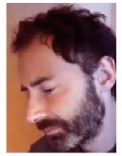
cytomine and BIAflows for data and computer scientists

May 11th 2021

Presenters : **Raphaël Marée** & **Sebastien Tosi**

Helpers : **Renaud Hoyoux**, **Ulysse Rubens**, **Romain Mormont**



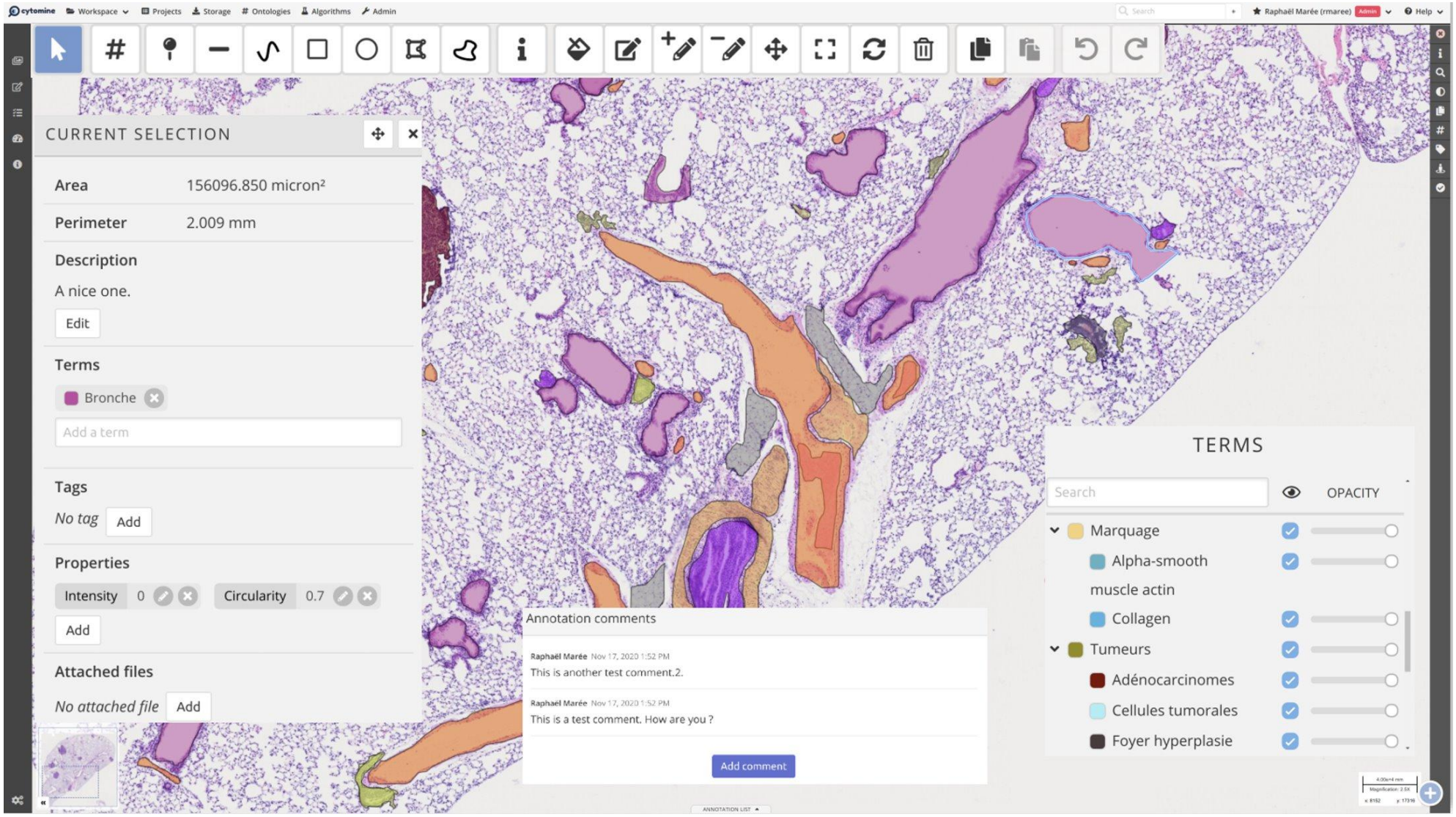
neubias network of european bioimage analysts

cytomine .ORG

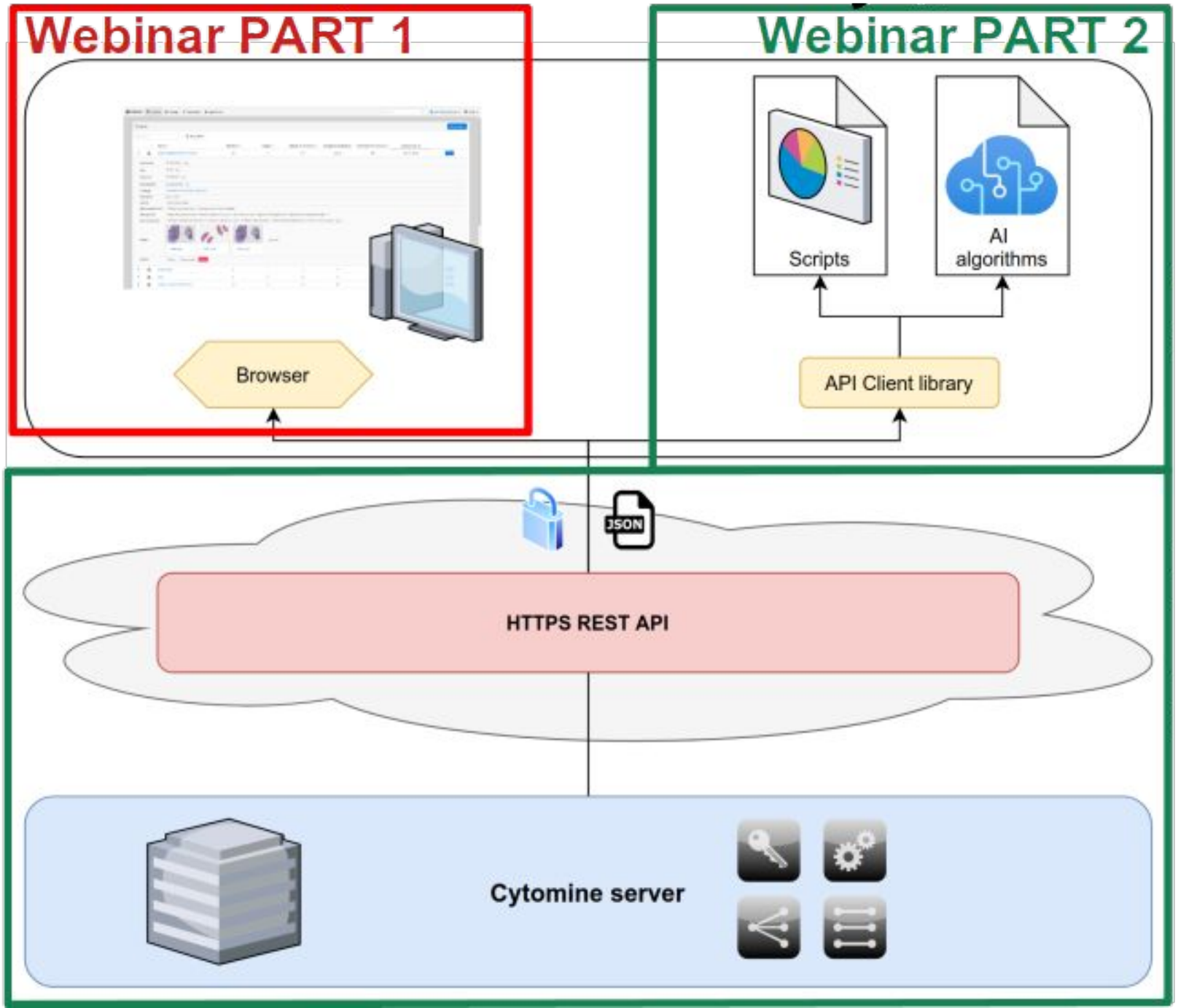
LIÈGE université

Last week: Cytomine-webUI for collaborative annotation

Video recording: https://www.youtube.com/watch?v=AFclLoaaekw



Single-Page Application, Written in VueJS, loaded in browser:
https://github.com/cytomine/Cytomine-Web-UI



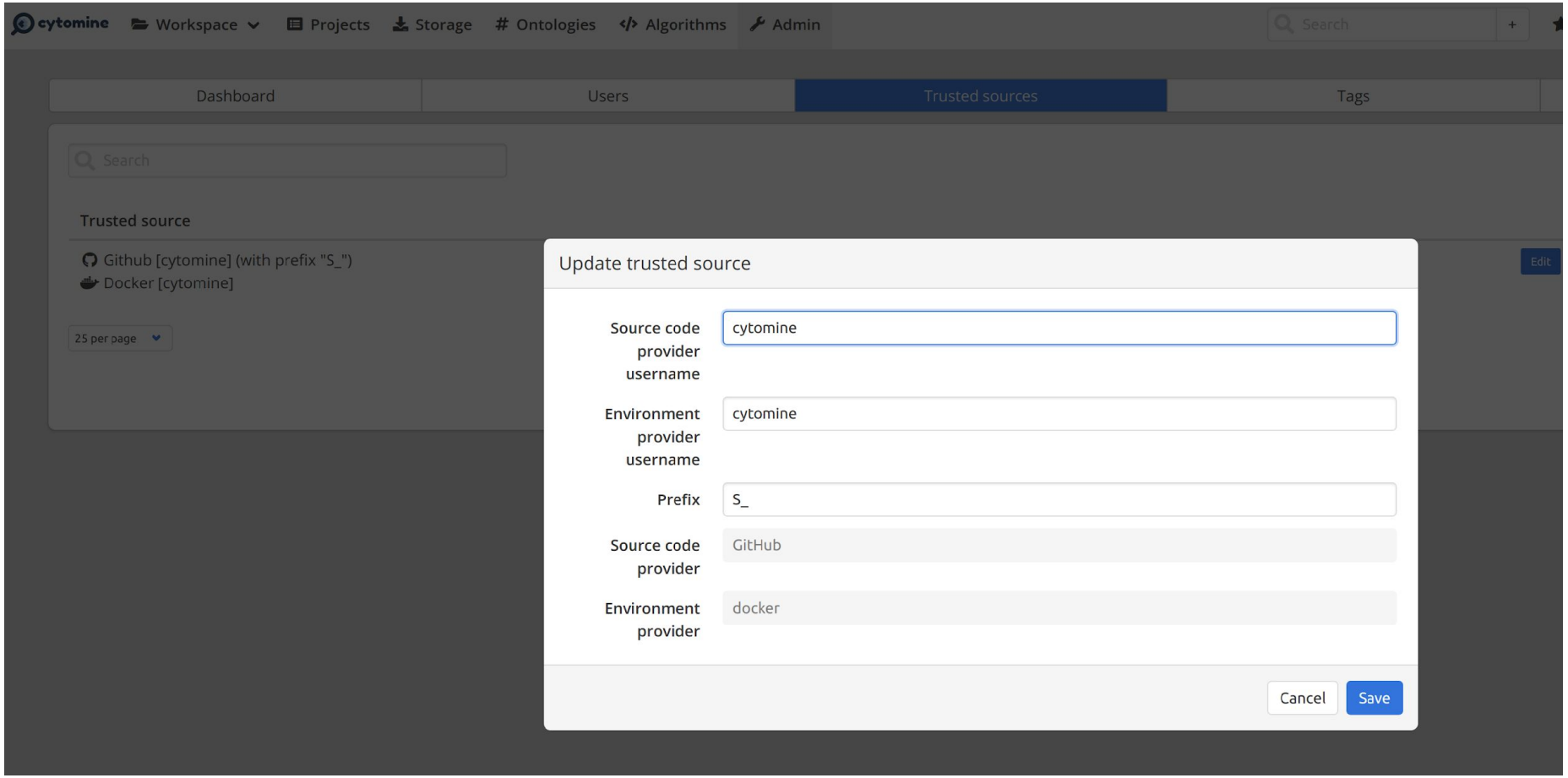
Cytomine
Overview of existing Apps
Architecture
Data models
API
API Clients
- Java/Javascript/Python

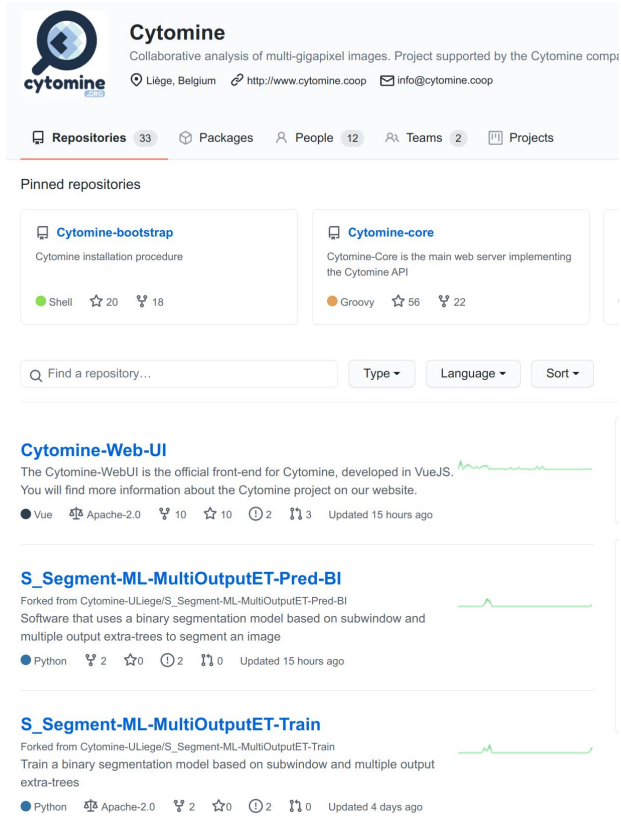
Creating Reproducible Apps
- Descriptor
- Containers

BIAFLOWS
Benchmarking with demo

Running algorithms from Cytomine Web-UI:

1. Configure “trusted sources” to fetch “Apps” from a source





Configuration to be done at the level of your Cytomine server (by admin e.g. core facility manager, PI)

Running algorithms from Cytomine Web-UI: 2. Enable “Apps” (“Algorithms”) in your project configuration

General | Members | Custom UI | **Algorithms** | Image filters

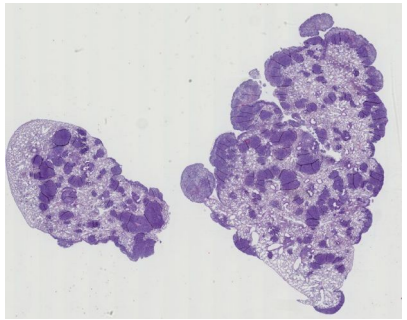
Search

Name	Version	Runnable	Status
Stats-TermArea (v1.1.1)	Last release	✔ Yes	Enabled
CellDetect_Stardist_HE_ROI (v1.0.3)	Last release	✔ Yes	Enabled
Segment-CV-AdaptThres-Sample (v1.3.0)	Last release	✔ Yes	Enabled
Segment-CV-AdaptThres-Sample (v1.2.1)	Deprecated	✔ Yes	Enabled
Segment-CV-AdaptThres-Sample (v1.2.0)	Deprecated	✔ Yes	Enabled
BckgrdSeg-ML-RandSubwET-Pred-BI	No information	✖ No	Enabled
Segment-CV-AdaptThres-Object-BI (v1.0.3)	Last release	✔ Yes	Enabled
Segment-CV-AdaptThres-Sample (v1.1.0)	Deprecated	✔ Yes	Enabled
TissueSegment_Model_Predict	No information	✖ No	Enabled
TissueSegment_Model_Builder	No information	✖ No	Enabled
TissueDetect	No information	✖ No	Enabled
S_Zebrafish_Head_Operculum_UNet_Segmentation (v3.0.4)	Last release	✔ Yes	Disabled

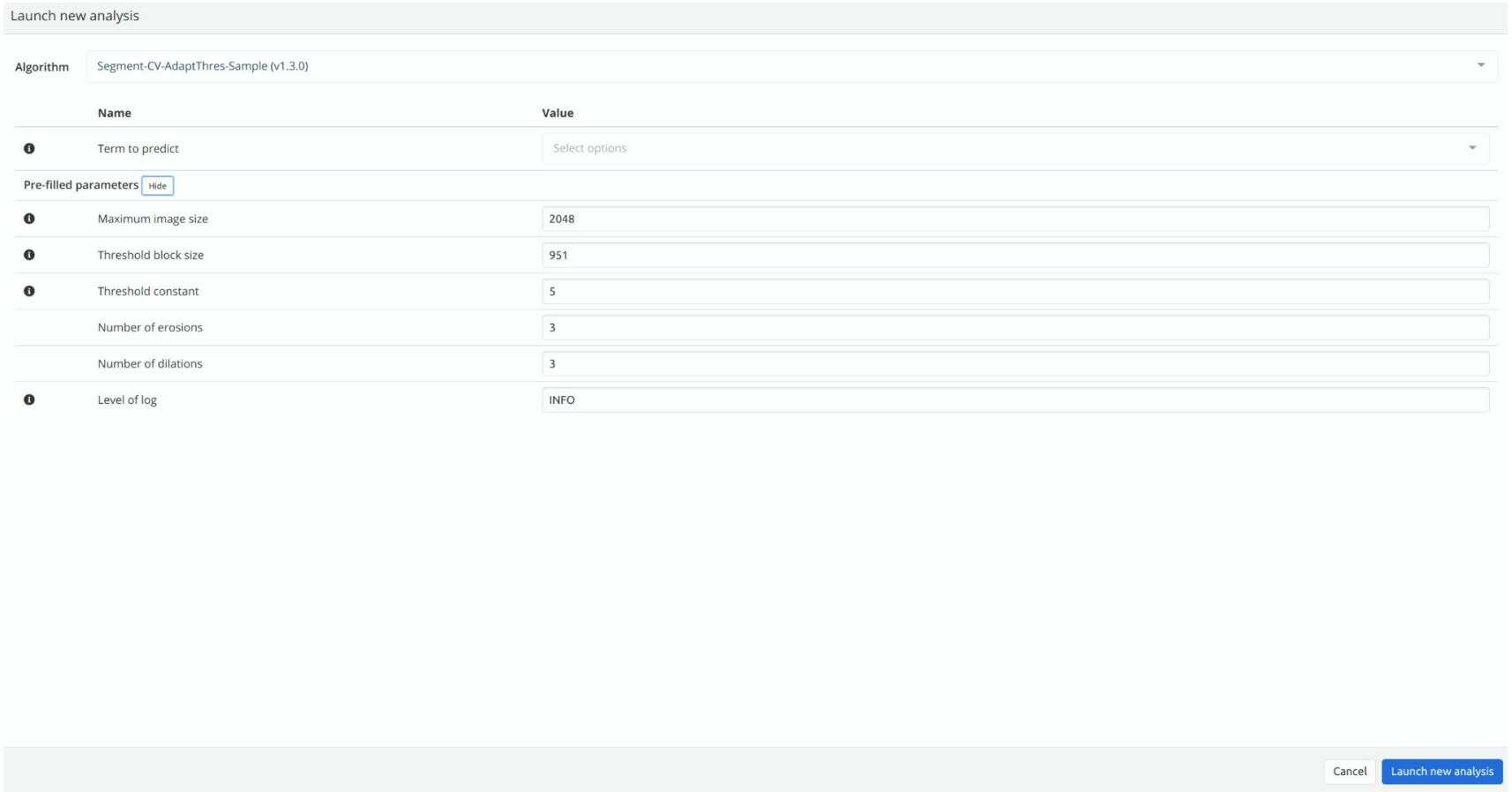
Configuration to be done at the project level (by project manager e.g. PI or researcher)

Example: Tissue sample detection (no AI, basic image processing)

- Code: https://github.com/cytomine/S_Segment-CV-AdaptThres-Sample
- Input: List of images, ontology term to predict
- Output: Annotations (contours) of tissue samples with ontology term

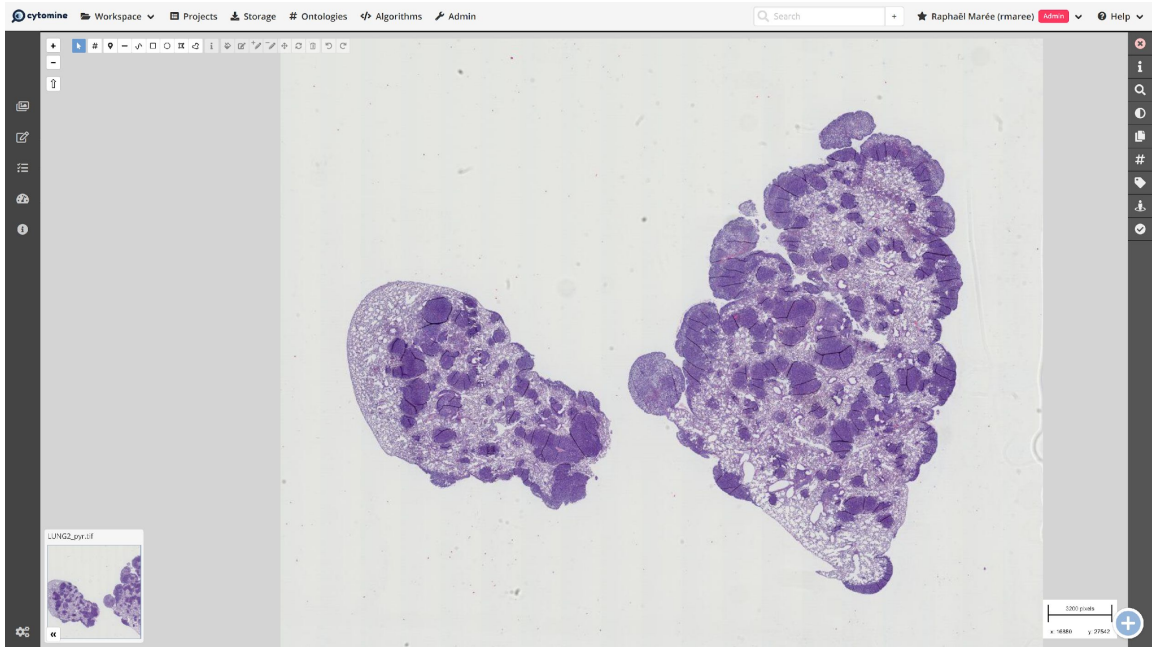


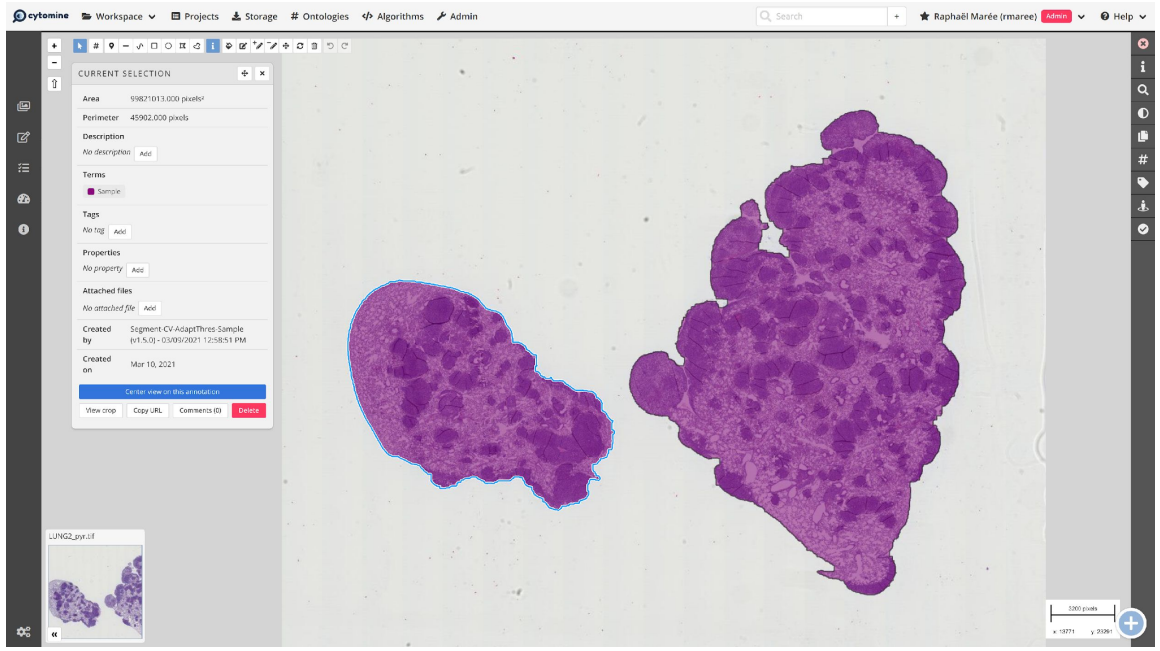
(40 000 x 35 000 pixels)



Example: Tissue sample detection (no AI, basic image processing)

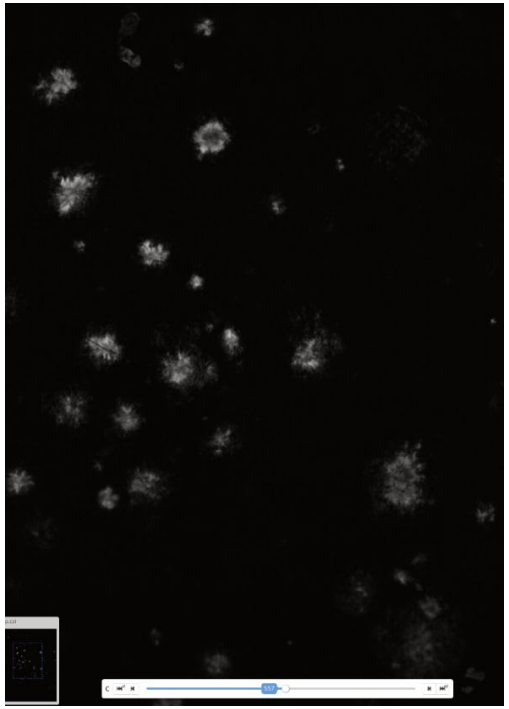
- Code: https://github.com/cytomine/S_Segment-CV-AdaptThres-Sample
- Input: List of images, ontology term to predict
- Output: Annotations (contours) of tissue samples with ontology term

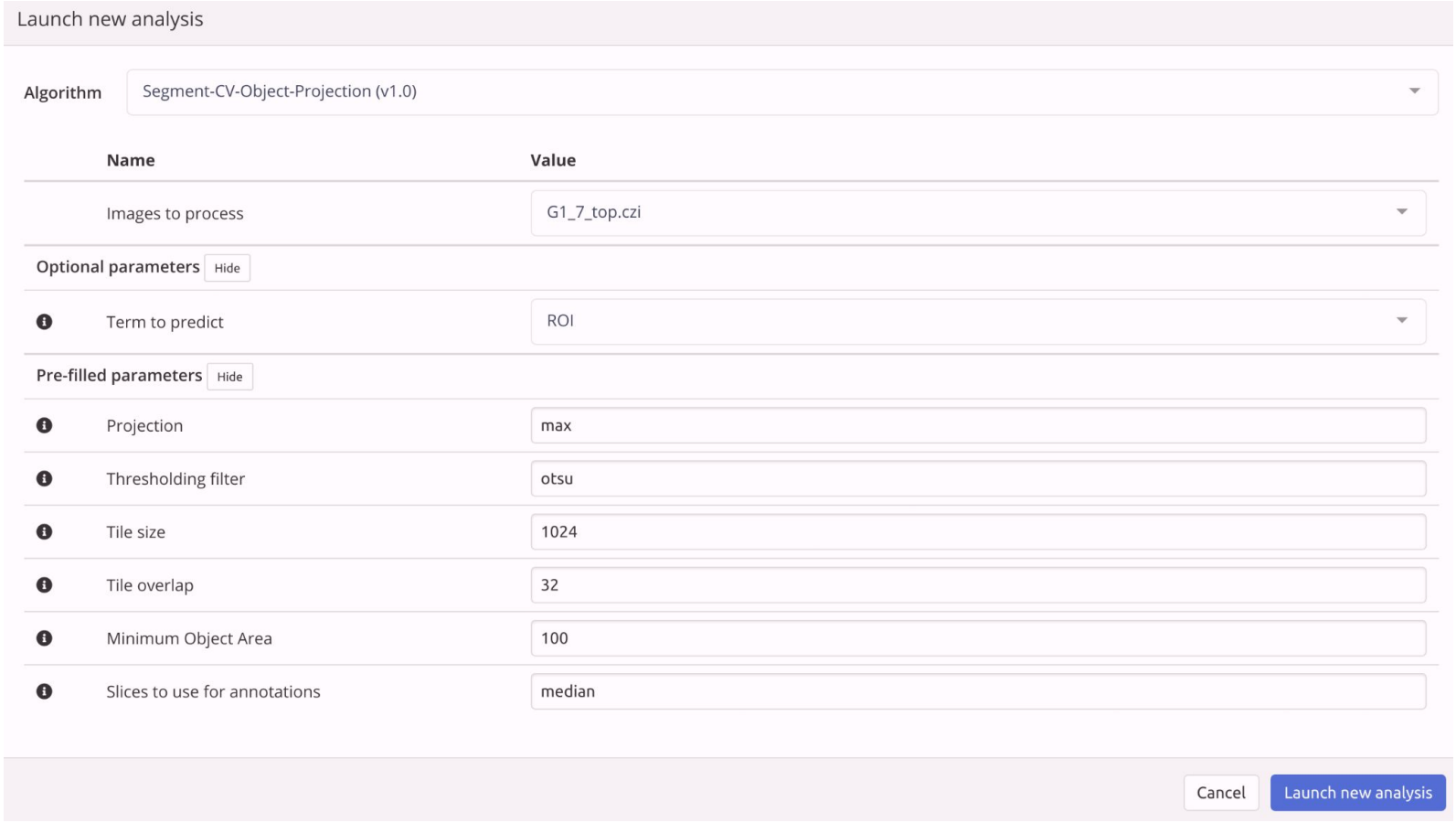




Example: Object detection in hyperspectral (2D+c) images

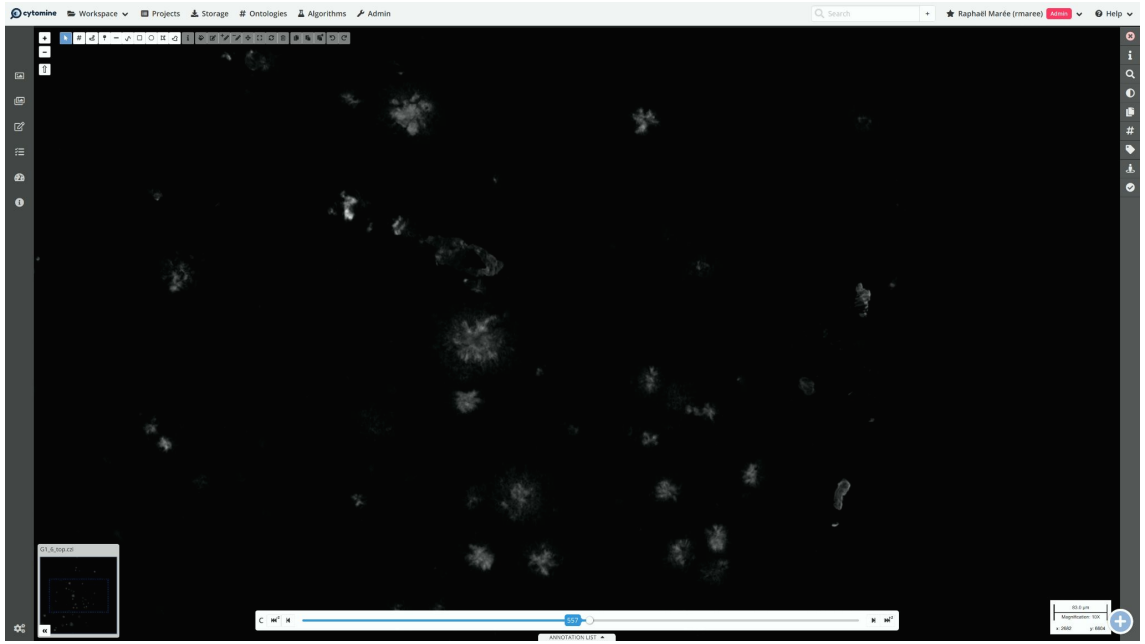
- Code: https://github.com/Cytomine-ULiege/S_Segment-CV-Object-Projection
- Input: List of images
- Output: Annotations (contours) of “objects”

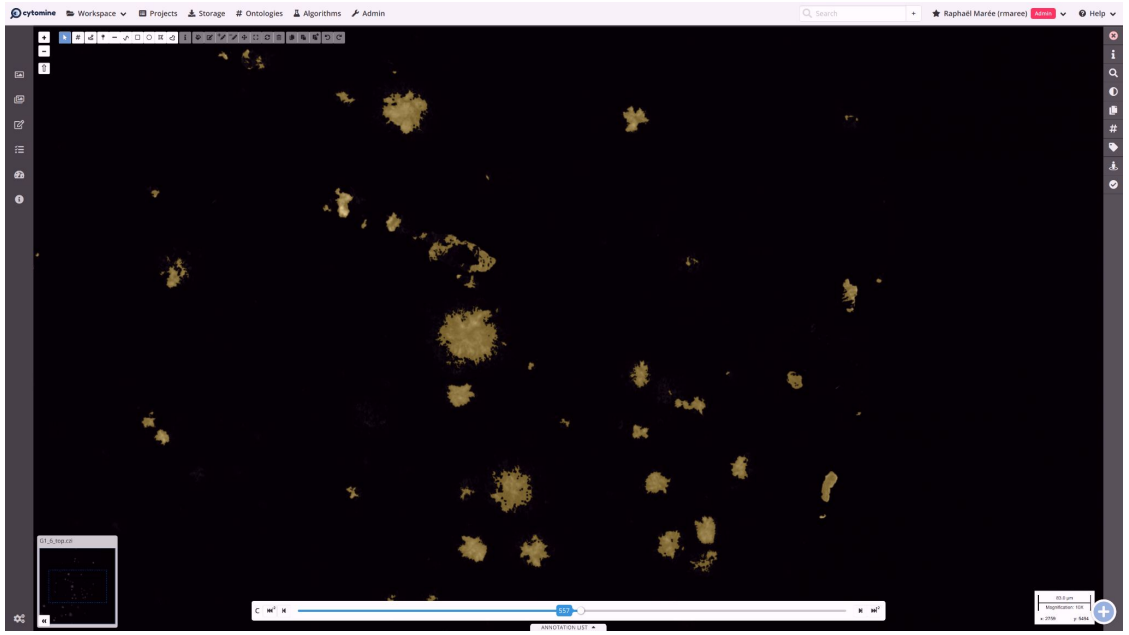




Example: Object detection in hyperspectral (2D+c) images

- Code: https://github.com/Cytomine-ULiege/S_Segment-CV-Object-Projection
- Input: List of images
- Output: Annotations (contours) of “objects”

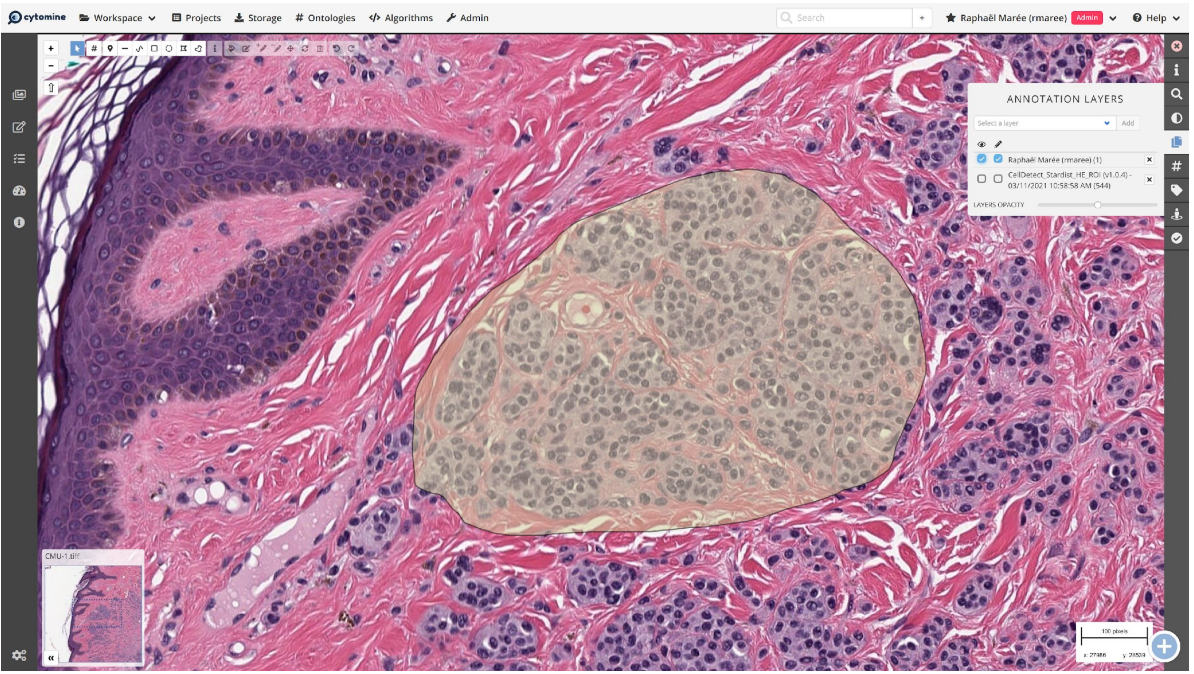




(Using Maximum Intensity Projection & Otsu)

Example: Cell segmentation (pre-trained StarDist H&E model)

- Code: https://github.com/cytomine/S_CellDetect_Stardist_HE_ROI
- Inputs: List of WSIs & ontology term for regions of interest, Stardist parameters
- Outputs: in each ROI, cell segmentations with ontology term



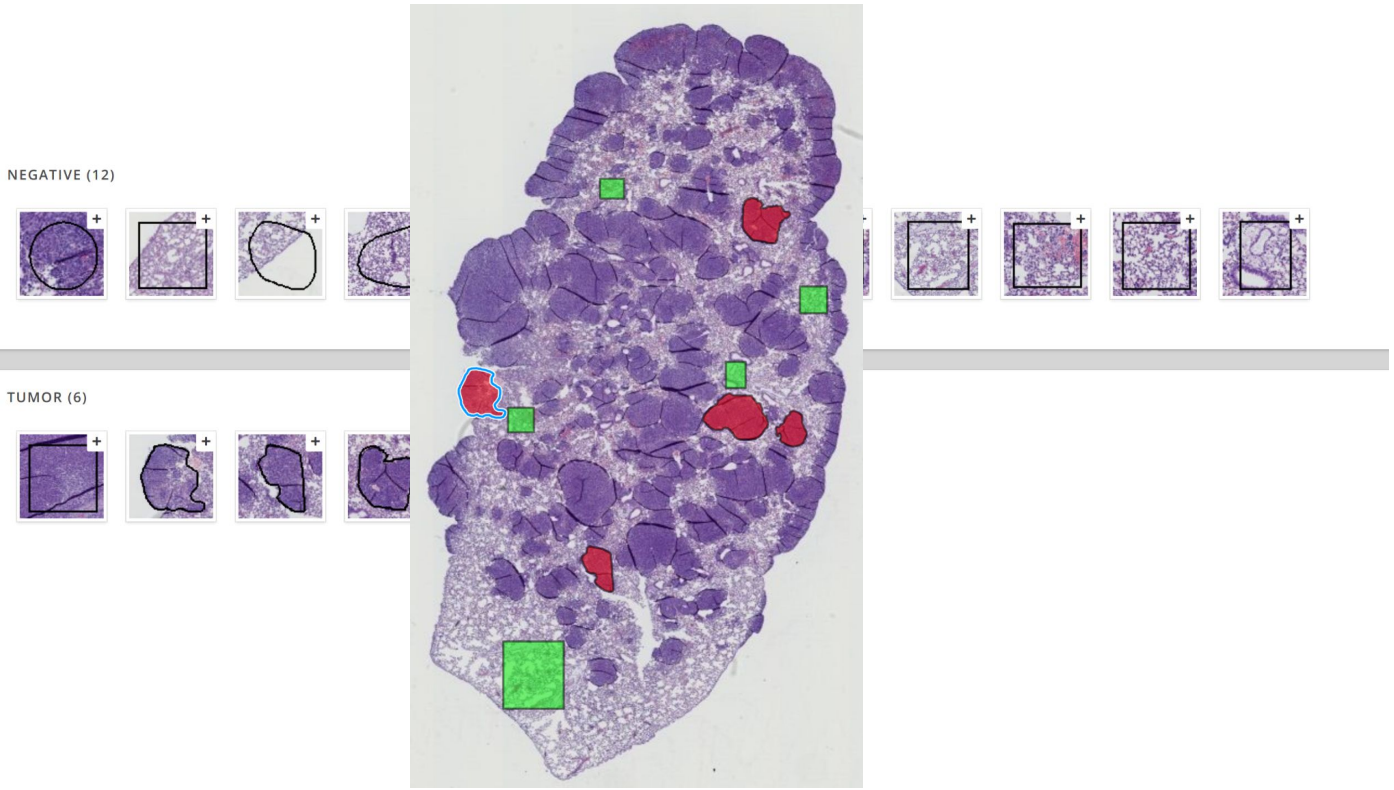
(based on Schmidt & Weigert, MICCAI 2018; https://github.com/stardist/stardist)

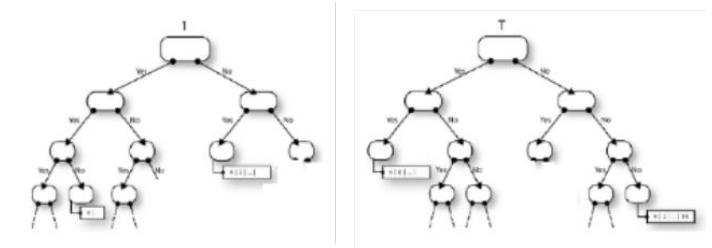
Example: Tissue area delineation: training (Random forests)

- Code: https://github.com/cytomine/S_Segment-ML-MultiOutputET-Train

- Inputs: images & ontology terms of positive/negative annotations, RF parameters (nb of trees, subwindow sizes,...)

- Outputs: a random forest-based segmentation model (saved in Cytomine database as “AttachedFile”)

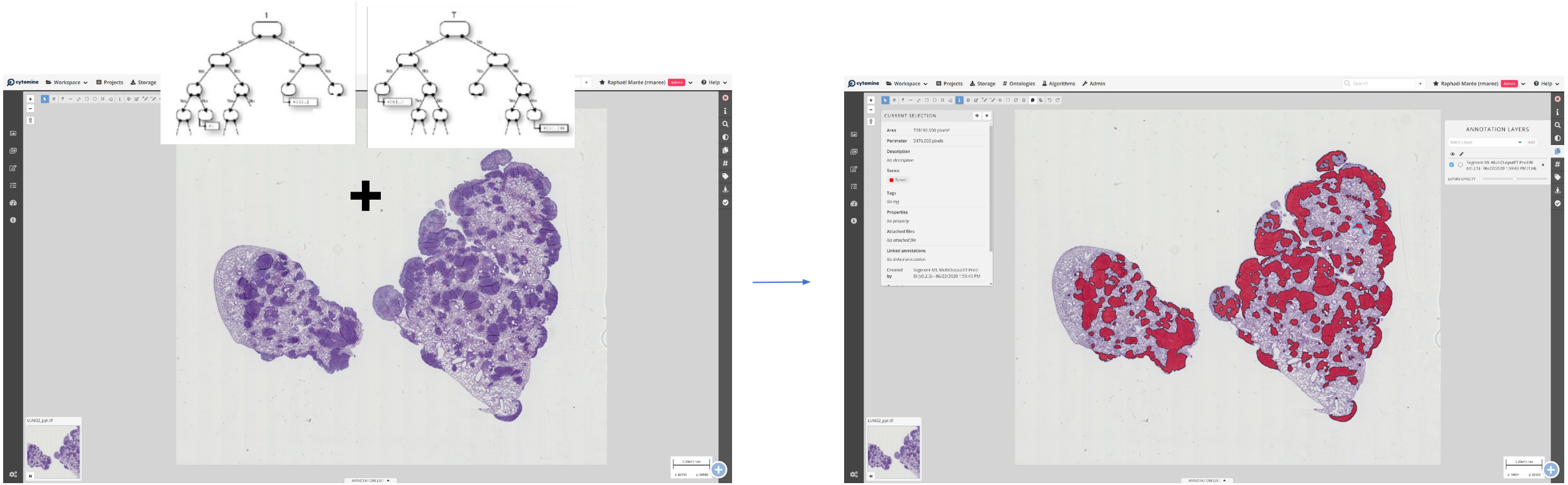




(Marée et al., ISBI 2014)

Example: Tissue area delineation: prediction (Random forests)

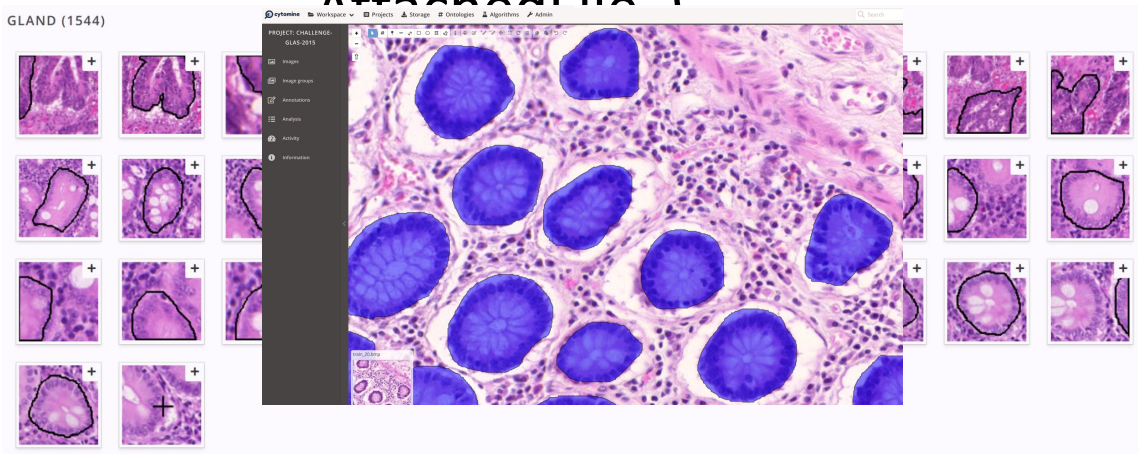
- Code: https://github.com/cytomine/S_Segment-ML-MultiOutputET-Pred-BI
- Inputs: list of images, a software identifier which generated a model, ontology term
- Outputs: in each image, segmented positive regions with ontology term

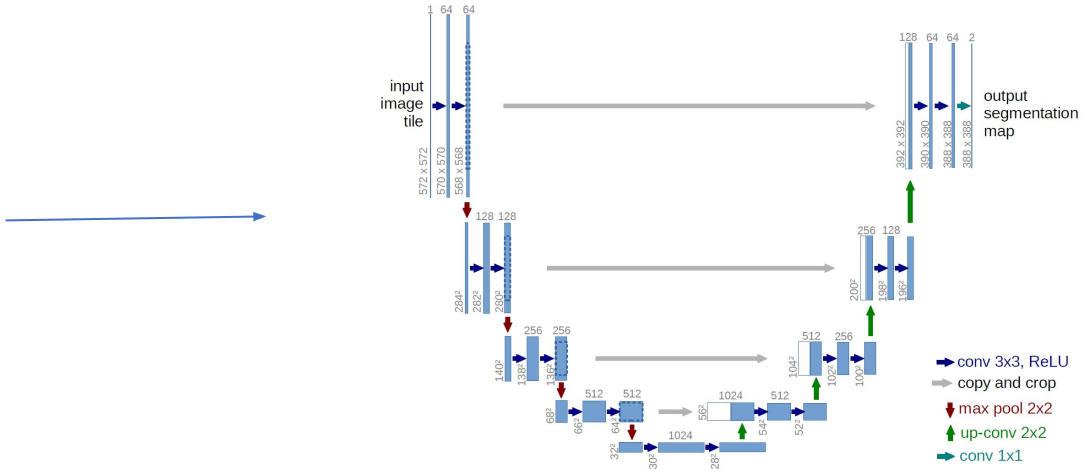


(Marée et al., ISBI 2014)

Example: Foreground segmentation (U-Net)

- Code: https://github.com/Cytomine-ULiege/S_Segment-ML-UNet-Binary-Train
  https://github.com/Cytomine-ULiege/S_Segment-ML-UNet-Binary-Pred
- Inputs: ontology terms for foreground annotations, U-Net parameters (nb. epochs, learning rate, weight decay, …)
- Outputs: a U-Net segmentation model (saved in Cytomine database as "AttachedFile")

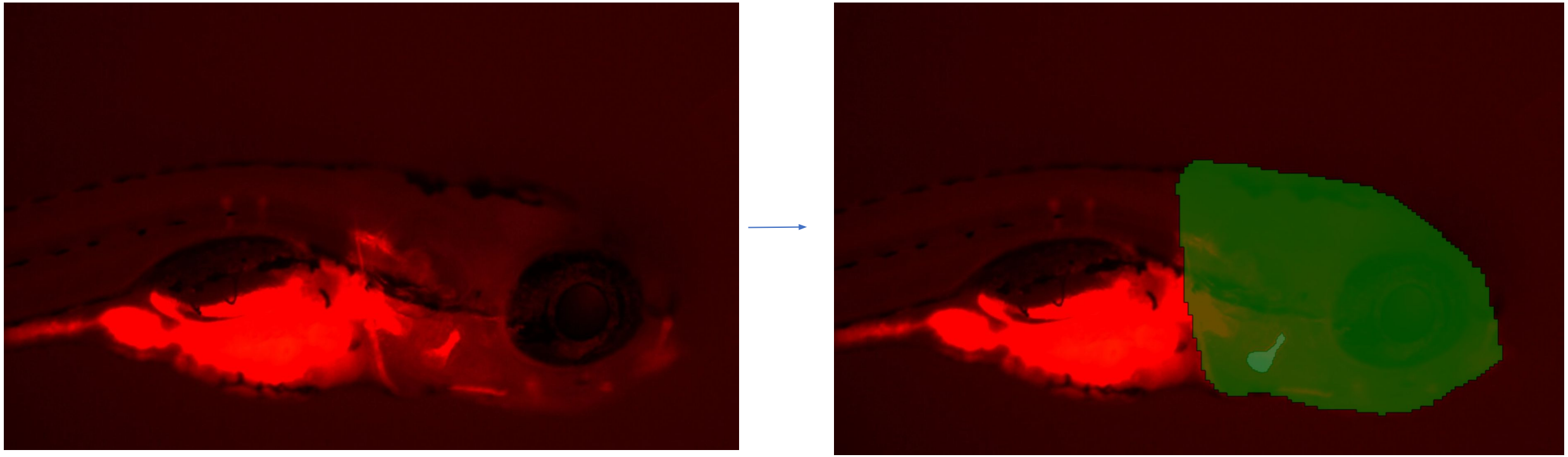




(Ronneberger, et al., ISBI 2015)

Example: Zebrafish Head/Operculum: prediction (pre-trained U-Net)

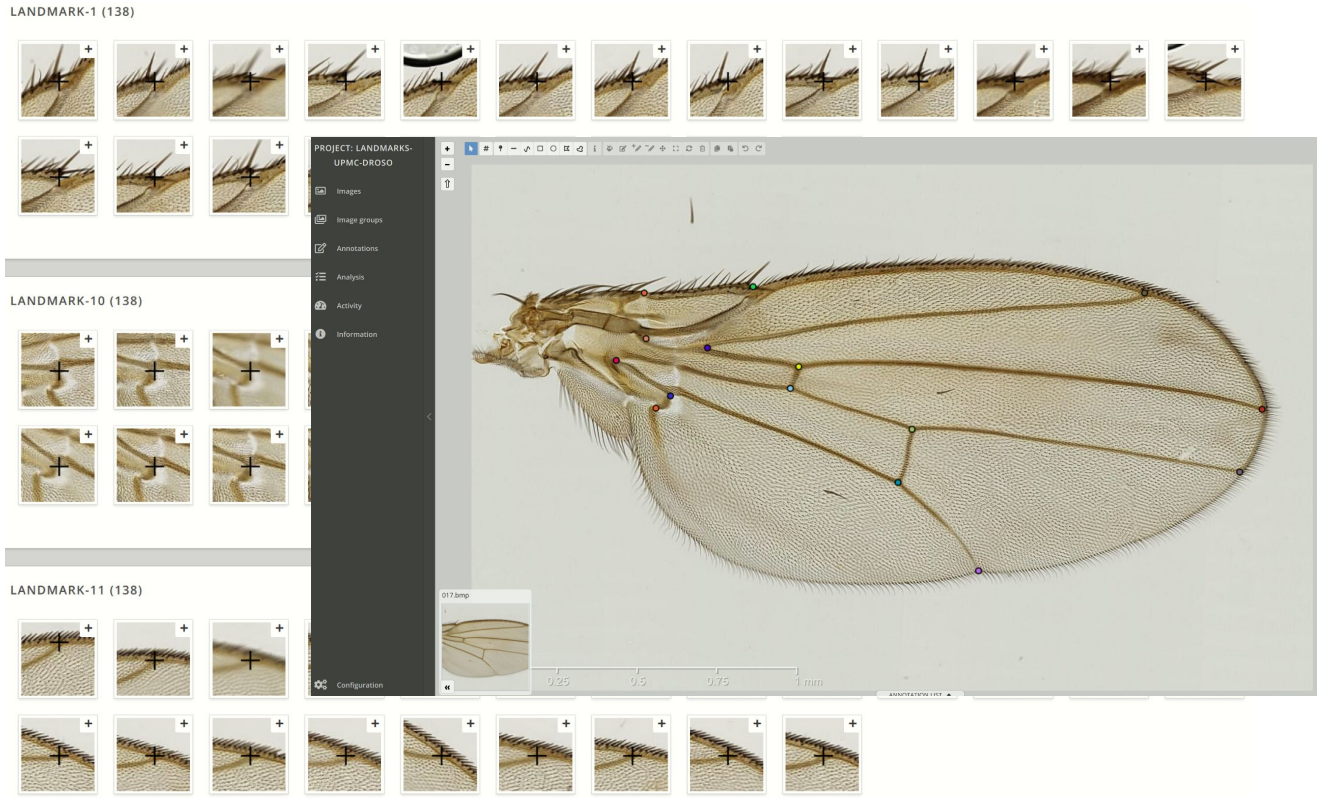
- Code: https://github.com/navdeepkaushish/S_Zebrafish_Head_Operculum_UNet_Segmentation
- Inputs: list of images, ontology terms for head & operculum
- Outputs: in each image, segmented head and operculum regions with corresponding terms

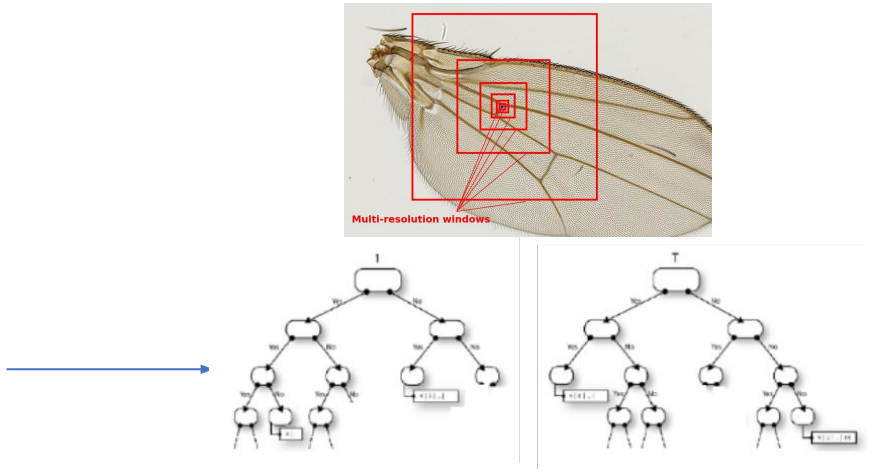


(Kumar et al., Submitted, 2021)

Example: Landmark detection : training (Random forests)

- Code: https://github.com/Cytomine-ULiege/S_LandmarkDetect-ML-MSET-Train
- Inputs: images & ontology term of manual annotations of anatomical landmarks
- Outputs: a random forest based detection model for each landmark
  (saved in Cytomine database as “AttachedFile”)

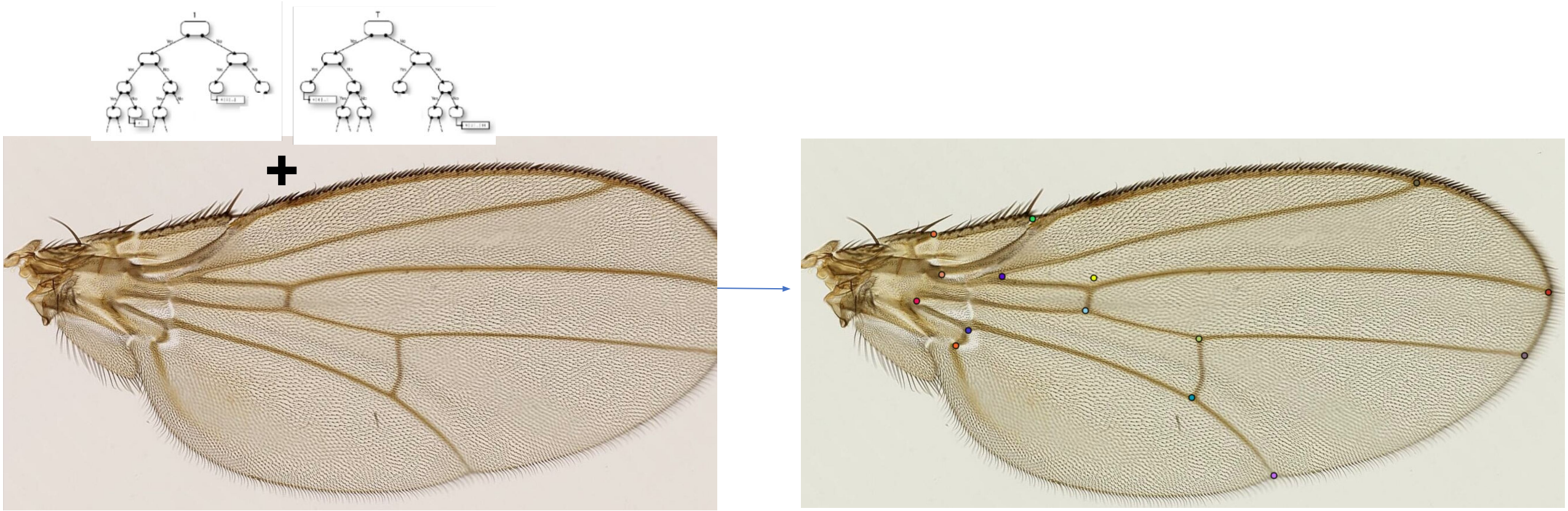




(Vandaele et al., Scientific Reports, 2019)

Example: Landmark detection : prediction (Random forests)

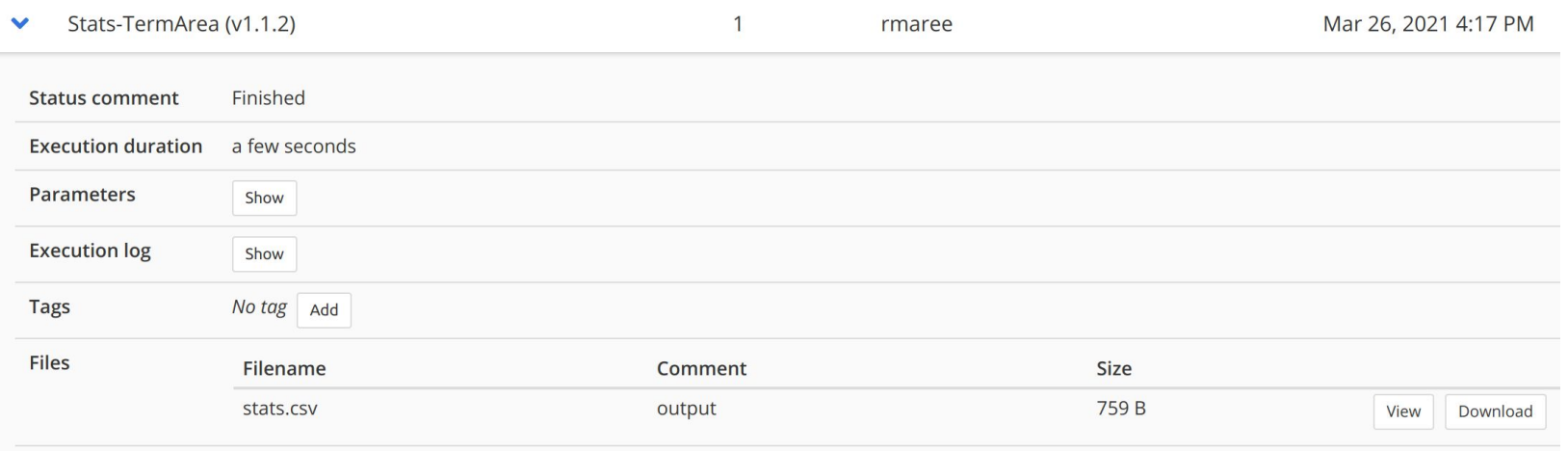
- Code: https://github.com/Cytomine-ULiege/S_LandmarkDetect-ML-MSET-Pred
- Inputs: images & ontology terms of manual annotations of anatomical landmarks
- Outputs: predicted annotation points for landmarks



(Vandaele et al., Scientific Reports, 2019)

Example: Statistics exporter

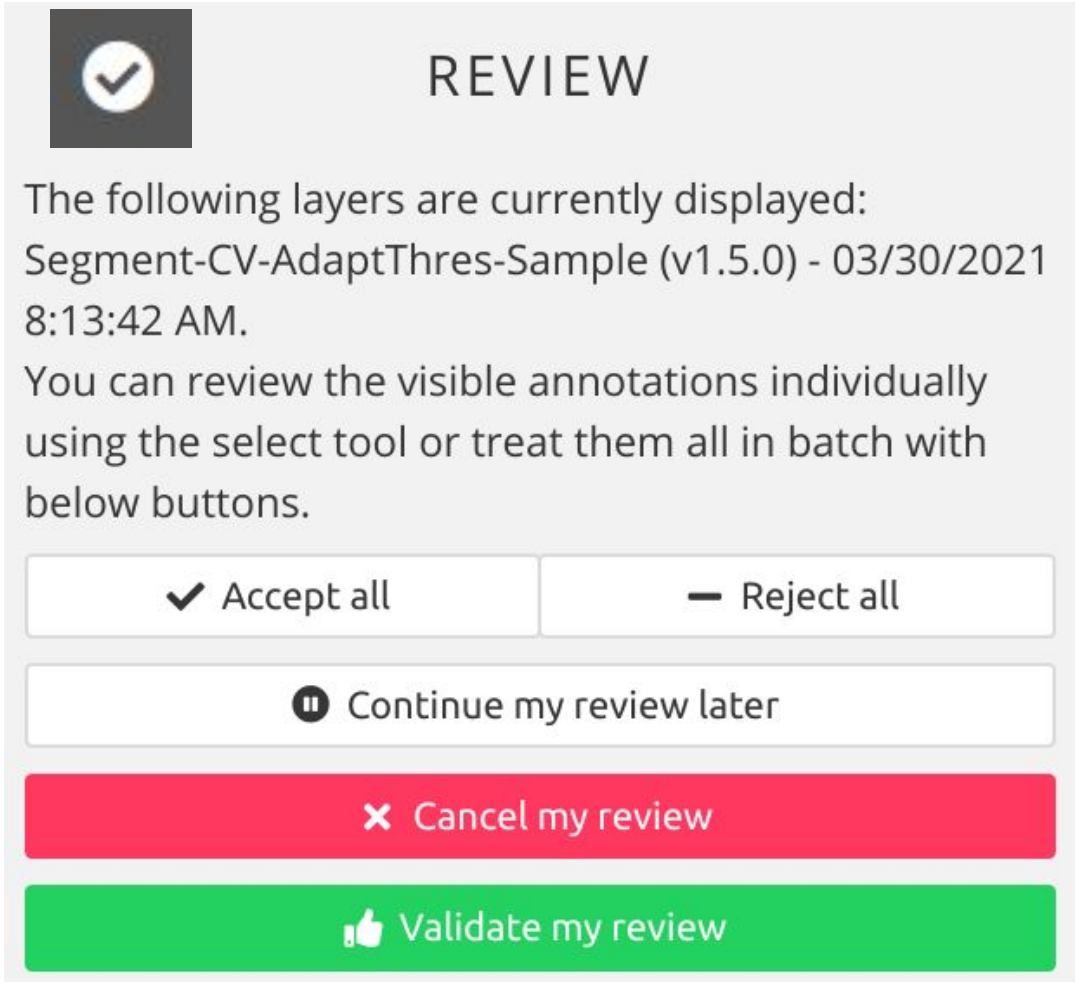
- Code: https://github.com/Cytomine-ULiege/S_Stats-TermArea
- Inputs: List of images, list of ontology terms, user/software ids
- Outputs: Statistics (counts, area, ratio, …)

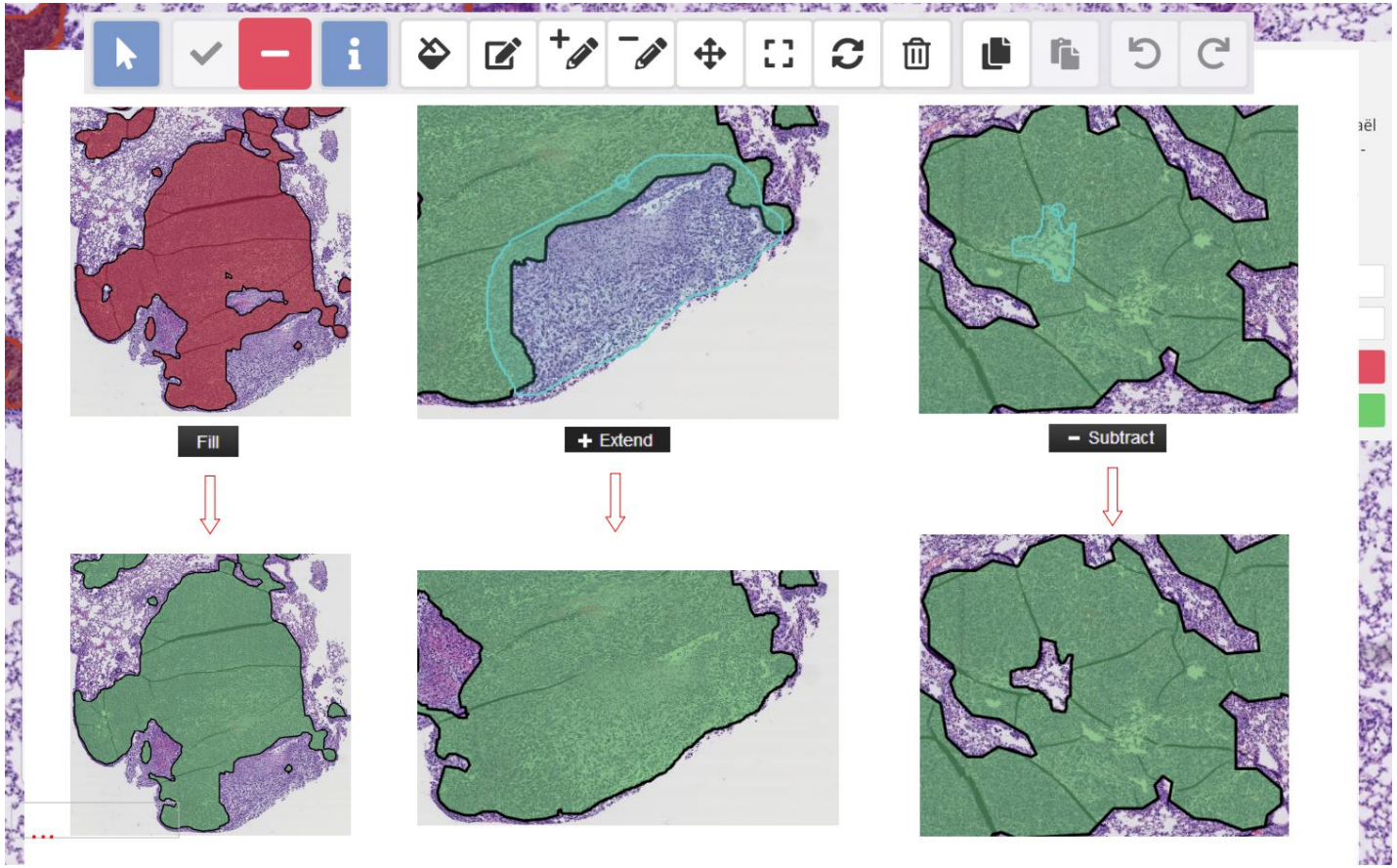


	A	B	C	D
1	Area statistics			
2	Image	Nucleus	ROI	Total
3	CMU-1.tiff	362989	362989	725978
4	Total	362989	362989	725978
5				
6				
7	Number statistics			
8	Image	Nucleus	ROI	Total
9	CMU-1.tiff	4	4	8
10	Total	4	4	8
11				
12				
13	Data ratio			
14	Image	Nucleus	ROI	Total
15	CMU-1.tiff	0.5	0.5	1
16				
17				
18	Details			
19	**********			
20	Image 1			
21	CMU-1.tiff			
22	**********			
23	#####			
24	Nucleus			
25	Created	Area		
26	2021.03.19 10:44:23	27689		
27	2021.03.19 10:43:54	0		
28	2021.03.19 09:50:49	0		
29	2021.03.11 09:57:26	335300		
30				
31	Number of annotations	4		
32	Average area	90747		
33	Total area	362989		
34				
35				
36	#####			
37	ROI			
38	Created	Area		
39	2021.03.19 10:44:23	27689		
40	2021.03.19 10:43:54	0		
41	2021.03.19 09:50:49	0		
42	2021.03.11 09:57:26	335300		
43				
44	Number of annotations	4		
45	Average area	90747		
46	Total area	362989		

Results are not satisfactory ? → Reviewing of algorithms predictions

- Results (e.g. cell detections, tissue delineation) can be reviewed/proofread (original & validated annotations are stored in Cytomine database)

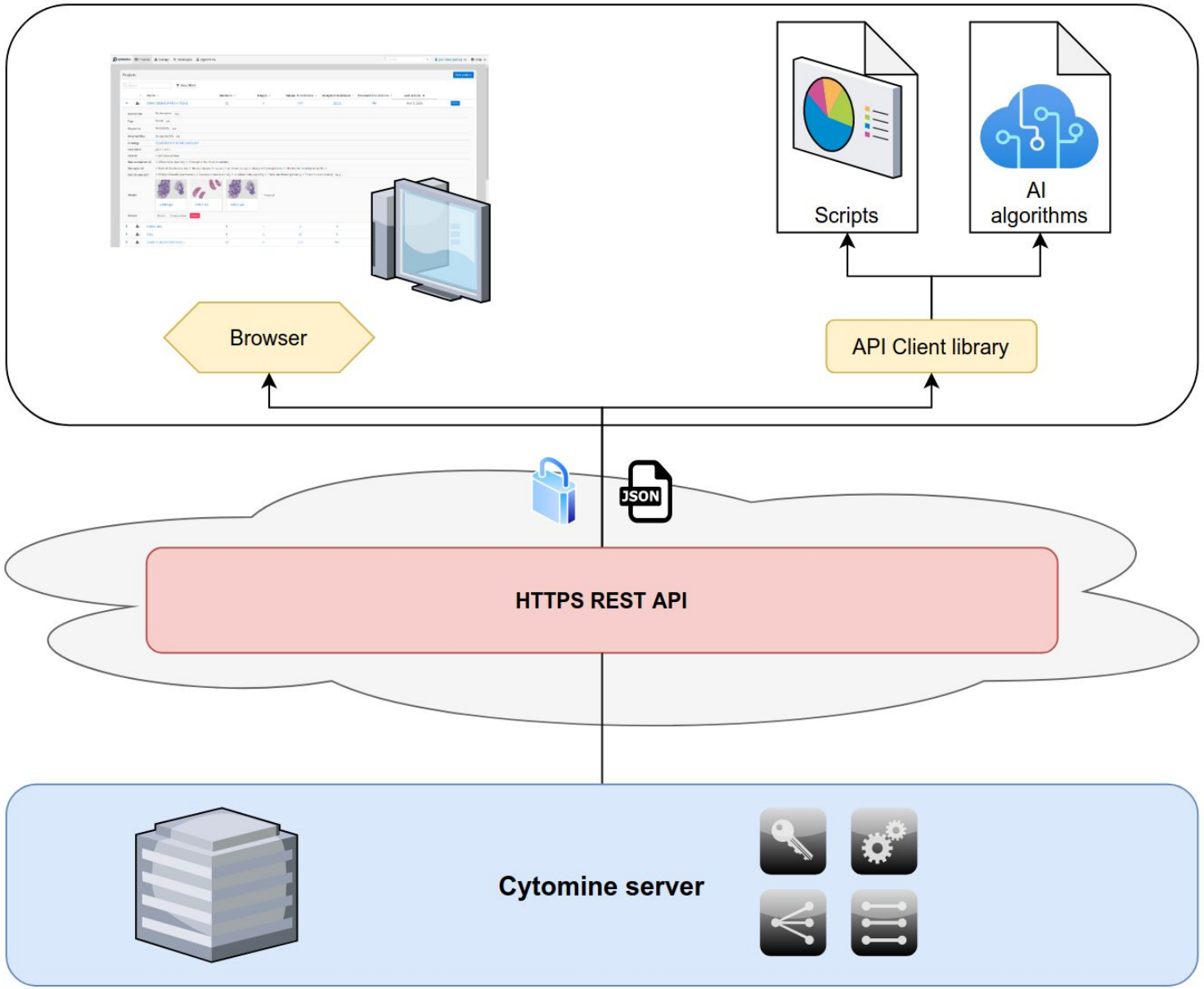




Results are not satisfactory ?

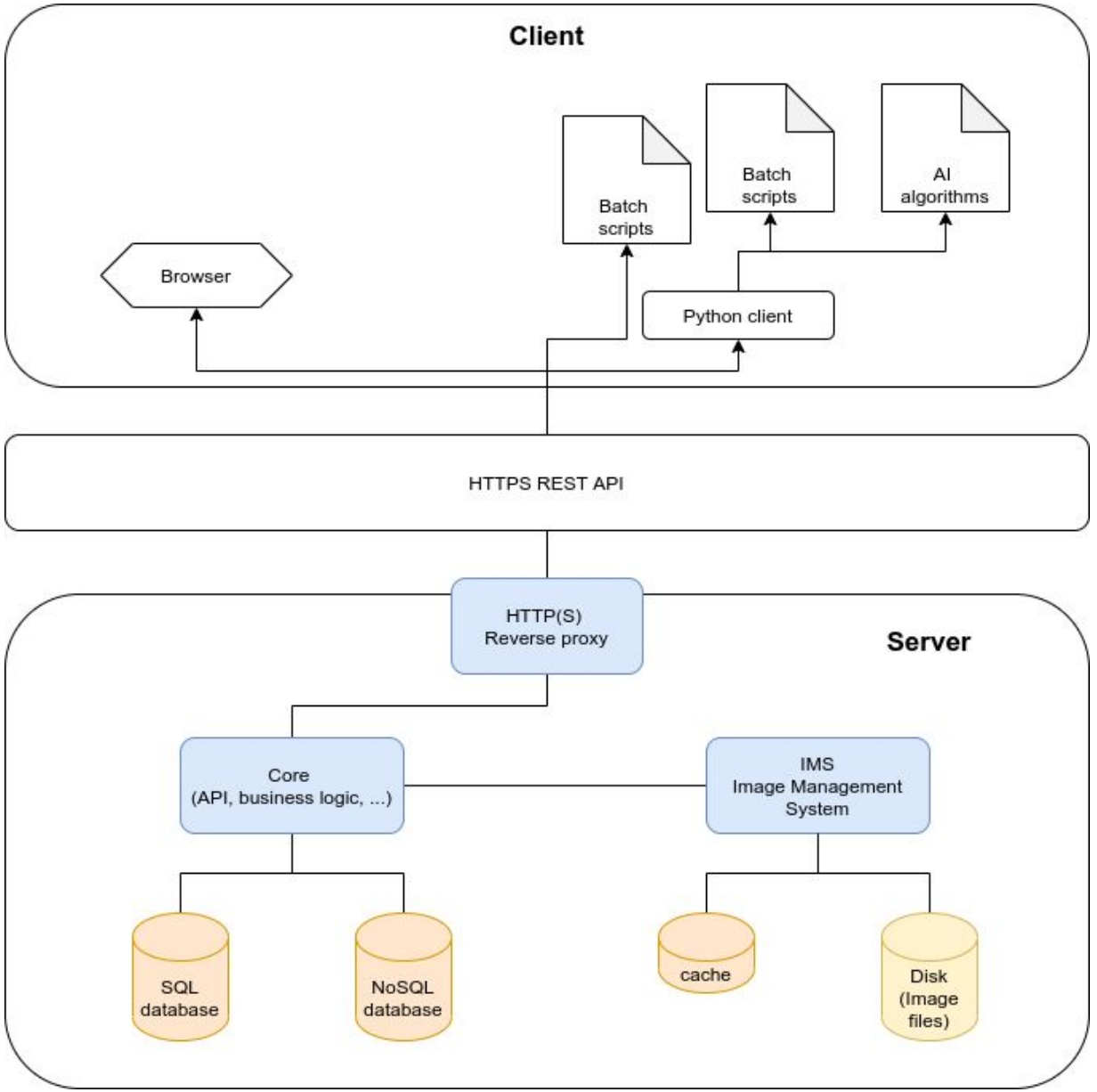
→ **Develop and integrate into Cytomine your own (better) algorithms !**

Beyond the scene: Cytomine architecture (simplified)



The data stored and managed by a Cytomine server can be obtained through specific URLs

Beyond the scene: Cytomine architecture (simplified)

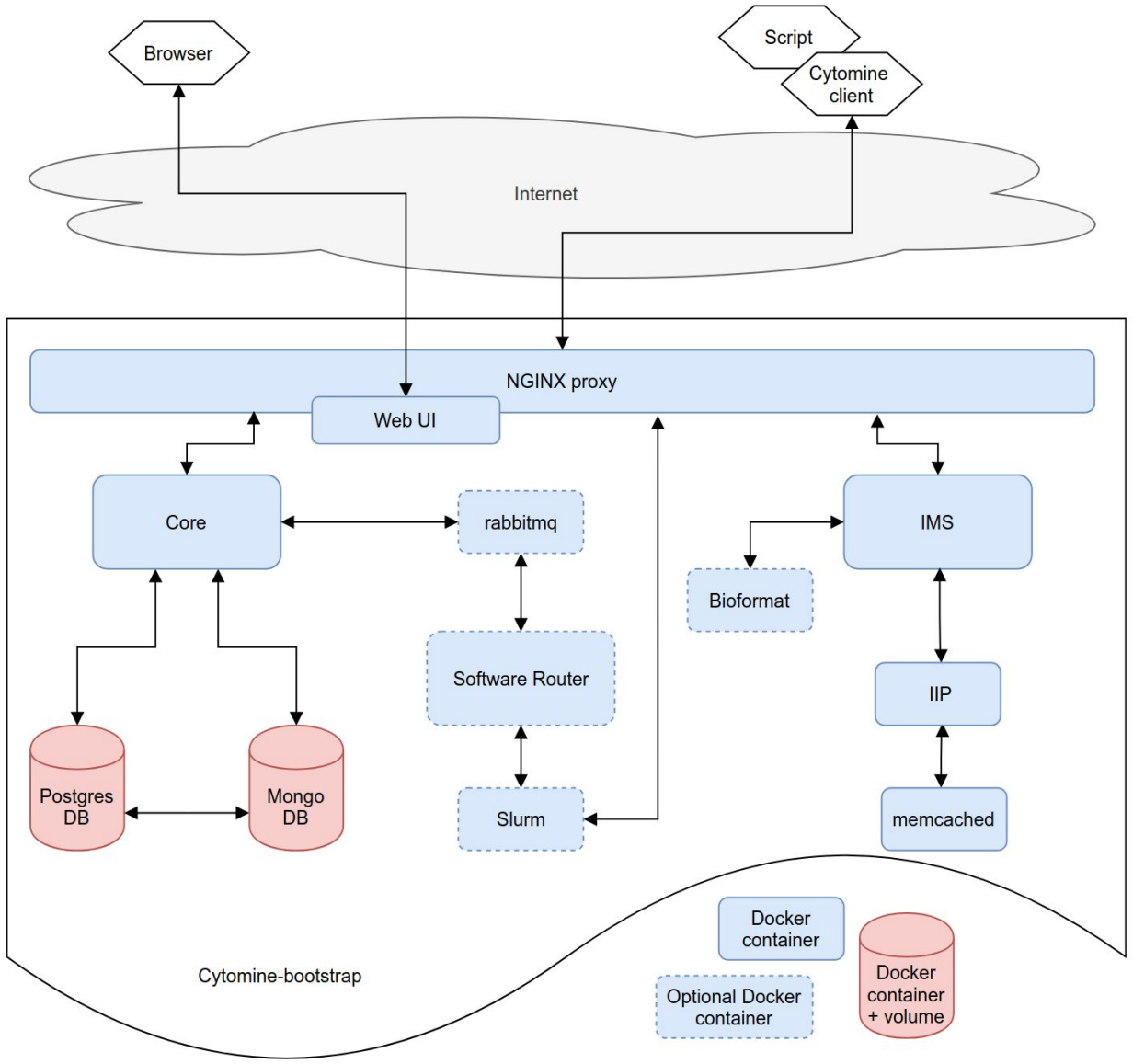


All the endpoints are secured. You need to be authenticated so that the Cytomine server accept to treat your request.

Data sources:

- SQL & NoSQL Databases (Docker volumes)
- Image files (Filesystem)

Beyond the scene: Cytomine architecture & libraries



Leveraging **Docker** technology, you can **install** the whole Cytomine platform on a personal laptop (isolated so without collaborative features), on a virtual machine or a dedicated server in your lab or at large scale (e.g. core facility or MOOC for thousands of users), by hosting containers on different machines to increase performance and stability (horizontal scaling).

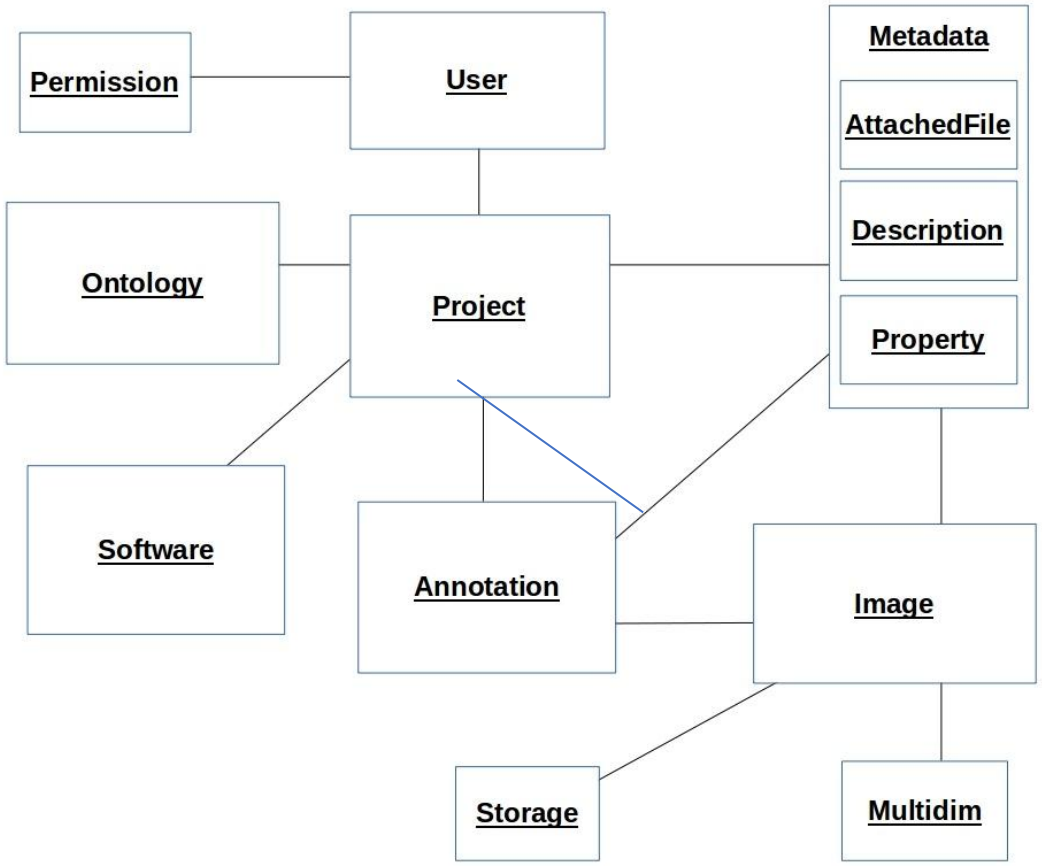
https://doc.cytomine.org/admin-guide/

"Toy" demo server:
https://demo.cytomine.be
username: jsnow
password: jsnow

To request a specific demo server access: info@cytomine.org

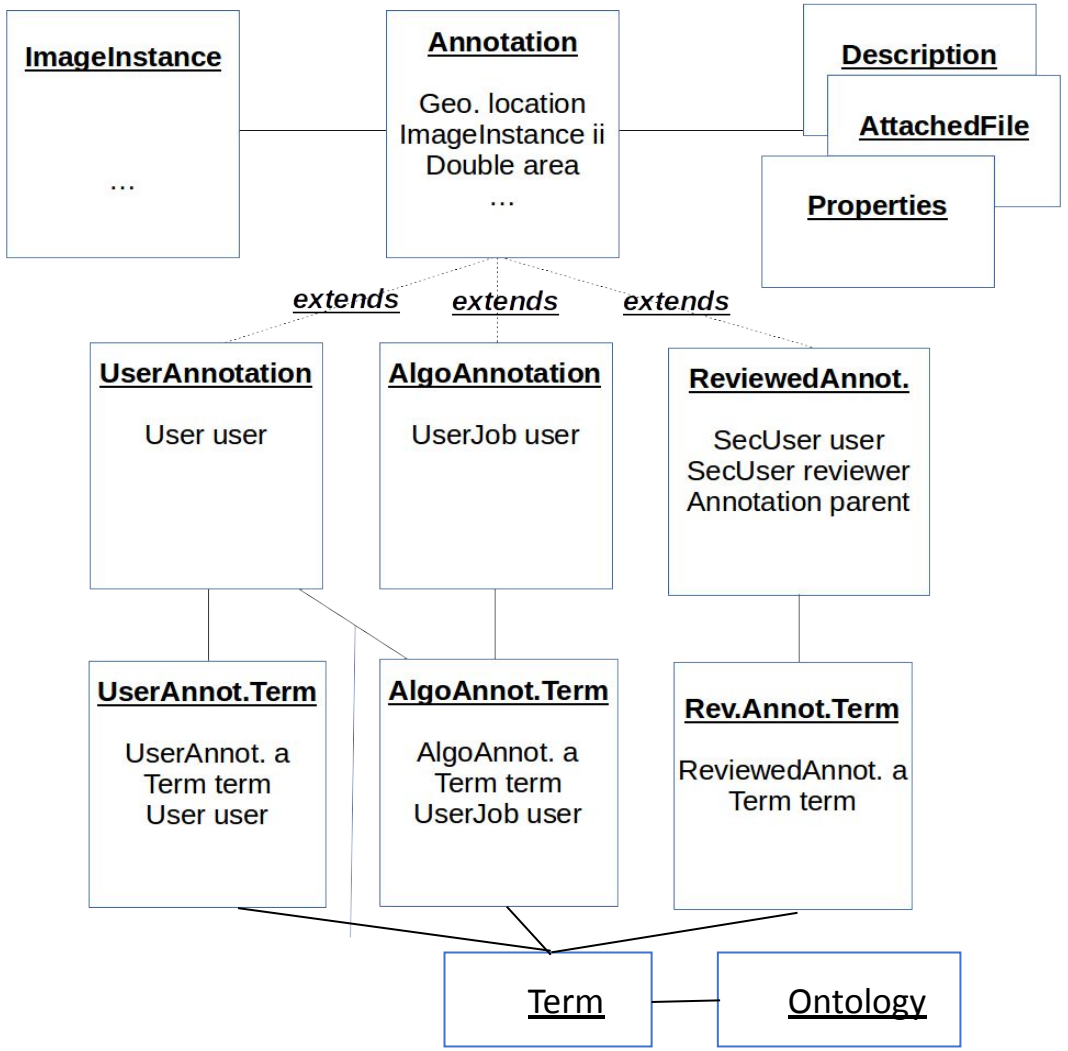
Beyond the scene: Cytomine data models (simplified)

- User: Me and You and Everyone We Know
- Permission:
  - Manage access rights
- Project:
  - Shared area between users
  - Group images & annotations
- Ontology:
  - Hierarchical list of terms to be associated to annotations
- Annotation:
  - Geometry + term within an image
- Image:
  - Reference to underlying image file (2D+c+z+t)
- Storage:
  - Virtual disk space
- Metadata:
  - AttachedFile: file attached as a metadata
  - Description: rich text description
  - Property: key-value pairs
  - Tags
- Software:
  - Runnable tools (e.g AI algorithm) from Cytomine



Beyond the scene: Cytomine annotation data model (simplified)

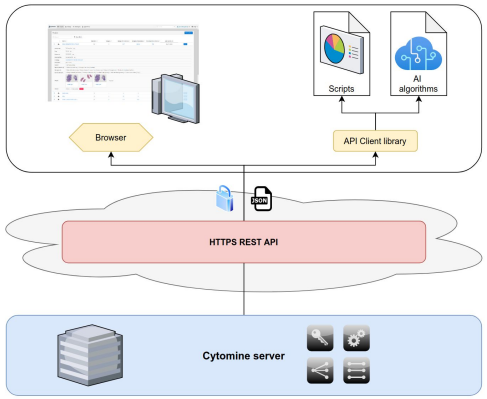
- 3 types
  - UserAnnotation: done by a human user
    - 1 annot layer per user in an image
  - AlgoAnnotation: done by a software execution (UserJob)
    - 1 annot layer per execution in an image
  - ReviewedAnnotation:
    - User or algo annotations
    - Reviewed by an “expert”
    - 0/1 review layer in an image
- Geometry format: WKT
  https://en.wikipedia.org/wiki/Well-known_text_representation_of_geometry
- Stored in a spatial database (PostGIS)



\+ AnnotationLinks to group annotations depicting same regions in multiple modalities: Under development (Cytomine ULiège R&D)

Cytomine Interoperability: HTTPS REST API

- API endpoints
  - Always require authentication
  - JSON



- Example: Project resource
  - List projects I have access to
    - GET https://demo.cytomine.be/api/project.json
    - Support filters like minImages, minAnnots, …
  - Get a specific project (if I have rights)
    - GET https://demo.cytomine.be/api/project/528050.json
  - Update project info (if I have rights)
    - PUT https://demo.cytomine.be/api/project/80741.json + JSON body
  - Delete project (if I have rights)
    - DELETE https://demo.cytomine.be/api/project/80741.json
  - Add new project (if I have rights)
    - POST https://demo.cytomine.be/api/project.json + JSON body

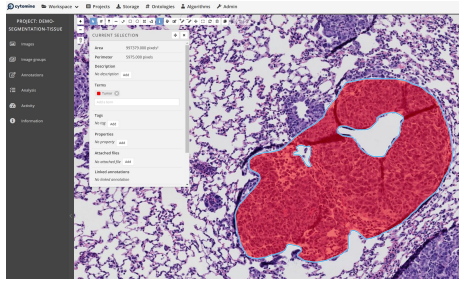

```
{
    class: "be.cytomine.project.Project",
    id: 80741,
    created: "1615273192548",
    updated: "1617106983480",
    deleted: null,
    name: "|           DEMO-VARIOUS",
    ontology: 80733,
    ontologyName: "           DEMO-VARIOUS",
    discipline: null,
    blindMode: false,
    areImagesDownloadable: false,
    disciplineName: null,
    numberOfSlides: 0,
    numberOfImages: 8,
    numberOfAnnotations: 46,
    numberOfJobAnnotations: 4108,
    retrievalProjects: [ ],
    numberOfReviewedAnnotations: 2,
    retrievalDisable: false,
    retrievalAllOntology: true,
    isClosed: false,
    isReadOnly: false,
    isRestricted: false,
    hideUsersLayers: false,
    hideAdminsLayers: false
}
```


- Same idea for other Cytomine resources: images, annotations, ontologies, users, …
- **Anything that can be done from Cytomine-WebUI can be done using the HTTPS REST API … and even more than that!**

Annotations in Cytomine REST API

- List annotations based on filters (image, user, ontology term,…): https://demo.cytomine.be/api/annotation/search.json?image=528132&user=701&term=528044

- Get a specific annotation: https://demo.cytomine.be/api/annotation/143584607.json

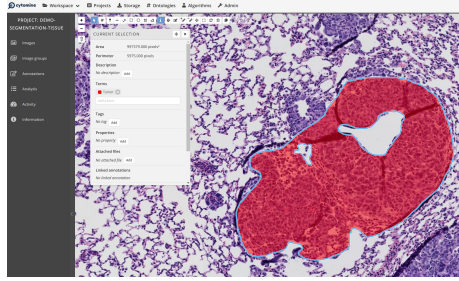


Annotation Geometry in WKT format

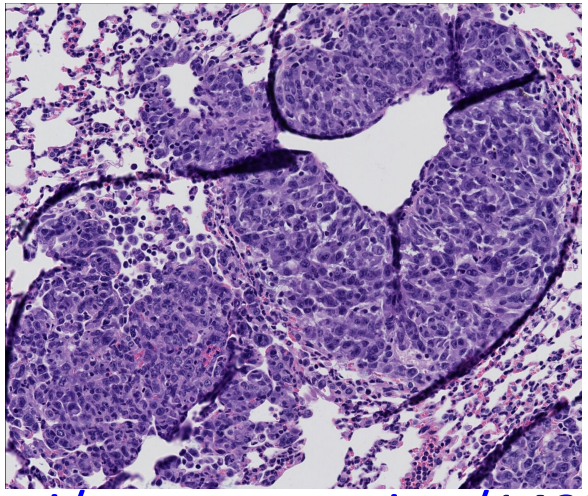
There are WKT parsers to convert WKT from and to geometry objects in several languages (e.g. Python: Shapely)


```
{
  "class": "be.cytomine.ontology.UserAnnotation",
  "id": 143584607,
  "created": "1575556669131",
  "updated": "1575556707121",
  "deleted": null,
  "slice": 146632788,
  "image": 528132,
  "project": 528050,
  "user": 701,
  "location": "POLYGON ((23569.544251797495 12583.846593241735, 23760.890496609507 12583.846593241735, 23827.23647439271 12572.308144907995, 23871.46743142396 12551.154313426061, 23916.6596921
23966.659615872202 12500.192742686315, 24008.96727883607 12473.269658427282, 24028.198159573374 12451.154294352575, 24058.96743142396 12389.615807871862, 24076.275018093882 12336.731229167028,
12247.308125834508, 24120.505746243296 12185.769639353795, 24128.198006985484 12107.885041575475, 24128.198006985484 12080.9620145369, 24123.390344021616 12058.846650462194, 24098.390496609507
24049.35187661439 11901.15416083817, 24024.35202920228 11865.577226023717, 23928.19808327943 11784.808144907995, 23892.621148464976 11743.4620145369, 23863.774941799937 11722.308183054967, 238
23806.082757351694 11663.654351573034, 23720.505898831187 11615.577264417069, 23679.159539578257 11598.26956305985, 23666.659615872202 11587.692704539342, 23613.77486550599 11569.423356394811,
11608.846536021276, 23502.236512539683 11615.577264417069, 23482.044213650523 11616.538911204381, 23464.736512539683 11581.923394541784, 23462.813447354136 11512.692590098424, 23460.89038216859
23413.77482735902 11462.69266639237, 23322.42877328187 11426.154313426061, 23268.58271920472 11415.577340464635, 23237.813447354136 11415.577340464635, 23180.121148464976 11425.19266639237, 23
11455.961938242956, 23137.81348550111 11465.57726417069, 23160.89038216859 11486.731210093541, 23161.8518003204444 11499.231248240514, 23158.00578439027 11505.961976389928, 23147.428696987925
23071.46727883607 11523.26956305985, 23059.928658840952 11493.461938242956, 23053.19793069154 11487.692628245397, 23022.428658840952 11474.231171946569, 22989.736550686655 11463.654313426061,
11448.26956305985, 22891.659539578257 11439.615884165807, 22858.00578439027 11439.615884165807, 22822.428849575816 11452.115807871862, 22810.890344021616 11464.615846018834, 22785.89038216859
22766.659615872202 11567.500291209264, 22757.04417550355 11588.654351573034, 22746.467316983042 11635.76956305985, 22738.774941799937 11713.654275279088, 22745.50578439027 11742.50036750321,
11797.308183054967, 22821.467316983042 11848.269524912877, 22853.198006985484 11880.000329356237, 22885.890344021616 11923.269753794713, 22899.351914761362 11948.26971564774, 22893.58260476380
22842.621148464976 12000.192704539342, 22831.082642910776 12010.769677500768, 22826.274979994691 12046.346612315221, 22831.082642910776 12067.500443797155, 22849.351876614439 12087.692742686315,
12113.654294352575, 22969.544251797495 12117.500481944127, 23023.390305874644 12130.000462870641, 23040.697892544566 12140.577321391149, 23090.69793069154 12181.923508982702, 23116.659539578257
23250.31348550111 12298.269639353795, 23303.198006985484 12341.538834910436, 23325.31348550111 12391.538930277868, 23360.89042031556 12437.692723612829, 23382.044251797495 12456.923547129674,
12557.885041575475, 23569.544251797495 12583.846593241735), (23646.46743142396 12335.769639353795, 23644.544137356577 12311.731210093541, 23638.774941799937 12304.038834910436, 23604.159654019
23570.50566994935 12258.846574168248, 23523.390344021616 12257.885041575475, 23504.15957772523 12260.769639353795, 23453.1981214264 12280.961995463414, 23422.428735134898 12282.885060648961,
12274.231210093541, 23391.659577772523 12256.923508982702, 23392.621110318003 12240.577340466635, 23425.313447354136 12221.346574168248, 23490.6981214264 12210.769658427282, 23521.46727883607
23541.65957772523 12138.654256205602, 23559.928696987925 12121.346555094762, 23583.967355130015 12105.962033610387, 23610.89038216859 12092.5005200911, 23630.121148464976 12087.692742686315,
12114.615826945348, 23704.159615872202 12121.346555094762, 23733.967355130015 12154.038834910436, 23747.428811428843 12179.038853983922, 23783.005746243296 12219.423508982702, 23792.6210721710
23800.313561795054 12301.154294352575, 23808.967355130015 12313.654275279088, 23812.813485550111 12326.154256205602, 23811.851952908335 12340.577302317663, 23789.73647439271 12355.961995463414,
12369.423508982702, 23708.005746243296 12363.654256205602, 23675.313409207163 12354.03881583695, 23646.46743142396 12335.769639353795), (23193.58271920472 12149.231171946569, 23163.77486550599
23138.774903652964 12153.077359538122, 23133.005708096323 12149.231171946569, 23132.04417550355 12143.462033610387, 23138.774903652964 12133.846593241735, 23140.69796883851 12120.385022501989,
12101.154313426061, 23114.73647439271 12098.269601206823, 23149.35176217347 12098.269601206823, 23159.928735134898 12092.5005200911, 23191.659654019175 12037.692704539342, 23205.121110318003
23220.505746243296 12027.115731577916, 23220.505746243296 12037.692704539342, 23207.04417550355 12060.76971564774, 23206.082642910776 12069.423508982702, 23218.58268105775 12096.346536021276,
12152.115826945348, 23211.851838467417 12154.038834910436, 23193.58271920472 12149.231171946569))",
  "geometryCompression": 2.25,
  "centroid": {
    "x": 23452.105097064356,
    "y": 11971.324686548254
  },
  "area": 997379.0,
  "areaUnit": "pixels²",
  "perimeter": 5975.0,
  "perimeterUnit": "pixels",
  "term": [
    528044
  ],
  "nbComments": 0,
  "cropURL": "https://research.cytomine.be/api/userannotation/143584607/crop.png",
  "smallCropURL": "https://research.cytomine.be/api/userannotation/143584607/crop.png?maxSize=256",
  "url": "https://research.cytomine.be/api/userannotation/143584607/crop.png",
  "imageURL": "https://research.cytomine.be/#/project/528050/image/528132/annotation/143584607",
  "reviewed": false
}
```

Annotations in Cytomine REST API

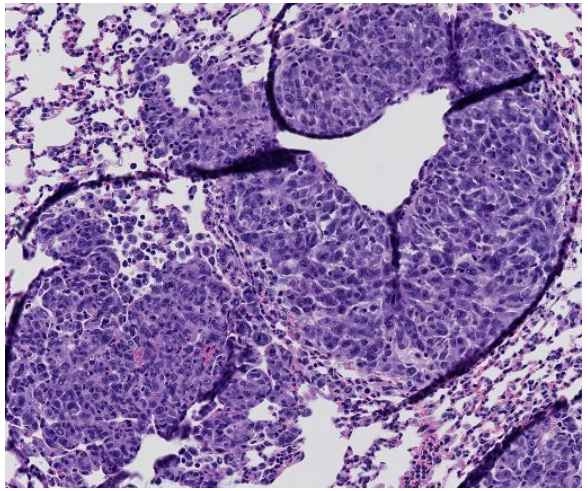


https://demo.cytomine.be/api/userannotation/143584607/crop.png



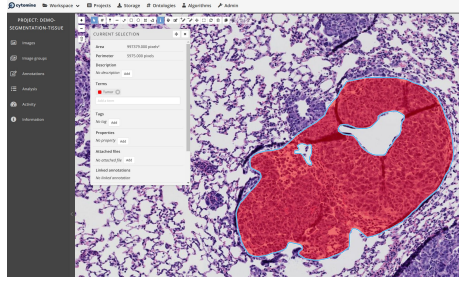
(5556 x 4672 pixels)

https://demo.cytomine.be/api/userannotation/143584607/crop.png?maxSize=512

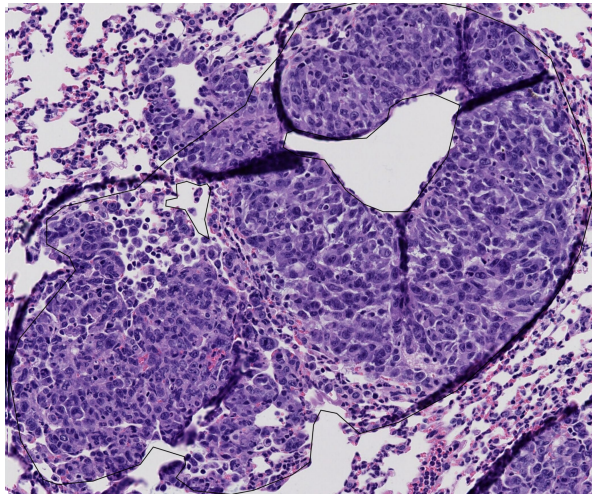


(512 x 430 pixels)

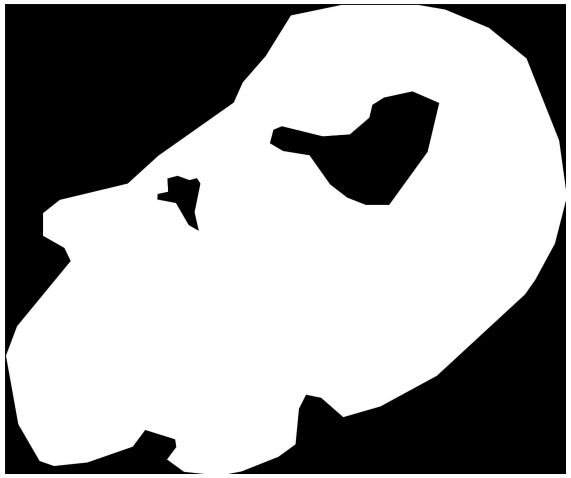
Annotations in Cytomine REST API



https://demo.cytomine.be/api/userannotation/143584607/crop.png&draw=true

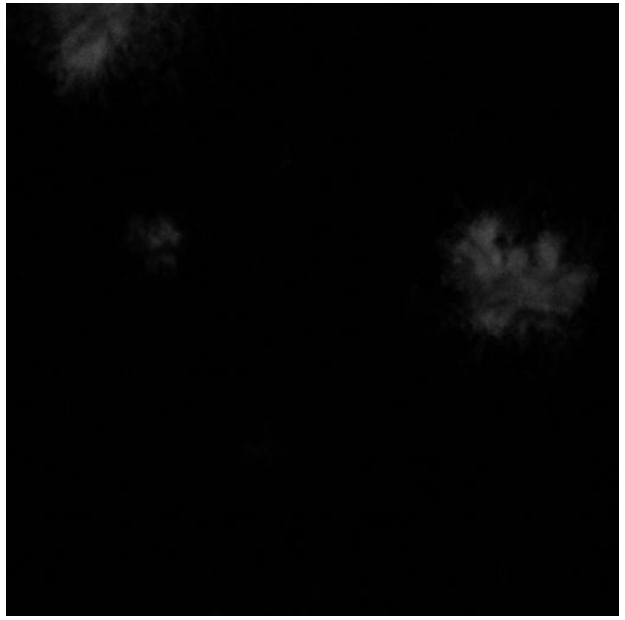


https://demo.cytomine.be/api/userannotation/143584607/crop.png?maxSize=512&mask=true

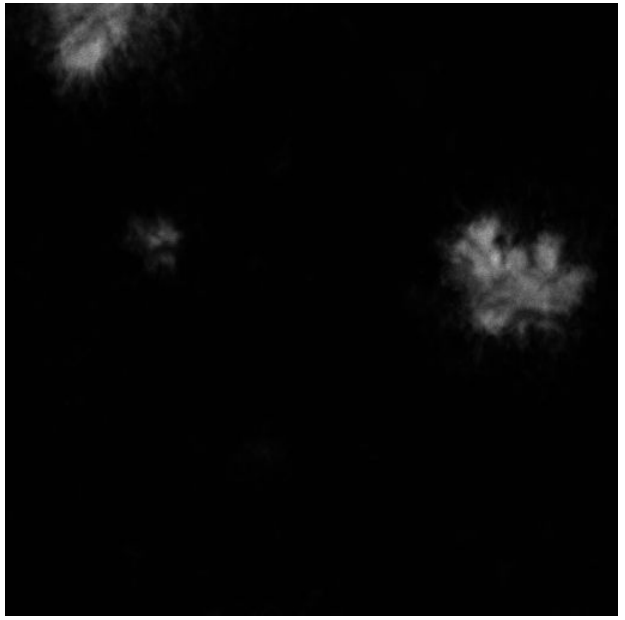


Other API examples: get a tile (window)

https://demo.cytomine.be/api/imageinstance/153042325/window-3000-3000-512-512.png?projection=max



raw sub-image (window)



maximum projection of the window

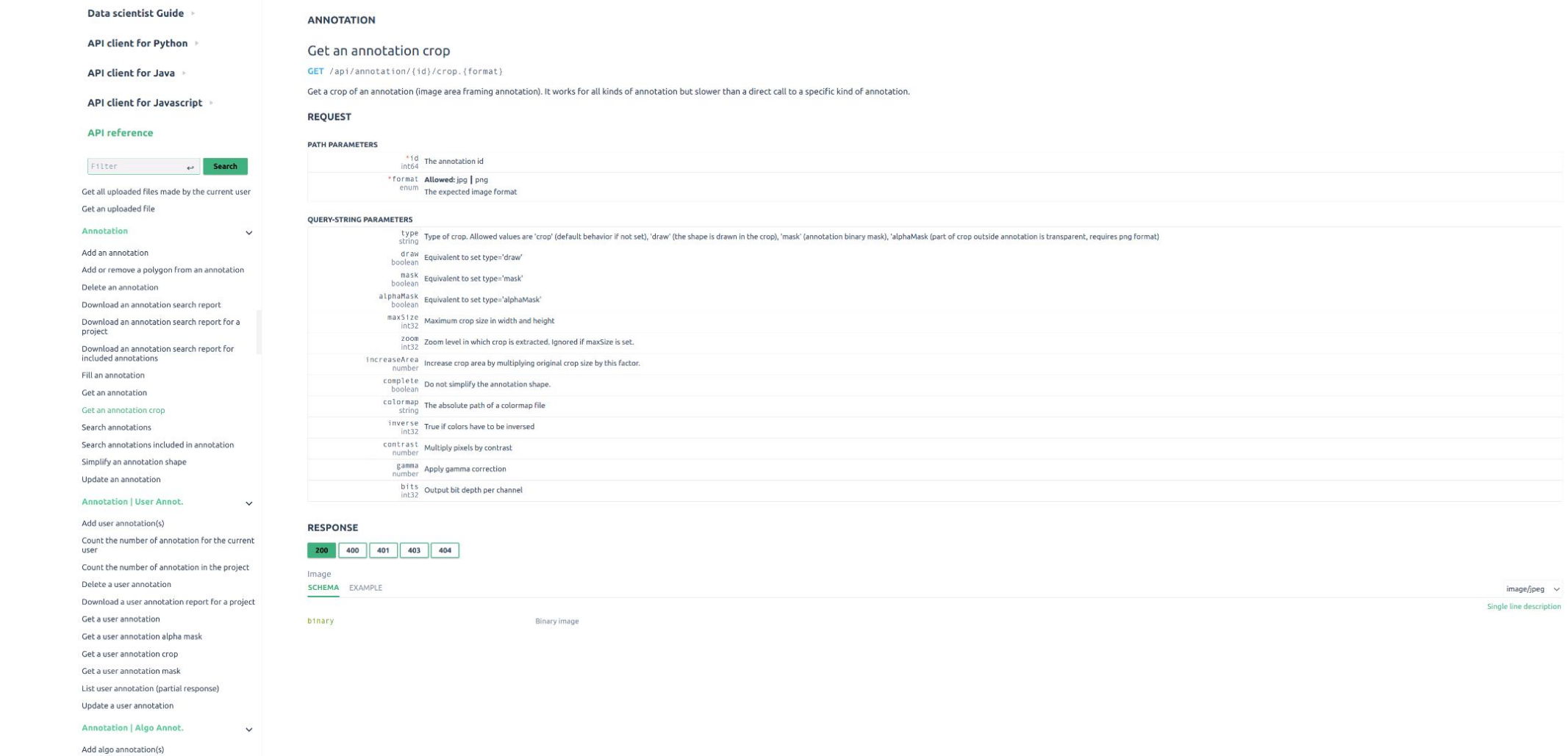
(Support of hyperspectral images is under development - Cytomine ULiège R&D Edition)

Important note: In Cytomine coordinate system (0,0) is bottom left corner
(x,y) → (x,image.height-y)

Complete Cytomine HTTPS REST API Reference

- https://doc.uliege.cytomine.org/dev-guide/api/reference

  In-house specification schema (move to OpenAPI ongoing)
  Still under validation (soon on doc.cytomine.org)



Cytomine API clients (Python/Java/Javascript)

- **Goal**: **to manipulate Cytomine resources as regular objects within your favorite language**
  - Clients wrap the HTTP API endpoints:
    - **Python**: https://github.com/cytomine/Cytomine-python-client
    - **Java**: https://github.com/cytomine/Cytomine-java-client
    - **JavaScript**: https://github.com/cytomine/Cytomine-js-client

- "Project" example with Python client:
  - List projects I have access to
    - projects = ProjectCollection().fetch()
  - Get a specific project (if I have rights)
    - project = Project().fetch(80741)
  - Update project info (if I have rights)
    - project.name = "My new name"
    - project.update()
  - Delete project (if I have rights)
    - project.delete()
  - Add new project (if I have rights)
    - other_project = Project(name="My new project").save()

Cytomine Python API client

- https://doc.cytomine.org/dev-guide/clients/python/usage#models

Models

A Cytomine **resource** (such as a project *STUDY-01* or an image *cell01.tiff*) is an instance of a **domain** (in the examples: a project, an image).

In the Cytomine API client for Python, **a resource is a Python object** which is an instance of **a model class** describing its domain. Each object (thus an instance of a model):

- has a set of attributes corresponding to resource attributes
- is managed itself through `fetch()`, `save()`, `update()` and `delete()` methods that communicate with the Cytomine server
- has some utilities to manage HTTP and JSON technical details

Examples of models are `Project`, `ImageInstance`, `Annotation`, `Term`, `User`, `CurrentUser`, etc.

Fetch a resource

To get data from server, we first need to fetch the resource. Indeed, when you create a new object (such as `myproject = Project()`), the object (here `myproject`) is empty. You need to populate this object with data coming from Cytomine.

> **TIP**
>
> The next sections show examples to fetch either a project or an image. The sections are very similar: the API client library is indeed very systematic.

Example: fetch a project

Supposing the Cytomine server has a project called *STUDY-01* with an identifier (ID) 42, the project resource is fetched in Python with:

```py
from cytomine.models import Project

# ... Assume we are connected

myproject = Project().fetch(id=42)

# Will print STUDY-01
print(myproject.name)
```

Cytomine Java API client

- https://doc.cytomine.org/dev-guide/clients/java/usage#models

Models

A Cytomine **resource** (such as a project *STUDY-01* or an image *cell01.tiff*) is an instance of a **domain** (in the examples: a project, an image).

In the Cytomine API client for Java, **a resource is a Java object** which is an instance of **a model class** describing its domain. Each object (thus an instance of a model):

- has a set of attributes corresponding to resource attributes
- is managed itself through `fetch()`, `save()`, `update()` and `delete()` methods that communicate with the Cytomine server
- has some utilities to manage HTTP and JSON technical details

Examples of models are `Project`, `ImageInstance`, `Annotation`, `Term`, `User`, `CurrentUser`, etc.

Fetch a resource

To get data from server, we first need to fetch the resource. Indeed, when you create a new object (such as `Project myproject = new Project()`), the object (here `myproject`) is empty. You need to populate this object with data coming from Cytomine.

> **TIP**
>
> The next sections show examples to fetch either a project or an image. The sections are very similar: the API client library is indeed very systematic.

Example: fetch a project

Supposing the Cytomine server has a project called *STUDY-01* with an identifier (ID) 42, the project resource is fetched in Java with:

Java | Groovy

```java
import be.cytomine.client.models.Project;

// ...
// Assume we are connected
// ...

Long projectId = 42L;
Project myProject = new Project().fetch(projectId);

// Will print STUDY-01
String name = myProject.getStr("name");
System.out.println(name);
```

Behind the scene, the API client library has sent a `GET /api/project/42.json` request to the Cytomine server and populated `myproject` with the response content.

Only the project meta information have been retrieved (such as name, ontology identifier, settings, ...). You need to perform other `fetch()` calls on other models to get the project images, annotations, project members, etc.See the API reference for all details.

Cytomine Java API client: example

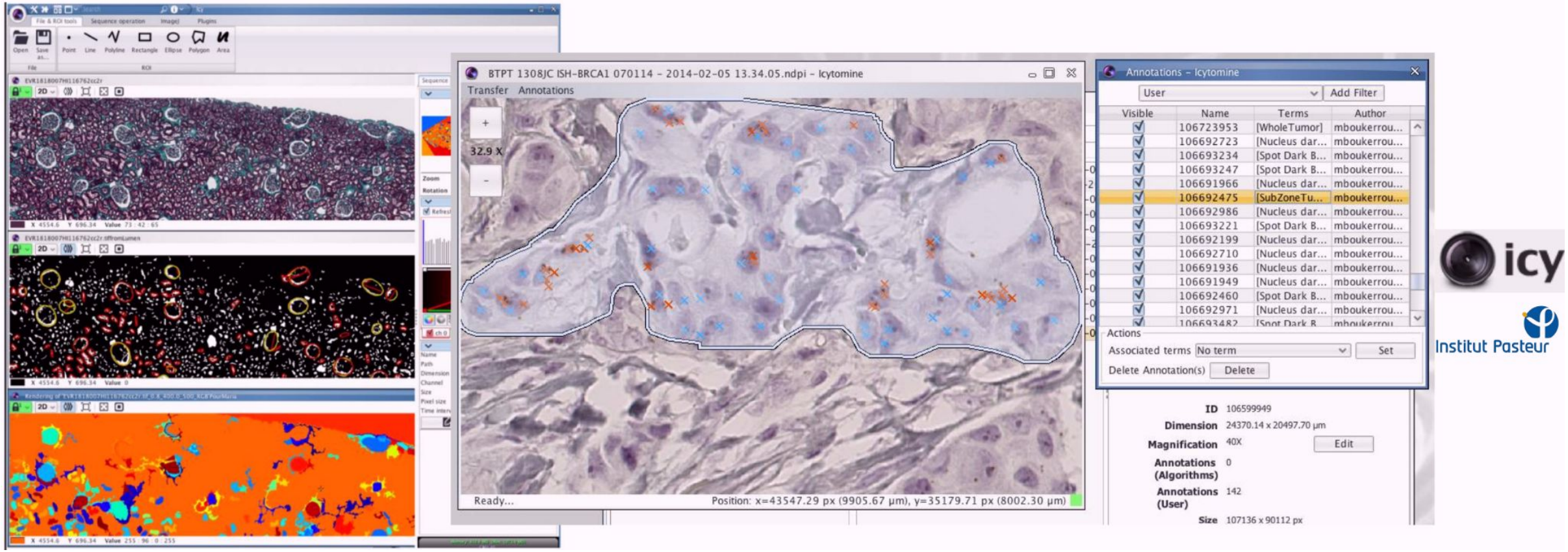
- Java client examples:
  https://github.com/cytomine/Cytomine-java-client/tree/master/src/test/java/client

```
Cytomine cytomine = new Cytomine(cytomineCoreUrl, publickey, privatekey, "./");


println "List the images of your project:"
ImageInstanceCollection images = cytomine.getImageInstances(proj.getId());
for(int i=0;i<images.size();i++) {
    ImageInstance image = images.get(i)
    println image.getId()
}


println "List the annotations of your project:"
AnnotationCollection annotations = cytomine.getAnnotationsByProject(proj.getId());
for(int i=0;i<annotations.size();i++) {
    Annotation annotation = annotations.get(i)
    List termNameList = annotation.getList("term").collect{ termId ->
        return terms.list.find{it.id==termId}.get("name")
    }
    println annotation.getId() + " " + annotation.get("location") + " " + annotation.get("image") + " " + annotation.get("name")
}
```

Cytomine Java API client: e.g. access Cytomine data from another tool

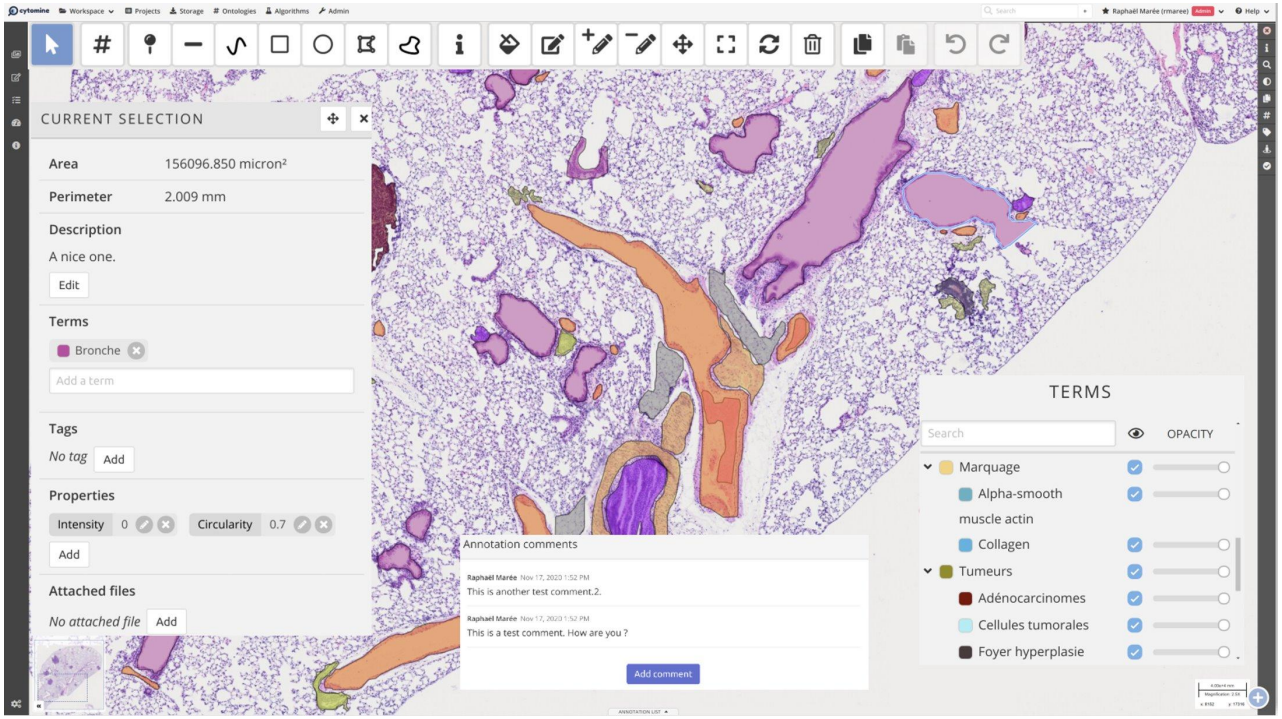


https://github.com/danyfel80/icytomine

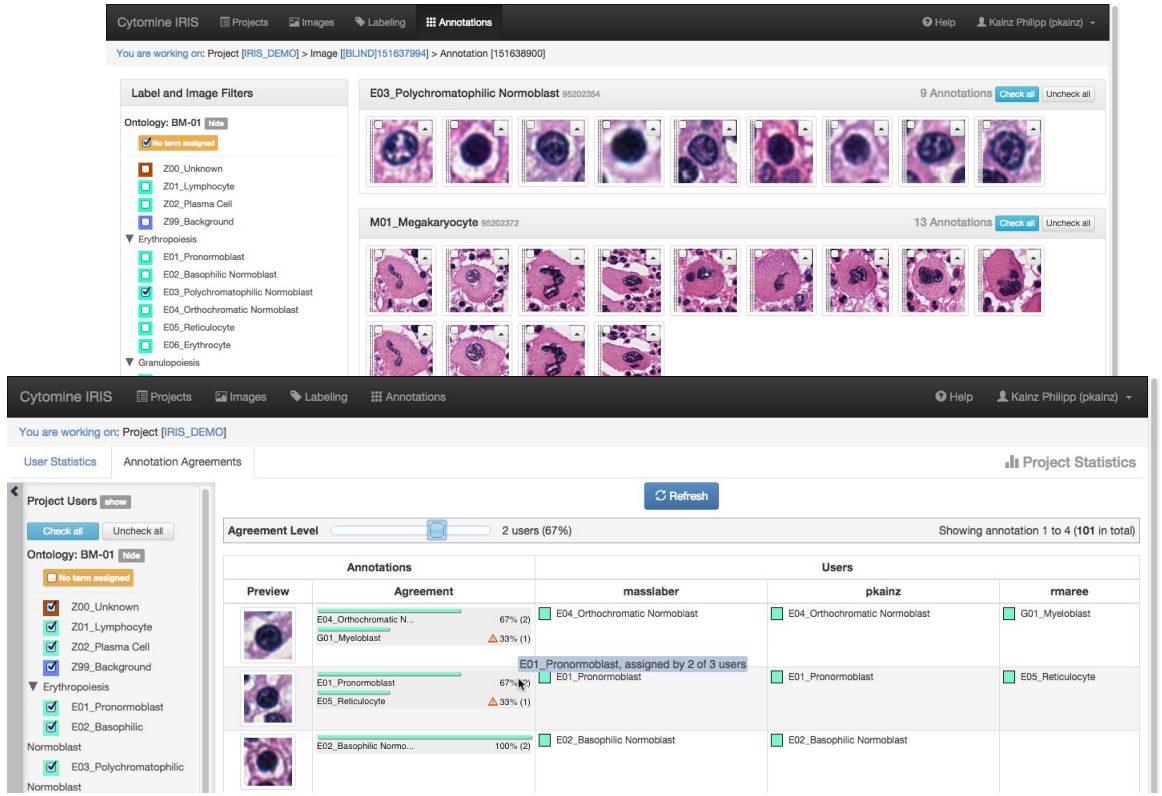
(Marée et al., ISBI 2016) (Gonzales et al., ECDP 2019)

Cytomine Javascript API client: extend or create Web-UI components

- https://doc.cytomine.org/dev-guide/clients/javascript/installation#how-to-install
- Examples on https://github.com/cytomine/Cytomine-js-client/tree/master/tests



Single-Page Application, Written in VueJS, loaded in browser:
https://github.com/cytomine/Cytomine-Web-UI



Cytomine IRIS (Inter-observer Reliability Study module) by Ph. Kainz
https://github.com/cytomine/Cytomine-IRIS (not using JS client)
https://www.youtube.com/watch?v=S0PhEJmqlmA

Cytomine Python client : quick demo

- Install Cytomine-Python-client or use “pre-built” Docker image https://doc.cytomine.org/dev-guide/clients/python/installation
- Authentication (using host, public & private keys from Account page)
- Fetch resources (e.g. project, ontology, terms, annotations, tile)
- Add resources (e.g. annotations, properties, upload image)

```
(base) maree@XPS-15-7590:~/cytomine/neubias-webinar$ sudo docker run -it -v "$PWD":/usr/src/myapp -w /usr/src/myapp cytomineuliege/software-python3-base:latest python webinar-get-data.py
[sudo] password for maree:
[2021-05-11 10:38:49,089][INFO] [GET] [currentuser] CURRENT USER - 701 : rmaree | 200 OK
[currentuser] CURRENT USER - 701 : rmaree
701
None
------------------------------------------------------------
[2021-05-11 10:38:50,115][INFO] [GET] [project collection] 527 objects | 200 OK
[project collection] 527 objects
[2021-05-11 10:38:50,189][INFO] [GET] [project] 10688948 : CHALLENGE-CAMELYON16-TRAIN | 200 OK
Project NAME: CHALLENGE-CAMELYON16-TRAIN | Ontology: 10688937
------------------------------------------------------------
[2021-05-11 10:38:50,275][INFO] [GET] [ontology] 10688937 : CAMELYON16-TRAIN | 200 OK
[2021-05-11 10:38:50,359][INFO] [GET] [term collection] 5 objects | 200 OK
Term ID: 528906181 | Name: Nucleus
Term ID: 528905747 | Name: ROI
Term ID: 10694501 | Name: Tumor
Term ID: 119767342 | Name: Sample
Term ID: 120684669 | Name: Sample_pred
------------------------------------------------------------
[2021-05-11 10:38:51,116][INFO] [GET] [imageinstance collection] 271 objects | 200 OK
Name: tumor_111.tif | Width: 97792 | Height: 220672 | ImageID: 125220817
Name: Tumor_009.tif | Width: 97792 | Height: 217088 | ImageID: 27712918
Name: Tumor_008.tif | Width: 97792 | Height: 221184 | ImageID: 27712752
Name: Tumor_007.tif | Width: 97792 | Height: 221184 | ImageID: 27712586
Name: Tumor_006.tif | Width: 97792 | Height: 221184 | ImageID: 27712408
Name: Tumor_005.tif | Width: 97792 | Height: 219648 | ImageID: 27712229
Name: tumor_110.tif | Width: 94208 | Height: 71680 | ImageID: 27712042
Name: tumor_109.tif | Width: 131072 | Height: 89600 | ImageID: 27711876
Name: tumor_108.tif | Width: 126976 | Height: 107520 | ImageID: 27711710
```

Cytomine Python client : applying a pre-trained Stardist model

- Stardist + Cytomine Python client to communicate with Cytomine server

1. Get ROIs (images with alphamask) from Cytomine server, save local PNG file

```
from csbdeep.utils import Path, normalize
from stardist import random_label_cmap
from stardist.models import StarDist2D
from cytomine import cytomine, models, CytomineJob
from cytomine.models import Annotation, AnnotationTerm, AnnotationCollection, ImageInstanceCollection, Job
from PIL import Image
```

```
roi_annotations = AnnotationCollection()
roi_annotations.project = conn.parameters.cytomine_id_project
roi_annotations.term = conn.parameters.cytomine_id_roi_term
roi_annotations.image = id_image
roi_annotations.showWKT = True
roi_annotations.fetch()
print(roi_annotations)
#Go over ROI in this image
for roi in roi_annotations:
    #Get Cytomine ROI coordinates for remapping to whole-slide
    #Cytomine cartesian coordinate system, (0,0) is bottom left corner
    roi_geometry = wkt.loads(roi.location)
    minx=roi_geometry.bounds[0]
    miny=roi_geometry.bounds[3]
    #Dump ROI image into local PNG file
    roi_path=os.path.join(working_path,str(roi_annotations.project)+'/'+str(roi_annotations.image)+'/'+str(roi.id))
    roi_png_filename=os.path.join(roi_path+'/'+str(roi.id)+'.png')
    roi.dump(dest_pattern=roi_png_filename,mask=True,alpha=True)
```

Cytomine Python client : applying a pre-trained Stardist model

- Stardist + Cytomine Python client to communicate with Cytomine server

2. Apply pre-trained Stardist model, convert & upload annotations to Cytomine

```
labels, details = model.predict_instances(img,
                                          prob_thresh=conn.parameters.stardist_prob_t,
                                          nms_thresh=conn.parameters.stardist_nms_t,)
print("Number of detected polygons: %d" %len(details['coord']))
cytomine_annotations = AnnotationCollection()
#Go over detections in this ROI, convert and upload to Cytomine
for pos,polygroup in enumerate(details['coord'],start=1):
    #Converting to Shapely annotation
    points = list()
    for i in range(len(polygroup[0])):
        #Cytomine cartesian coordinate system, (0,0) is bottom left corner
        #Mapping Stardist polygon detection coordinates to Cytomine ROI in whole slide image
        p = Point(minx+polygroup[1][i],miny-polygroup[0][i])
        points.append(p)

    annotation = Polygon(points)
    #Append to Annotation collection
    cytomine_annotations.append(Annotation(location=annotation.wkt,
                                           id_image=id_image,
                                           id_project=conn.parameters.cytomine_id_project,
                                           id_terms=[conn.parameters.cytomine_id_cell_term]))
#Send Annotation Collection (for this ROI) to Cytomine server in one http request
ca = cytomine_annotations.save()
```

Cytomine Python client : applying a pre-trained Stardist model

- Stardist + Cytomine Python client to communicate with Cytomine server

2. Apply pre-trained Stardist model, convert & upload annotations to Cytomine

OK but…

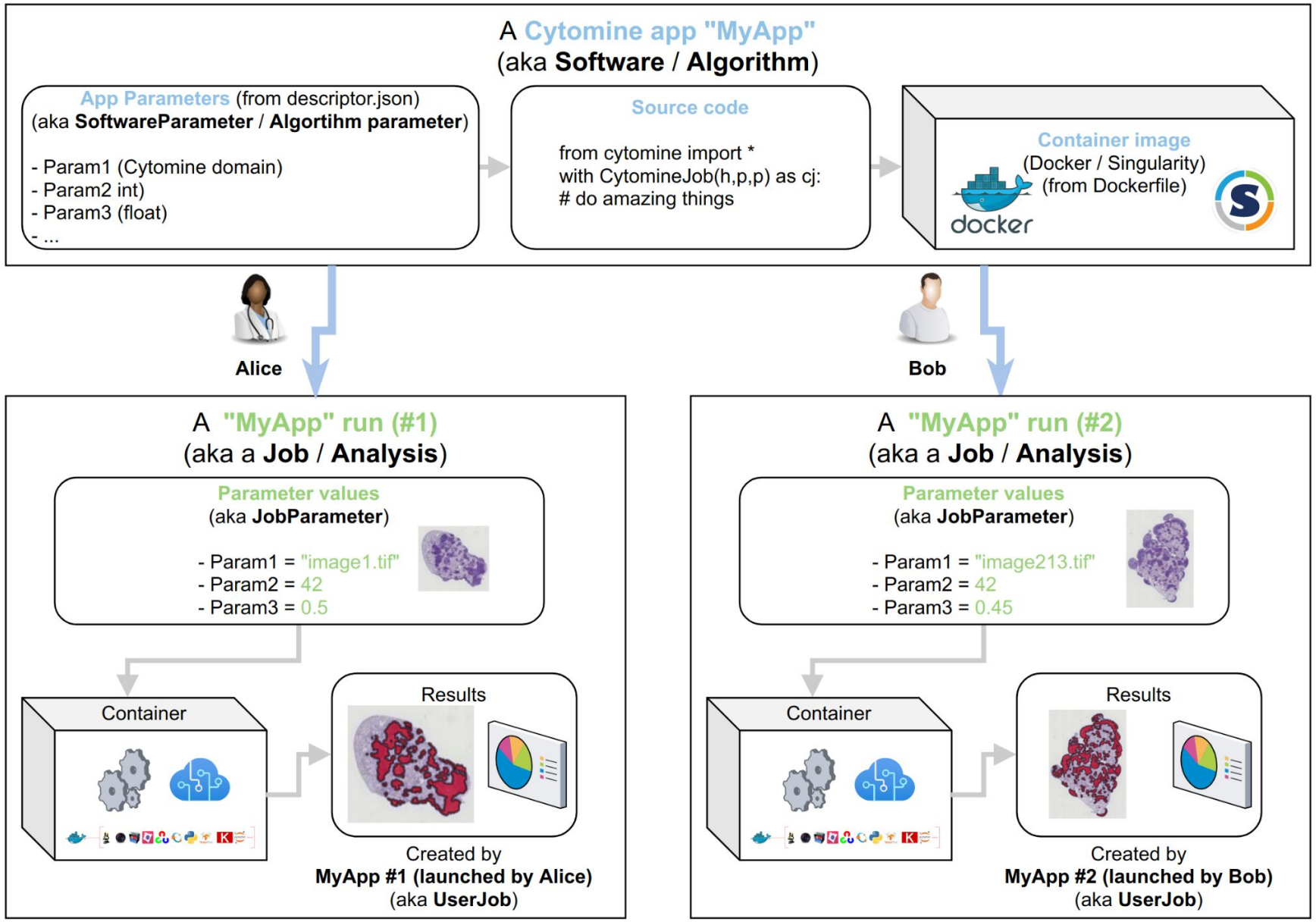
- It runs with user credentials and create data in Cytomine user own layer (UserAnnotation)
- It is neither traceable nor reproducible
  - Executed on my own computer with specific OS/libraries
  - Code is not versioned
  - Parameter values are not saved

→ We need more formalism to improve usability & reproducibility

```
                                                id_project=conn.parameters.cytomine_id_project,
                                                id_terms=[conn.parameters.cytomine_id_cell_term]))
#Send Annotation Collection (for this ROI) to Cytomine server in one http request
ca = cytomine_annotations.save()
```

Cytomine “Apps” concepts: Software, Job, Parameters

https://doc.cytomine.org/dev-guide/algorithms/



Cytomine “Apps”: Parameter Descriptor and Containerized Execution Environment

e.g. https://github.com/cytomine/S_CellDetect_Stardist_HE_ROI

Dockerfile

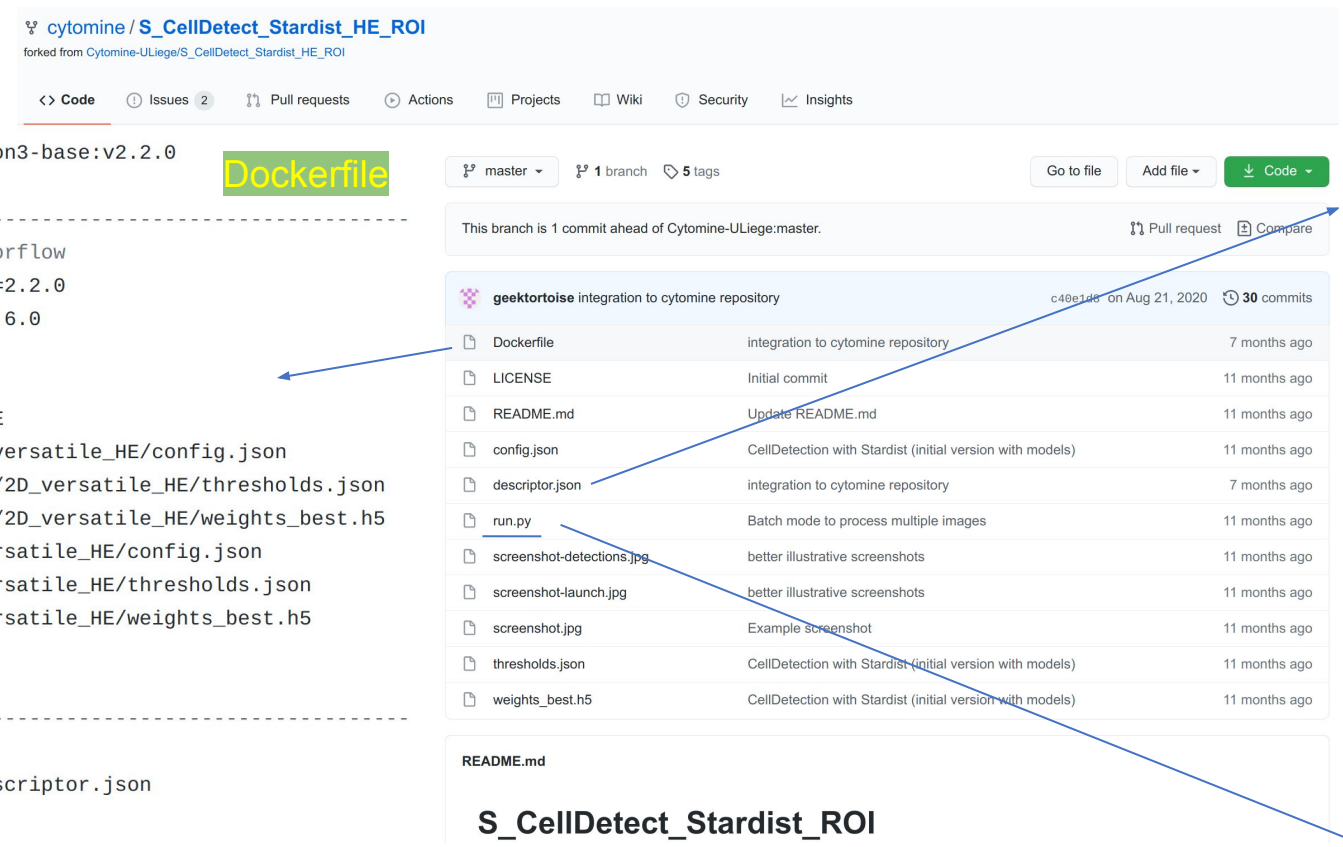
```
FROM cytomine/software-python3-base:v2.2.0

# ----------------------------------------------------------
# Install Stardist and tensorflow
RUN pip install tensorflow==2.2.0
RUN pip install stardist==0.6.0
RUN mkdir -p /models && \
    cd /models && \
    mkdir -p 2D_versatile_HE
ADD config.json /models/2D_versatile_HE/config.json
ADD thresholds.json /models/2D_versatile_HE/thresholds.json
ADD weights_best.h5 /models/2D_versatile_HE/weights_best.h5
RUN chmod 444 /models/2D_versatile_HE/config.json
RUN chmod 444 /models/2D_versatile_HE/thresholds.json
RUN chmod 444 /models/2D_versatile_HE/weights_best.h5



# ----------------------------------------------------------
# Install scripts
ADD descriptor.json /app/descriptor.json
RUN mkdir -p /app
ADD run.py /app/run.py

ENTRYPOINT ["python3", "/app/run.py"]
```



Descriptor

```
{
  "id": "stardist_prob_t",
  "value-key": "@ID",
  "command-line-flag": "--@id",
  "name": "Stardist Probability Threshold",
  "description": "Probability Threshold in range [0.0, 1.0] - higher values lead to fewer segmented objects, but will likel
  "default-value": 0.5,
  "set-by-server": false,
  "optional": true,
  "type": "Number"
},
{
  "id": "stardist_nms_t",
  "value-key": "@ID",
  "command-line-flag": "--@id",
  "name": "Stardist Non-Maximum Suppression Overlap threshold",
  "description": "Overlap Threshold in range [0.0, 1.0] - higher values allow segmented objects to overlap substantially.",
  "default-value": 0.5,
  "set-by-server": false,
  "optional": true,
  "type": "Number"
},
```

Code (e.g. Python)

```
#Stardist model prediction with thresholds
labels, details = model.predict_instances(img,
                                          prob_thresh=conn.parameters.stardist_prob_t,
                                          nms_thresh=conn.parameters.stardist_nms_t)
print("Number of detected polygons: %d" %len(details['coord']))
cytomine_annotations = AnnotationCollection()
#Go over detections in this ROI, convert and upload to Cytomine
for pos,polygroup in enumerate(details['coord'],start=1):
    #Converting to Shapely annotation
    points = list()
    for i in range(len(polygroup[0])):
        #Cytomine cartesian coordinate system, (0,0) is bottom left corner
        #Mapping Stardist polygon detection coordinates to Cytomine ROI in whole slide imag
        p = Point(minx+polygroup[1][i],miny-polygroup[0][i])
        points.append(p)

    annotation = Polygon(points)
    #Append to Annotation collection
    cytomine_annotations.append(Annotation(location=annotation.wkt,
                                           id_image=id_image,#conn.parameters.cytomine_id_i
                                           id_project=conn.parameters.cytomine_id_project,
```

Cytomine “Apps” *Parameter* Descriptor in JSON

e.g. https://github.com/cytomine/S_CellDetect_Stardist_HE_ROI/blob/master/descriptor.json

follows: https://doc.cytomine.org/dev-guide/algorithms/descriptor-reference

Execution command line + mandatory Cytomine parameters (host, private/public key, project id, software id)

```
{
    "name": "CellDetect_Stardist_HE_ROI",
    "container-image": {
        "image": "cytomine/s_celldetect_stardist_he_roi",
        "type": "singularity"
    },
    "schema-version": "cytomine-0.1",
    "description": "Cell (nuclei) detection using StarDist with versatile H&E pre-trained model",
    "command-line": "python run.py CYTOMINE_HOST CYTOMINE_PUBLIC_KEY CYTOMINE_PRIVATE_KEY CYTOMINE_ID_PROJECT CYTOMINE_ID_SOFTWARE CYTOMINE_ID_IMAGES
    "inputs": [
        {
                "id": "cytomine_host",
                "value-key": "@ID",
                "command-line-flag": "--@id",
                "name": "Cytomine host",
                "set-by-server": true,
                "optional": false,
                "type": "String"
        },
        {
                        "id": "cytomine_public_key",
                        "value-key": "@ID",
                        "command-line-flag": "--@id",
                        "name": "Cytomine public key",
                        "set-by-server": true,
                        "optional": false,
                        "type": "String"
        },
        {
                "id": "cytomine_private_key",
                        "value-key": "@ID",
                        "command-line-flag": "--@id",
                        "name": "Cytomine private key",
                        "set-by-server": true,
                        "optional": false,
                        "type": "String"
        },
                {
                        "id": "cytomine_id_project",
                        "value-key": "@ID",
                        "command-line-flag": "--@id",
                        "name": "Cytomine project ID",
                        "set-by-server": true,
                        "optional": false,
```

\+ algorithm specific parameters (e.g. thresholds,...)

```
{
    "id": "stardist_prob_t",
    "value-key": "@ID",
    "command-line-flag": "--@id",
    "name": "Stardist Probability Threshold",
    "description": "Probability Threshold in range [0.0, 1.0] - higher values lead to fewer segmented objects, but will likel
    "default-value": 0.5,
    "set-by-server": false,
    "optional": true,
    "type": "Number"
},
{
    "id": "stardist_nms_t",
    "value-key": "@ID",
    "command-line-flag": "--@id",
    "name": "Stardist Non-Maximum Suppression Overlap threshold",
    "description": "Overlap Threshold in range [0.0, 1.0] - higher values allow segmented objects to overlap substantially.",
    "default-value": 0.5,
    "set-by-server": false,
    "optional": true,
    "type": "Number"
},
```

The JSON descriptor is parsed by Cytomine server to create a Software in Cytomine database tables
→ “App” resource: https://demo.cytomine.be/api/software/155177168.json
→ “App” webpage: https://demo.cytomine.be/#/algorithm/155177168

See https://doc.cytomine.org/dev-guide/algorithms/descriptor-reference#complete-example
https://doc.cytomine.org/dev-guide/algorithms/write-app

Cytomine “Apps” *Environment* Descriptor (Dockerfile)

e.g. https://github.com/cytomine/S_CellDetect_Stardist_HE_ROI/blob/master/Dockerfile

```
FROM cytomine/software-python3-base:v2.2.0

# ----------------------------------------------------------
# Install Stardist and tensorflow
RUN pip install tensorflow==2.2.0
RUN pip install stardist==0.6.0
RUN mkdir -p /models && \
    cd /models && \
    mkdir -p 2D_versatile_HE
ADD config.json /models/2D_versatile_HE/config.json
ADD thresholds.json /models/2D_versatile_HE/thresholds.json
ADD weights_best.h5 /models/2D_versatile_HE/weights_best.h5
RUN chmod 444 /models/2D_versatile_HE/config.json
RUN chmod 444 /models/2D_versatile_HE/thresholds.json
RUN chmod 444 /models/2D_versatile_HE/weights_best.h5


# ----------------------------------------------------------
# Install scripts
ADD descriptor.json /app/descriptor.json
RUN mkdir -p /app
ADD run.py /app/run.py

ENTRYPOINT ["python3", "/app/run.py"]
```

Base container image that includes Cytomine Python client

Specific libraries & files (e.g. pre-trained model) needed by your code and installed in the container image

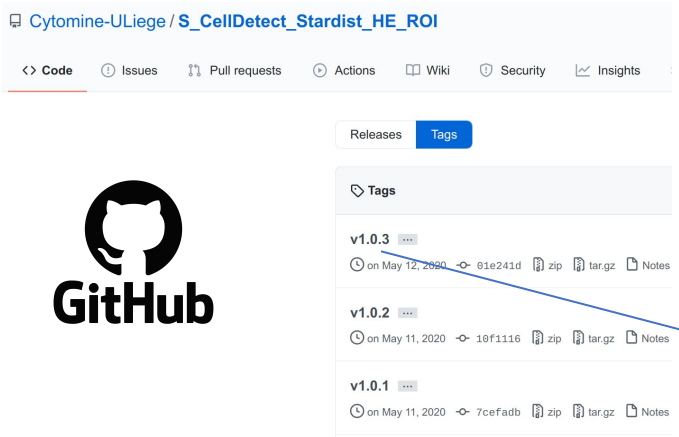
Command to be executed when container image is instanciated

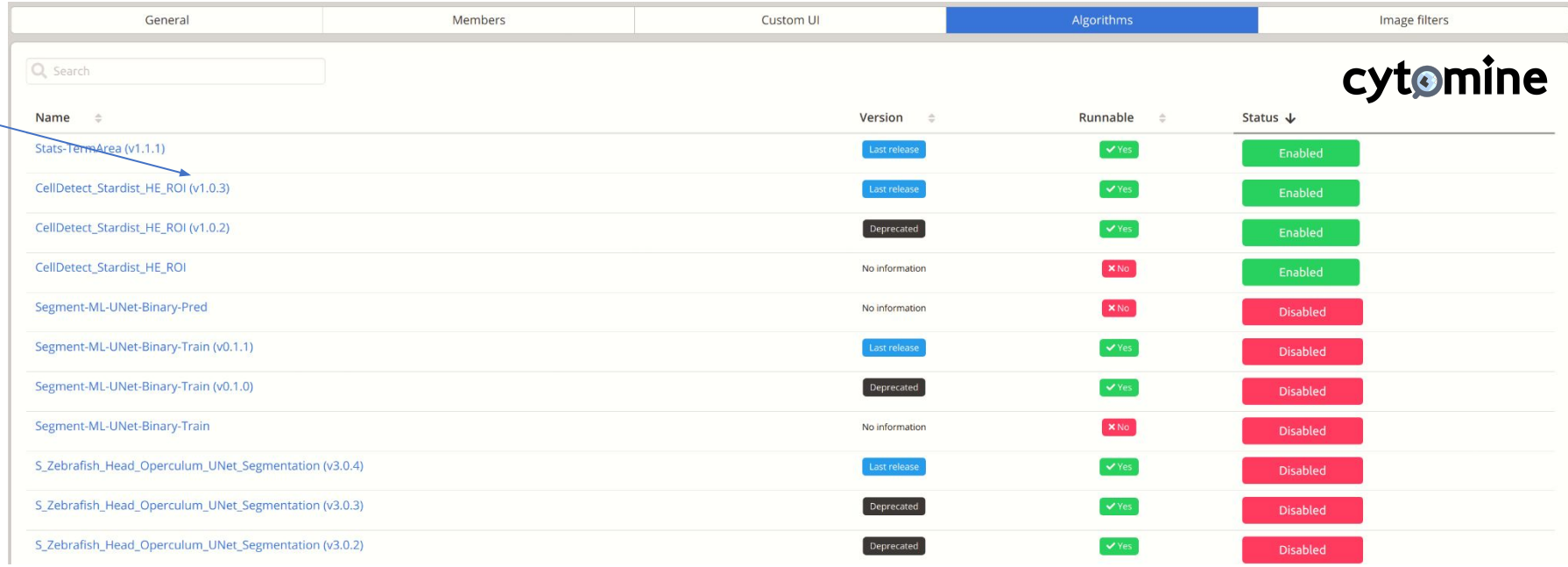
The Dockerfile is used by Docker to build a reproducible image of your software environment.
It allows to execute almost anything e.g. codes from:

Cytomine “Apps” versioning

- **Versioning** is achieved by creating **Github release tags** for important new code versions

- Automatic Builds configuration (DockerHub ⟷ Github) are used to create Docker images for each new release and automatically import them into Cytomine server (“trusted sources” configuration)





Cytomine “Apps” execution logging

- **Traceability** is achieved by logging software stack trace

Execution log

Hide

```
[2020-12-03 16:25:40,460][INFO] [GET] [currentuser] CURRENT USER - 525457679 : JOB[rmaree ], 2020-12-03 16:25:30:631 | 200 OK
[2020-12-03 16:25:40,525][INFO] [GET] [project] 138124787 : DEMONSTRATION-VARIOUS | 200 OK
[2020-12-03 16:25:40,594][INFO] [GET] [software] 155177168 : CellDetect_Stardist_HE_ROI | 200 OK
[2020-12-03 16:25:40,642][INFO] [GET] [softwareparameter collection] 12 objects | 200 OK
[2020-12-03 16:25:40,712][INFO] [GET] [job] 525457631 : None | 200 OK
[2020-12-03 16:25:40,712][INFO] Job (id:525457631) status update: "None" (status: RUNNING, progress: 0%)
[2020-12-03 16:25:40,839][INFO] [PUT] [job] 525457631 : None | 200 OK
[2020-12-03 16:25:40,840][INFO] Job (id:525457631) status update: "Initialization..." (status: RUNNING, progress: 0%)
[2020-12-03 16:25:40,960][INFO] [PUT] [job] 525457631 : None | 200 OK
Using TensorFlow backend.
WARNING:tensorflow:From /usr/local/lib/python3.7/site-packages/tensorflow_core/python/ops/resource_variable_ops.py:1630: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops) with constraint is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
[2020-12-03 16:25:41,795][WARNING] From /usr/local/lib/python3.7/site-packages/tensorflow_core/python/ops/resource_variable_ops.py:1630: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops) with constraint is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
WARNING:tensorflow:From /usr/local/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:4070: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

[2020-12-03 16:25:41,839][WARNING] From /usr/local/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:4070: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

2020-12-03 16:25:42.396999: W tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could not load dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1: cannot open shared object file: No such file or directory; LD_LIBRARY_PATH: /.singularity.d/libs
2020-12-03 16:25:42.397090: E tensorflow/stream_executor/cuda/cuda_driver.cc:318] failed call to cuInit: UNKNOWN ERROR (303)
2020-12-03 16:25:42.397141: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be running on this host (cytomine-slurm): /proc/driver/nvidia/version does not exist
2020-12-03 16:25:42.397453: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: FMA
2020-12-03 16:25:42.458346: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 2600085000 Hz
2020-12-03 16:25:42.469674: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x55c4cbf43880 initialized for platform Host (this does not guarantee that XLA will be used). Devices:
2020-12-03 16:25:42.469876: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default Version
Loading network weights from 'weights_best.h5'.
Loading thresholds from 'thresholds.json'.
Using default values: prob_thresh=0.692478, nms_thresh=0.3.
[2020-12-03 16:25:42,993][INFO] [GET] [imageinstance collection] 5 objects | 200 OK
[2020-12-03 16:25:42,994][INFO] Job (id:525457631) status update: "Running detection on image (1/1)." (status: RUNNING, progress: 0%)
[2020-12-03 16:25:43,108][INFO] [PUT] [job] 525457631 : None | 200 OK
[2020-12-03 16:25:43,189][INFO] [GET] [annotation collection] 1 objects | 200 OK
[annotation collection] 1 objects
----------------------------ROI----------------------------
ROI Geometry from Shapely: POLYGON ((3.341142933359549 508.4867133364421, 675.7250540661721 508.4867133364421, 675.7250540661721 4.675006793473358, 3.341142933359549 4.675006793473358, 3.341142933359549 508.4867133364421))
ROI Bounds
(3.3411429333595493, 4.675006793473358, 675.7250540661721, 508.4867133364421)
roi_png_filename: /home/cytomine/525457631/138124787/525455833/525455855/525455855.png
[2020-12-03 16:25:44,041][INFO] File downloaded successfully from https://research.cytomine.be/api/userannotation/525455855/alphamask.png
WARNING:tensorflow:From /usr/local/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

Normalizing image channels independently.
------------------- Processing ROI file 0: /home/cytomine/525457631/138124787/525455833/525455855/525455855.tif
[2020-12-03 16:25:44,215][WARNING] From /usr/local/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

Number of detected polygons: 60
.............................................................[2020-12-03 16:25:45,211][INFO] [POST] algoannotations  added | 200 OK
[2020-12-03 16:25:47,052][INFO] [POST] algoannotations 525457740,525457760,525457784,525457838,525457888,525457932,525457995,525458061,525458114,525458141,525458170,525458213,525458252,525458291,525458357 added | 200 OK
[2020-12-03 16:25:47,217][INFO] [POST] algoannotations 525457756,525457810,525457866,525457907,525457955,525458023,525458088,525458133,525458187,525458226,525458285,525458319,525458374,525458412,525458450 added | 200 OK
[2020-12-03 16:25:47,281][INFO] [POST] algoannotations 525457757,525457794,525457853,525457919,525457971,525458006,525458049,525458105,525458162,525458244,525458309,525458349,525458387,525458438,525458473 added | 200 OK
[2020-12-03 16:25:47,427][INFO] [POST] algoannotations 525457758,525457824,525457875,525457943,525457981,525458037,525458075,525458135,525458208,525458269,525458334,525458401,525458462,525458495,525458508 added | 200 OK
[2020-12-03 16:25:47,429][INFO] Job (id:525457631) status update: "Finished." (status: SUCCESS/TERMINATED, progress: 100%)
[2020-12-03 16:25:47,556][INFO] [PUT] [job] 525457631 : None | 200 OK
[2020-12-03 16:25:47,557][INFO] Job (id:525457631) status update: "Job successfully terminated" (status: SUCCESS/TERMINATED, progress: 100%)
[2020-12-03 16:25:47,675][INFO] [PUT] [job] 525457631 : None | 200 OK
```

Cytomine “Apps” reproducibility

- **Reproducibility** is further achieved by storing each execution parameter values (done by Cytomine-Core in its internal database for each execution)

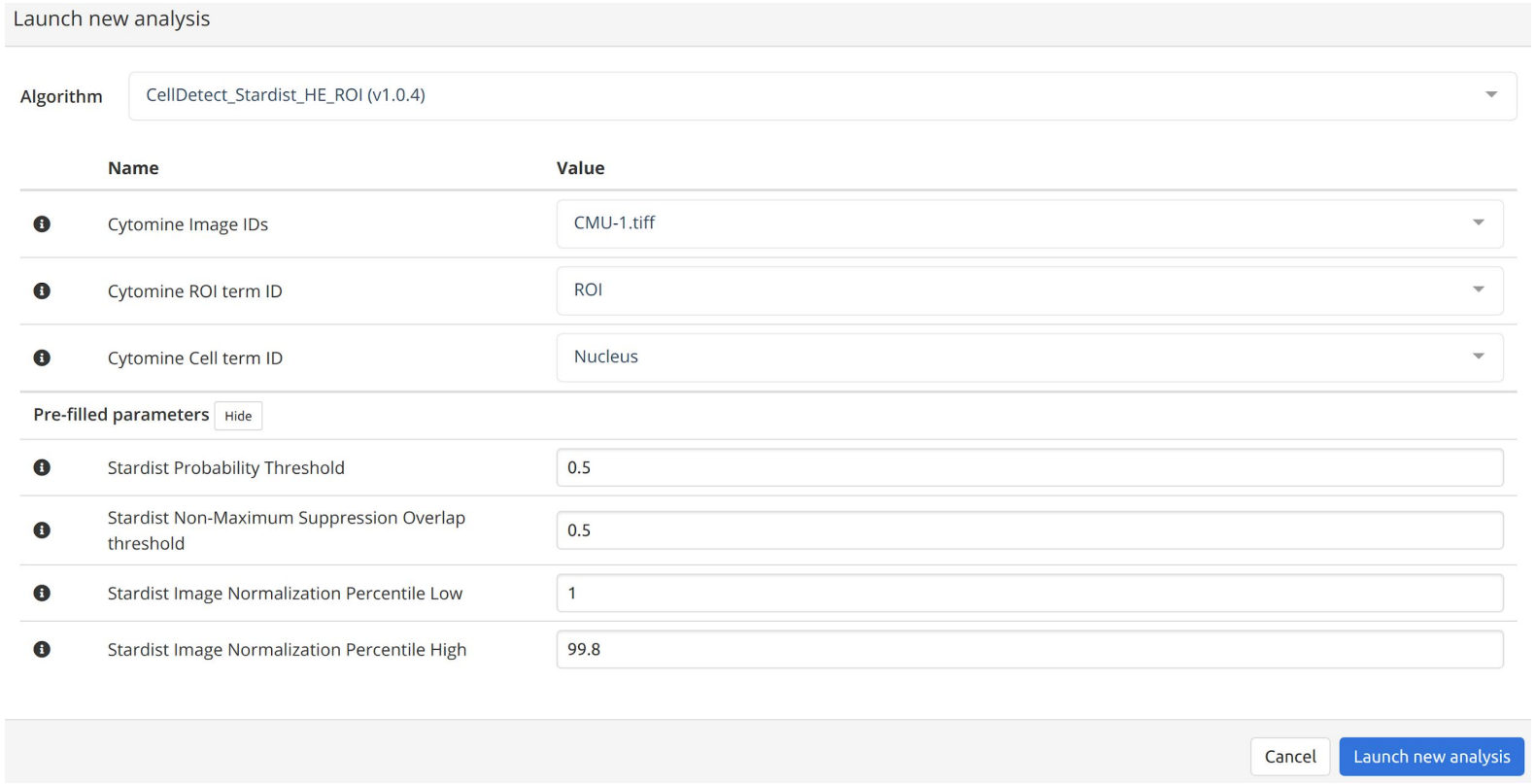
Segment-ML-MultiOutputET-Pred-BI (v0.2.7) | 4 | rhoyoux | Mar 29, 2021 10:51 AM | Success

Status comment	Job successfully terminated
Execution duration	18 minutes
Parameters	Hide

Name	Value	Type
Zoom level	5	Number
Tile size	256	Number
Tile overlap	0	Number
Number of jobs	8	Number
Union enabled	true	Boolean
Min annotations area	0	Number
Max annotations area	-1	Number
Minimum standard deviation for tile filtering	0	Number
Maximum mean for tile filtering	255	Number
Training job identifier	207885	Cytomine domain
Prediction step	1	Number
Cytomine host		String
Cytomine public key		String
Cytomine private key		String
Cytomine software id	202996	Number
Cytomine project id	1397	Number

→ Reproducible execution of Cytomine “Apps”

- From Cytomine WebUI



When you execute a Cytomine “App” from Cytomine Web-UI, Cytomine server sends parameter values to the “App” software environment (Docker container image) that is running on a “Processing Server”.
This “App” interacts with Cytomine using its embedded Cytomine Python client and upload results to the server

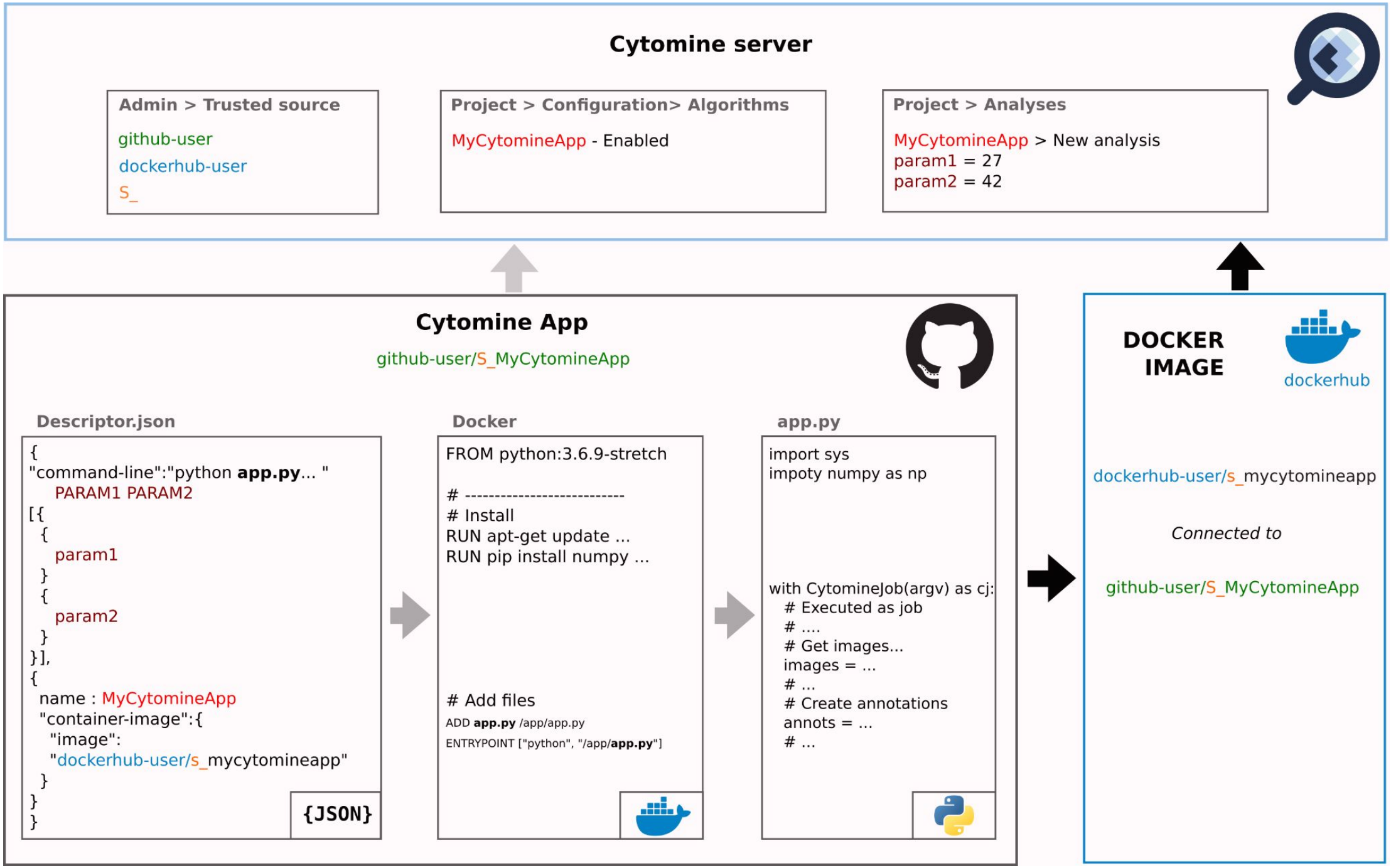
→ Reproducible execution of Cytomine "Apps"

From remote command-line (on your own computer/server/cluster):

- sudo docker pull cytomine/s_celldetect_stardist_he_roi:v1.0.4 ← Github/Dockerhub release tag

- sudo docker run -it cytomine/s_celldetect_stardist_he_roi:v1.0.4
  --cytomine_host https://demo.cytomine.be \
  --cytomine_public_key XXXXX \
  --cytomine_private_key XXXXX \ ← Cytomine credentials from Account page
  --cytomine_id_project 6870 \
  --cytomine_id_software 2762 \
  --cytomine_id_images 6967\
  --cytomine_id_roi_term 6880\ ← Cytomine resource identifiers
  --cytomine_id_cell_term 6888\
  --stardist_prob_t 0.8\ ← Your algorithm specific parameters for this execution

→ Then results are also accessible in Cytomine WebUI:
https://demo.cytomine.be/#/project/10688948/analysis


Cytomine server
Admin > Trusted source
github-user
dockerhub-user
S_
Project > Configuration> Algorithms
MyCytomineApp - Enabled
Project > Analyses
MyCytomineApp > New analysis
param1 = 27
param2 = 42
Cytomine App
github-user/S_MyCytomineApp
Descriptor.json
{
"command-line":"python app.py... "
PARAM1 PARAM2
[{
{
param1
}
{
param2
}
}],
{
name : MyCytomineApp
"container-image":{
"image":
"dockerhub-user/s_mycytomineapp"
}
}
}
{JSON}
Docker
FROM python:3.6.9-stretch

Install
RUN apt-get update ...
RUN pip install numpy ...
Add files
ADD app.py /app/app.py
ENTRYPOINT ["python", "/app/app.py"]
app.py
import sys
impoty numpy as np
with CytomineJob(argv) as cj:
# Executed as job
# ....
# Get images...
images = ...
# ...
# Create annotations
annots = ...
# ...
DOCKER IMAGE
dockerhub
dockerhub-user/s_mycytomineapp
Connected to
github-user/S_MyCytomineApp

Cytomine and BIAFLOWS for data and computer scientists

Cytomine might look “complex” compared to regular desktop tools. But it allows to make image-based research truly **collaborative and reproducible** by sharing almost anything (images, annotations, apps, code, results, …) **over the web**. Both end-users (WebUI) and computer scientists (API/clients) are able to manipulate, share, and publish all data.

Step-by-step documentation: https://doc.cytomine.org/
Any user issue ? https://forum.image.sc/tag/cytomine
Any code issue ? https://github.com/cytomine/

How to contribute ?

- Integrate your own “apps” and share them (thousands of potential users)
- Other ways: https://doc.cytomine.org/community/how-to-contribute

BIAFLOWS is a specific instance of Cytomine:
It uses same architecture/technologies, it comes with a slightly different UI, pre-populated workflows & datasets (2D+c+z+t); standardized inputs/outputs & metrics for benchmarking.

Acknowledgments

ULiège & Cytomine.coop : Ulysse Rubens, Romain Mormont, Navdeep Kumar, Remy Vandaele, Renaud Hoyoux, Grégoire Vincke, Pierre Geurts [cytomine.org]

Neubias and Comulis EU network : Sebastien Tosi, Benjamin Pavie, Volker Backer, Natasa Sladoje [neubias.org & comulis.eu]

Fundings (since 2010):