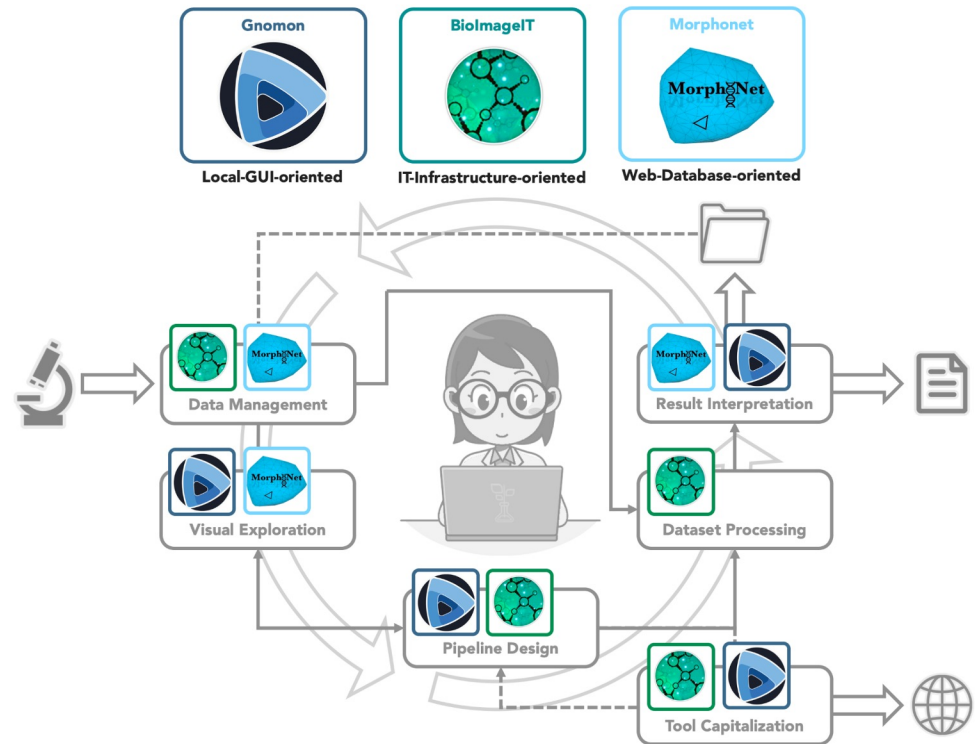


01

Gnomon & Naviscope

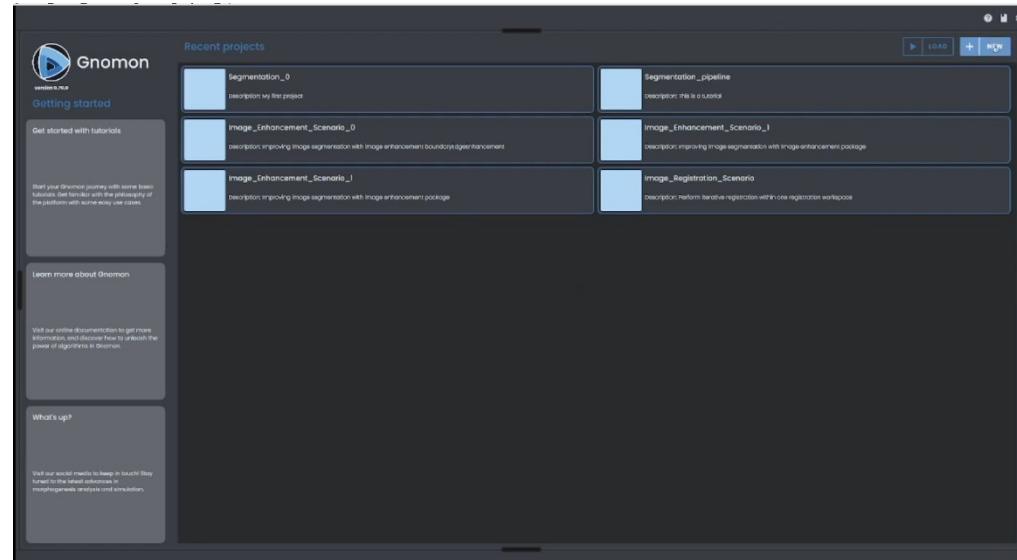
Naviscope

- Provide an easily accessible computational tool for the exploration of morphogenesis
- Bridge the gap between experimental data and computational simulations by offering the possibility to go from one to the other in the same platform
- Ensure the interoperability of computational libraries within the platform and its extensibility by a generalized plugin-based architecture



Gnomon

- **Environnement intégré d'expérimentation numérique reproductible pour l'étude des dynamiques des systèmes vivants à différentes échelles**
- **Données locales**

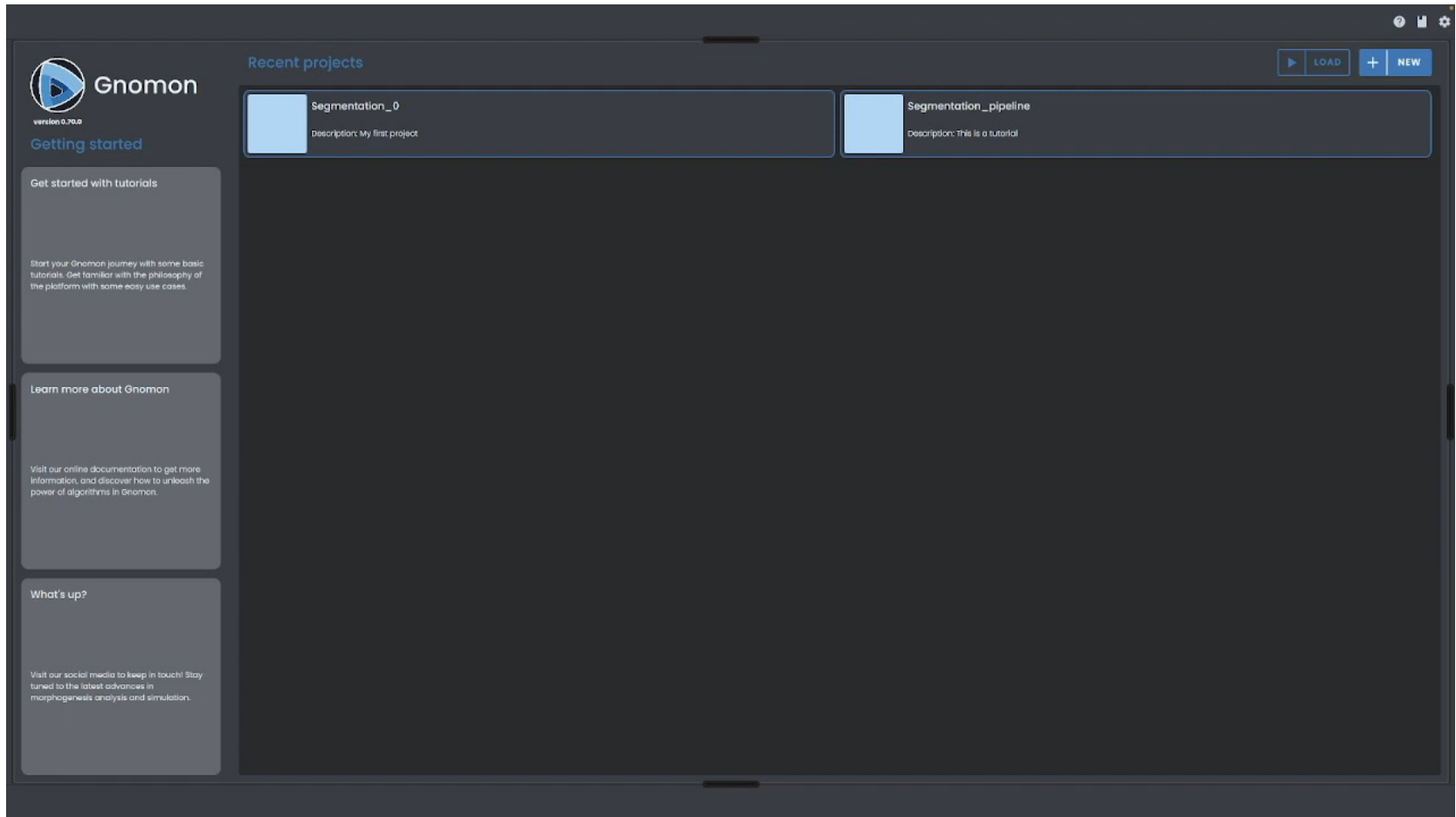


2 IJD depuis 12/2021 au sein de l'EPI MOSAIC
 1 ingénieur SED en coordination
 60 % 1 ingénieur INRAE

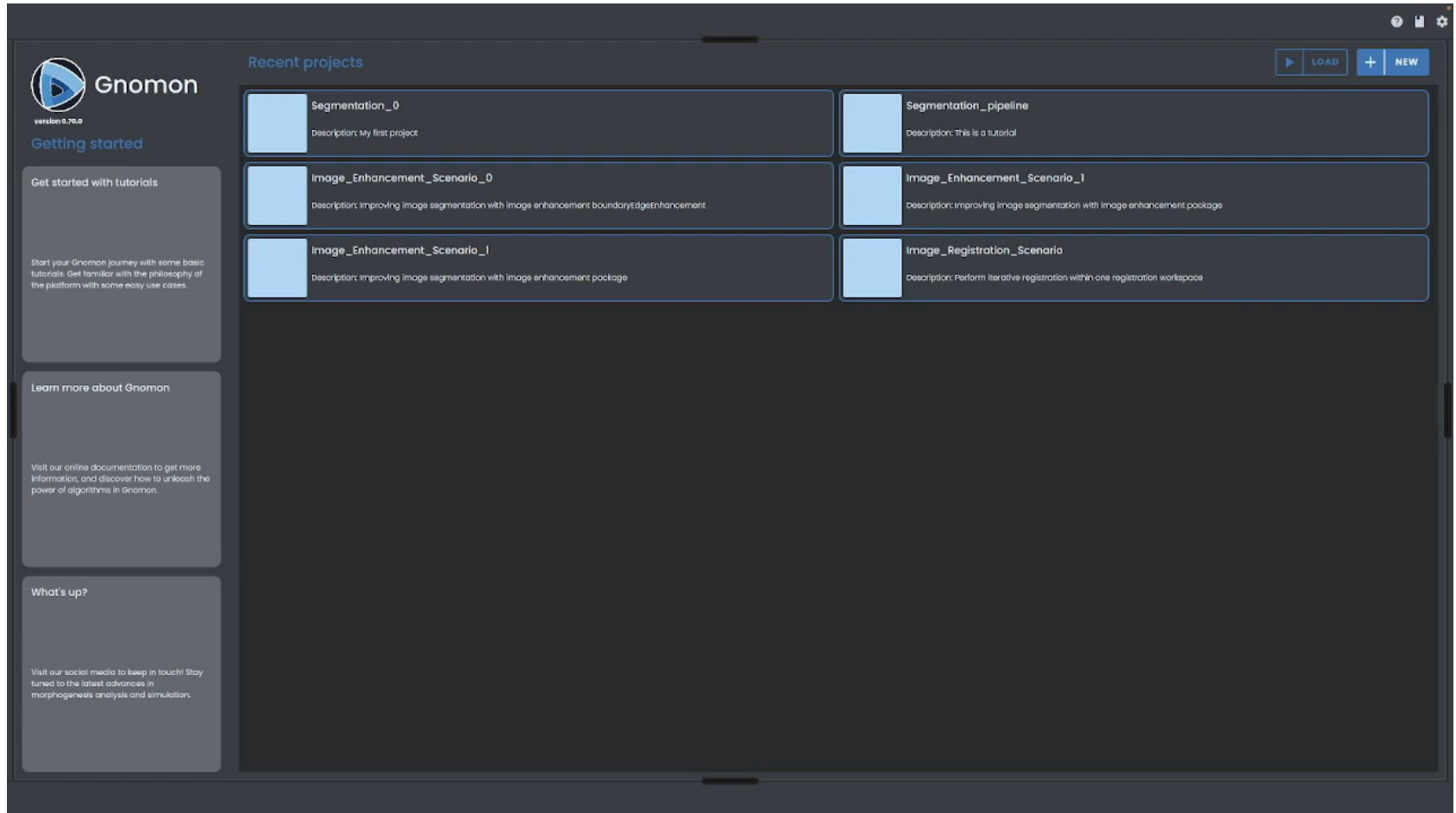
02

Etat actuel

Example: Image Enhancement



Example: Python Plugin



The screenshot displays the Gnomon software interface. On the left, there is a sidebar with the Gnomon logo (version 0.70.0) and sections for 'Getting started', 'Get started with tutorials', 'Learn more about Gnomon', and 'What's up?'. The main area is titled 'Recent projects' and contains six project cards arranged in a 3x2 grid. Each card has a blue square icon and a title with a description below it. In the top right corner of the main area, there are buttons for 'LOAD', '+', and 'NEW'.

Project Name	Description
Segmentation_0	Description: My first project
Segmentation_pipeline	Description: This is a tutorial
Image_Enhancement_Scenario_0	Description: Improving image segmentation with image enhancement boundaryedgeenhancement
Image_Enhancement_Scenario_1	Description: Improving image segmentation with image enhancement package
Image_Enhancement_Scenario_1	Description: Improving image segmentation with image enhancement package
Image_Registration_Scenario	Description: Perform iterative registration within one registration workspace

Example: Simulation Lpy

The screenshot displays the gnomon software interface. The main window is divided into several panels:

- Code Editor (Left):** Contains L-System code:


```

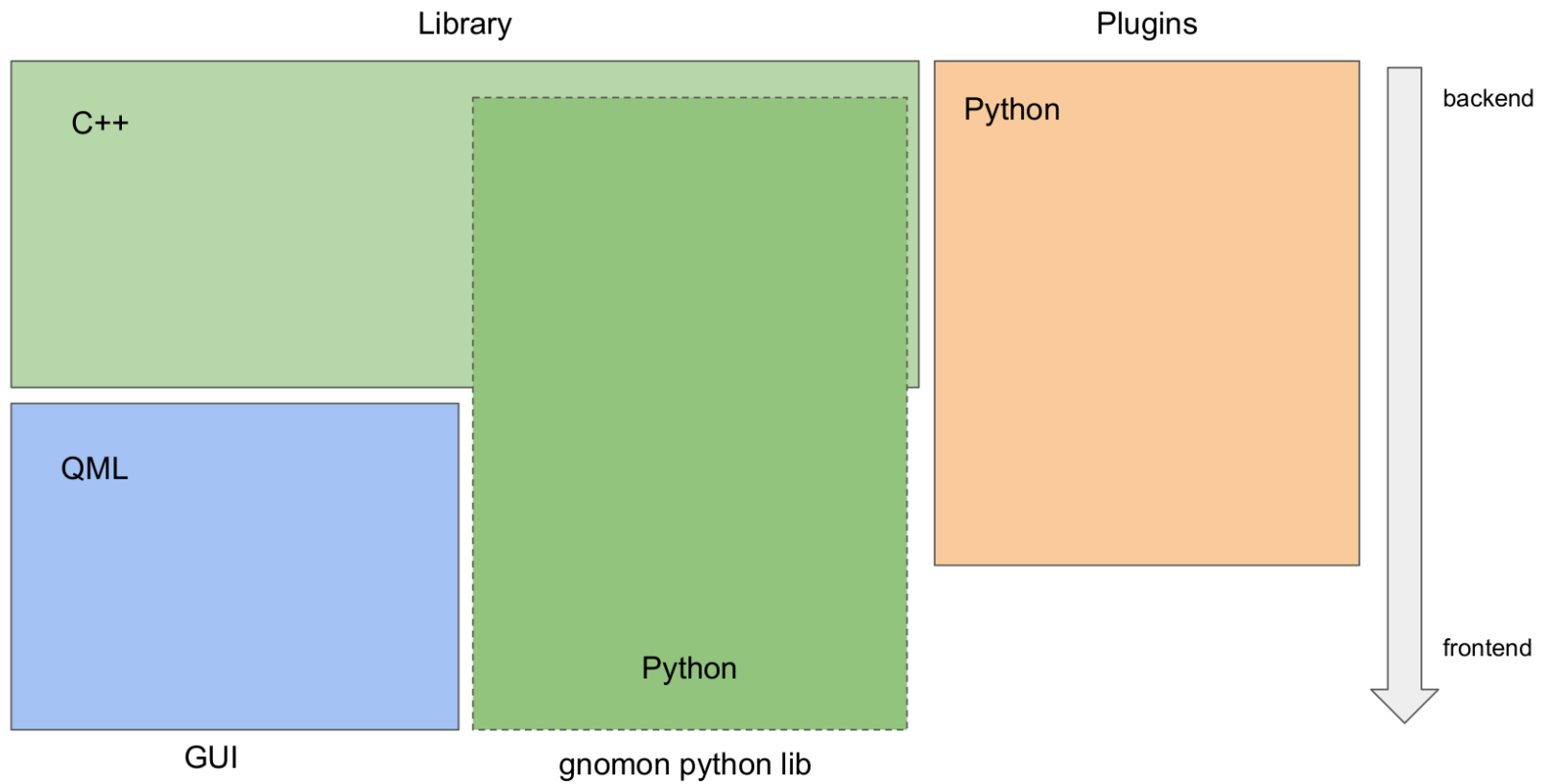
1 Axiom: -(90)_ (0.1)F(50.)
2 production:
3 F(x) : produce F(x/3.)+(60)F(x/3.)-(120)F(x/3.)+(60)F(x/3.)
4 endsystem
5 ##### INITIALISATION #####
6
7 def __initialiseContext__(context):
8     from openalea.plantgl.all import Material,Color3
9     context.options.setSelection("Selection Required",0)
10    context.options.setSelection("Module declaration",0)
11
      
```
- File Browser (Center):** A dialog window titled "gnomon" showing the file structure:

Name	Size	Type	Date Modified
lilac-from-treex.lpy	3.11 KIB	pla...ent	07/12/... 10:23
lilac-to-treex.lpy	4.06 KIB	MAT...ile	07/12/... 10:23
lilac.lpy	2.96 KIB	pla...ent	07/12/... 10:23
- Model View (Right):** Shows the current model: "IStringEvolutionModelLPy".
- Simulation Controls (Bottom):** Includes buttons for "SAVE", "LOAD", "RESET", "STEP", "RUN", and "NEW WORKSPACE". A "DERIVATION LENGTH" indicator is also present.

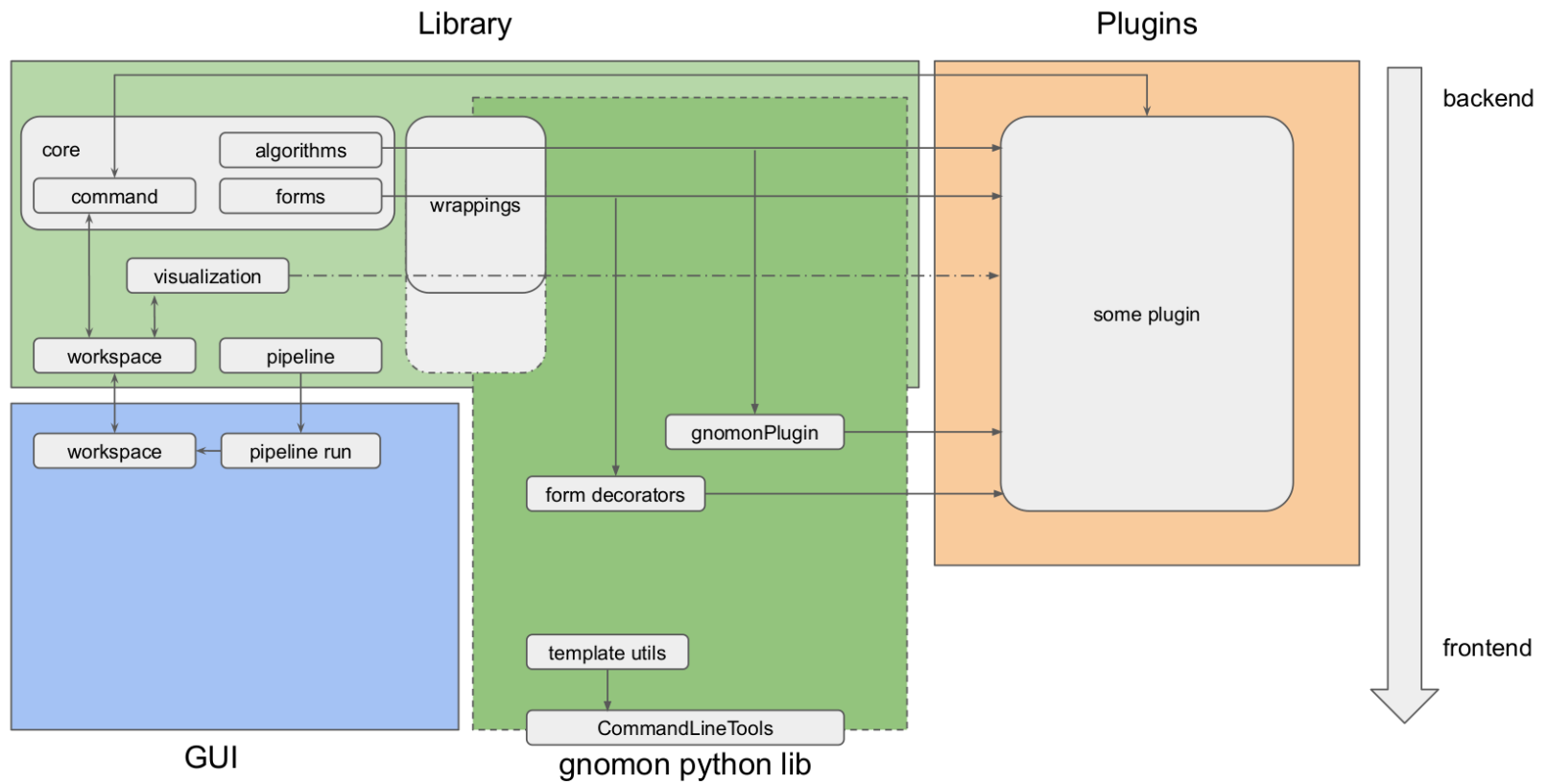
03

Architecture

Languages



Architecture

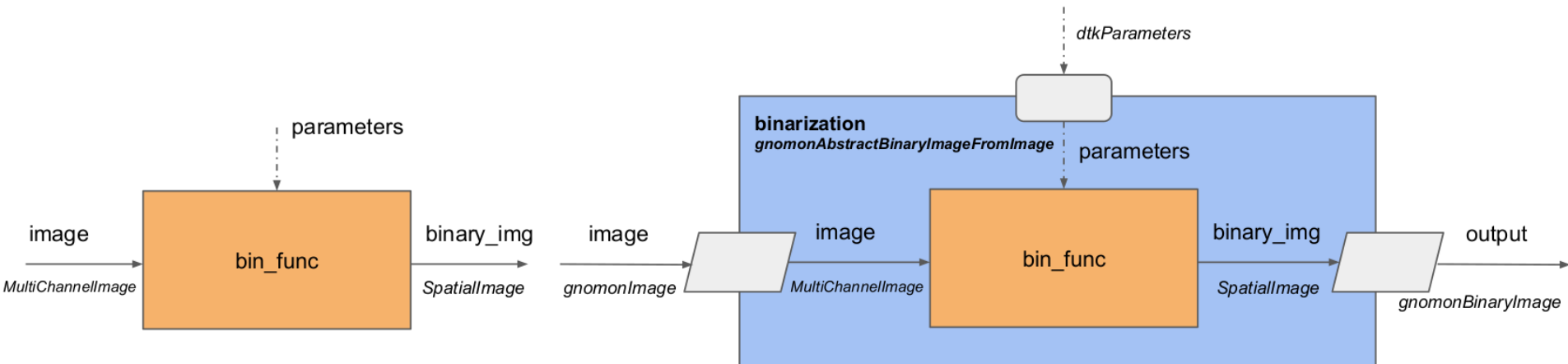


Plugins: binarization

```

69 def bin_func(image: SpatialImage, threshold: int, channel: str, op: str) -> SpatialImage:
70     if op == 'greater >':
71         binary_img = image[channel] > threshold
72     else:
73         binary_img = image[channel] < threshold
74     return binary_img.astype('bool')

```



Plugins: binarization

```

12 @algorithmPlugin(version="0.3.1", coreversion="0.71.0")
13 @imageInput('img_dict', data_plugin="gnomonImageDataMultiChannelImage")
14 @binaryImageOutput('b_img', data_plugin="binaryImageDataSpatialImage")
15 class binarization(gnomonAbstractBinaryImageFromImage):
16     """
17     Image Binarization Plugin
18     """
19     def __init__(self):
20         super().__init__()
21
22         self._parameters = {
23             'channel': d_inliststring("Channel", "", [""], "Channel on which to apply the algorithm"),
24             'greater_or_lower': d_inliststring("> or <", "greater >", ["greater >", "lower <"], "Channel on which t
25             'threshold': d_int("Threshold", 127, 0, 255, "Image is true where intensity > threshold"),
26         }
27
28         self.img_dict = {}
29         self.b_img = {}

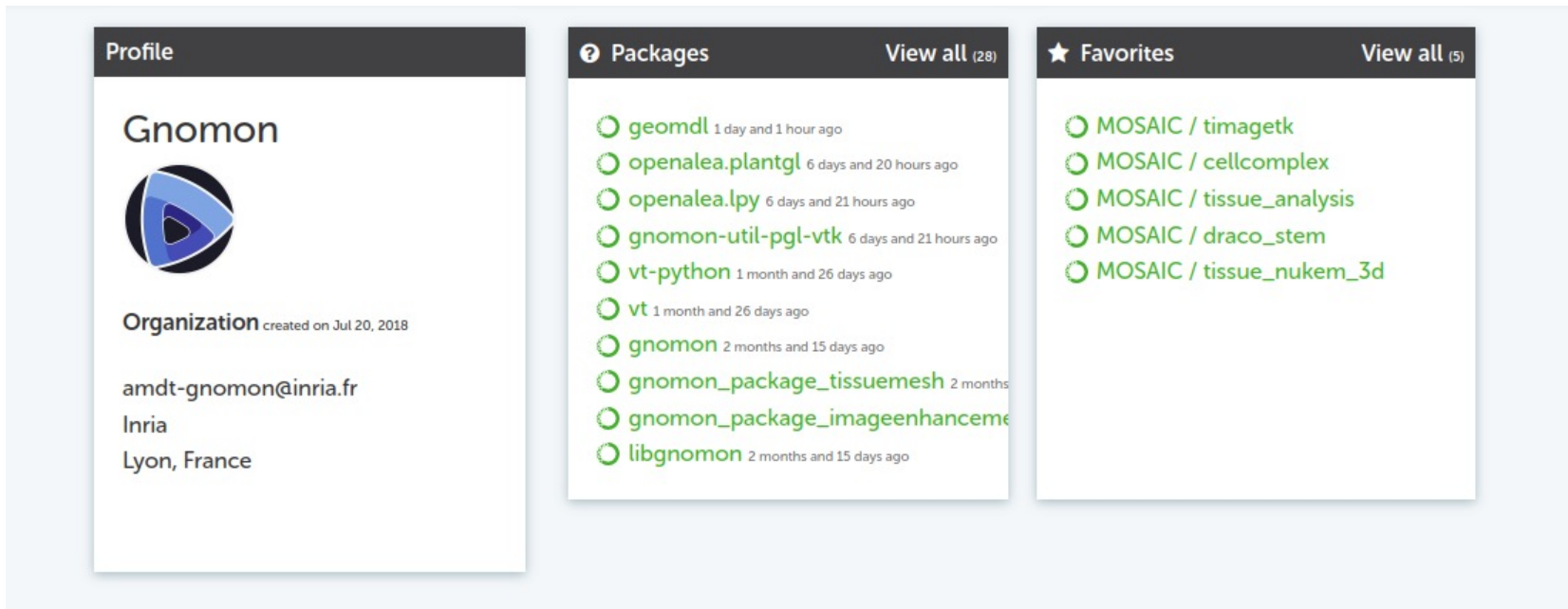
```

The diagram illustrates the data flow and component structure of the binarization plugin. It shows an input `image` (gnomonImage) being processed by a `binarization` block (gnomonAbstractBinaryImageFromImage). Inside this block, the `image` is converted to `MultiChannellImage`, which is then processed by a `bin_func` block. The `bin_func` block also receives `parameters` from a `dtkParameters` block. The output of `bin_func` is `binary_img` (SpatialImage), which is then converted to the final `output` (gnomonBinaryImage).

04

Dépendences

Installation par conda



The screenshot shows a Conda profile page for the 'Gnomon' organization. The profile includes the organization's name, logo, email address (amdt-gnomon@inria.fr), and location (Inria, Lyon, France). The 'Packages' section lists several installed packages with their installation times, such as 'geomdl', 'openlea.plantgl', 'openlea.lpy', 'gnomon-util-pgl-vtk', 'vt-python', 'vt', 'gnomon', 'gnomon_package_tissuemesh', 'gnomon_package_imageenhanceme', and 'libgnomon'. The 'Favorites' section lists five favorite packages: 'MOSAIC / timagetk', 'MOSAIC / cellcomplex', 'MOSAIC / tissue_analysis', 'MOSAIC / draco_stem', and 'MOSAIC / tissue_nukem_3d'.

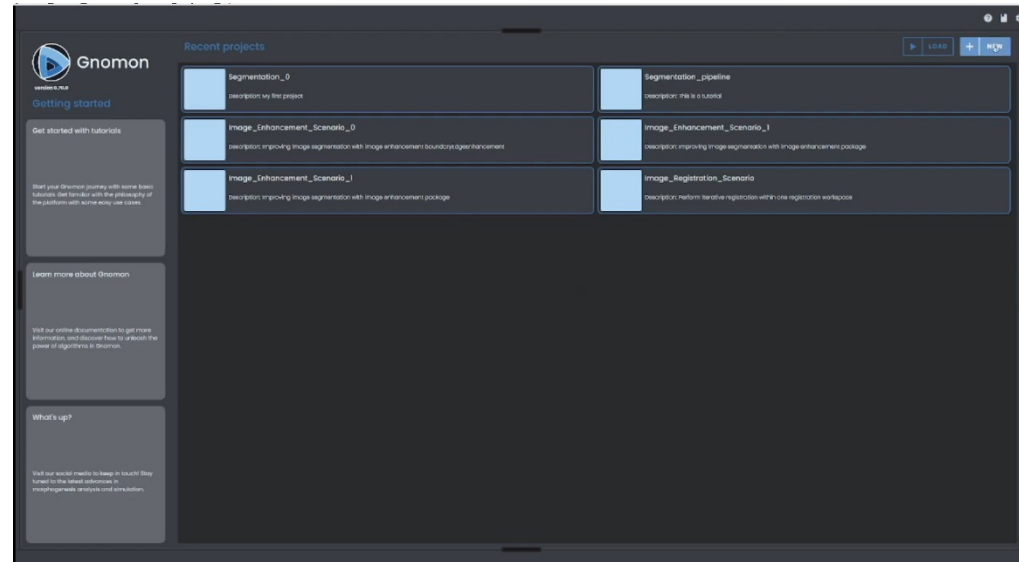
```
Mamba install gnomon -c gnomon -c mosaic -c morpheme -c dtk-forge6 -c conda-forge
```



Qt, PySide6, VTK, python, SWIG, dtk, ...

Factorisation de code dans dtk (SED Sophia)

- **Interface qml**
- **Visu VTK**
- **Gestion des parametres**
- **Logs**
- **Système de plugins c++**



Merci !

Remerciements

- **Arthur Luciani "Creating an interface to run pipelines in python"**
- **Karamoko Samassa "Integrating Simulations in Gnomon LPy as a Proof of Concept"**