDC at work: thyroid nodule malignancy diagnosis

TODO

SLDC at work: workflow



SLDC at work: workflow (cont'd)



SLDC at work: results

TODO

Future works

???

Thank you for your attention !
Any question ?