Supplementary information for BIAFLOWS: A collaborative framework to benchmark and deploy bioimage analysis workflows

Ulysse Rubens° (ULiège), Romain Mormont° (ULiège), Volker Baecker (MRI, Biocampus Montpellier), Gino Michiels (HEPL/ULiège), Lassi Paavolainen (FIMM, HiLIFE, UHelsinki), Graeme Ball (University of Dundee), Devrim Ünay (IUE), Benjamin Pavie (VIB), Anatole Chessel (Ecole Polytechnique), Leandro A. Scholz (Universidade Federal do Paraná), Martin Maška (Masaryk University), Renaud Hoyoux (Cytomine SCRL FS), Rémy Vandaele (ULiège), Stefan G. Stanciu (Politehnica Bucarest), Ofra Golani (Life Sciences Core Facilities, Weizmann Institute of Science, Israel), Natasa Sladoje (Uppsala University), Perrine Paul-Gilloteaux (Structure Fédérative de Recherche François Bonamy, Université de Nantes, CNRS, INSERM, Nantes, France), Raphaël Marée* (ULiège), Sébastien Tosi* (IRB Barcelona).

° These authors contributed equally to this work
* These authors jointly supervised this work (+ corresponding authors)

Supplementary Section 1. BIAFLOWS online instance: current content

This section reports the current content of BIAFLOWS in terms of projects illustrating specific bioimage analysis problems and workflows to process the image datasets from these projects. All workflows are referenced in BISE (NEUBIAS online repository of bioimage analysis resources).

Problem class	BIAFLOWS Project	Workflow GitHub	BISE link
1. Object detection / counting	**SPOT-COUNTING-2D**	W_SpotDetection-IJ	http://biii.eu/spot-detection-imagej
		W_SpotDetection-Icy	http://biii.eu/spot-detection-icy
	SPOT-COUNTING-3D	W_SpotDetection3D-IJ	http://biii.eu/node/1458
2. Object segmentation	**NUCLEI-SEGMENTATION**	W_NucleiSegmentation-ImageJ	http://biii.eu/nuclei-segmentation-2d-imagej
		W_NucleiSegmentation-CellProfiler	https://biii.eu/nuclei-segmentation-cellprofiler
		W_NucleiSegmentation--Python	http://biii.eu/nuclei-segmentation-python
		W_NucleiSegmentation-ilastik	https://biii.eu/nuclei-segmentation-ilastik
		W_NucleiSegmentation-MaskRCNN	https://biii.eu/node/1487
	DATA-SCIENCE-BOWL-2018	W_NucleiSegmentation-MaskRCNN	https://biii.eu/node/1487
		W_NucleiSegmentation-ilastik	https://biii.eu/nuclei-segmentation-ilastik
	NUCLEI-SEGMENTATION-3D	W_NucleiSegmentation3D-ImageJ	http://biii.eu/nuclei-segmentation-3d-imagej
3. Pixel classification	**GLAND-SEGMENTATION**	W_PixCla-UNet-GlaS	https://biii.eu/pixel-classification-glas-challenge-unet
4. Particle tracking	**NUCLEI-TRACKING-NODIVISION**	W_NucleiTracking-ImageJ	https://biii.eu/nuclei-tracking-imagej
5. Tree network tracing	**NEURON-TRACING-3D**	W_NeuronTracing_vaa3d	https://biii.eu/app-all-path-pruning
6. Filament network tracing	**VESSEL-TRACING-3D**	W_FilamentTracing3D-ImageJ	http://biii.eu/node/1453
7. Landmark detection	**LANDMARKS-DROSO**	W_LandmarkDetect-ML-MSET-Pred	https://biii.eu/landmark-detection-mset-models-prediction
		W_LandmarkDetect-ML-LC-Pred	https://biii.eu/landmark-detection-lc-models-prediction
		W_LandmarkDetect-ML-DMBL-Pred	https://biii.eu/node/1485
8. Object tracking	**NUCLEI-TRACKING-DIVISION**	**Framework available**	

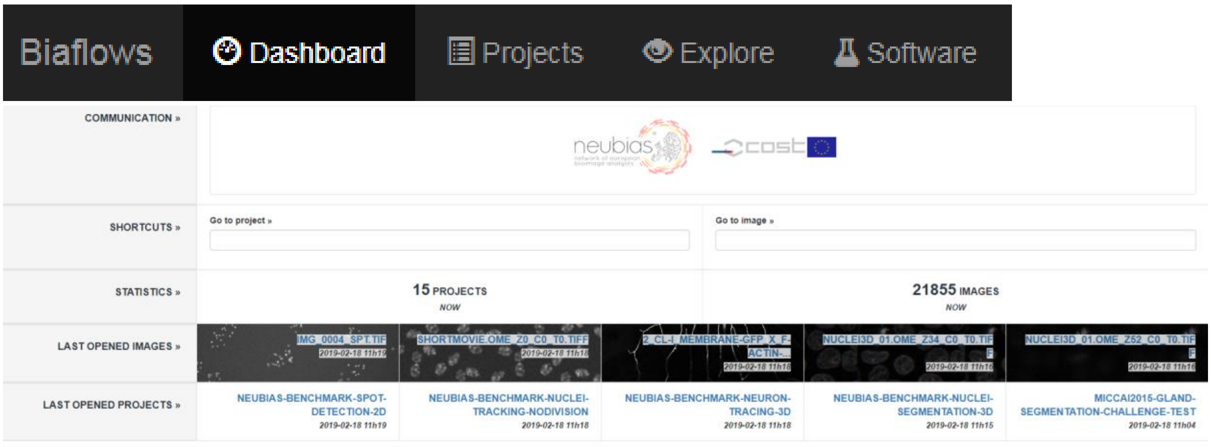
Table 1. BIA problems and workflows currently available in BIAFLOWS online instance. The name of the code repository on https://github.com/Neubias-WG5 and a link to the webpage where each workflow is referenced and described in Bioimage Informatics Search Index (BISE) is provided.

Supplementary Section 2. BIAFLOWS user guide

BIAFLOWS online instance can be accessed at https://biaflows.neubias.org. It is possible to browse the content in read only mode from the guest account (username: *guest*; password: *guest*). The platform has been tested for Chrome, Chromium, Safari and Firefox.

Dashboard

The *Dashboard* (Fig. S2.1) is BIAFLOWS landing page. It provides an overview of the content (projects, images) and recent activity. Notably, **Go to project »** is helpful to quickly access a project by typing one or several keywords (e.g. *Nuclei*).



COMMUNICATION »

SHORTCUTS » Go to project » Go to image »

STATISTICS » 15 PROJECTS NOW | 21855 IMAGES NOW

LAST OPENED IMAGES »

- IMG_0004_SPT.TIF 2019-02-18 11h19
- SHORTMOVIE.OME_Z0_C0_T0.TIFF 2019-02-18 11h18
- 2_CL-I_MEMBRANE-GFP_X_F-ACTIN-... 2019-02-18 11h18
- NUCLEI3D_01.OME_Z34_C0_T0.TIFF 2019-02-18 11h16
- NUCLEI3D_01.OME_Z52_C0_T0.TIFF 2019-02-18 11h16

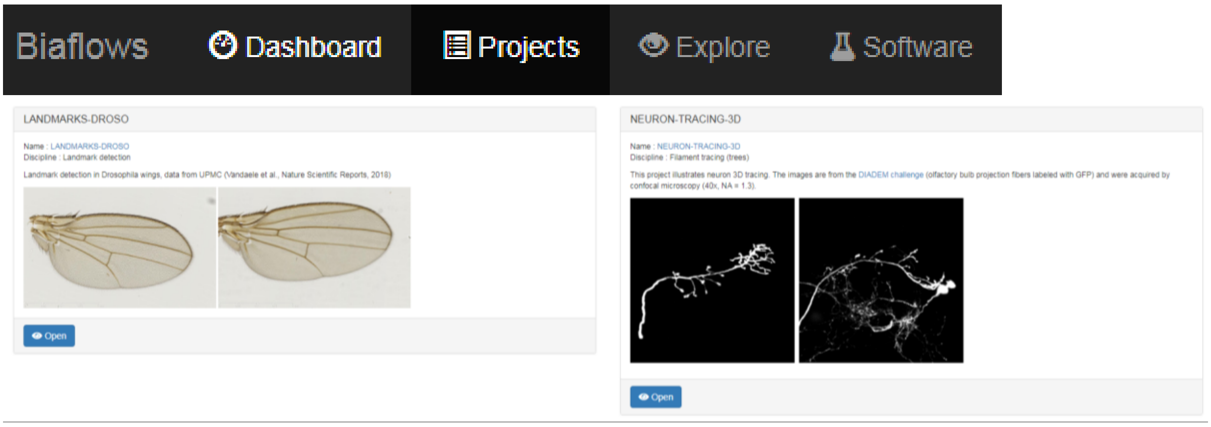
LAST OPENED PROJECTS »

- NEUBIAS-BENCHMARK-SPOT-DETECTION-2D 2019-02-18 11h19
- NEUBIAS-BENCHMARK-NUCLEI-TRACKING-NODIVISION 2019-02-18 11h18
- NEUBIAS-BENCHMARK-NEURON-TRACING-3D 2019-02-18 11h18
- NEUBIAS-BENCHMARK-NUCLEI-SEGMENTATION-3D 2019-02-18 11h15
- MICCAI2015-GLAND-SEGMENTATION-CHALLENGE-TEST 2019-02-18 11h04

Figure S2.1 BIAFLOWS Dashboard

Projects

The *Projects* section (Fig. S2.2) brings an overview of BIAFLOWS projects. Each project illustrates a common BIA problem and gathers a set of annotated images and compatible workflows to process them. A BIA problem class, a.k.a. *Discipline* (e.g. *Object Segmentation*), is associated to every project; BIAFLOWS currently supports 8 disciplines (see Supplementary Table 1). Disciplines specify a format for ground truth annotations (identical to expected workflow output), as well as the benchmark metrics to be evaluated on the annotated images of a project. Clicking on a project redirects to the *Explore* section.



LANDMARKS-DROSO

Name : LANDMARKS-DROSO
Discipline : Landmark detection

Landmark detection in Drosophila wings, data from UPMC (Vandaele et al., Nature Scientific Reports, 2018)

NEURON-TRACING-3D

Name : NEURON-TRACING-3D
Discipline : Filament tracing (trees)

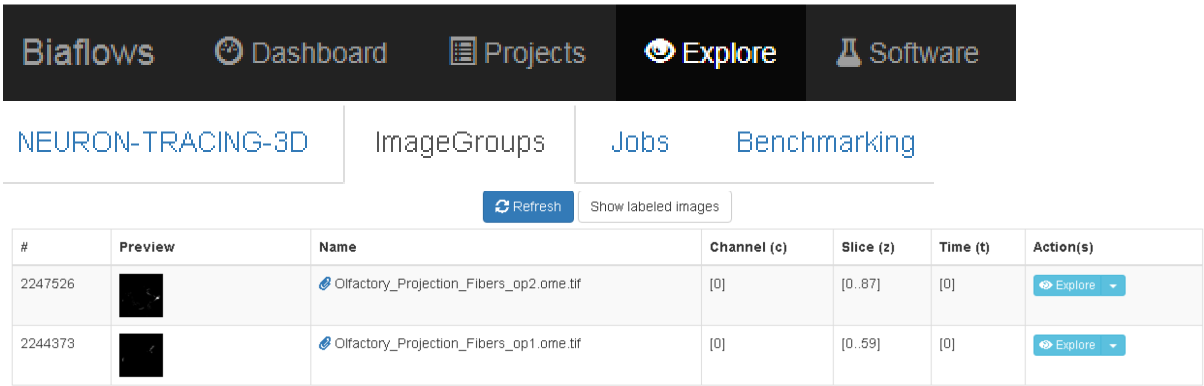
This project illustrates neuron 3D tracing. The images are from the DIADEM challenge (olfactory bulb projection fibers labeled with GFP) and were acquired by confocal microscopy (40x, NA = 1.3).

Figure S2.2 Projects section

Explore

This section displays information about a project, more specifically its associated discipline, a description of its image analysis task and a reference to the original source of the annotated images. All images from a project are either 2D or multidimensional (C, Z, T), they can be viewed by clicking on the *Images* tab (2D images) or *ImageGroups* tab (multidimensional images).

Images / ImageGroups



#	Preview	Name	Channel (c)	Slice (z)	Time (t)	Action(s)
2247526		Olfactory_Projection_Fibers_op2.ome.tif	[0]	[0..87]	[0]	Explore
2244373		Olfactory_Projection_Fibers_op1.ome.tif	[0]	[0..59]	[0]	Explore

Figure S2.3 Images tab

The table displayed on this page lists all the images from the current project (Fig. S2.3). Images can be filtered by name by typing a keyword in the search bar and visualized by clicking on their thumbnails.

In case ground truth annotations are stored as image masks (this depends on the discipline of the project), they can be displayed by clicking Show labeled images. If ground truth annotations are in a different format (e.g. SWC, text file), they can be managed from 📎.

Finally, all images can be downloaded from their associated drop down menus.

Image Viewer

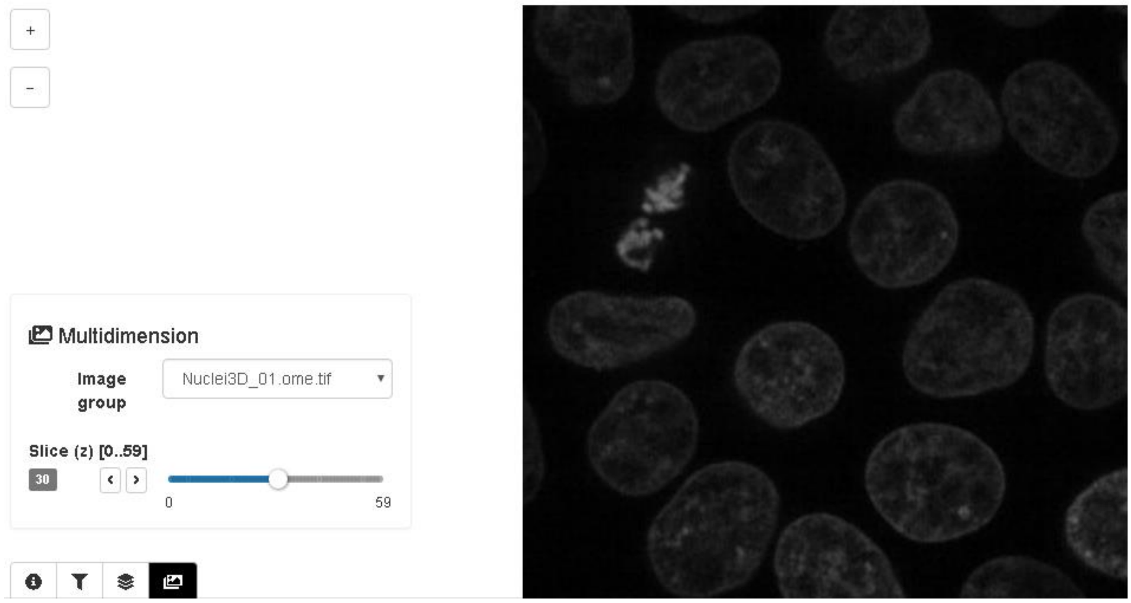


Image group: Nuclei3D_01.ome.tif

Slice (z) [0..59]: 30

Figure S2.4 Image viewer

Navigation:
Use mouse scroll to zoom in/out, click + drag to move around. For multidimensional images (ImageGroups), the panel 🖼 is used to navigate through the slices.

Layers:
To overlay an annotation layer, click ≋.

Annotation layers are generated from the results of a workflow run (a.k.a. job), several layers can be overlaid and they can be removed / hidden from the button next to their names.

Show/hide a layer from Show and remove it from ✕Remove. Layer opacity can be adjusted with the bottom slider. If you do not know which annotation layer to display, click ★ Add all starred jobs. ★ jobs are workflow runs validated by their maintainer as representative results.

It is also possible to have multiple viewers opened at once and synchronize them (Fig. S2.5). This can be helpful for instance to open the same image several times and compare several workflow results (annotation layers). To do so, select an image from the upper left drop down
menu and click Add a viewer, then use 🔗 to synchronize the views.

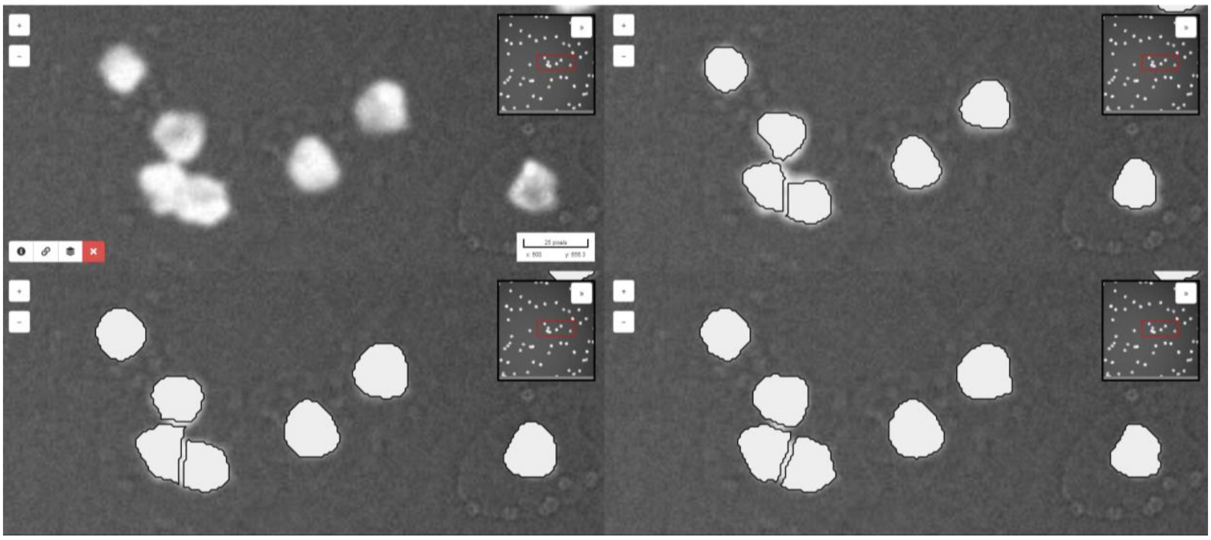


Figure S2.5 Image viewer: 4 synchronized views
(top left: original image, others: workflow results overlays)

The current image can be downloaded by clicking on information panel ℹ and the Download button.

For 2D images, it is also possible to switch to previous/next image (‹ Previous image | Next image ›) in the project from this panel.

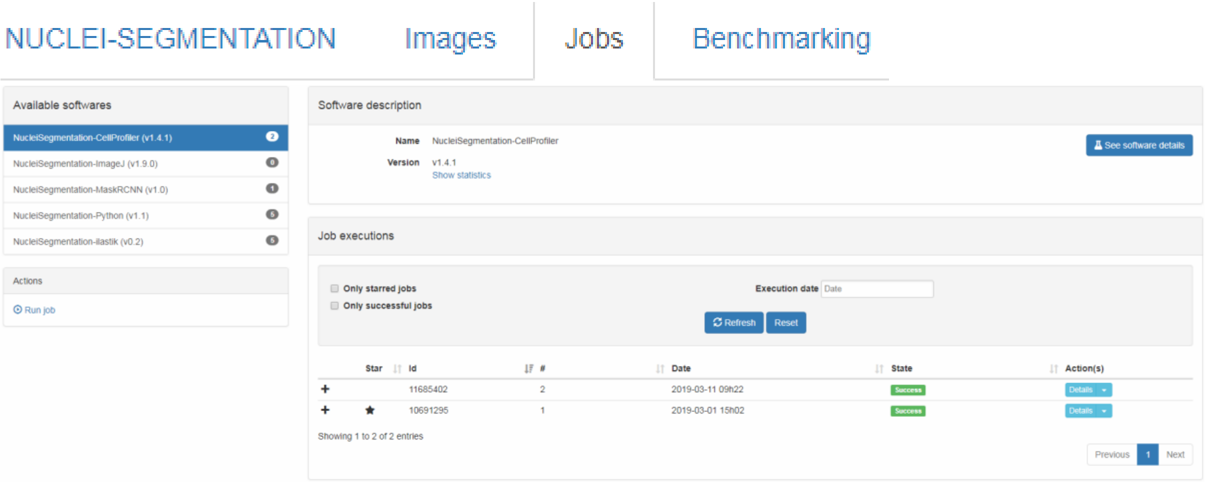
Finally, clicking on an annotation in the viewer will display some information.

Image adjustments:

Contrast and gamma correction can be adjusted from ▼ . The image can also be video inverted from this panel.

Jobs

This tab lists all software available in the current project (Fig. S2.6, left section), i.e. available workflows to process the images according to task associated to the project. All workflows are versioned and deprecated versions are struck through.



NUCLEI-SEGMENTATION | Images | Jobs | Benchmarking

Available softwares

- NucleiSegmentation-CellProfiler (v1.4.1) 2
- NucleiSegmentation-ImageJ (v1.9.0) 0
- NucleiSegmentation-MaskRCNN (v1.0) 1
- NucleiSegmentation-Python (v1.1) 5
- NucleiSegmentation-ilastik (v0.2) 6

Actions

Run job

Software description

Name NucleiSegmentation-CellProfiler

Version v1.4.1

Show statistics

See software details

Job executions

□ Only starred jobs

□ Only successful jobs

Execution date Date

Refresh Reset

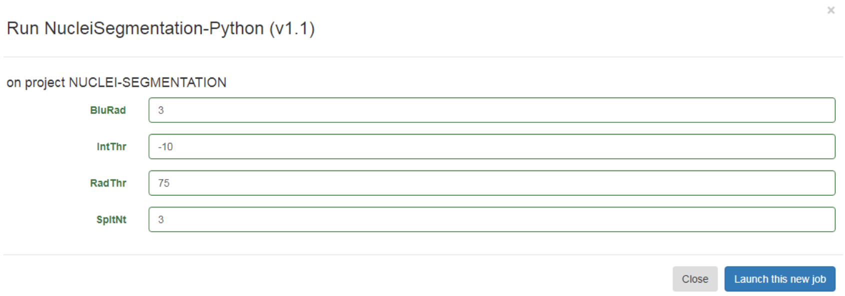
	Star	Id	#	Date	State	Action(s)
+		11685402	2	2019-03-11 09h22	Success	Details
+	★	10691295	1	2019-03-01 15h02	Success	Details

Showing 1 to 2 of 2 entries

Previous 1 Next

Figure S2.6 Jobs tab

A workflow can be launched by clicking on its name, and then Run job (if the account used brings execution rights). This opens a workflow parameter dialog box. Default parameters (Fig. S2.7) are set so as to bring meaningful results for the images of the project.



Run NucleiSegmentation-Python (v1.1)

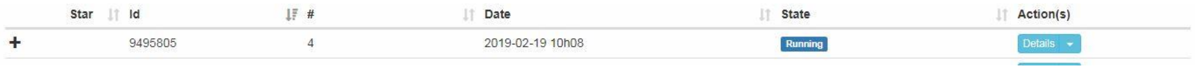
on project NUCLEI-SEGMENTATION

BluRad	3
IntThr	-10
RadThr	75
SpltNt	3

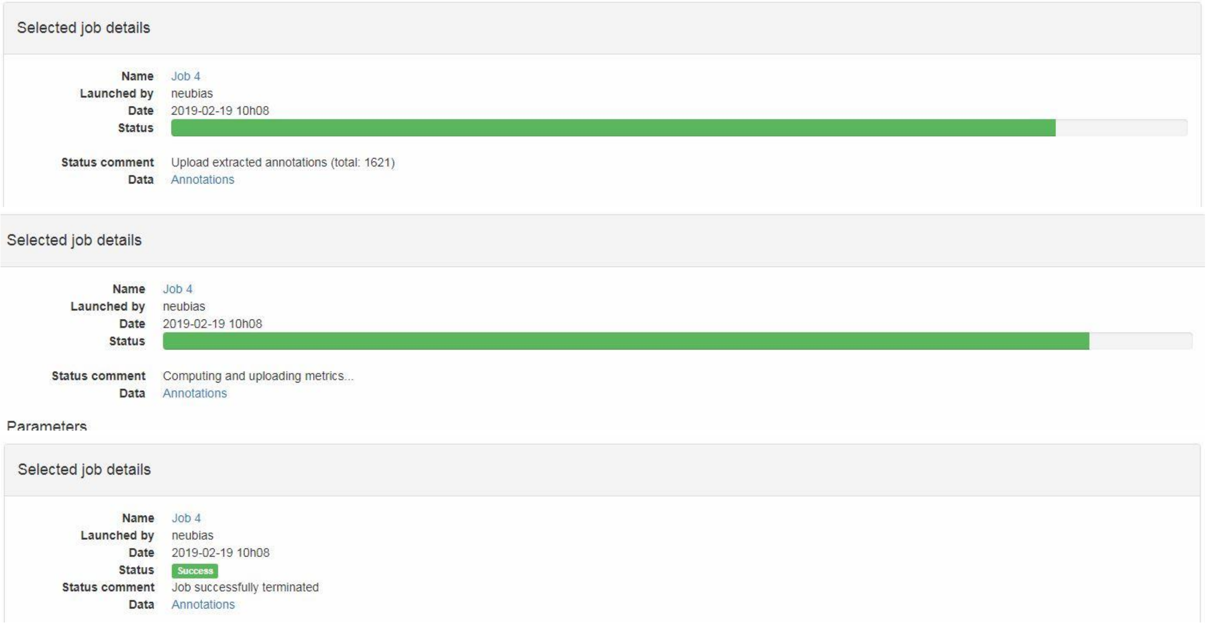
Close Launch this new job

Figure S2.7 Parameter dialog box for workflow execution

Upon launch, a job if flagged as *running* in the job list (Fig. S2.8) and a progress bar displays information on the current execution step. This progress bar can be recalled at any time by clicking Details next to a running workflow.



	Star	Id	#	Date	State	Action(s)
+		9495805	4	2019-02-19 10h08	Running	Details



Selected job details

Name Job 4
Launched by neubias
Date 2019-02-19 10h08
Status

Status comment Upload extracted annotations (total: 1621)
Data Annotations

Selected job details

Name Job 4
Launched by neubias
Date 2019-02-19 10h08
Status

Status comment Computing and uploading metrics...
Data Annotations

Parameters

Selected job details

Name Job 4
Launched by neubias
Date 2019-02-19 10h08
Status Success
Status comment Job successfully terminated
Data Annotations

Figure S2.8 Job status (up and middle) and job details (bottom)

Upon completion, the job status should either turn to success Success or failed Failed .

If a job is successful, workflow results (annotation layers for the image explorer) and benchmark metrics (benchmark tab) should be available for this run.

A complete report (log file) of the job is also accessible from *Job attached file* Download.

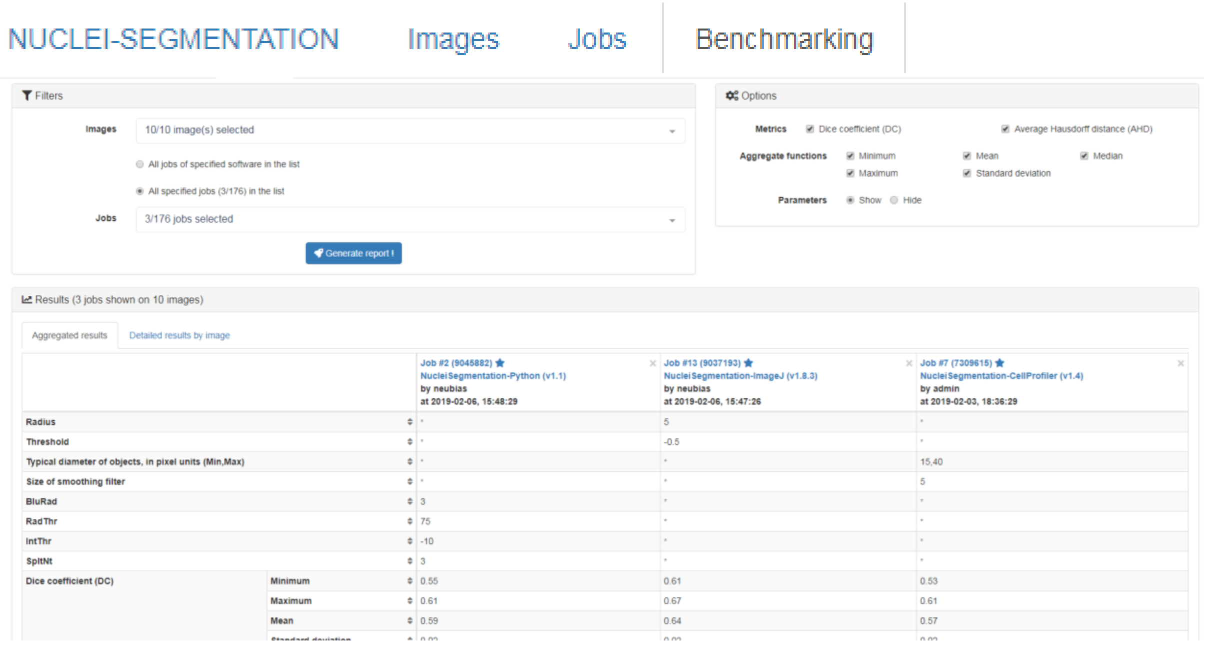
Benchmarking

The *Benchmarking* tab (Fig. S2.9) is used to browse benchmark results (metrics) from past workflow runs (jobs). To generate a benchmark report click Generate report !.

By default, a report is generated for all starred jobs and all images from the project. Both selections can be refined from the **Images** and **Jobs** drop down menus and the choice selector:

○ All jobs of specified software in the list

● All specified jobs (3/176) in the list



Images: 10/10 image(s) selected

○ All jobs of specified software in the list

● All specified jobs (3/176) in the list

Jobs: 3/176 jobs selected

Metrics: ☑ Dice coefficient (DC) ☑ Average Hausdorff distance (AHD)

Aggregate functions: ☑ Minimum ☑ Mean ☑ Median ☑ Maximum ☑ Standard deviation

Parameters: ● Show ○ Hide

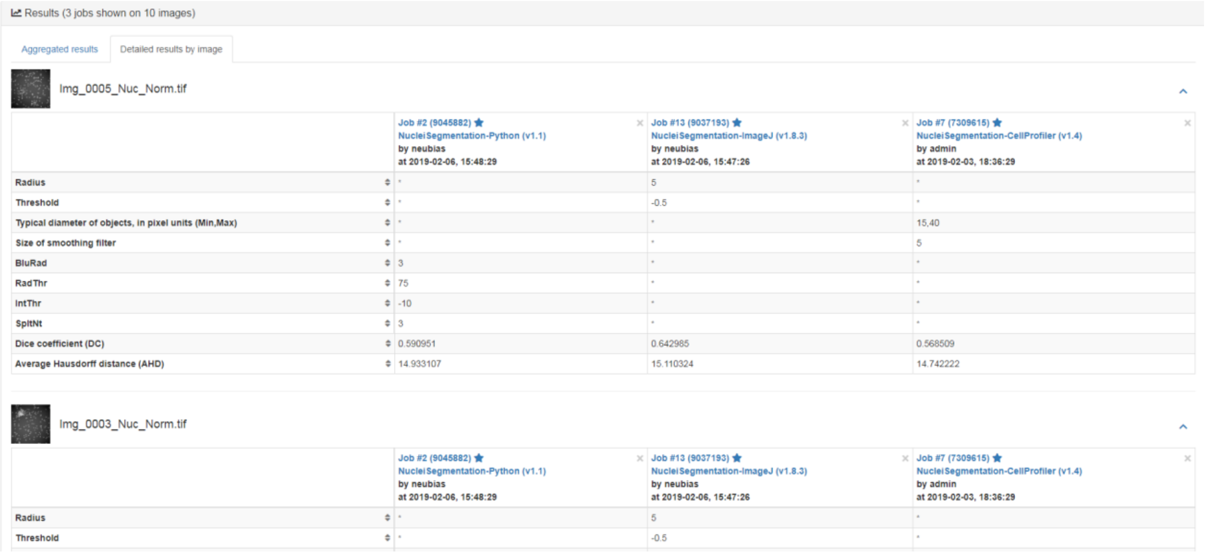
Results (3 jobs shown on 10 images)

		Job #2 (9045882) NucleiSegmentation-Python (v1.1) by neubias at 2019-02-06, 15:48:29	Job #13 (9037193) NucleiSegmentation-ImageJ (v1.8.3) by neubias at 2019-02-06, 15:47:26	Job #7 (7309615) NucleiSegmentation-CellProfiler (v1.4) by admin at 2019-02-03, 18:36:29
Radius		*	5	*
Threshold		*	-0.5	*
Typical diameter of objects, in pixel units (Min,Max)		*	*	15,40
Size of smoothing filter		*	*	5
BluRad		3	*	*
RadThr		75	*	*
IntThr		-10	*	*
SpltNt		3	*	*
Dice coefficient (DC)	Minimum	0.55	0.61	0.53
	Maximum	0.61	0.67	0.61
	Mean	0.59	0.64	0.57
	Standard deviation	0.02	0.02	0.02

Figure S2.9 Benchmarking tab (aggregated view)

Metrics results and workflow parameters can be displayed from the right panel Options . The metrics can be explored as aggregated (all images, Fig. S2.9) or on a per image basis (Fig. S2.10) by switching between tabs Aggregated results Detailed results by image .

The *detailed results by image* view is useful to, e.g., detect jobs with poor metrics and inspect the associated annotation layer to understand where the workflow has failed. To quickly access an image, its thumbnail should be clicked from the list.



Results (3 jobs shown on 10 images)

Img_0005_Nuc_Norm.tif

	Job #2 (9045882) NucleiSegmentation-Python (v1.1) by neubias at 2019-02-06, 15:48:29	Job #13 (9037193) NucleiSegmentation-ImageJ (v1.8.3) by neubias at 2019-02-06, 15:47:26	Job #7 (7309615) NucleiSegmentation-CellProfiler (v1.4) by admin at 2019-02-03, 18:36:29
Radius	*	5	*
Threshold	*	-0.5	*
Typical diameter of objects, in pixel units (Min,Max)	*	*	15,40
Size of smoothing filter	*	*	5
BluRad	3	*	*
RadThr	75	*	*
IntThr	-10	*	*
SpltNt	3	*	*
Dice coefficient (DC)	0.590951	0.642985	0.568509
Average Hausdorff distance (AHD)	14.933107	15.110324	14.742222

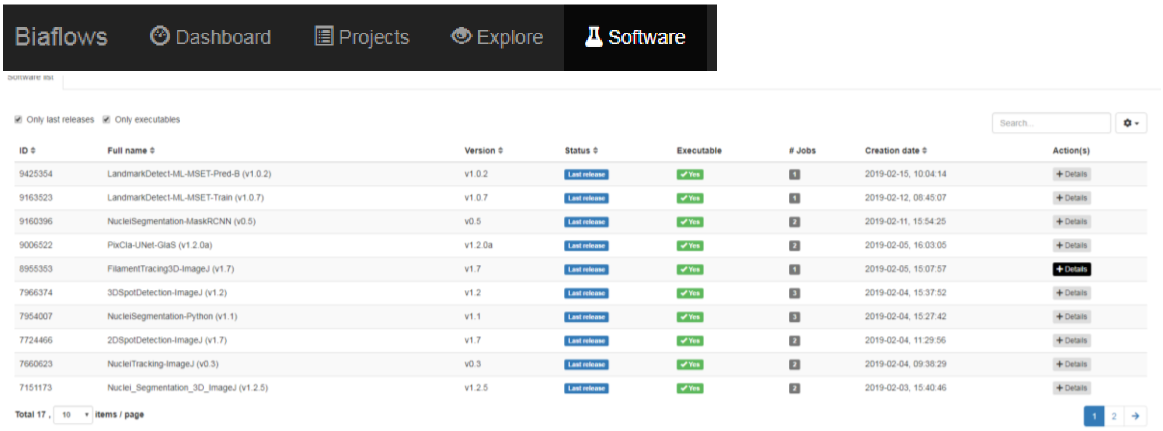
Img_0003_Nuc_Norm.tif

	Job #2 (9045882) NucleiSegmentation-Python (v1.1) by neubias at 2019-02-06, 15:48:29	Job #13 (9037193) NucleiSegmentation-ImageJ (v1.8.3) by neubias at 2019-02-06, 15:47:26	Job #7 (7309615) NucleiSegmentation-CellProfiler (v1.4) by admin at 2019-02-03, 18:36:29
Radius	*	5	*
Threshold	*	-0.5	*

Figure S2.10 Benchmarking tab (image view)

Software

The software section gives an overview of all BIA workflows currently available in the system (Fig S2.11). In principle, the most relevant software is the last executable version but all past versions are also displayed as reference. Workflows can be filtered by name from the search bar.



☑ Only last releases ☑ Only executables

ID ⇕	Full name ⇕	Version ⇕	Status ⇕	Executable	# Jobs	Creation date ⇕	Action(s)
9425354	LandmarkDetect-ML-MSET-Pred-B (v1.0.2)	v1.0.2	Last release	✓ Yes	1	2019-02-15, 10:04:14	+ Details
9163523	LandmarkDetect-ML-MSET-Train (v1.0.7)	v1.0.7	Last release	✓ Yes	1	2019-02-12, 08:45:07	+ Details
9160396	NucleiSegmentation-MaskRCNN (v0.5)	v0.5	Last release	✓ Yes	2	2019-02-11, 15:54:25	+ Details
9006522	PixCla-UNet-GlaS (v1.2.0a)	v1.2.0a	Last release	✓ Yes	2	2019-02-05, 16:03:05	+ Details
8955353	FilamentTracing3D-ImageJ (v1.7)	v1.7	Last release	✓ Yes	1	2019-02-05, 15:07:57	+ Details
7966374	3DSpotDetection-ImageJ (v1.2)	v1.2	Last release	✓ Yes	3	2019-02-04, 15:37:52	+ Details
7954007	NucleiSegmentation-Python (v1.1)	v1.1	Last release	✓ Yes	3	2019-02-04, 15:27:42	+ Details
7724466	2DSpotDetection-ImageJ (v1.7)	v1.7	Last release	✓ Yes	2	2019-02-04, 11:29:56	+ Details
7660623	NucleiTracking-ImageJ (v0.3)	v0.3	Last release	✓ Yes	2	2019-02-04, 09:38:29	+ Details
7151173	Nuclei_Segmentation_3D_ImageJ (v1.2.5)	v1.2.5	Last release	✓ Yes	2	2019-02-03, 15:40:46	+ Details

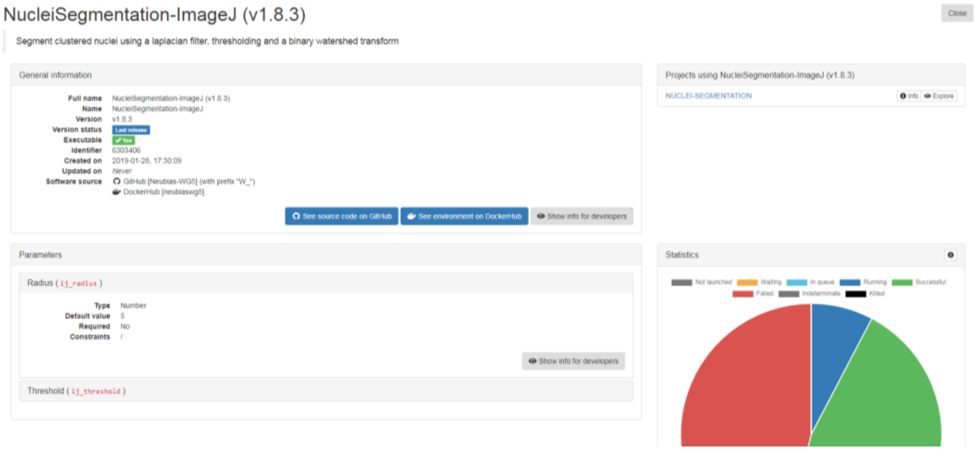
Total 17 , 10 items / page

1 2 →

Figure S2.11 Software (BIA workflows) tab

Selecting a workflow brings some useful information (Fig. S2.12); in particular:

- Its input parameters and default values (expandable list)
- The projects using this workflow (with links to projects)
- **GitHub** and **DockerHub** links respectively for the workflow source code and Docker image builds
- The command line invoked by BIAFLOWS to launch the workflow image (Show info for developers)



NucleiSegmentation-ImageJ (v1.8.3)

Segment clustered nuclei using a laplacian filter, thresholding and a binary watershed transform

General information

Full name	NucleiSegmentation-ImageJ (v1.8.3)
Name	NucleiSegmentation-ImageJ
Version	v1.8.3
Version status	Last release
Executable	✓ Yes
Identifier	6303406
Created on	2019-01-28, 17:30:09
Updated on	*Never*
Software source	GitHub [Neubias-WG5] (with prefix "W_") DockerHub [neubiaswg5]

Projects using NucleiSegmentation-ImageJ (v1.8.3)

NUCLEI-SEGMENTATION

Parameters

Radius (`ij_radius`)

Type	Number
Default value	5
Required	No
Constraints	/

Threshold (`ij_threshold`)

Figure S2.12 Software information

Supplementary Section 3. Installing and populating BIAFLOWS locally

Installing a local instance of BIAFLOWS

To apply a workflow on local images or to benchmark custom workflows on images stored in the online instance, it can be convenient to install a local instance of BIAFLOWS and populate it. This procedure is described below. Local instances of BIAFLOWS can be installed on UNIX-based operating systems. The installation procedure which follows is for Linux Ubuntu. Some specific details related to deployment on Mac OS can be found here: Install Cytomine on Mac OS.

1/ Install requirement

Cytomine runs in Docker containers, so that the only requirement is to install Docker. Check official Docker documentation to install Docker for Ubuntu. In particular, choose Install using the repository, set up the repository and install Docker CE.

2/ Retrieve BIAFLOWS installation files

```
mkdir Biaflows/
cd Biaflows/
git clone https://github.com/Neubias-WG5/Cytomine-bootstrap.git
cd Cytomine-bootstrap
```

3/ Configure the local instance

Edit configuration.sh file and if needed, update URLs (CORE_URL, IMS_URL, UPLOAD_URL). Be sure to use URLs that are not already used by other applications (avoid localhost) to prevent conflicts. Add the XXX_URL variable values into the `/etc/hosts` of the host machine. In the /etc/hosts, add the following lines and don't forget to adapt them with values chosen for XXX_URL variables.

```
127.0.0.1   localhost-core
127.0.0.1   localhost-ims
127.0.0.1   localhost-upload
127.0.0.1   rabbitmq
```

If needed, update data paths (IMS_STORAGE_PATH, …). All data paths must be valid and mappable in the Docker engine. Don't forget to create all these directories (mkdir) if they don't exist.

Configure `BIAFLOWS_WORKFLOWS_METRICS` to *true* or *false* depending if you want to perform benchmarking (annotated images required) or only plan to process images.

4/ Initialize the deployment

Generate the installation script with the command:

```
sudo bash init.sh
```

5/ Deploy the local instance

Run the generated deployment script with the command

```
sudo bash start.sh
```

6/ Check the running instance

When start up is finished, check the application is running in your browser on the URL specified in your CORE_URL variable (by default: localhost-core).

An admin account is created by default (username: admin ; password: admin). The password should be updated once connected in the Account page at the top right.

7/ Install sample projects (images and ground-truth data)

After BIAFLOWS successfully installed locally, this local instance is still empty of data. Most of projects available on BIAFLOWS maintained by NEUBIAS can be imported to this local instance.

Get the public and private keys of the admin account (at the end of the Account page). Then, run

```
cd Cytomine-bootstrap
sudo bash ./inject_demo_data.sh ADMIN_PUBLIC_KEY ADMIN_PRIVATE_KEY
```

where `ADMIN_PUBLIC_KEY` and `ADMIN_PRIVATE_KEY` have been substituted by their value.

The script starts to download projects and import them in your local BIAFLOWS.

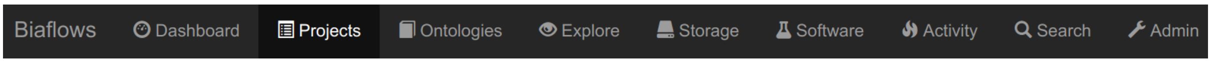
The list of imported projects can be tweaked by editing the file

```
 Cytomine-bootstrap/configs/project_migrator/projects.txt
```

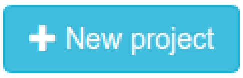
The whole data injection procedure can take several minutes, depending on your internet connection and the number of projects being imported.

Creating a new project in a local instance of BIAFLOWS

Click ***Projects*** tab from upper banner

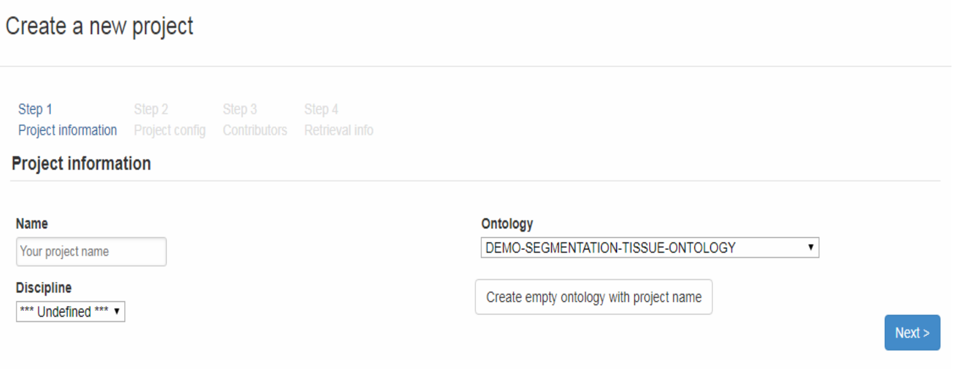


Click ***New Project***



Fill project information:

a) Choose a meaningful project name
b) Click “Create empty ontology with project name”
c) Choose the Discipline corresponding to the project BIA task (see Supplementary Section 5)
d) Click Next.



Create a new project

Step 1 Project information | Step 2 Project config | Step 3 Contributors | Step 4 Retrieval info

Project information

Name: Your project name

Discipline: *** Undefined ***

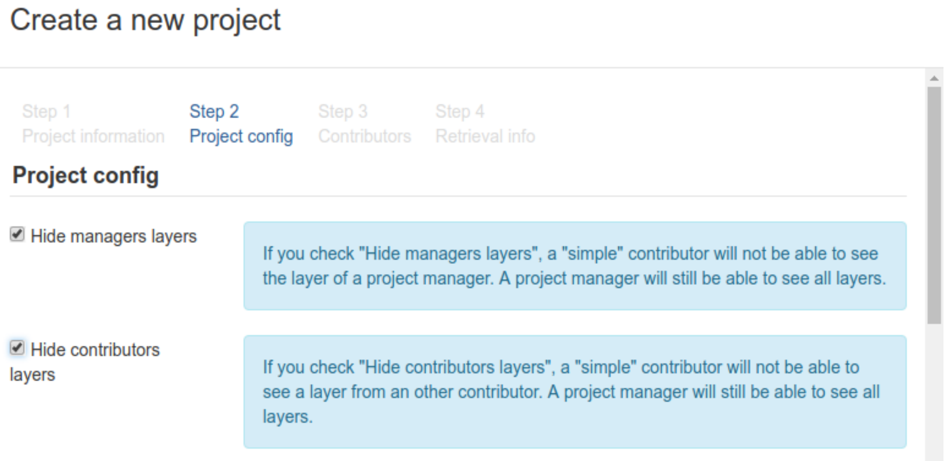
Ontology: DEMO-SEGMENTATION-TISSUE-ONTOLOGY

Create empty ontology with project name

Next >

Configure the project, notably:

e) Tick “Hide managers layers”
f) Tick “Hide contributors layers”



Create a new project

Step 1 Project information | Step 2 Project config | Step 3 Contributors | Step 4 Retrieval info

Project config

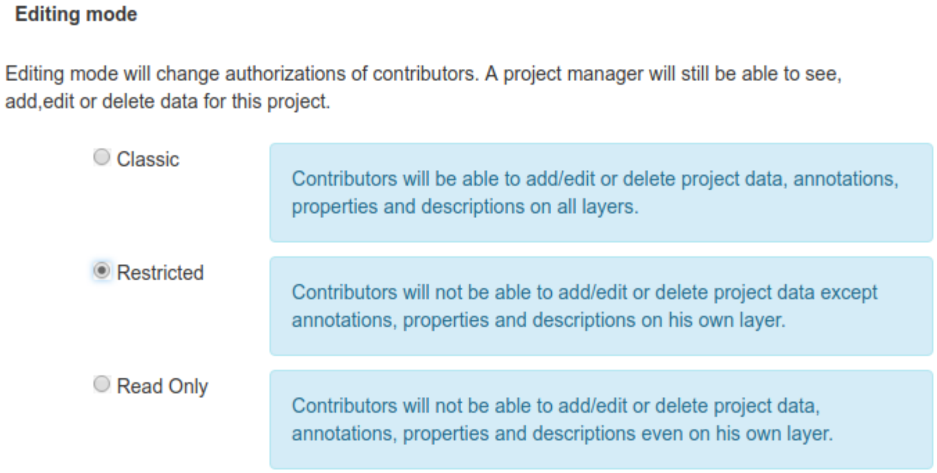
☑ Hide managers layers

If you check "Hide managers layers", a "simple" contributor will not be able to see the layer of a project manager. A project manager will still be able to see all layers.

☑ Hide contributors layers

If you check "Hide contributors layers", a "simple" contributor will not be able to see a layer from an other contributor. A project manager will still be able to see all layers.

g) Scroll down and select “Restricted editing mode”



Editing mode

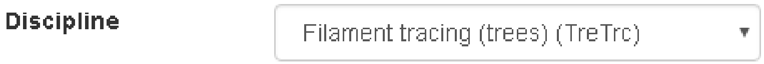
Editing mode will change authorizations of contributors. A project manager will still be able to see, add,edit or delete data for this project.

○ Classic — Contributors will be able to add/edit or delete project data, annotations, properties and descriptions on all layers.

● Restricted — Contributors will not be able to add/edit or delete project data except annotations, properties and descriptions on his own layer.

○ Read Only — Contributors will not be able to add/edit or delete project data, annotations, properties and descriptions even on his own layer.

h) Set the Discipline according to the BIA task



Discipline Filament tracing (trees) (TreTrc) ▾

Disciplines specify the format of ground truth annotations (and workflow outputs) as well as the associated benchmark metrics (see Supplementary Section 5 for details).

i) Click Next.

Configure project members. If you work alone, you can leave contributors and project managers to default user.

j) Click Next and disable retrieval information

k) Create the project by clicking Save

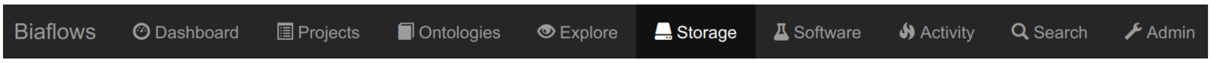
A description of the project can be added from Explore Tab by pressing Edit description .

<u>Note</u>: Projects can be fully configured to display or hide panels / tabs / tools in the user interface. This is achieved from the *Configuration* tab.

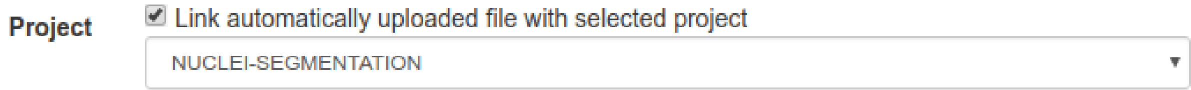
Uploading images to an existing project of a local instance of BIAFLOWS

Very important:
2D images must be uploaded as 8-bit TIFF and multi-dimensional images (Z, C, T) must be uploaded as single file 8-bit OME-TIFF. Images should not have the text string _lbl in their name.

1) Click ***Storage*** tab from upper banner



2) **Tick "Link automatically uploaded file with selected project"** and select the project you want to link the images to, for example:


Project
☑ Link automatically uploaded file with selected project
NUCLEI-SEGMENTATION

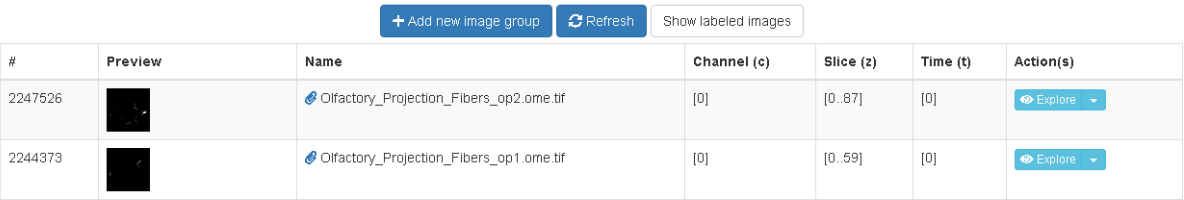
Note: If you don't see a project in the list, be sure that you are member of the project!

3) Click + Add files...

4) Select the files from the file browser

5) Click + Start upload and wait until completion.

Note: images can also be associated to a project once in storage without ticking link to project from the *Images* (2D images) / *ImageGroups* (multidimensional images) tabs. This can be useful to associate the same image to several projects. To do so use + Add new image and + Add new image group.



#	Preview	Name	Channel (c)	Slice (z)	Time (t)	Action(s)
2247526		Olfactory_Projection_Fibers_op2.ome.tif	[0]	[0..87]	[0]	Explore
2244373		Olfactory_Projection_Fibers_op1.ome.tif	[0]	[0..59]	[0]	Explore

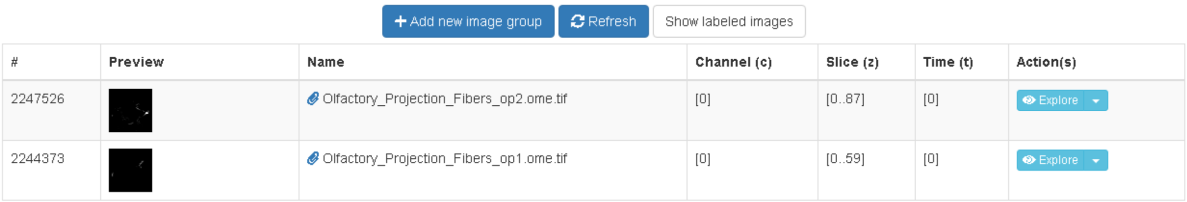
Uploading ground truth annotations to an existing project of a local instance of BIAFLOWS

If you plan to perform benchmarking, ground truth annotations should also be uploaded and associated to the images of the project. The format of these annotations depends on the project discipline (see Supplementary Methods section 5).

Image annotations (e.g. binary masks) should be uploaded as 16-bit TIFF for 2D images and single file 16-bit OME-TIFF for multidimensional images. These masks can for instance be created with ImageJ. Annotation images should be uploaded as described in previous section, and with same name as their corresponding image + *_lbl* suffix (e.g. MyImage.ome.tif and MyImage_lbl.ome.tif).

Other types of annotations (e.g. SWC, text file) should be added to the images as attached files, which can be achieved from the 📎 icon in the image list from *Images* / *ImageGroups* tabs.

Note: Only the last uploaded file will be taken into account during metrics computation.



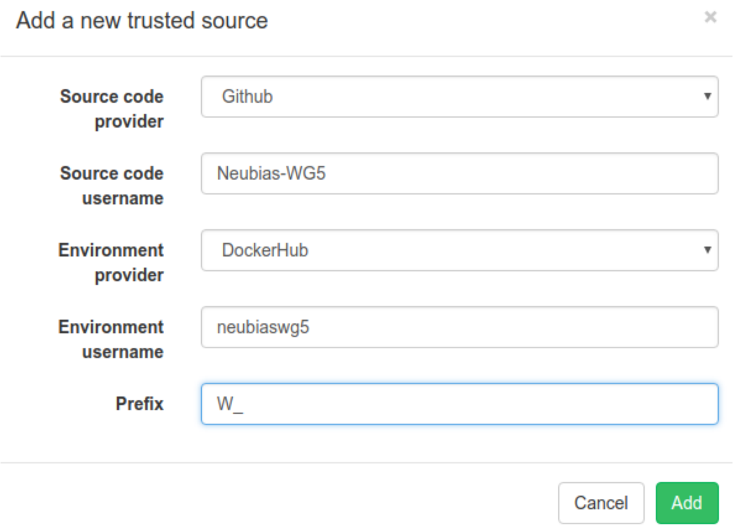
#	Preview	Name	Channel (c)	Slice (z)	Time (t)	Action(s)
2247526		📎 Olfactory_Projection_Fibers_op2.ome.tif	[0]	[0..87]	[0]	Explore
2244373		📎 Olfactory_Projection_Fibers_op1.ome.tif	[0]	[0..59]	[0]	Explore

Adding existing workflows from trusted sources to a local instance of BIAFLOWS

It is possible to integrate existing BIAFLOWS ready workflows to any BIAFLOWS instance. This operation requires configuring an external trusted source composed of (1) a source code registry (typically a Github user space) and (2) an execution environment registry (typically a DockerHub user space). If you have a user space where workflow repositories are mixed with other repositories, you can specify a prefix to distinguish workflows repositories from other ones.

For instance, all bioimage analysis workflows developed by NEUBIAS are prefixed by W_ and available here: https://github.com/Neubias-WG5.

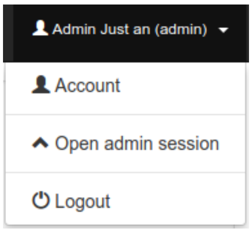
The trusted source has to be configured as follows:



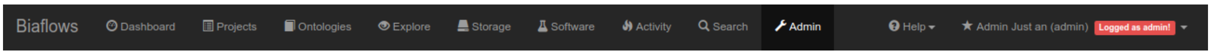
Add a new trusted source

Source code provider	Github
Source code username	Neubias-WG5
Environment provider	DockerHub
Environment username	neubiaswg5
Prefix	W_

To add a new trusted source, connect as an administrator and grant administration privileges by clicking Open admin session at the top-right:



Then go to the Admin page:



In the Softwares tab, click on + Add trusted source and fill the form. Send the form and click on Refresh software from trusted sources to add workflows from this trusted source to the local BIAFLOWS instance. Trusted sources are periodically checked to automatically add new workflows (new versions of existing workflows or completely new workflows).

Supplementary Section 4. Adding new image analysis workflows to a local instance of BIAFLOWS

Sample workflows running in FIJI, ICY, CellProfiler, ilastik, Vaa3D, or Python can be found in this GitHub repository: https://github.com/neubias-wg5.

Users willing to get help on packaging a workflow to this format to add it to a BIAFLOWS instance are redirected to https://forum.image.sc. Interested users can contact biaflows@neubias.org to include a packaged workflow to the online instance of BIAFLOWS.

Requirements

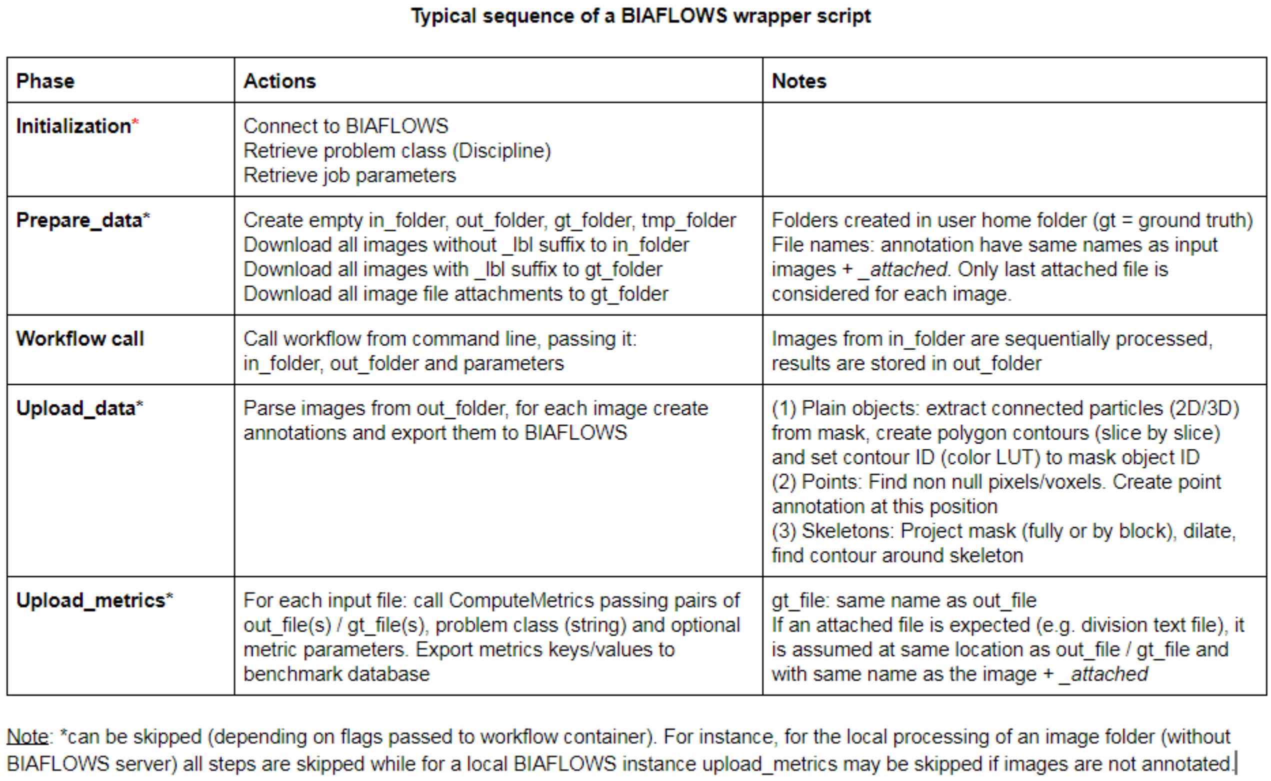
All image analysis workflows in BIAFLOWS must:

- Run headless from command line
- Take an input image folder: 8-bit TIFF images (2D) or single file 8-bit OME-TIFF images (C,Z,T)
- Expose functional parameters and parse them from the command line call
- Export results in an output folder (parsed from command line call). The format of the results is problem class (discipline) dependent and described in Supplementary Methods section 5.

The workflow and its software execution environment should be described in an authorized GitHub repository. The repository should be made of 4 files:

- A Docker recipe (Dockerfile) configuring the software execution environment (required OS/libraries,...). All necessary software dependencies must be specified in DockerFile (plugins, libraries to install, etc.); some examples can be found here: https://github.com/Neubias-WG5
- A JSON descriptor (descriptor.json) specifying the workflow functional parameters. This file also holds URL and credentials to communicate with a specific BIAFLOWS instance. A reference for this descriptor can be found here: https://doc.cytomine.be/display/ALGODOC/Software+JSON+descriptor+reference
- The workflow executable or, more commonly, a script running on a BIA platform callable from command line. So far these scripts were tested: ImageJ macros, ICY protocols, CellProfiler pipelines, ilastik pipelines, Vaa3D plugins, Python 2.X or 3.X code.
- A Python wrapper script (wrapper.py), the Docker entry point sequencing all operations. The steps executed by typical BIAFLOWS wrappers are described in figure S4.1.

Important: Workflow repositories must have a name starting by a fixed prefix which is chosen by the trusted source owner (e.g. W_ for the online instance of BIAFLOWS).



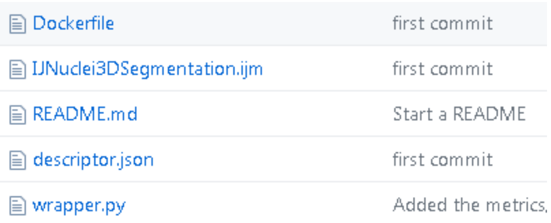
Typical sequence of a BIAFLOWS wrapper script

Phase	Actions	Notes
Initialization*	Connect to BIAFLOWS Retrieve problem class (Discipline) Retrieve job parameters	
Prepare_data*	Create empty in_folder, out_folder, gt_folder, tmp_folder Download all images without _lbl suffix to in_folder Download all images with _lbl suffix to gt_folder Download all image file attachments to gt_folder	Folders created in user home folder (gt = ground truth) File names: annotation have same names as input images + *_attached*. Only last attached file is considered for each image.
Workflow call	Call workflow from command line, passing it: in_folder, out_folder and parameters	Images from in_folder are sequentially processed, results are stored in out_folder
Upload_data*	Parse images from out_folder, for each image create annotations and export them to BIAFLOWS	(1) Plain objects: extract connected particles (2D/3D) from mask, create polygon contours (slice by slice) and set contour ID (color LUT) to mask object ID (2) Points: Find non null pixels/voxels. Create point annotation at this position (3) Skeletons: Project mask (fully or by block), dilate, find contour around skeleton
Upload_metrics*	For each input file: call ComputeMetrics passing pairs of out_file(s) / gt_file(s), problem class (string) and optional metric parameters. Export metrics keys/values to benchmark database	gt_file: same name as out_file If an attached file is expected (e.g. division text file), it is assumed at same location as out_file / gt_file and with same name as the image + *_attached*

Note: *can be skipped (depending on flags passed to workflow container). For instance, for the local processing of an image folder (without BIAFLOWS server) all steps are skipped while for a local BIAFLOWS instance upload_metrics may be skipped if images are not annotated.

Figure S4.1. Typical sequence of BIAFLOWS wrapper script

In this document we will illustrate how to create these four files by following this existing sample workflow: https://github.com/Neubias-WG5/W_NucleiSegmentation3D-ImageJ



Dockerfile	first commit
IJNuclei3DSegmentation.ijm	first commit
README.md	Start a README
descriptor.json	first commit
wrapper.py	Added the metrics,

Installing software required for development

As a workflow will run into a Docker container and interact with a BIAFLOWS instance, it is required to install Docker, Python 3, and Cytomine Python client in your local machine:

Docker

Linux https://goo.gl/Q9A5n9
Windows: https://goo.gl/gHi3Kr

Python 3 + Cytomine python

See https://doc.cytomine.be/display/ALGODOC/Data+access+using+Python+client

Step 1. **Uploading a workflow descriptor to BIAFLOWS**

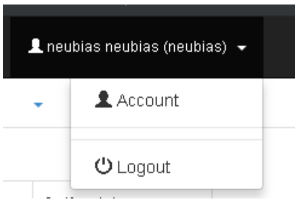
Workflows have first to be described through a JSON descriptor, e.g.: https://github.com/Neubias-WG5/W_NucleiSegmentation3D-ImageJ/blob/master/descriptor.json

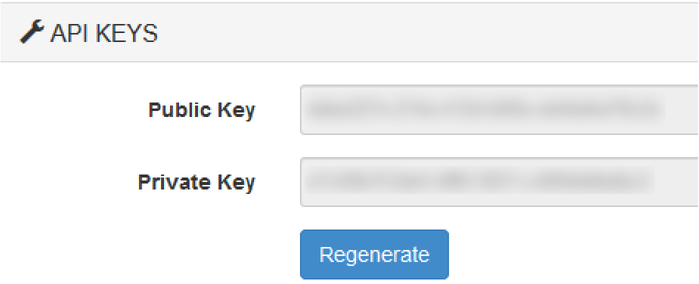
Currently, some sections have to be customized manually, especially the software parameters (and default values). We are planning a tool to create and publish a descriptor from the UI.

To make a workflow available in a BIAFLOWS instance, its descriptor currently needs to be published from the Python client. This can be performed by customizing the following code and running it inside the folder containing the JSON descriptor:

```
from cytomine import Cytomine
from cytomine.utilities.descriptor_reader import read_descriptor
with Cytomine(host, public_key, private_key) as c:
      read_descriptor("descriptor.json")
```

`host` is the url of your BIAFLOWS server, e.g. https://biaflows.neubias.org
`public_key` and `private_key` can be found from user Account page (section API KEYS)





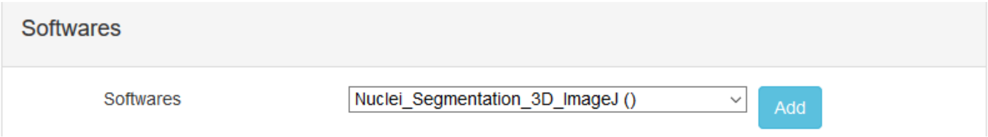
API KEYS

Public Key [redacted]

Private Key [redacted]

Step 2. **Linking a workflow to a BIAFLOWS project**

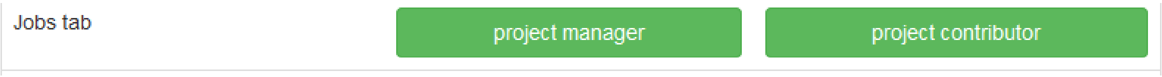
- In BIAFLOWS, *Projects*, select the project to which you want to add the workflow
- Go to *Projects* > *Configuration* > *Software* and select the software to add to the project from the list (e.g., Nuclei_Segmentation_3D_ImageJ-)



Softwares

Softwares: Nuclei_Segmentation_3D_ImageJ ()

- Go to *Projects* > *Configuration* and make sure that Jobs tab is activated (green)



Jobs tab

Step 3. Creating the Dockerfile

Docker files specify the execution environment. They typically start by creating (FROM) a layer from an existing Docker image with basic operating system. Then they execute commands (RUN) to install specific software and libraries, and copy (ADD) files (e.g. the Python wrapper script and workflow script) into the execution environment the workflow will be called from. Finally, the ENTRYPOINT is set to the wrapper script.

A sample DockerFile can be found here:
https://github.com/Neubias-WG5/W_NucleiSegmentation-ImageJ/blob/master/Dockerfile

If you do not know how to configure your Dockerfile it is recommended to adapt the Dockerfile from an existing BIAFLOWS workflow using the same target software you are planning to use. Workflows available in BIAFLOWS online instance are all stored here: https://github.com/Neubias-WG5/

Step 4. Creating the wrapper script

The same hold for wrapper scripts, if possible, it is recommended to adapt a wrapper script from 1) the same problem class (discipline), 2) image dimensionality (e.g. 3D image) and 3) matching the software you are planning to use. In this case, only the workflow call (command line) should have to be adapted.

A sample wrapper script is available here:
https://github.com/Neubias-WG5/W_NucleiSegmentation-ImageJ/blob/master/wrapper.py

Step 5. Building the workflow image, running it in a container and debugging

A workflow can be directly pushed to GitHub and linked to BIAFLOWS, but it is preferable to test it locally beforehand. For this, it is required to build and run the Docker image locally.

Building the container (you need around 5GB disk space for this operation)

From a directory where you gathered the 4 files required to describe the workflow:

```
cd ~/Documents/Code/NEUBIAS/W_Nuclei_Segmentation_3D_ImageJ$
```

```
sudo docker build -t seg3d .
```

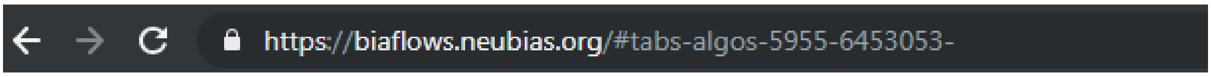
Note: `seg3d` is the name of the Docker image to build.

Running the container

After building the image, the workflow can be run locally using:

```
sudo docker run -it seg3d --host host --public_key public_key --private_key private_key --software_id software_id --project_id project_id --ij_min_threshold 15 --ij_radius 4
```

The list of command-line parameters should exactly match the parameters defined in the descriptor file. Biaflows instance URL and credentials should be filled as well as software_id / project_id. These can be retrieved as followed from *Jobs* tab by clicking on the name of the published workflow from available software section.


https://biaflows.neubias.org/#tabs-algos-5955-6453053-

The URL of this page contains 2 IDs: The first one is the project ID (e.g. **5955**), the second is the software ID (e.g. **6453053**)

Integrating new workflows to BIAFLOWS can lead to their runtime failure. In such situation, the execution of the workflow is reported as failed in the web interface, some log file is downloadable, and no associated benchmark metric is associated to this run. There is hence no risk that this would be left unnoticed by the user. In that situation, the problem should be solved as collaboration between the workflow developer and BIAFLOWS administrators. We will reflect this in the text. If the software run was not successful, you can run the Docker from an interactive session:

```
sudo docker run --entrypoint bash -it seg3d
```

If needed, it is also possible to launch with X capabilities, e.g. to debug imageJ macro more easily:

```
xhost + sudo docker run --entrypoint bash -v
/home/yourusername/tmp/test:/data -e DISPLAY=$DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix -it seg3d
```

If you want to access local images without having to download them each time from BIAFLOWS, you can attach a local folder to a folder inside the Docker container (-v option), for instance using:

```
sudo docker run --entrypoint bash -v /home/yourusername/tmp/test:/data -it seg3d
```

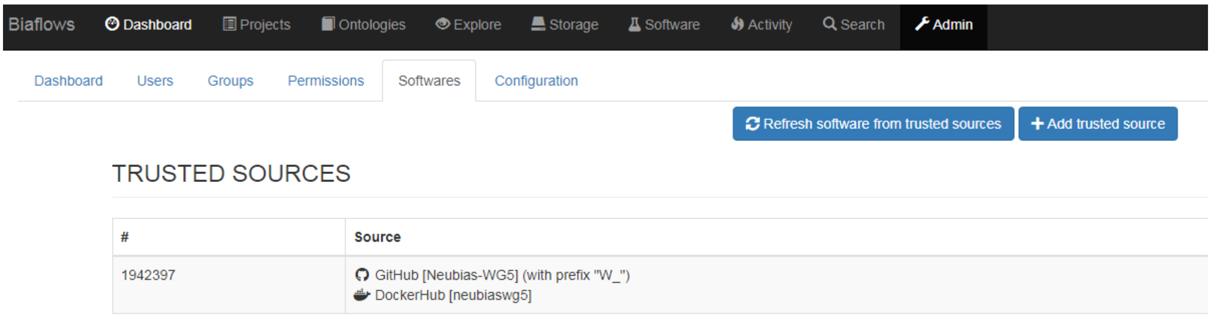
Some other useful Docker commands:

Check if an image is running:	ps -a
Kill a running container:	sudo docker rm 65e88b2015df
Kill all running containers:	sudo docker rm $(sudo docker ps -a -q)
Download a specific container	sudo docker pull neubiaswg5/fiji-base:latest

Note: to download a newly updated workflow image, is necessary to remove older versions manually first.

Step 6. **Publishing a workflow with version control**

Once your workflow is running properly, you can officially publish it with version control. To allow automatic import to BIAFLOWS, the set of files previously described should be stored in a GitHub repository (linked to DockerHub) from an account trusted by the BIAFLOWS instance. Adding / checking trusted repositories is performed from Admin / Software tab from an admin account (see Supplementary Methods section 3):



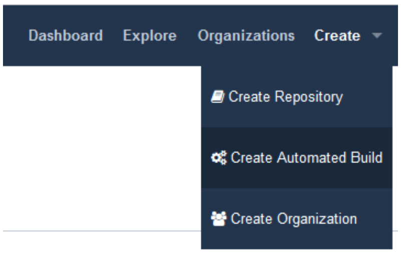
TRUSTED SOURCES

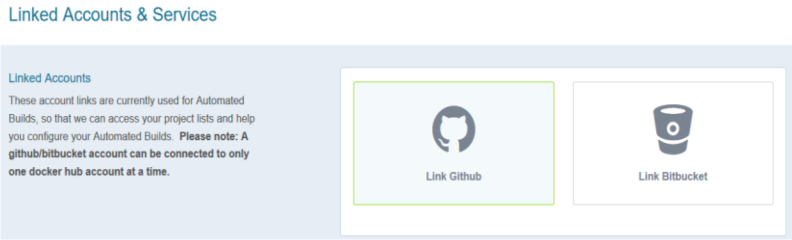
#	Source
1942397	GitHub [Neubias-WG5] (with prefix "W_") DockerHub [neubiaswg5]

Step 6. **Linking a GitHub repository to DockerHub**

Now we assume you created a GitHub organization (e.g. neubias-wg5) and a repository (e.g. W_NucleiSegmentation3D-ImageJ) containing the workflows files. It is now required to link DockerHub to GitHub. This operation has to be done only once for a given GitHub account:

1. Create an account on DockerHub : https://hub.docker.com/ and login.
2. Create an automatic build by linking Docker account to GitHub organization account.
3. In DockerHub website, click on Create>Create Automated Build.





These account links are currently used for Automated Builds, so that we can access your project lists and help you configure your Automated Builds. **Please note: A github/bitbucket account can be connected to only one docker hub account at a time.**

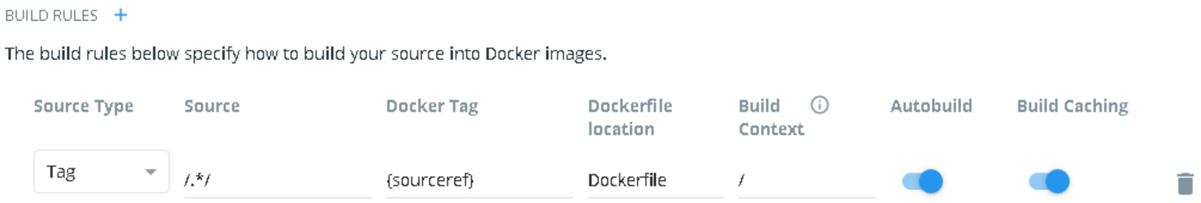
1. In **Linked Accounts,** click on **Link Github**
2. Click **Select**
3. Make sure Organization access (e.g. **Neubias-WG5)** is selected (green check mark) and click on **Authorize docker**
4. Enter your GitHub password to enable access

Step 7. **Associating a new workflow repository to DockerHub**

Once your Github organization account and DockerHub are linked, it is now possible to create and automated build procedure for each workflow. This procedure will build a workflow image that will later be downloaded automatically by a BIAFLOWS instance.

To do so, from DockerHub:

1. Click on Create>Create Repository+
2. In build settings click on GitHub icon
3. Select organization (e.g. **neubiaswg5**) and workflow Github repository (e.g. W_NucleiSegmentation3D-ImageJ) at the bottom of the page
4. Enter a short description (less than 100 characters) and click **Create**
5. Click on *Click here to customize the build settings* and configure as in figure below
6. Click on Save



BUILD RULES +

The build rules below specify how to build your source into Docker images.

Source Type	Source	Docker Tag	Dockerfile location	Build Context	Autobuild	Build Caching
Tag	/.*/	{sourceref}	Dockerfile	/	●	●

Step 8. **Creating a versioned release on GitHub**

Following this procedure, a Docker image of your workflow will only be built once a GitHub release is triggered and tagged from GitHub.

To create versioned releases of your workflow, go to GitHub and draft a new release (see https://goo.gl/bFz66N). This will add a new tag to the last commit. As we configured automatic build in previous step (with a new Docker Hub tag) for every new Github tag, a new Docker image will now be automatically built and published with the same tag as provided on Github.

The BIAFLOWS instance that trusts this Github/Docker Hub repository will automatically retrieve and add this new version. The workflow will then be executable from the BIAFLOWS instance.

Supplementary Section 5. BIAFLOWS ground truth annotations (and workflow output) expected formats and reported metrics

To perform benchmarking, ground truth annotations associated to the images of a BIAFLOWS project should be encoded in a format that is specific to its discipline (BIA problem class). Most annotations are stored as mask images as TIFF / OME-TIFF images. The BIA workflows of a project are also expected to output results in the same format. Currently 8 disciplines are supported and their respective annotation formats and associated benchmark metrics are described below.

Note: each discipline has a long name (explicit) and short name, for instance Object Segmentation (ObjSeg). The same hold for metrics, for instance DICE (DC).

See supplementary section 7 for metrics description.

Discipline: Object Segmentation (ObjSeg)

Task: Delineate objects or isolated regions

Object Encoding: 2D/3D binary masks with foreground > 0, background = 0

Reported metrics: DICE (DC), AVERAGE_HAUSDORFF_DISTANCE (AHD), computed by VISCERAL [1] executable (archived here).

Discipline: Spot / object counting (SptCnt)

Task: Estimate the number of objects

Object Encoding: 2D/3D binary masks, exactly 1 spot/object per non null pixel

Reported metrics: RELATIVE_ERROR_COUNT (REC), computed by custom Python code.

Discipline: Spot / object detection (ObjDet)

Task: Detect objects in an image (e.g. nucleus)

Object Encoding: 2D/3D binary masks, exactly 1 object per non null pixel.

Reported metrics: CONFUSION_MATRIX (TP, FN, FP), F1_SCORE (F1), PRECISION (PR), RECALL (RE), Distance RMSE (RMSE), computed by Particle Tracking Challenge metric Java code (particle matching only) (archived here in bin/DetectionPerformance.jar).

Discipline: Pixel/Voxel Classification (PixCla)

Task: Estimate pixels class

Object Encoding: 2D/3D class masks, gray level encodes pixel/voxel class, background = 0

Reported metrics: F1_SCORE (F1), ACCURACY (ACC), PRECISION (PR), RECALL (RE), computed by custom Python code.

Discipline: Filament Tree Tracing (TreTrc)

Task: Estimate the medial axis of a single filament tree

Object Encoding: SWC file

Reported metrics: DIADEM metric (DM) computed by DIADEM challenge code (archived here).

Discipline: Filament Networks Tracing (LooTrc)

Task: Estimate the medial axis of one or several network(s) of filaments

Object Encoding: 2D/3D skeleton binary masks with skeleton pixels > 0, background = 0

Reported metrics:
UNMATCHED_VOXEL_RATE (UVR), computed by custom Python code.
NetMets metrics: Geometric False Negative rate (FNR), Geometric False Positive rate (FPR) computed by NetMets Python code.

Metrics parameters:
UVR: GATING_DIST (default = 5, maximum distance between skeleton voxels in reference and prediction skeletons to be considered as matched).
NetMets: skeleton pixel_sampling (default = 3), sigma (= GATING_DIST), subdiv (set to 4).

Discipline: Landmark Detection (LndDet)

Task: Estimate the position of specific feature points

Object Encoding: 2D/3D class masks, exactly 1 landmark per non null pixel, gray level encodes landmark class (1 to N, N is the number of landmarks).

Reported metrics: Number of reference / predicted landmarks (NREF, NPRED), Mean distance from predicted landmarks to closest reference landmarks with same class (MRE). All metrics computed by custom Python code.

Discipline: Particle Tracking (PrtTrk)

Task: Estimate the tracks of object centroid (no division)

Object Encoding: 2D/3D label masks, exactly 1 particle per non null pixel, gray level encodes particle track ID.

Reported metrics: Pairing distance (PD), Normalized pairing score alpha (NPSA), Full normalized pairing score beta (FNPSB), Number of reference tracks (NRT), Number of candidate tracks (NCT), Jaccard Similarity Tracks (JST), Number of paired tracks (NPT), Number of missed tracks (NMT), Number of spurious tracks (NST), Number of reference detections (NRD), Number of candidate detections (NCD), Jaccard similarity detections (JSD), Number of paired detections (NPD), number of missed detections (NMD), Number of spurious detections (NSD). All computed from Particle Tracking Challenge metric Java code (archived here in bin/TrackingPerformance.jar).

Metrics parameters: GATING_DIST (default = 5, maximum distance between particle detections in reference / prediction tracks to be considered as matched).

Discipline: Object Tracking (ObjTrk)

Task: Estimate object tracks and contours (with possible divisions)

Encoding: 2D/3D TIFF label masks, gray level encodes object ID + division text file (see Cell Tracking Challenge format)

Reported metrics: Segmentation measure (SEG), Tracking measure (TRA). All computed from Cell Tracking Challenge metric command-line executables (archived here in bin/SEGMeasure and bin/TRAMeasure).

Supplementary Section 6. BIAFLOWS Additional features

Executing a BIAFLOWS workflow without BIAFLOWS server

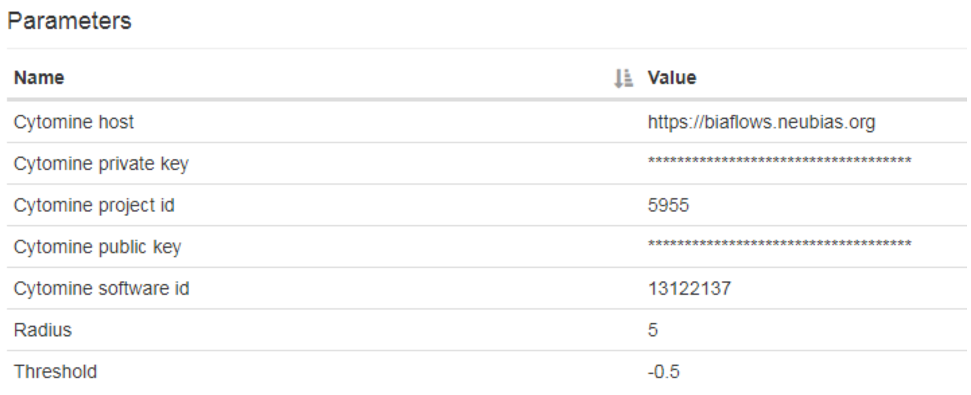
It is possible to run a workflow image independently of any BIAFLOWS server. This can for instance be useful to process a local folder of images. For this, first install Docker on the target workstation, then:

- Get the docker image of the workflow from Dockerhub:
  ```
  docker pull {remote_image}
  ```
  Or alternatively build its Docker image from source (GitHub repository)
  Inside repository folder: `docker build -t {local_image} .`

- Prepare an empty folder {DATA_PATH} with a subfolder **/data** itself with subfolders:
  - ○ **{DATA_PATH}/data/in**: add input images to this folder. Images should be 8-bit TIFF (2D) or 8-bit single file OME-TIFF (C,Z,T). String _lbl is forbidden in image name.
  - ○ **{DATA_PATH}/data/out**: workflow results are exported to this folder
  - ○ **{DATA_PATH}/data/gt**: leave empty

- Run the workflow with these specific flags:
  ```
  docker run -v {DATA_PATH}/data:/data -it {image_name}
  {WORKFLOW_PARAMETERS} --infolder /data/in --gtfolder /data/gt --
  outfolder /data/out --nodownload --noexport --nometrics
  ```

The workflow parameters that were used for a specific run (job) can be found from Details ▾ in BIAFLOWS (Jobs tab). Concretely parameters are all non Cytomine variables. For the example below **{WORKFLOW_PARAMETERS}** should be --Radius 5 --Threshold -0.5.



Parameters

Name	Value
Cytomine host	https://biaflows.neubias.org
Cytomine private key	************************************
Cytomine project id	5955
Cytomine public key	************************************
Cytomine software id	13122137
Radius	5
Threshold	-0.5

Notes on the flags:

`--nodownload`: inputs images (and files) are not downloaded from a BIAFLOWS server, instead they are read locally
`--noexport`: results are not uploaded to any BIAFLOWS instance
`--nometrics`: metrics are not computed, nor uploaded to any BIAFLOWS instance.

Importing existing datasets from a BIAFLOWS instance

Content migration from an existing instance (e.g. BIAFLOWS online instance) to a local instance is possible. To migrate data, we developed tools that rely on the BIAFLOWS RESTful programming interface to export project data (including images and object annotations) from the source instance to the destination instance. Corresponding code and documentation is available here and can be adapted for specific purposes:
https://github.com/Neubias-WG5/Cytomine-project-migrator

Manual Import of annotations

In order to be able to compute metrics for benchmarking (optional), ground-truth annotations should also be provided for all uploaded images and encoded with format specified in Supplementary Methods section 5.

Workflow results use the same formats as ground truth annotations, but to be visualized in BIAFLOWS they are internally converted to a spot / contour format. If needed, it is possible to manually convert annotations from a provided Python library (e.g. from image masks or CSV files). Supported formats include 2D, 3D/2D+t and 3D+t objects. The implementation is a part of the neubiaswg5-utilities library (https://github.com/Neubias-WG5/neubiaswg5-utilities). These annotations can then be uploaded to BIAFLOWS and displayed using overlays in the web image viewer as any annotation automatically created from the output of a workflow.

RESTful API documentation

All the interactions with BIAFLOWS are performed through a RESTful API. It allows linking the server side and a BIAFLOWS client, such as the web user interface or a Java / Python client (used in workflows to communicate with the core server). All available services provided by the API are summarized in a user-friendly way on a website automatically installed with BIAFLOWS installation procedure. On this interface, the documentation can be browsed and API responses directly tested in a playground area.

BIAFLOWS RESTful API documentation can be found here:
https://biaflows.neubias.org/restApiDoc/?doc_url=https://biaflows.neubias.org/restApiDoc/api#

Supplementary Section 7. BIAFLOWS Benchmarking Metrics

Object Segmentation (ObjSeg)

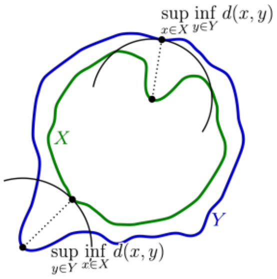
DICE (DICE)

DICE coefficient is computed as twice the overlap area (hence similarity) between the ground truth objects (***X***) and the objects estimated by the workflow (***Y***) normalized to the sum of the overall areas of both ground truth and estimated objects, that is:

$$\frac{2 * |X \cap Y|}{|X| + |Y|}$$

DICE coefficient ranges from 0 (no overlap) to 1 (perfect overlap).

AVERAGE HAUSDORFF DISTANCE (AHD)


$\sup_{x\in X}\inf_{y\in Y} d(x,y)$
X
Y
$\sup_{y\in Y}\inf_{x\in X} d(x,y)$

Given a point on the contour of an object and the contour of another object, the minimum distance is the distance from that point to any other point on the other contour. The Hausdorff distance between the contours of a ground truth object (***X***) and an object estimated by the workflow (***Y***) is defined as the maximum of the minimum distances computed for any point of any of the two contours (respect the other contour). The Average Hausdorff distance is the average of the Hausdorff distance computed for all the objects. The average Hausdorff distance (AHD) is a positive unbounded number in pixel units. AHD is equal to 0 only for perfect object contour estimation. Whereas DICE coefficient takes into account objects as a whole, a large AHD may only reflect that an object as a large deviation to the ground truth object for a single point (big “bump” or “dip”).

Spot / object counting (ObjCnt)

RELATIVE_ERROR_COUNT (REC)

The relative error count is computed as the absolute difference between the number of ground truth and the number of estimated objects, normalized to the count of the ground truth objects. REC is a positive unbounded number only equal to 0 for perfect count. A larger REC means a worse object count.

Spot / object detection (ObjCnt)

CONFUSION_MATRIX (TP, FN, FP)
F1_SCORE (F1)
PRECISION (PR)
RECALL (RE)
Distance RMSE (RMSE)

Given a set of ground truth detections and a set of detections produced by an evaluated algorithm, these two sets are first associated by solving a distance-constrained assignment problem using the Hungarian algorithm. This procedure pairs reference detections with candidate detections, not being farther than a given maximum distance, by minimizing their overall distance. Unpaired reference detections correspond to false negatives (FN), whereas unpaired candidate detections correspond to false positives (FP). Finally, paired detections are considered true positives (TP). As true negatives (TN) are not applicable here, their number is fixed at 0. See Pixel/Voxel classification (binary classification) for the definition of CONFUSION_MATRIX, F1_SCORE, PRECISION and RECALL.

Distance RMSE, the root mean square error, indicates the overall localization accuracy of optimally paired detections (i.e., TP detections), being a nonnegative number with the upper bound given by the maximum distance specified.

Pixel/Voxel Classification (PixCla)

F1_SCORE (F1)
ACCURACY (ACC)
PRECISION (PR)
RECALL (RE)

Given a classification problem with C distinct classes, a confusion matrix is a square matrix of dimensions C x C where the element aij (i ∈ [0, C[, j ∈ [0, C[) is the number of samples/pixels of class j that are predicted to be class i. The matrix diagonal contains correctly predicted samples.

In binary classification (C = 2), the elements of the confusion matrix have a particular meaning:

- a00: true negative (TN)
- a01: false negative (FN)
- a10: false positive (FP)
- a11: true positive (TP)

The accuracy (ACC) is the proportion of correctly predicted samples (the matrix trace divided by the total number of samples). The accuracy ranges from 0 (all samples misclassified) to 1 (all samples correctly classified). A classifier that would pick a class at random typically yields accuracy around 1/C. In binary classification, we have:

ACC = (TP + TN) / (TP + TN + FP + FN)

And precision (PR):

PR = TP / (TP + FP)

Intuitively, the precision is the ability of the classifier not to label as positive a sample that is negative. It ranges from 0 (all positive samples misclassified) to 1 (no false positive). In multiclass classification (C > 2), we use the weighted averaged precision: precision is computed for each class separately and then the resulting precisions are averaged weighted by support (number of positive samples for each class). This weighting strategy accounts for class imbalance. In binary classification the recall (RE) is:

RE = TP / (TP + FN)

Intuitively, the recall is the ability of the classifier to find all positive samples. It ranges from 0 (all positive samples misclassified) to 1 (no false negative). In multiclass classification, we use the same approach as for PR (i.e. weighted average recall).

Recall and precision must be analyzed jointly and it makes no sense to benchmark one or the other independently. For instance, for a balanced binary classification problem, it is easy to get a perfect recall of 1 with a constant classifier that would always predict the positive class but this classifier would yield a precision of 0.5 (which is the score a random classifier would get). When tuning a classifier, there is usually a trade-off between precision and recall. In binary classification, the f1-score (F1) is:

F1 = 2 * (PR * RE) / (PR + RE)

Intuitively, it is the weighted average of precision and recall. It ranges from 0 (precision and/or recall are 0) to 1 (all samples correctly classified). In multiclass classification, we use the same approach as for PR and RE (average weighted f1-score).

Filament Tree Tracing (TreTrc)

DIADEM metric

Full description: http://diademchallenge.org/metric.html

Citation: *Gillette, T. A., Brown, K. M., & Ascoli, G. A. (2011). The DIADEM metric: comparing multiple reconstructions of the same neuron. Neuroinformatics, 9(2-3): 233-245*.

Summary: The DIADEM metric compares topologically two registered digital morphological reconstructions of one similar neuron. The metric is a multi-step process that scores the connection between each node in the gold standard (e.g., manual) reconstruction based on whether or not the test (e.g. automated) reconstruction captures that connection. Each connection is weighted by the size of the subtree to which a connection leads in terms of terminal degree. An option exists to consider excess of nodes not present in the gold standard and therefore lower the score. This produces a score between 0 and 1 that reflects topological similarity, with a score of 1 reflecting a perfect score.

Filament Networks Tracing (LooTrc)

NetMets metric

Geometric False Negative rate (G_{FNR})
Geometric False Positive rate (G_{FPR})

Citations: Mayerich, D., Bjornsson C.,Taylor J., Roysam B. (2012). NetMets: software for quantifying and visualizing errors in biological network segmentation. BMC Bioinformatics, 13(Suppl 8): S7.

Summary: Similarly to the DIADEM metric, NetMets relies on mapping of branch points and end points between the ground truth and test case. Given two fiber networks N_1 and N_2, it estimates the ratio of the length of fiber in N_1 that has no correspondence in N_2 to the total fiber length in N_1. This estimate is computed by placing an implicit Gaussian envelope around N_2 and integrating along the set of curves representing fibers in N_1. In order to quantify both missed fibers and false positives, a bi-directional measurement, comparing N_1 to N_2 as well as comparing N_2 to N_1. In general terms, the more distant nodes in N_2 are from the nearest nodes in N_1, the value of M decreases. Thus, this value reflects that if nodes in a tree are distant from the other tree, there is a geometry error in that region where those nodes are. In addition, due to the bidirectional nature of this calculation, it is possible to obtain both the false negative rate G_{FNR} and false positive rate G_{FPR} of a test tree compared to the ground truth.

Equation:

$$\boldsymbol{M(N_1, N_2) = \frac{1}{n}\sum_{x\in N_1}^{n}(1 - e^{-\frac{d(x,N_2)^2}{2\sigma^2}})}$$

N_1,N_2 : fibers network
n : number of grid points in N_1
$d(x,N_2)$: distance between x (belonging to N_1) and the closest point in N_2, used to weight the distance field based on the geometry of N_2.
σ : sensitivity parameter use to increase/decrease the weighted distance field.
$G_{FNR} = \boldsymbol{M}(\boldsymbol{N_{GT}}, \boldsymbol{N_T})$: Fraction of Ground True not present in Test (False Negative Rate), between 0 and 1, lower values are better
$G_{FPR} = \boldsymbol{M}(\boldsymbol{N_T}, \boldsymbol{N_{GT}})$: Fraction of Test not present in Ground True (False Positive Rate) , between 0 and 1, lower values are better.

Landmark Detection (LndDet)

Number of ground truth landmarks (NREF)
Number of predicted landmarks (NPRED)

Mean distance (MRE):
Mean distance from predicted landmarks to ground truth landmarks (in pixels).

Particle Tracking (PrtTrk)

14 metrics from PTC challenge, see supplementary note 3 for complete description in: https://media.nature.com/original/nature-assets/nmeth/journal/v11/n3/extref/nmeth.2808-S1.pdf.

5 main metrics are derived from these 14 metrics:

1. $\alpha(X, Y) = 1 - d(X, Y)/d(X, \varnothing)$. Ø denotes a set of dummy tracks; hence, $d(X, \varnothing)$ is the maximum possible total distance (error) from the ground truth. The measure ranges from 0 (worst) to 1 (best), indicating the overall degree of matching of ground truth and estimated tracks without taking into account spurious (non-paired estimated) tracks.

2. $\beta(X, Y) = (d(X, \varnothing) - d(X, Y))/(d(X, \varnothing) + d(Y, \varnothing))$. Y denotes the set of spurious tracks, and $d(Y, \varnothing)$ is the corresponding penalty term. The measure ranges from 0 (worst) to α (best) and is essentially α with a penalization of non-paired estimated tracks.

3. JSC = TP/(TP + FN + FP). This is the Jaccard similarity coefficient for track points. It ranges from 0 (worst) to 1 (best) and characterizes overall particle detection performance. TP (true positives) denotes the number of matching points in the optimally paired tracks; FN (false negatives), the number of dummy points in the optimally paired tracks; and FP (false positives), the number of non-matching points including those of the spurious tracks.

4. $JSC\theta = TP\theta/(TP\theta + FN\theta + FP\theta)$. This is the Jaccard similarity coefficient for entire tracks instead of single track points. Similarly to JSC, it ranges from 0 (worst) to 1 (best). TPθ denotes the number of estimated tracks paired with ground-truth tracks; FNθ, the number of dummy tracks paired with ground-truth tracks; and FPθ, the number of spurious tracks.

5. RMSE, the r.m.s. error, indicates the overall localization accuracy of matching points in the optimally paired tracks (the TP as in JSC), being a nonnegative number with the upper bound given by the maximum distance specified.

Object Tracking (ObjTrk)

The metrics are taken from the Cell Tracking Challenge (http://celltrackingchallenge.net/) and are described in detail in https://doi.org/10.1038/nmeth.4473. The segmentation accuracy measure (SEG) evaluates the average amount of overlap, being expressed by the Jaccard similarity index, between the reference segmentation ground truth and the segmentation masks computed by an evaluated algorithm. The tracking accuracy measure (TRA) is a normalized weighted distance between the tracking solution submitted by the participant and the reference tracking ground truth, with weights chosen to reflect the effort it takes a human curator to carry out the edits manually (see https://doi.org/10.1371/journal.pone.0144959 for details).

Both SEG and TRA take values in the interval [0, 1], with higher values corresponding to better performance.

References

1. http://www.visceral.eu/resources/evaluatesegmentation-software/