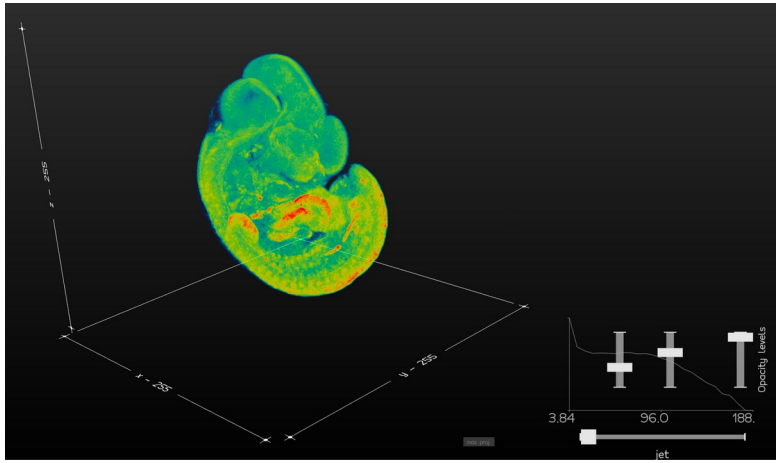
vedo

A python module for scientific analysis **and visualization of** ɜd o**bjects**



Laura Avinyó, Marco Musy

Summer School on Computational Modelling of Multicellular Systems
June 2023

SHARPE LAB

EMBL
European Molecular Biology Laboratory

Where?

vedo sits somewhere in here

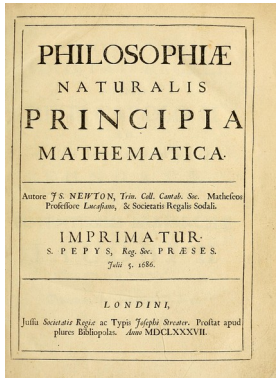
TB

Preprocessing
Raw Data

GB

Postprocessing
Data analysis

MB


PHILOSOPHIÆ
NATURALIS
PRINCIPIA
MATHEMATICA.

Autore *IS. NEWTON, Trin. Coll. Cantab. Soc.* Matheseos Professore *Lucasiano*, & Societatis Regalis Sodali.

IMPRIMATUR.
S. PEPYS, *Reg. Soc.* PRÆSES.
Julii 5. 1686.

LONDINI,
Jussu *Societatis Regiæ* ac Typis *Josephi Streater*. Prostat apud plures Bibliopolas. *Anno* MDCLXXXVII.

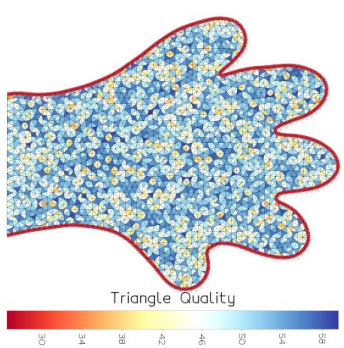
"A 3D-powered version of `matplotlib`"

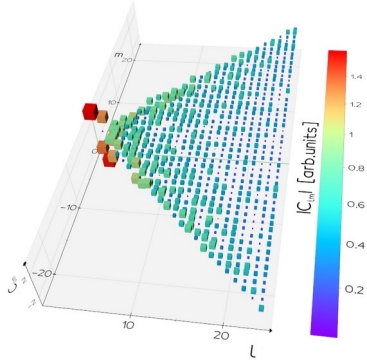
"A handy day-to-day tool for the researcher"

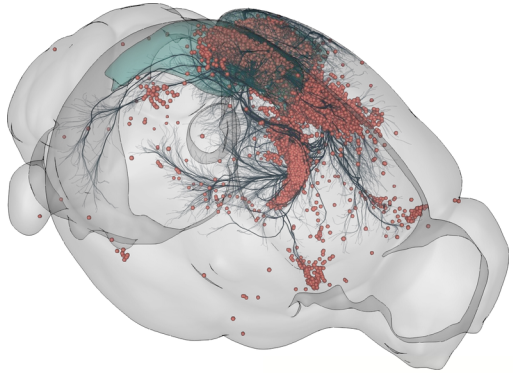
It can prove useful with any type of data having a spatio-temporal structure.

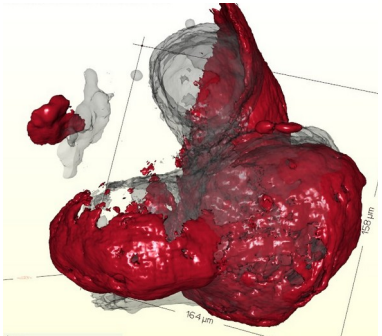
What can *you* do with it?

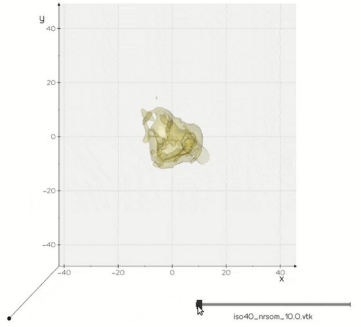
- Work with polygonal meshes and point clouds
- Morphometrics (mesh warp, cut, connect, …)
- Analysis of 3D images and tetrahedral meshes
- 2D/3D plotting and histogramming.
- Integration with other external libraries (`Qt`, `napari`, `trimesh`, `pymeshlab`, `SHTools` …)
- Jupyter and Colab environments are supported
- `Command Line Interface` (CLI) as quick viz tool
- Export/exchange 3D interactive scenes to file
- Create interactive animations
- Generate publication-quality renderings

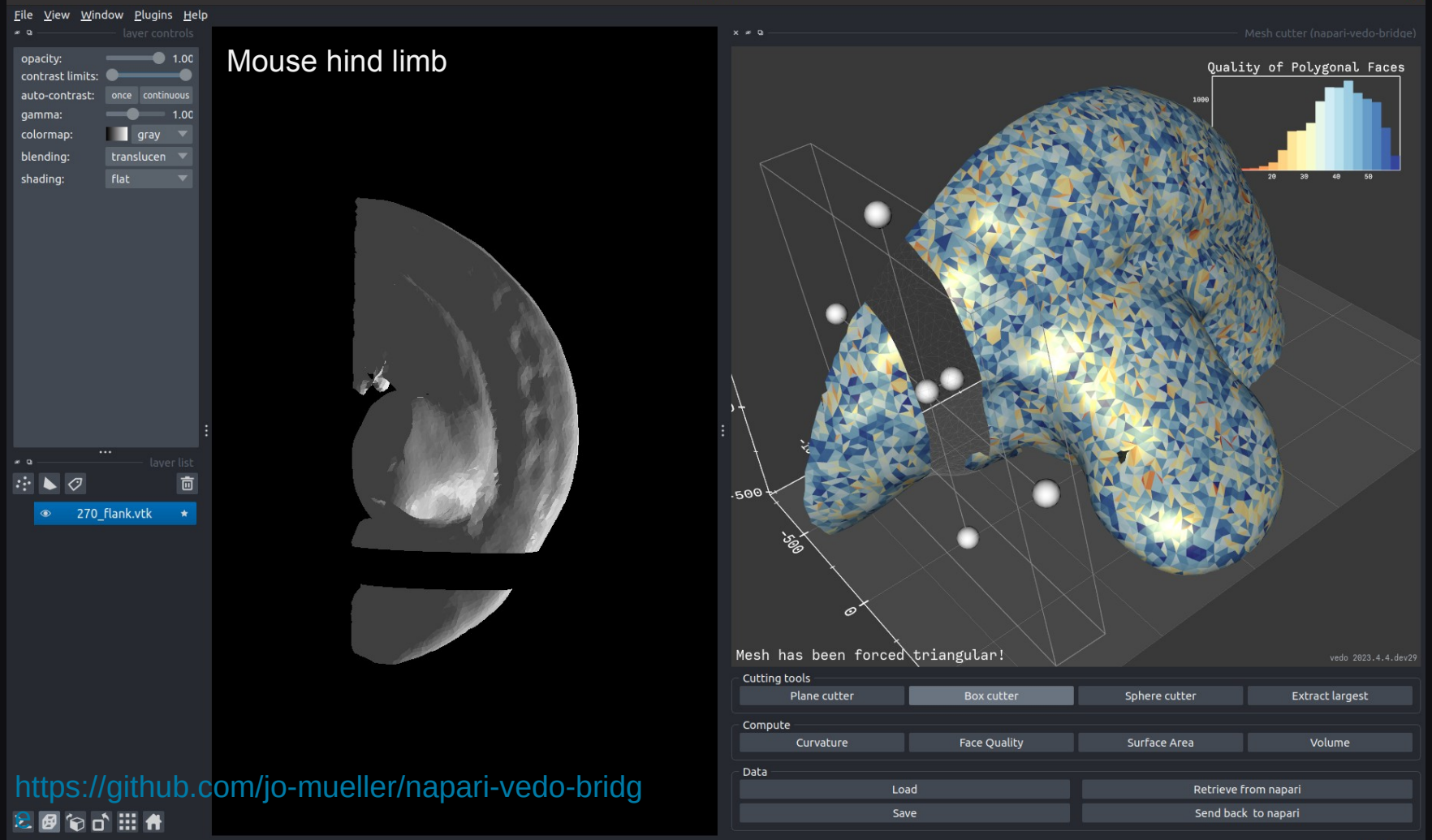




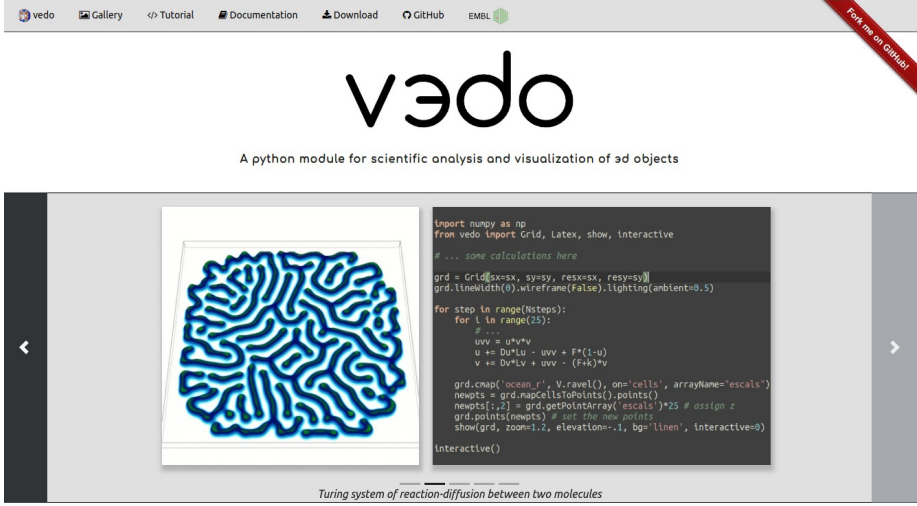







napari
File View Window Plugins Help
layer controls
opacity: 1.00
contrast limits:
auto-contrast: once continuous
gamma: 1.00
colormap: gray
blending: translucen
shading: flat
layer list
270_flank.vtk
Mouse hind limb
Mesh cutter (napari-vedo-bridge)
Quality of Polygonal Faces
1000
20 30 40 50
-500
-500
0
Mesh has been forced triangular!
vedo 2023.4.4.dev29
Cutting tools
Plane cutter
Box cutter
Sphere cutter
Extract largest
Compute
Curvature
Face Quality
Surface Area
Volume
Data
Load
Retrieve from napari
Save
Send back to napari
activity
https://github.com/jo-mueller/napari-vedo-bridg

How?



- Install:

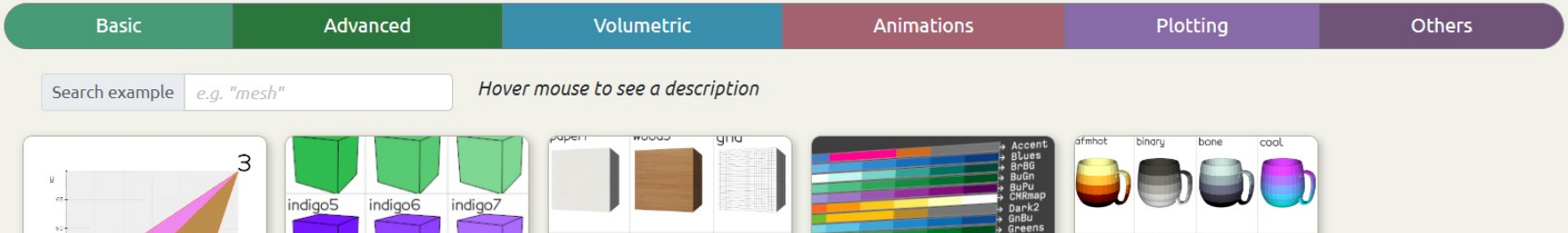
  `pip install vedo`

- Documentation:

  https://vedo.embl.es/

- **350+** examples as reference
  - Designed to be short and intuitive (most are <30 lines)
  - Searchable `vedo --search string`
  - Runnable `vedo --run examplename`

  API documentation is found at vedo.embl.es/docs

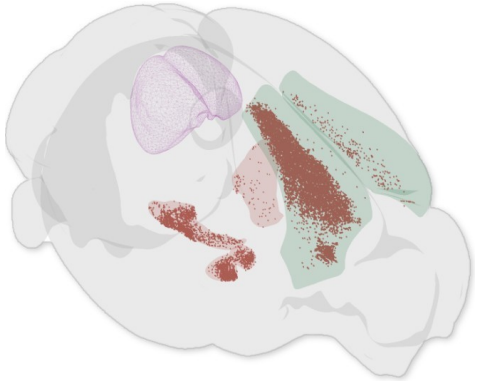


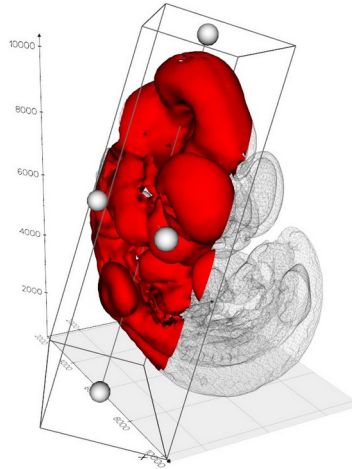
Conclusion

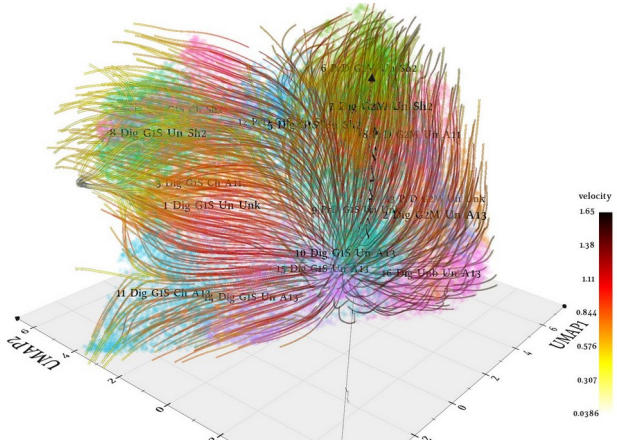
- Proved very useful in diverse applications
- Documented API with many examples
- **Happy to offer support!**

marco.musy@embl.es
laura.avino@embl.es

https://vedo.embl.es/

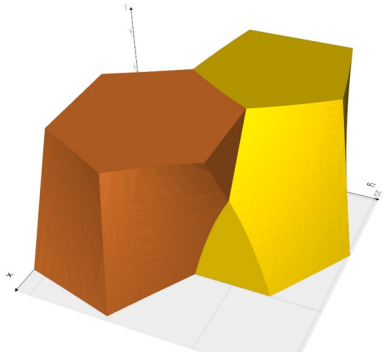






Summer School on Computational Modelling of Multicellular Systems
June 2023

SHARPE LAB
EMBL European Molecular Biology Laboratory



practicals

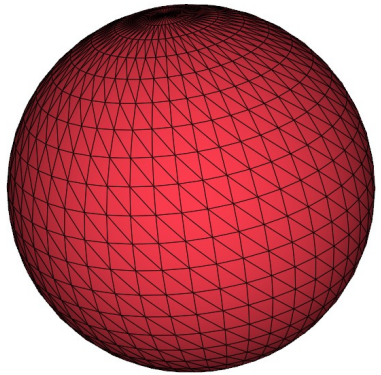
Installing steps

1. Get the repo.
   Opening a terminal (anaconda).
   ```
   > git clone https://github.com/LauAvinyo/vedo-embo-course.git
   > cd vedo-embo-course
   ```
2. Install dependencies:
   ```
   > pip install vedo -U
   > pip install scipy
   ```
3. Open the files:
   If you have a preference use it.
   Otherwise, install jupyterlab.
   ```
   > pip install jupyterlab
   > jupyter-lab
   ```

Basic Geometric Objects

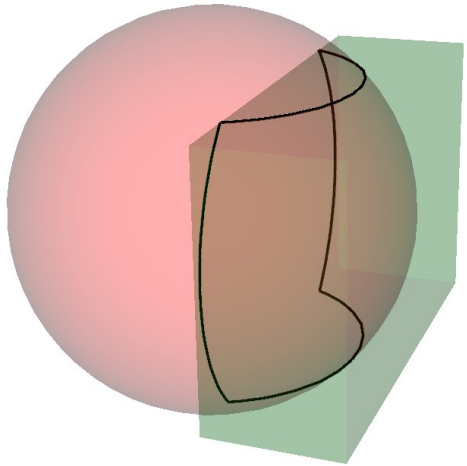
```
1  from vedo import *
2
3  settings.default_backend = "vtk"
4
5  sphere = Sphere().linewidth(1)
6
7  plt = Plotter()
8  plt += sphere
9  plt.show()
10 plt.close()
```

In Jupyter notebooks



Press “h” in rendering window

```
# Create a sphere and a box
sphere = Sphere(r=1.5).c("red5", 0.2)
box = Box(pos=(1,0,0)).triangulate().c("green5", 0.2)

# Find the intersection between the two
intersection = sphere.intersect_with(box).lw(4)

plt += [sphere, box, intersection]
```



Plotting made simple

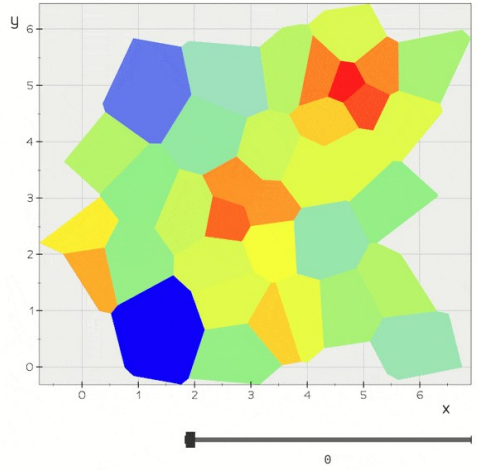
Example from Tuesday Chaste tutorial

```
> vedo -s data/chaste/Practical_2_3/results_*.vtu
```

(CLI)

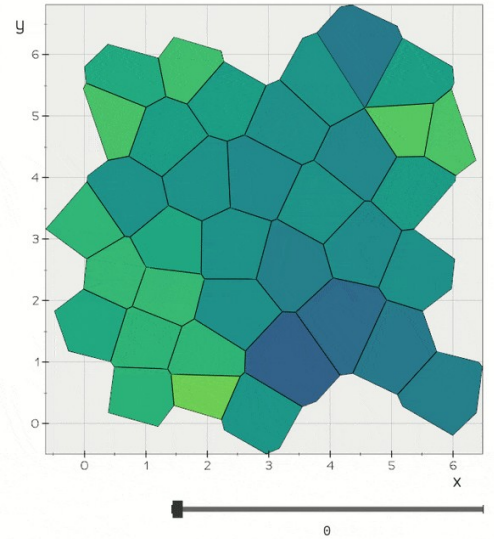
```
from vedo.applications import Browser
Browser("data/chaste/Practical_2_3/results_*.vtu").show()
```

(script)



or:

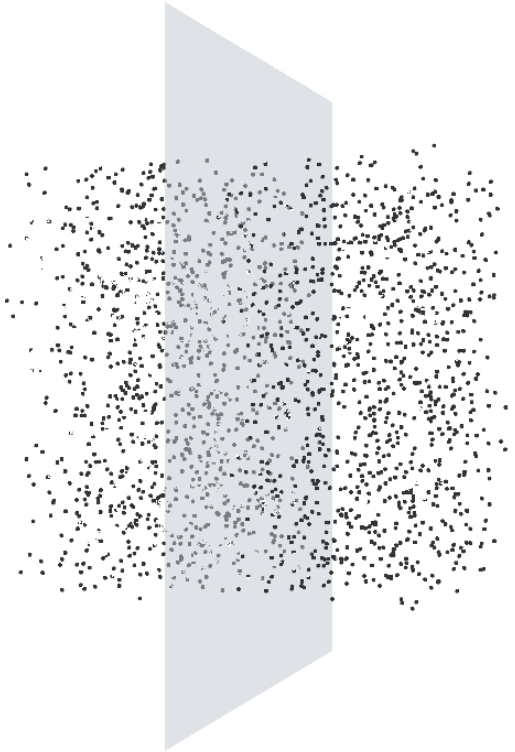
```
from vedo import load
from vedo.applications import Browser

ugrids = load("chaste/Practical_2_3/results_*.vtu")

meshes = []
for u in ugrids:
    m = u.alpha(1).tomesh().linewidth(1)
    m.cmap("viridis_r", vmin=0.2, vmax=2)
    meshes.append(m)

Browser(meshes).show()
```



Point Clouds & Polygonal Meshes

Create a point cloud and cut it

```
points = np.random.rand(2000, 3)

pts = Points(points)
pln = Plane(pos=(0.5, 0.5, 0.6), normal=(1, 0, 0), s=(1.5, 1.5))

show(pts, pln).close()
```



Create a point cloud and cut it

```
pts.cut_with_plane((0.5, 0.5, 0.6))
```

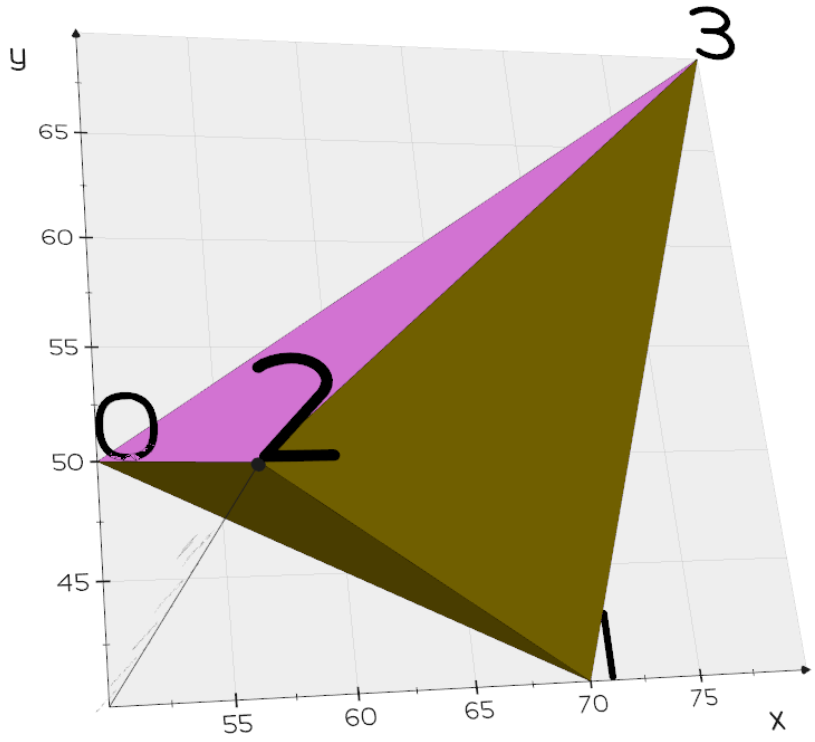
Can you create a normal distributed cloud and cut the points inside a cylinder?

Hint: search the API docs for “cylinder”

Build a polygonal mesh manually

```python
from vedo import Mesh, show

# Define the vertices and faces that make up the mesh
verts = [(50,50,50), (70,40,50), (50,40,80), (80,70,50)]
faces = [(0,1,2), (2,1,3), (1,0,3)]

# Build the polygonal Mesh object from the vertices and faces
mesh = Mesh([verts, faces])

# Set the backcolor of the mesh to violet
# and show edges with a linewidth of 2
mesh.backcolor('violet').linecolor('tomato').linewidth(2)

# Create labels for all vertices in the mesh showing their ID
labs = mesh.labels('id').c('black')

# Print the points and faces of the mesh as numpy arrays
print('points():', mesh.points())
print('faces() :', mesh.faces())

# Show the mesh, vertex labels, and docstring
show(mesh, labs, viewup='z', axes=1).close()
```

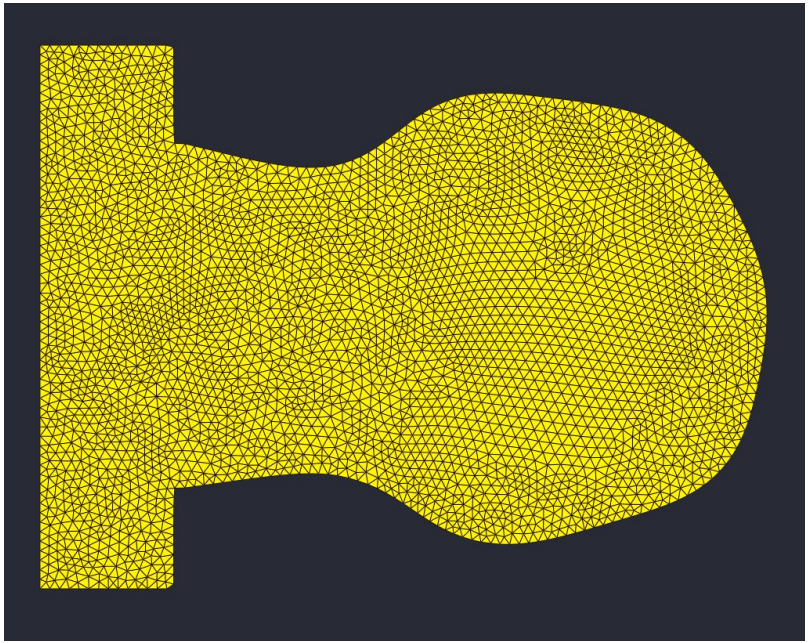
vedo --run buildmesh



Create a polygonal mesh

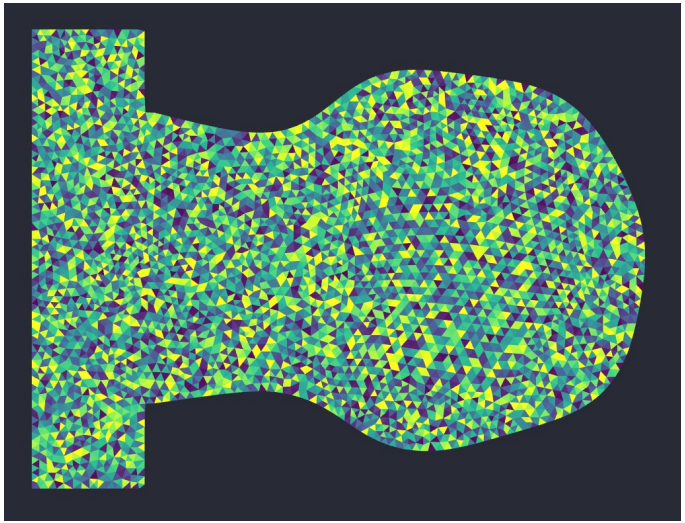
```
# Read data
faces = np.load(faces_path)
verts = np.load(verts_path)
```

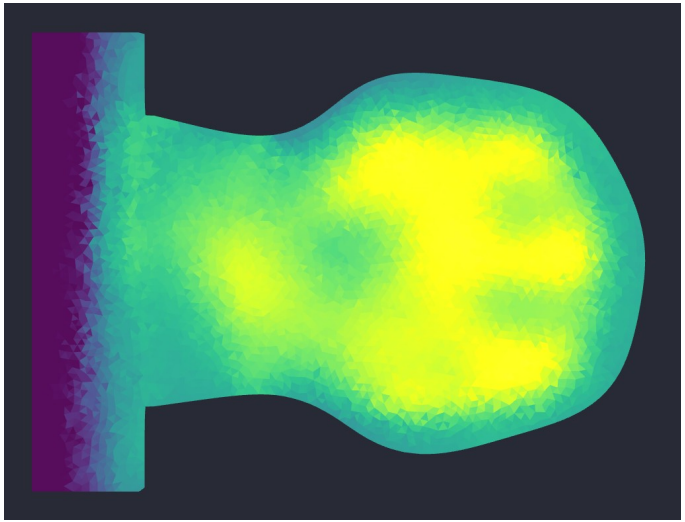
```
msh = Mesh([verts, faces]).linewidth(1)
```



Add gene data associated with cells

```
# Adding scalar values
n = faces.shape[0] # number of faces
values = np.random.random(n)
msh.celldata["fake_data"] = values
```





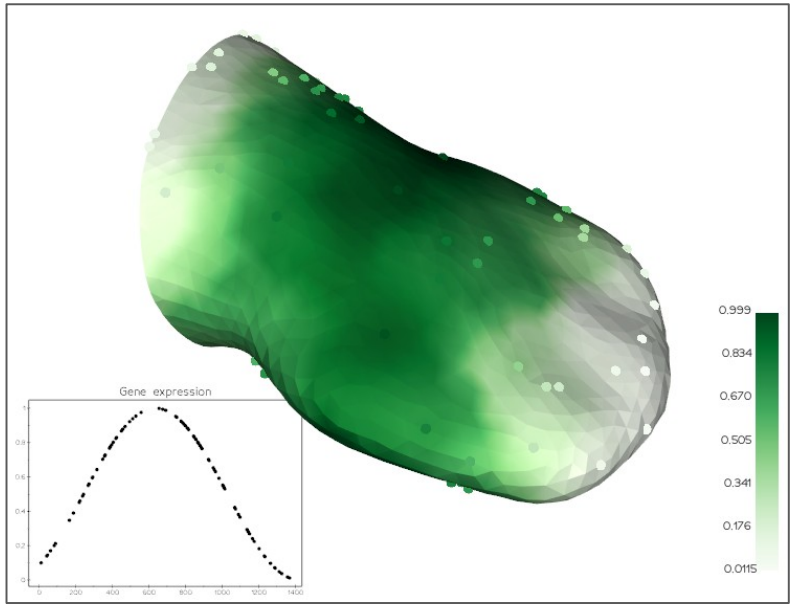
Can you add the gene expression data to the mesh?

Use `../data/sox9_data/gene_data.npy`

Interpolate data from sparse measurements in 3D

Let’s assume that we know the expression of a gene in 100 positions

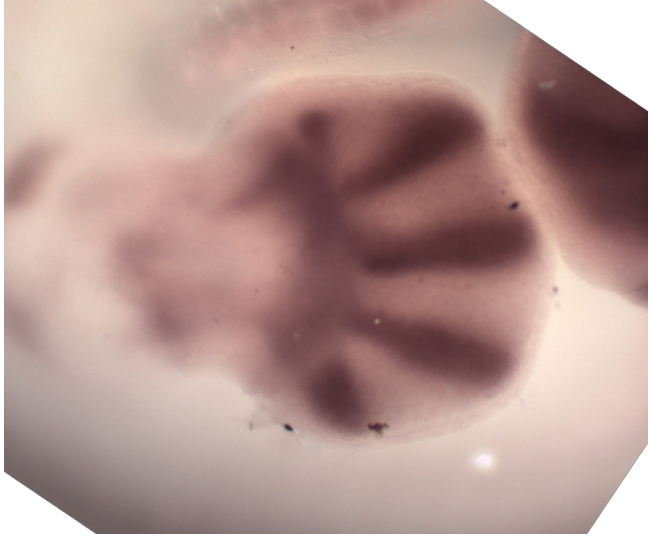
```python
import numpy as np
from vedo import dataurl, Mesh, Points, show
from vedo.pyplot import plot

# Load a mesh of a mouse limb at 12 days of development
msh = Mesh(dataurl + "290.vtk")

# Pick 100 points where we measure the value of a gene expression
ids = np.random.randint(0, msh.npoints, 100)
pts = msh.points()[ids]        # slice the numpy array
x = pts[:, 0]                  # x coordinates of the points
gene = np.sin((x+150)/500)**2  # we are making this up!

# Create a set of points with those values
points = Points(pts, r=10).cmap("Greens", gene)

# Interpolate the gene data onto the mesh, by averaging the 5 closest points
msh.interpolate_data_from(points, n=5).cmap("Greens").add_scalarbar()

# Create a graph of the gene expression as function of x-position
gene_plot = plot(x, gene, lw=0, title="Gene expression").as2d(scale=0.5)

# Show the mesh, the points and the graph
show(msh, points, gene_plot)
```

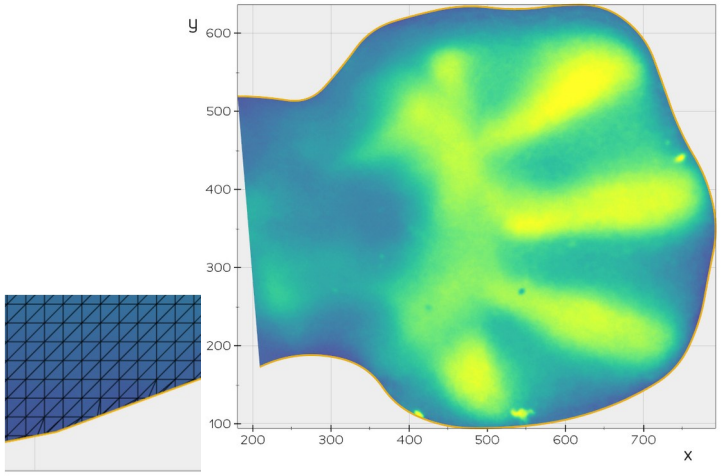


Mesh from a JPG image

Manually select a contour and extract a **polygonal mesh** from the input image

```
from vedo import Picture, settings, show
from vedo.applications import SplinePlotter

settings.default_backend = "vtk"

pic = Picture("data/sox9_exp.jpg").bw()

plt = SplinePlotter(pic)
plt.show(mode="image", zoom='tight')
outline = plt.line
plt.close()

print("Cutting with outline...")
msh = pic.tomesh().triangulate().cmap("viridis_r")
cut_msh = msh.clone().cut_with_point_loop(outline)

cut_msh.interpolate_data_from(msh, n=3)
show(cut_msh, outline, axes=1).close()
```

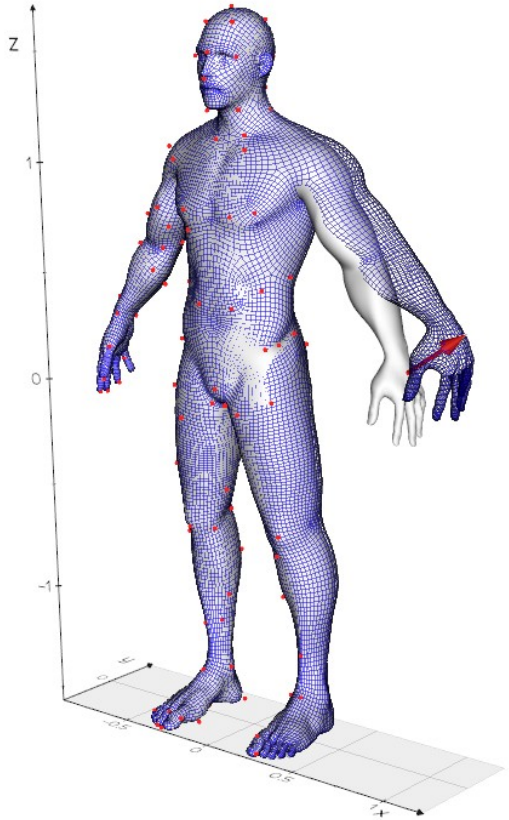




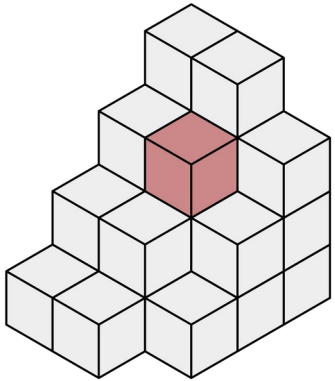
Warp a Mesh (non-linear registration)

All points stay fixed while a single point in space moves as the arrow indicates

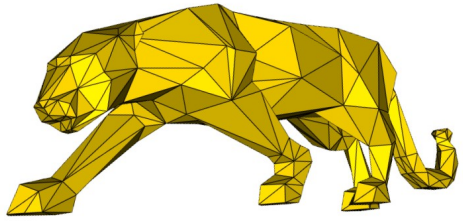
```
from vedo import dataurl, Mesh, Arrows, show

# Load a mesh
mesh = Mesh(dataurl+"man.vtk").color("white")

# Create a heavily decimated copy with about 200 points
# (to speed up the computation)
mesh_dec = mesh.clone().triangulate().decimate(n=200)

sources = [[0.9, 0.0, 0.2]]  # this point moves
targets = [[1.2, 0.0, 0.4]]  # ...to this.
for pt in mesh_dec.points():
    if pt[0] < 0.3:          # while these pts don't move
        sources.append(pt)   # (e.i. source = target)
        targets.append(pt)

# Create the arrows representing the displacement
arrow = Arrows(sources, targets)

# Warp the mesh
mesh_warped = mesh.clone().warp(sources, targets)
mesh_warped.c("blue").wireframe()

# Show the meshes and the arrow
show(mesh, mesh_warped, arrow, axes=1)
```



Volumes (eg TIFF stacks)

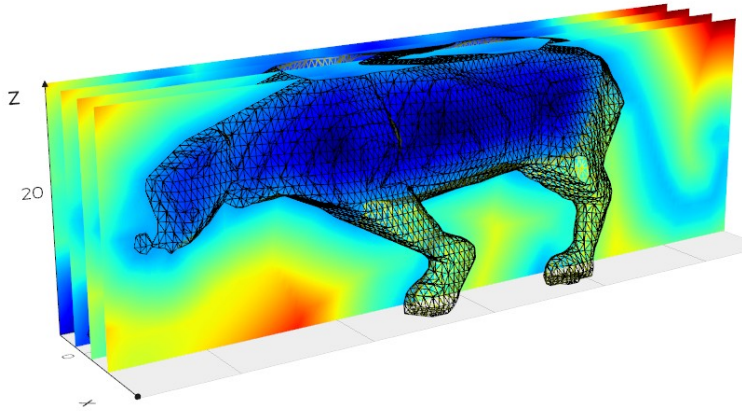


Compute distance from a mesh

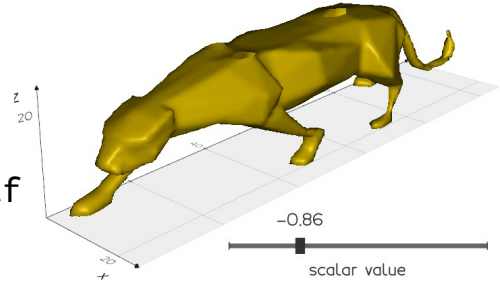


..and save it as a tiff stack

```
1   from vedo import *
2
3   msh = Mesh(dataurl + "panther.stl")
4
5   vol = msh.signed_distance(dims=[25,125,25])
6   iso = vol.isosurface(0.0)
7
8   plt = Plotter()
9   plt += iso.wireframe()
10
11  for i in range(0, 25, 5):
12      plt += vol.xslice(i).cmap("jet")
13
14  vol.write("panther.tif")
15  plt.show(axes=1)
```

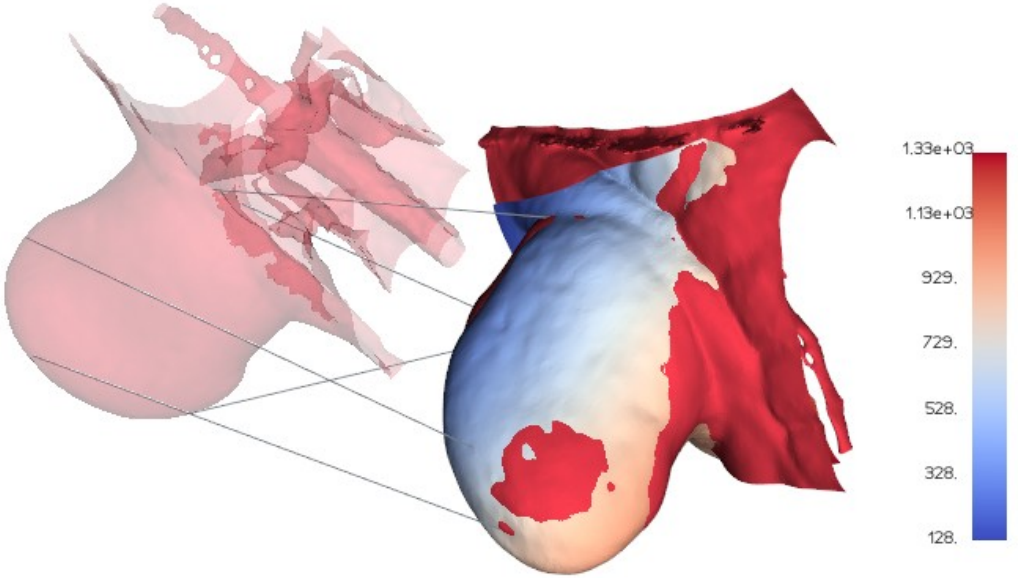


> vedo panther.tif



Extras: Linear vs Non-Linear Registration

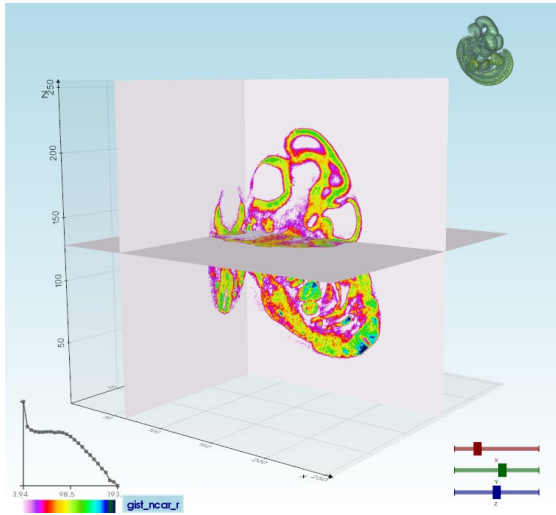
Naive registration (ICP) might fail!

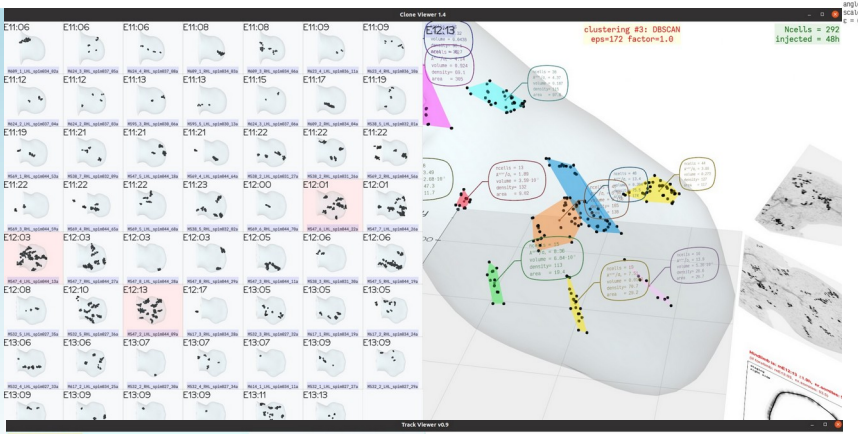


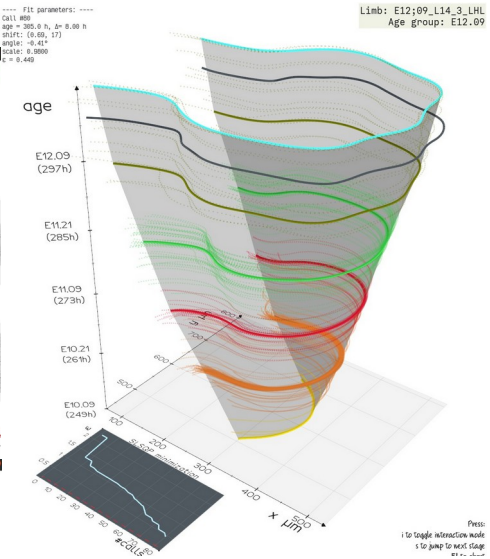
Try to:

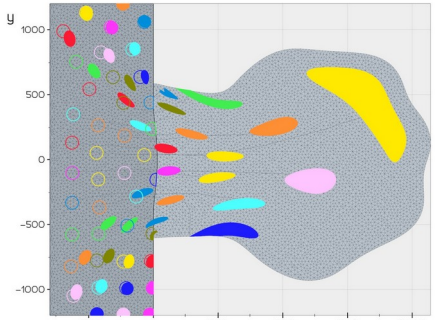
1. Inspect the meshes
2. Identify landmark points
3. Warp one mesh onto the other

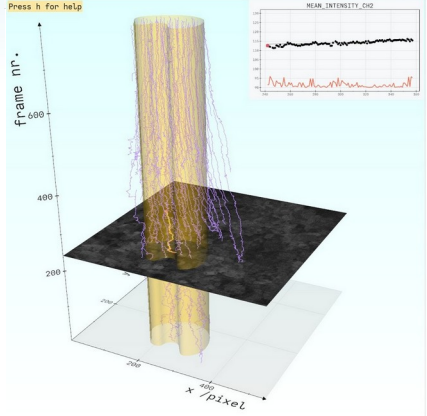
Play with your own data (if you have it at hand!)

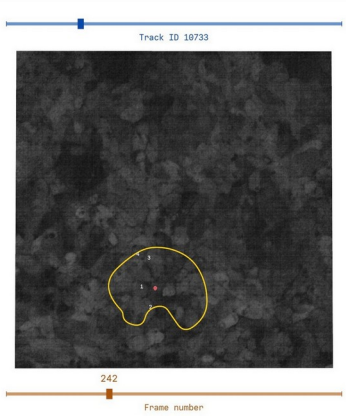


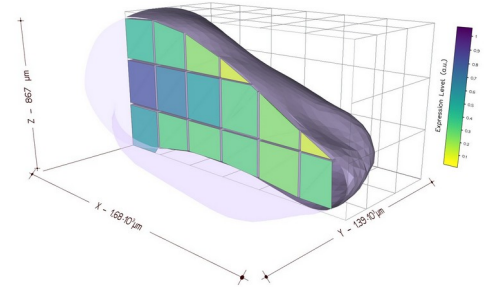










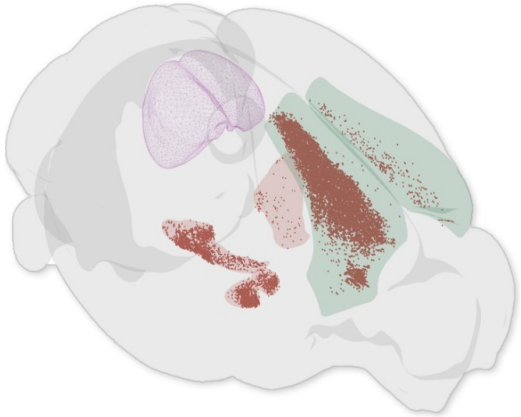


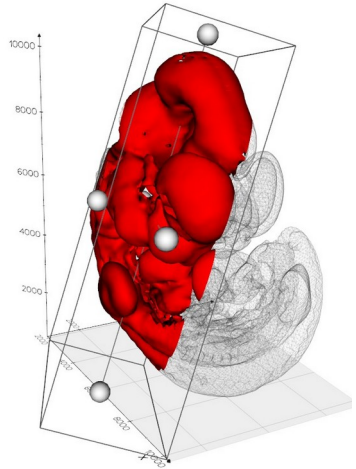
Conclusion

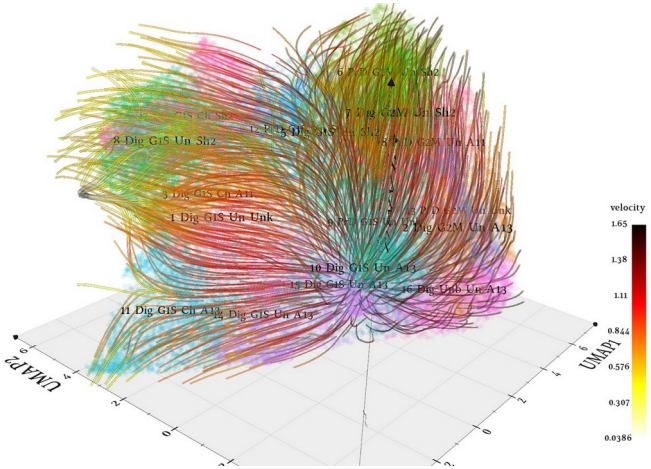
- Proved very useful in diverse applications
- Documented API with many examples
- **Happy to offer support!**

marco.musy@embl.es
laura.avino@embl.es

https://vedo.embl.es/







Summer School on Computational Modelling of Multicellular Systems
June 2023

SHARPE LAB
EMBL European Molecular Biology Laboratory

Installation links

Conda (you should have it already from Tuesday):
https://docs.conda.io/en/latest/miniconda.html

Jupyterlab:
https://jupyterlab.readthedocs.io/en/latest/getting_started/installation.html#conda

The repo:
https://github.com/LauAvinyo/vedo-embo-course/tree/main