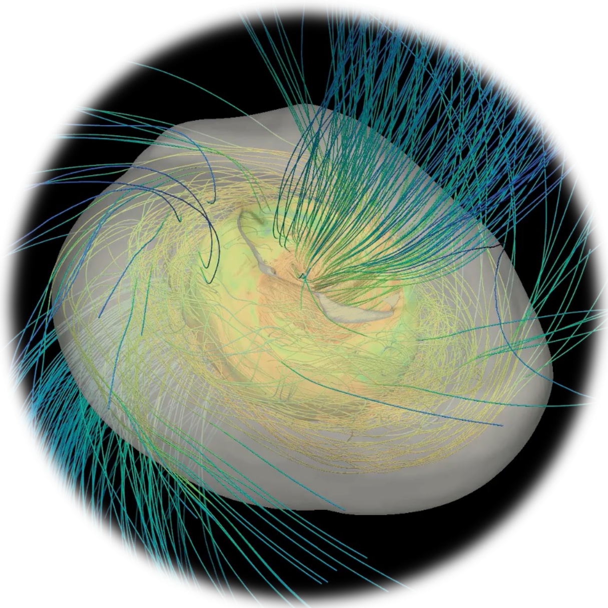
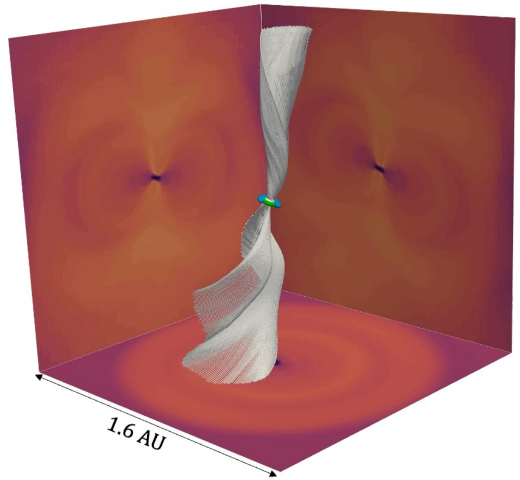
# A brief tutorial on 3D rendering for AMR data

**Adnan Ali Ahmad**

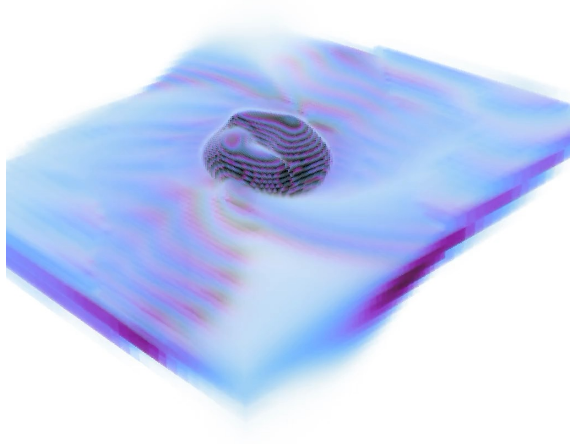
**RAMSES SNO DAYS 2025**



t = -0.02 (yr)




t = 0.000 kyr



ENS DE LYON

CRAL
CENTRE DE RECHERCHE ASTROPHYSIQUE DE LYON

agence nationale de la recherche

- A set of techniques you can use to visualize your RAMSES/SHAMROCK/DYABLO data

  - Particle renderings
  - Volume renderings
  - Iso-contouring
  - 3D printing
  - Streamlines

- Available softwares

  ParaView

  UNREAL ENGINE

  blender

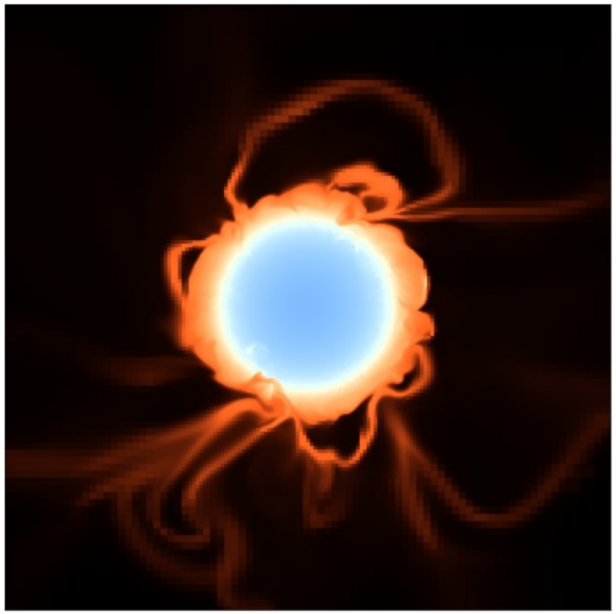
  VisIt

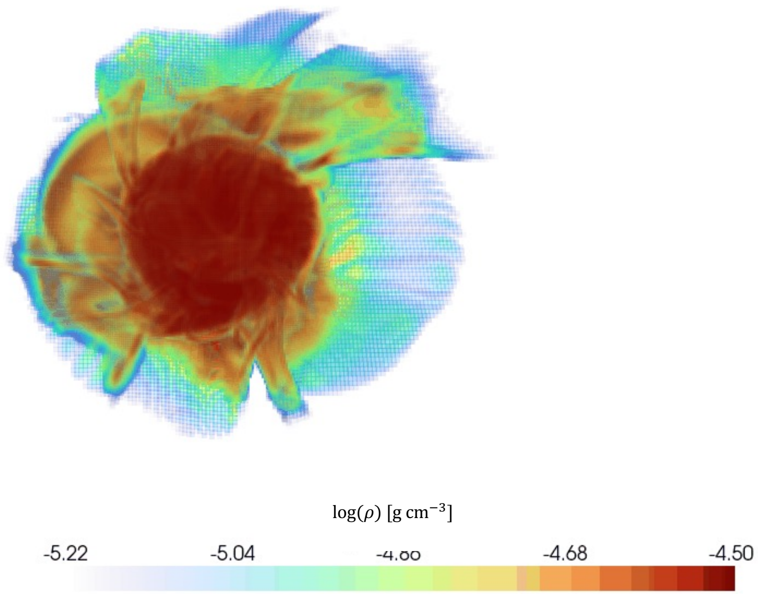
  unity

  PyVista

# Why 3D renders?

- Easier to understand 3D structutres with 3D images…





# The difficulties

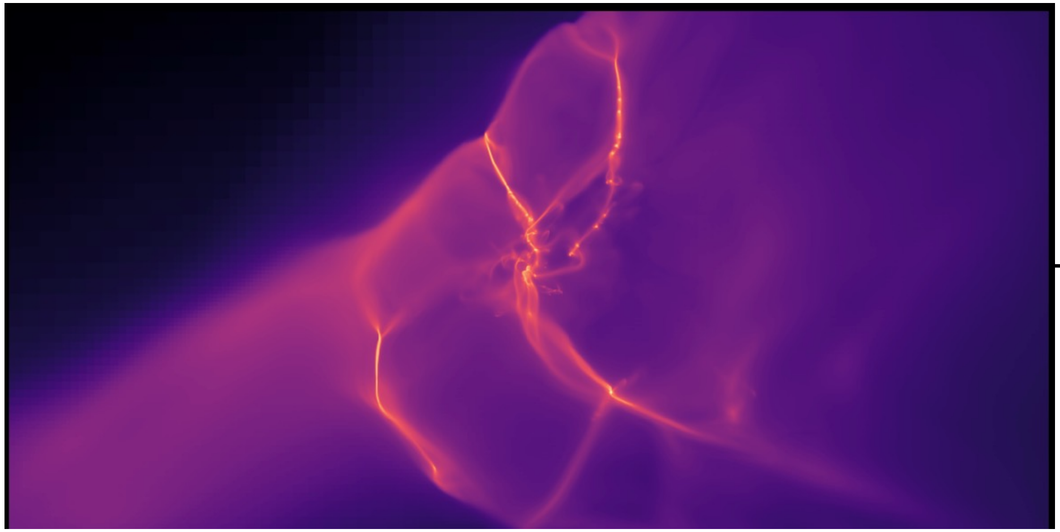
- Most 3D rendering softwares work with **point clouds** or **regular grids**
- Most objects are **embedded** within continious spatial distributions of matter
- 3D rendering is **RAM-heavy**
- 3D renders are easier to understand when interactive: how does one **make a publication-ready 3D figure**?

# I – Particle rendering

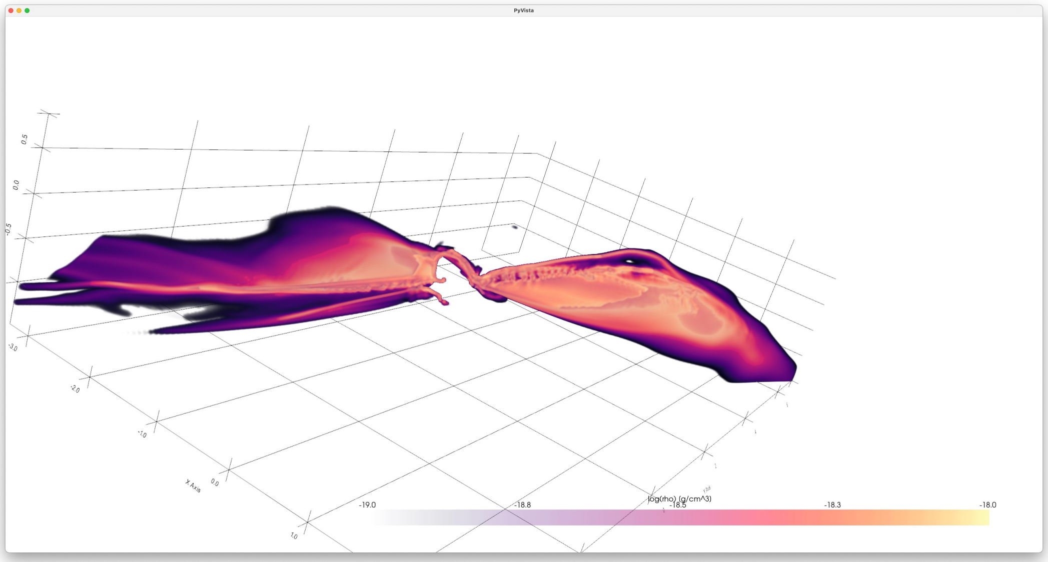
- Very easy to do
- Works well with:
    - Tracer particles
    - Cell center positions

- Problem: too many particles in simulations, **might need to be selective**

# I – Particle rendering



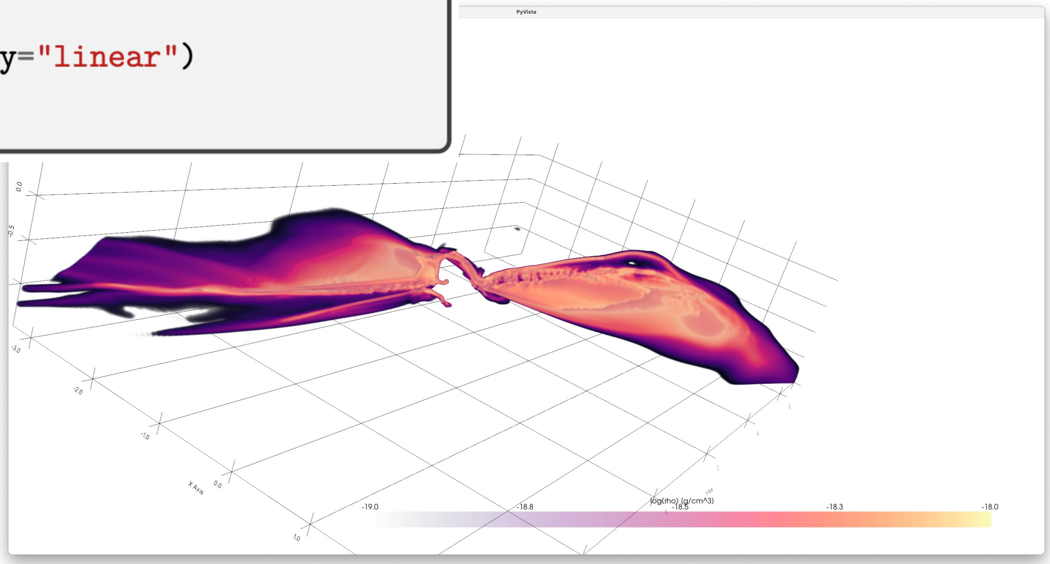
Extract cells exceeding density threshold



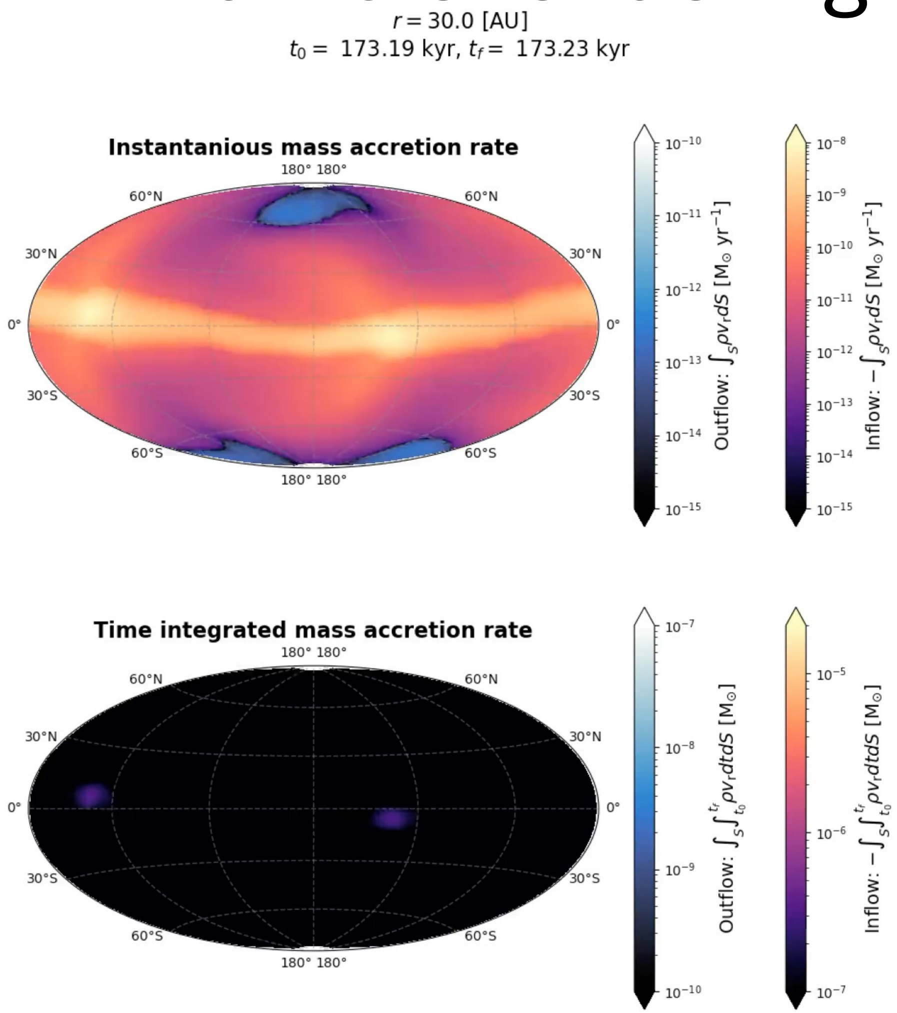
# I – Particle rendering

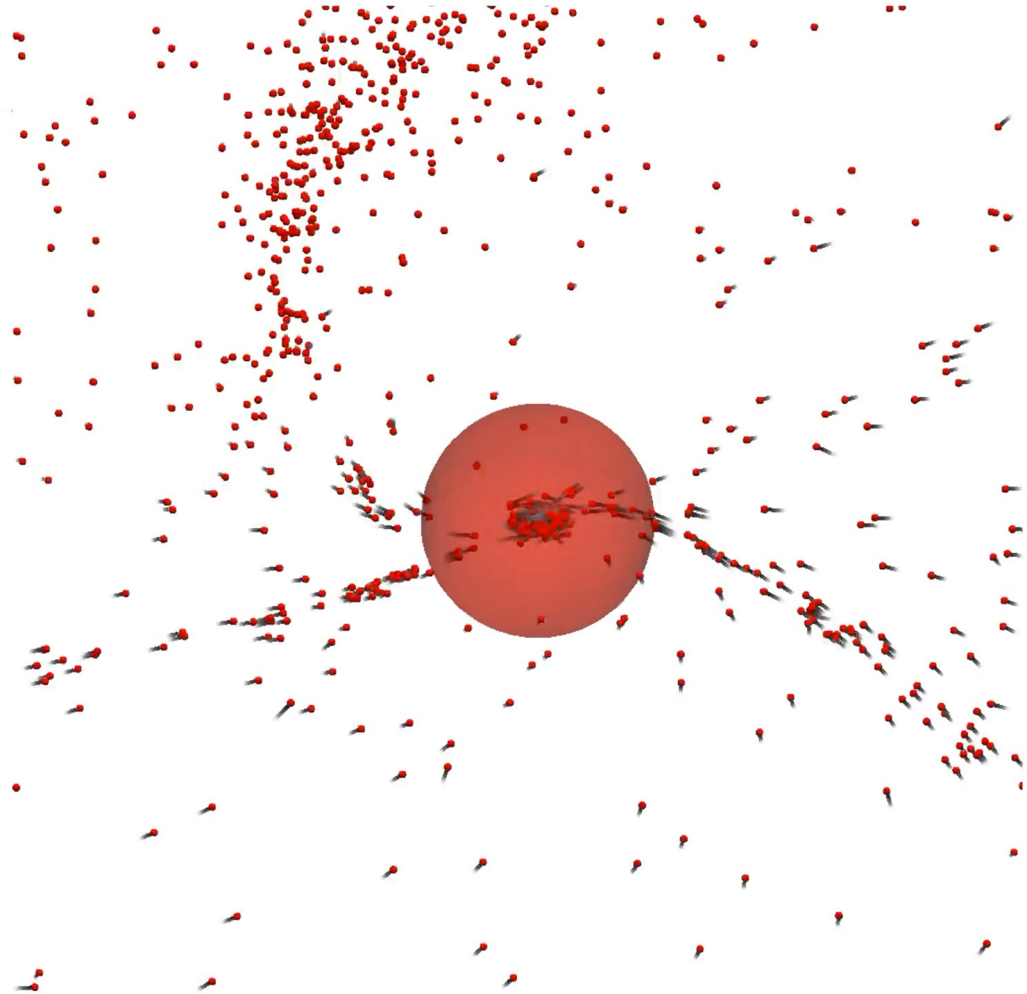
**Snippet 9: ROI render using Pyvista**

```
rho = np.load(path+"rho.npy")
pos = np.load(path+"pos.npy")
c = # numpy boolean array
points = np.transpose([pos[0][c],pos[1][c],pos[2][c]])
cloud = pv.PolyData(points)
cloud['log(rho) [g/cm^3]'] = np.log10(rho[c])
p = pv.Plotter()
p.add_mesh(cloud, cmap="magma", clim=None, opacity="linear")
p.show()
```



# I – Particle rendering



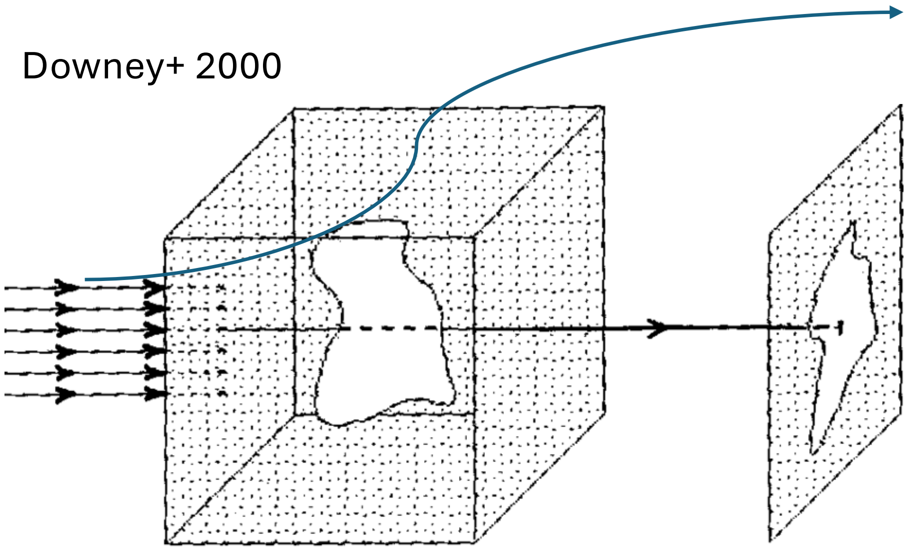



# II – Volume rendering

- Generates 2D image from 3D scalar field

- Allows for visualization of data without explicit extraction of geometrical surface

Downey+ 2000



Transfer function is applied (e.g., sigmoid, linear, logarithmic…)

**Needs uniformly-spaced dataset**

# II – Volume rendering

**Step 1:** Create a uniformly-spaced grid

**Subset of simulation domain**

AMR2CUBE

Stacked set of slices

etc...

**Step 2:** Load data into software

**Snippet 3: Loading a scalar field onto a Pyvista uniform grid**

```
import pyvista as pv
stack = np.load(path)
xmin = -100; xmax = 100;
ymin = -100; ymax = 100;
zmin = -100; zmax = 100;
dx = (xmax-xmin)/stack.shape[0]
dy = (ymax-ymin)/stack.shape[1]
dz = (zmax-zmin)/stack.shape[2]
# Create the grid on which PyVista can deposit the data
grid = pv.ImageData()
grid.dimensions = stack.shape
grid.origin = [xmin, ymin, zmin]
grid.spacing = [dx, dy, dz]
grid.point_data['scalar'] = stack.flatten(order='C')
```

# II – Volume rendering

**Step 1:** Create a uniformly-spaced grid

**Subset of simulation domain**

AMR2CUBE

Stacked set of slices

etc…

**Step 2:** Load data into software

**Step 3:** Visualize

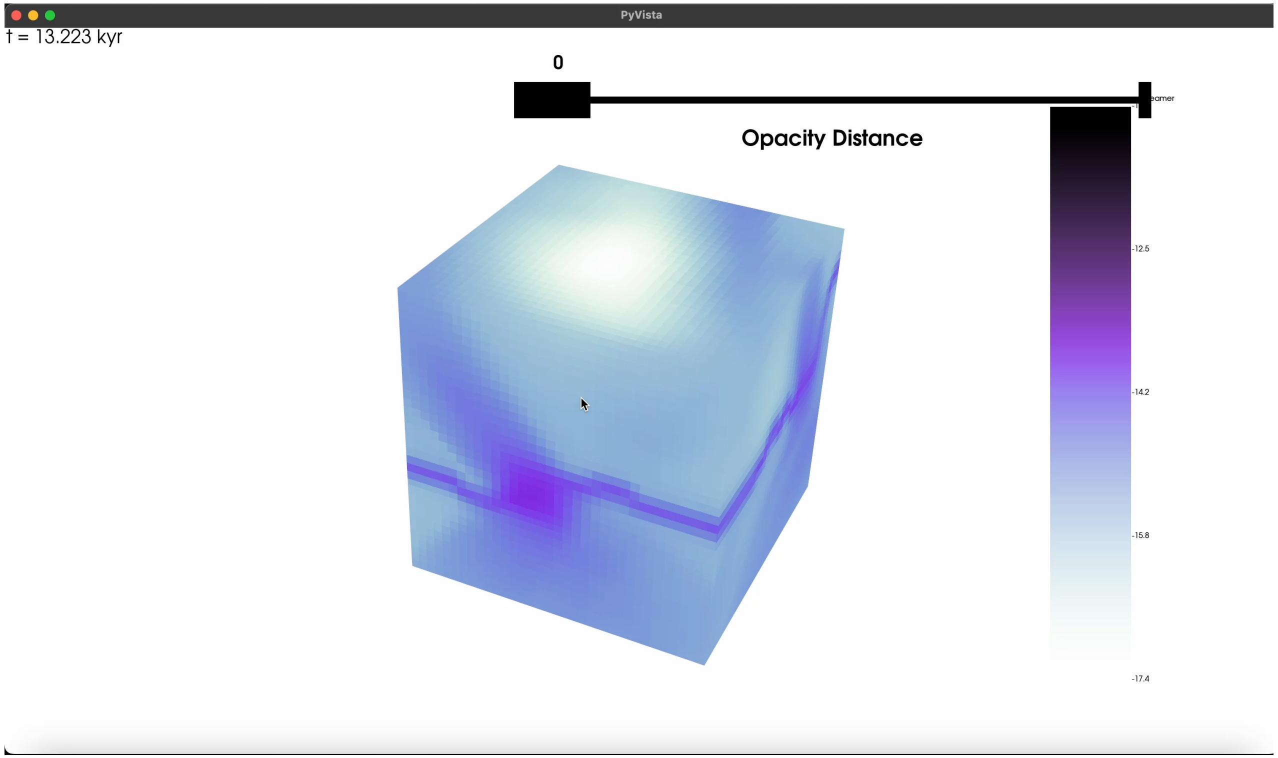
**Snippet 5: Volume render of a scalar field with interactive opacity using Pyvista**

```python
def update_opacity_distance(val):
        vr.GetProperty().SetScalarOpacityUnitDistance(val)
        return

p = pv.Plotter()
vr = p.add_volume(grid, scalars="scalar", cmap="magma", clim=[-4, -1],
             opacity="linear", mapper="gpu",
             opacity_unit_distance=grid.length / 25,
             shade=True, scalar_bar_args={"interactive":True})
f = lambda val: vr.GetProperty().SetScalarOpacityUnitDistance(val)
p.add_slider_widget(rng=[0, grid.length/4],
    callback=update_opacity_distance, title="Opacity Distance")
p.show_grid()
```

# II – Volume rendering

$137^3$

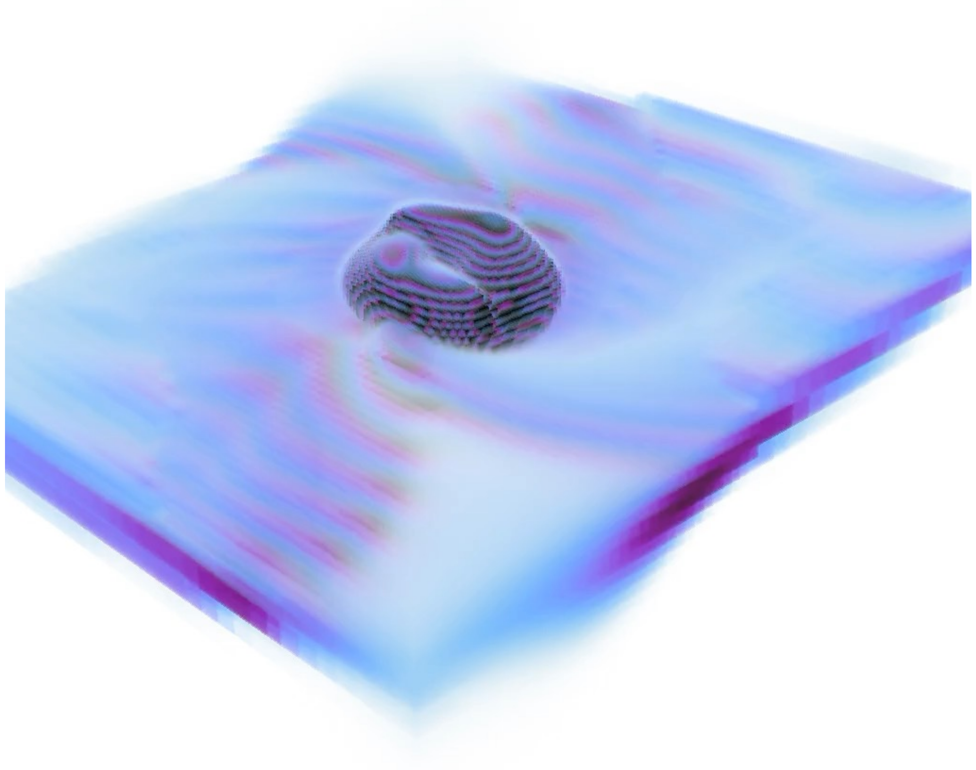


# II – Volume rendering

t = 0.000 kyr

**Script run on cluster to produce animation frames**

**Ahmad+ 2025c (submitted)**



# II – Volume rendering

t = 0.000 days

**Ahmad+ 2025b: Birth of a Brown Dwarf**

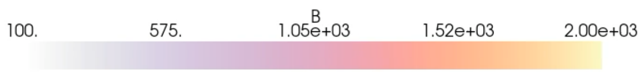
B field

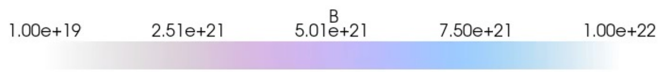
Electric current



$\mathbf{435^3}$

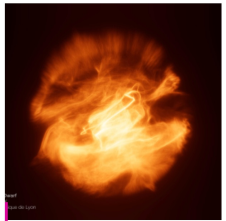
$\approx$ 0.02 AU



# Collaboration with an Alex Andrix (Artist)

TERRE & UNIVERS

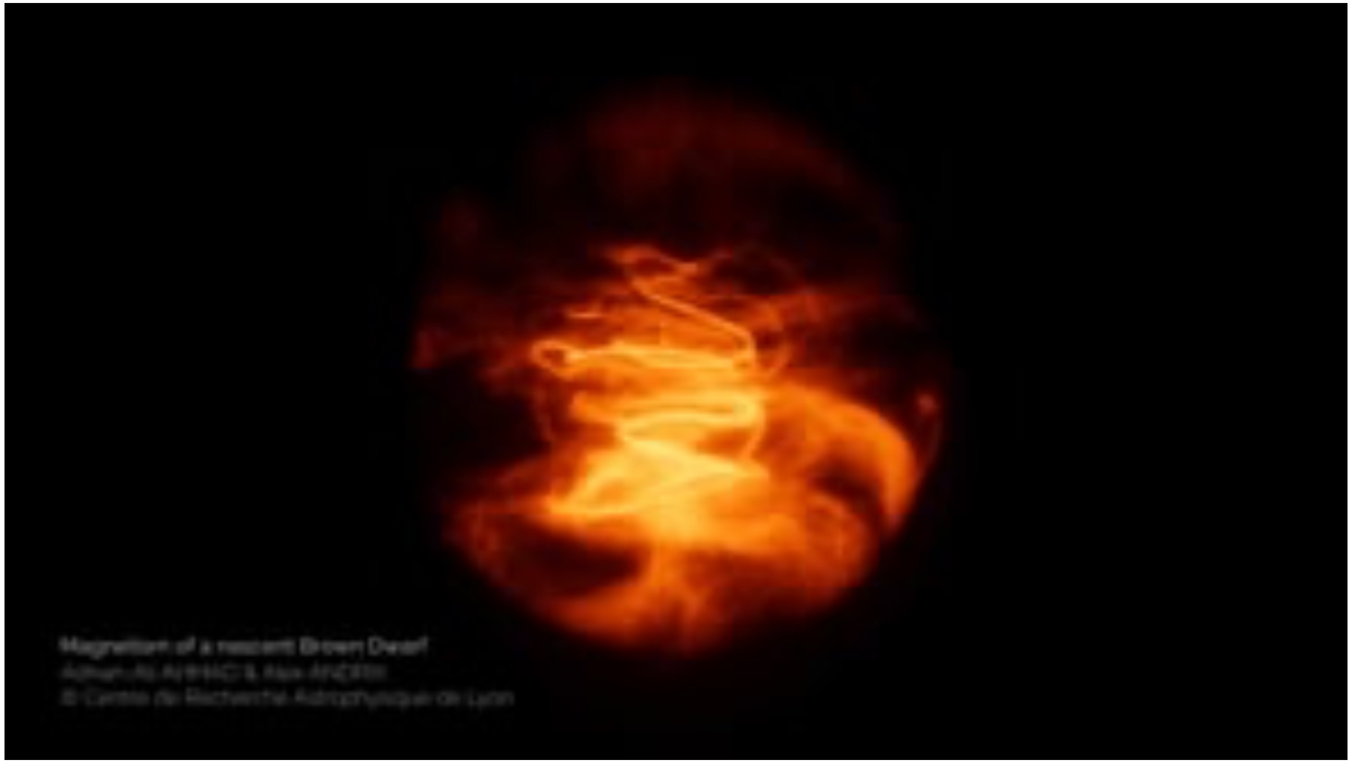


© Adnan-Ali AHMAD & Alex ANDRIX - CRAL

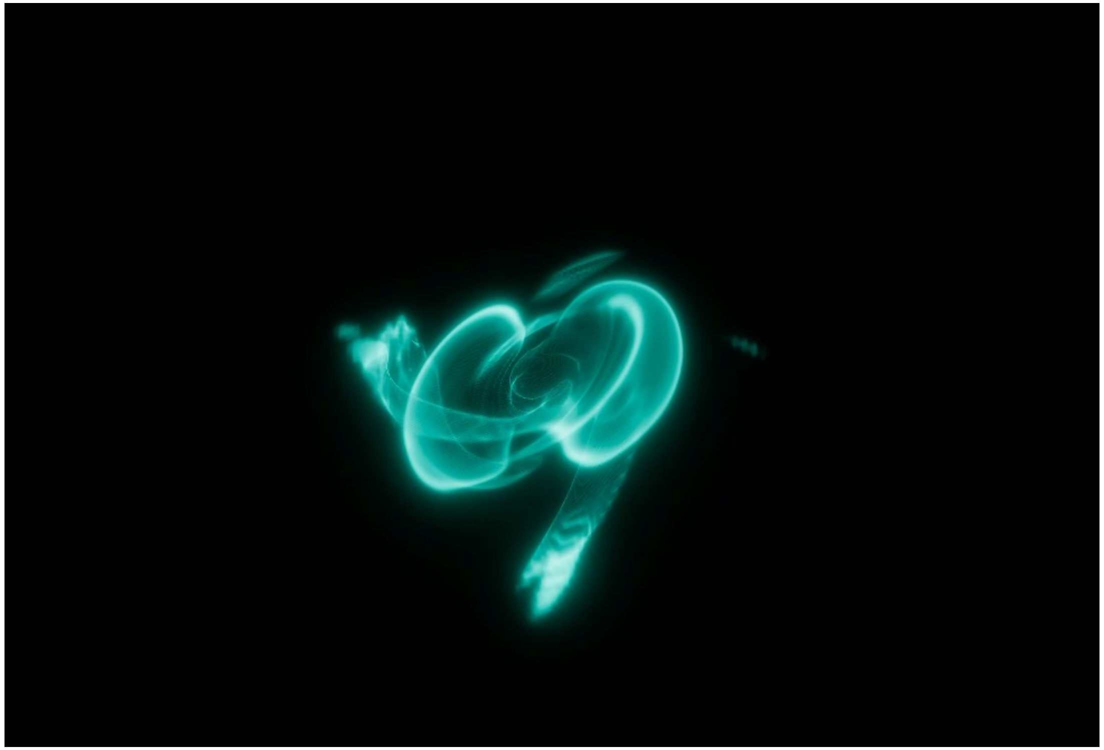
Accueil › Actualités

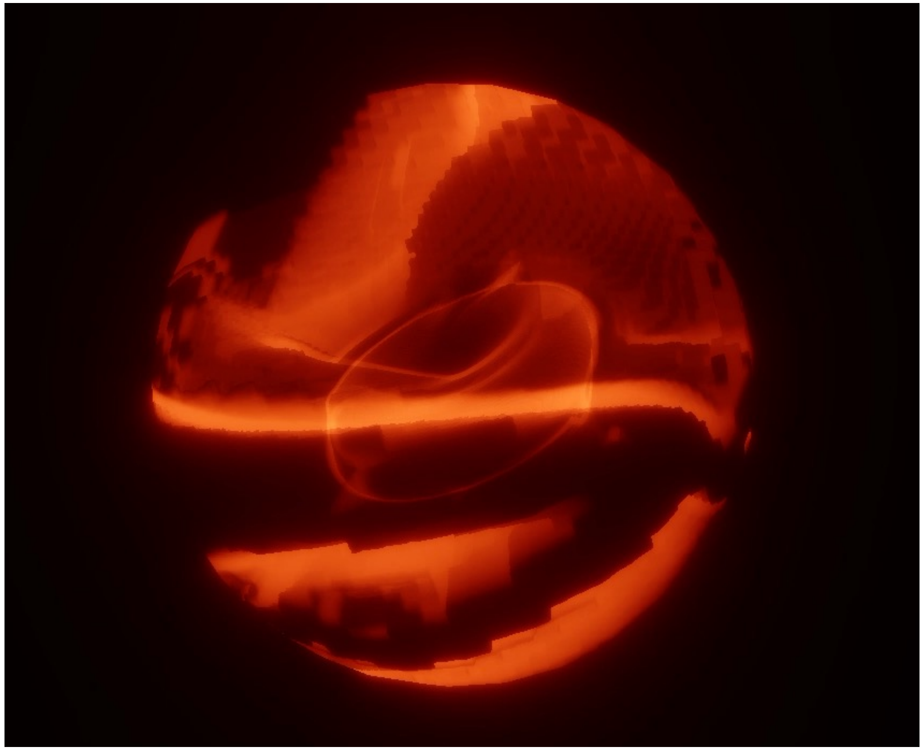
## Première simulation de la formation d'une naine brune par effondrement gravitationnel

*30 septembre 2025* RÉSULTAT SCIENTIFIQUE UNIVERS



# Collaboration with an Alex Andrix (Artist)

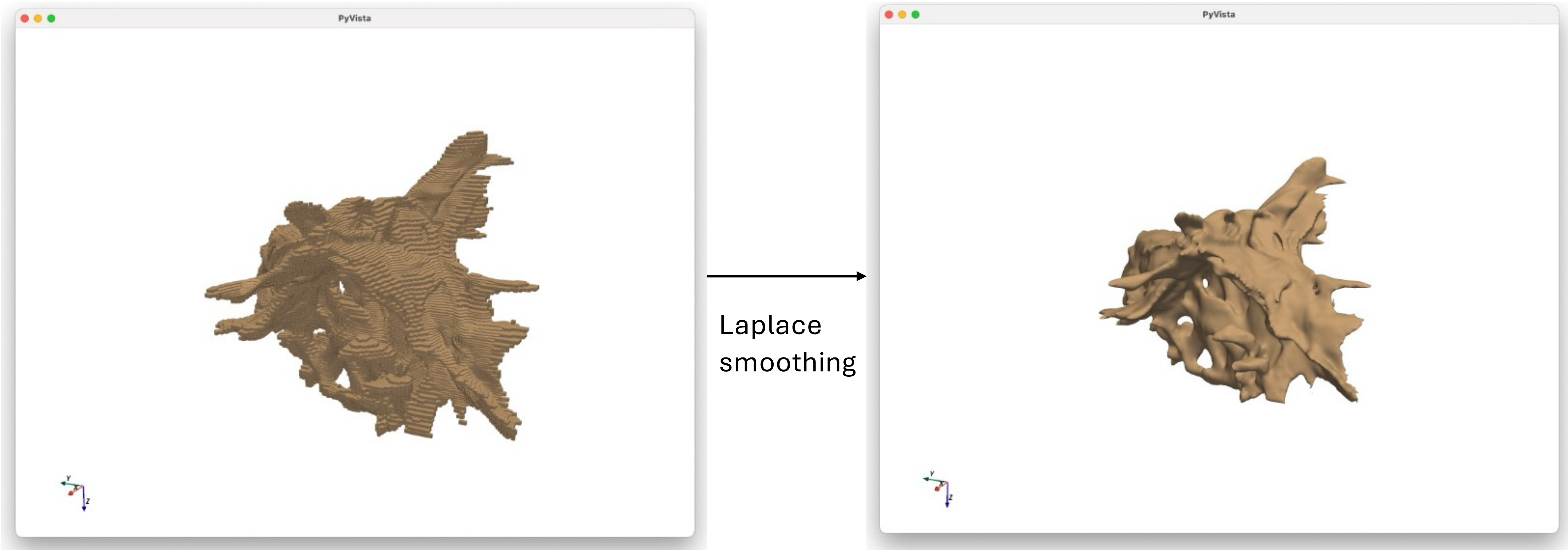




# III – Iso-contouring

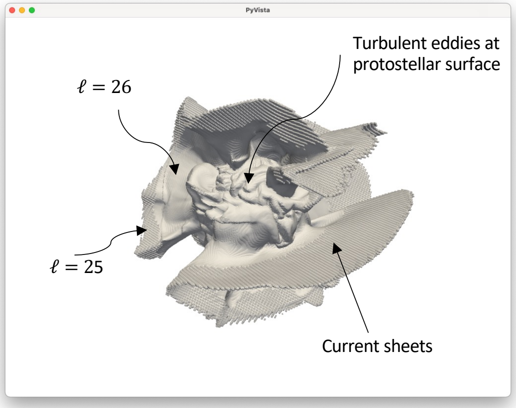
**Snippet 8: Iso-contouring using Pyvista**

```python
contours = grid.contour(np.arange(rhomin, rhomax+1, .5))
smooth_contour = contours..smooth(n_iter=2000, progress_bar=True)
p = pv.Plotter()
p.add_mesh(contours, opacity=.4, clim=[rhomin, rhomax], cmap="viridis")
p.show()
```

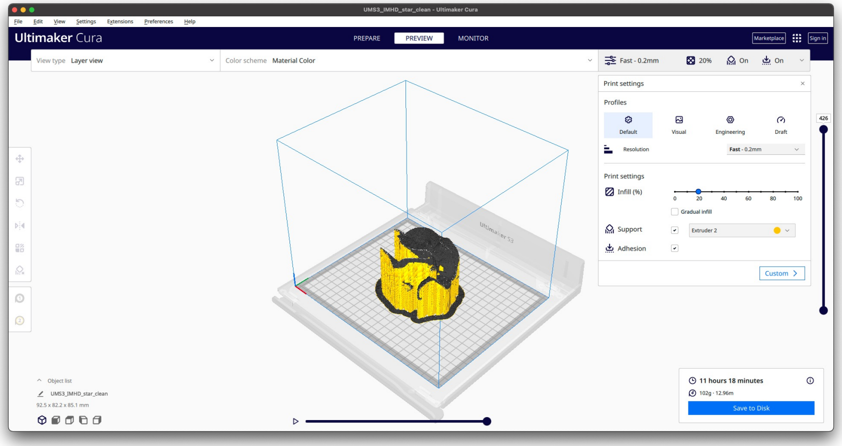


# III – 3D printing

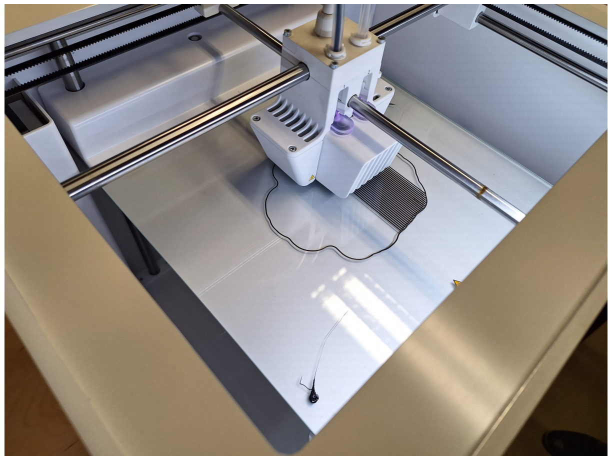
*a*




*b*

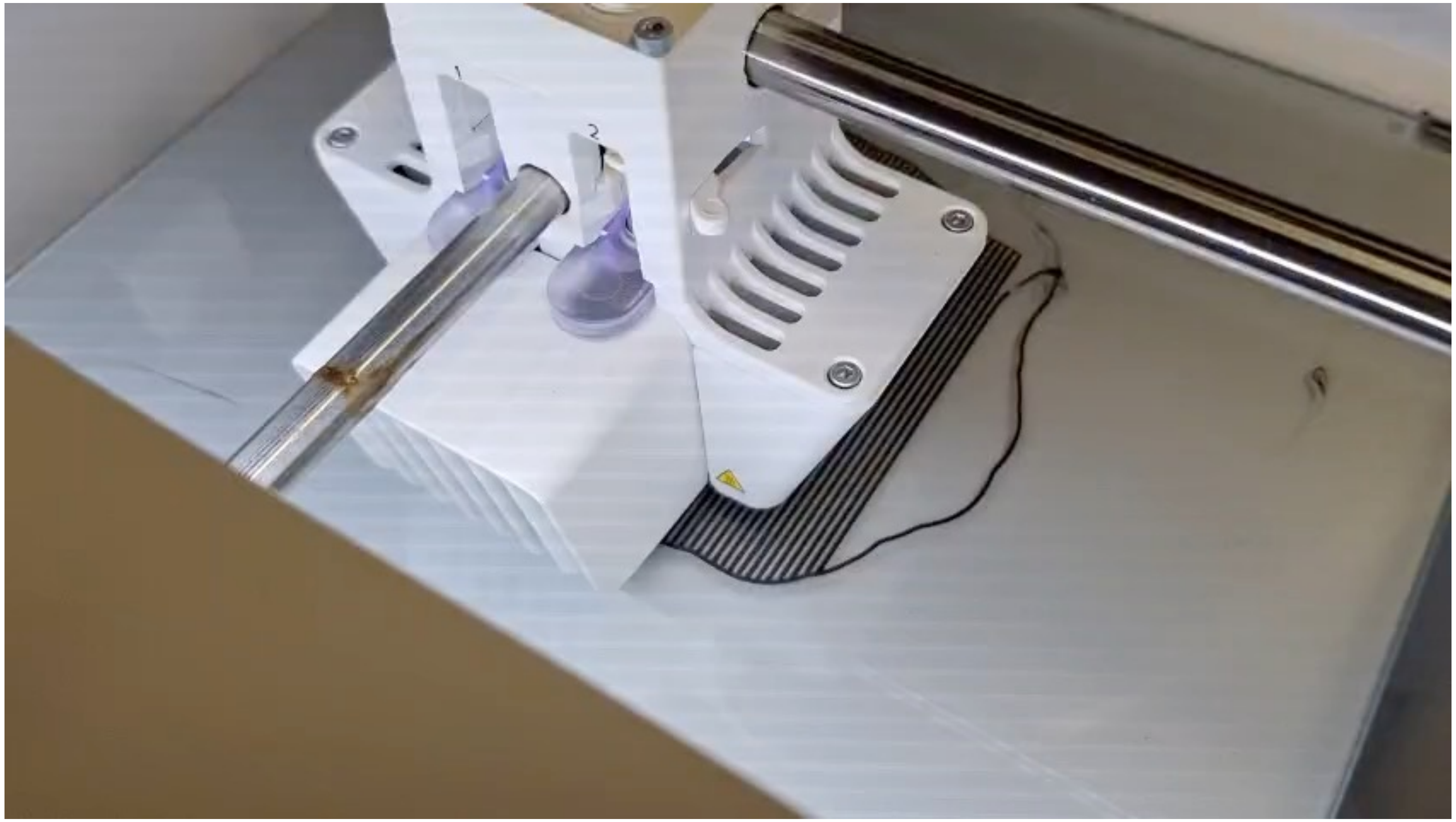



*c*

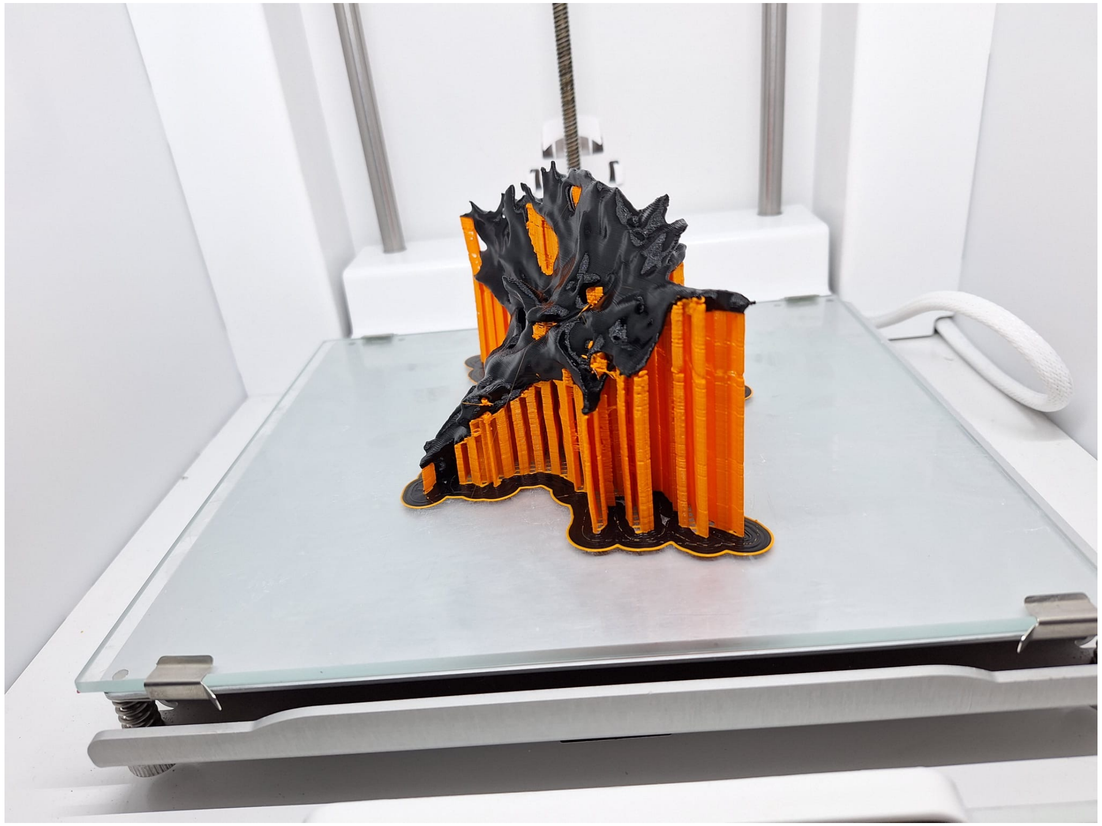


*d*

# III – 3D printing

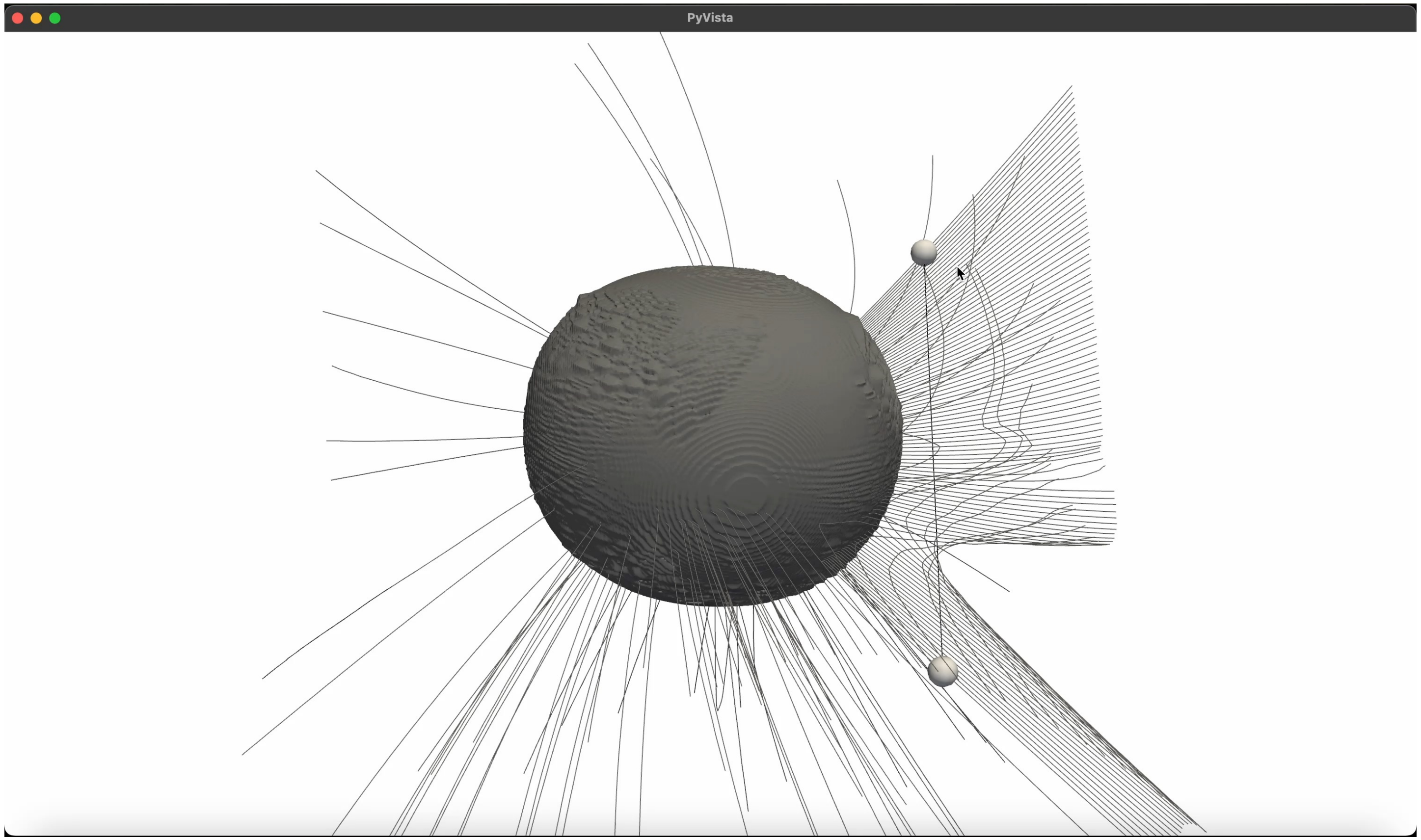


# III – 3D printing



# III – Streamlines

**Interactive streamlines of magnetic vector field near brown dwarf**

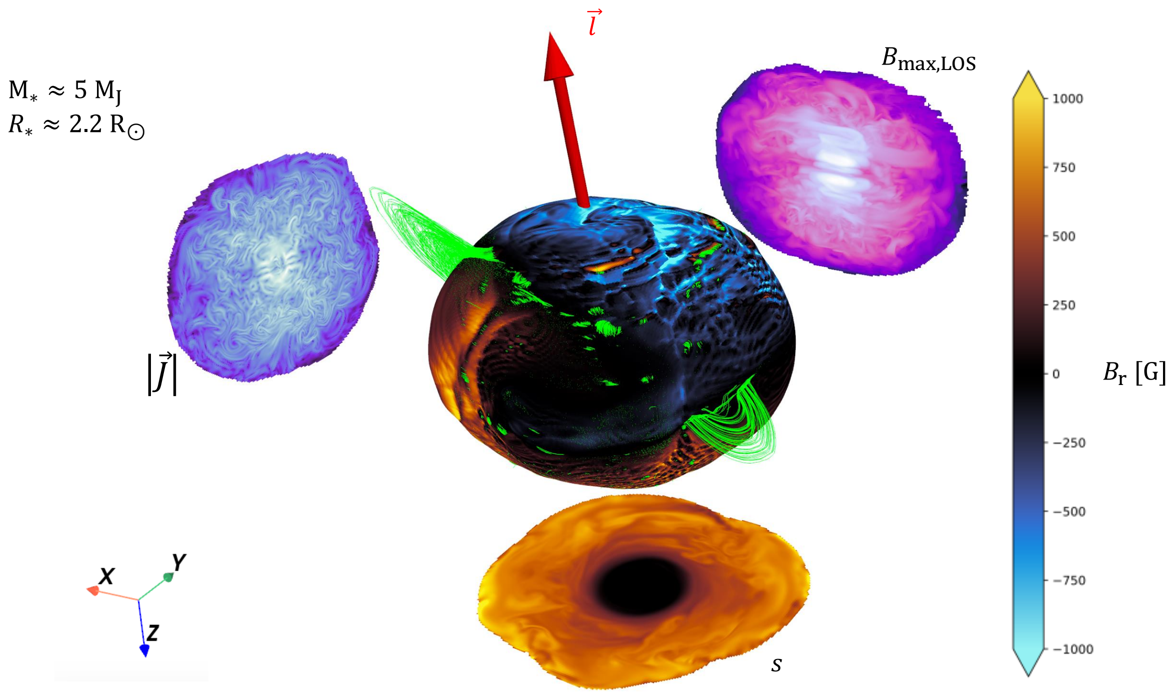



- Vector field instead of scalar field
- Requires quite a bit of fine-tuning to find good starting positions

# III – Streamlines

- Vector field instead of scalar field
- Requires quite a bit of fine-tuning to find good starting positions

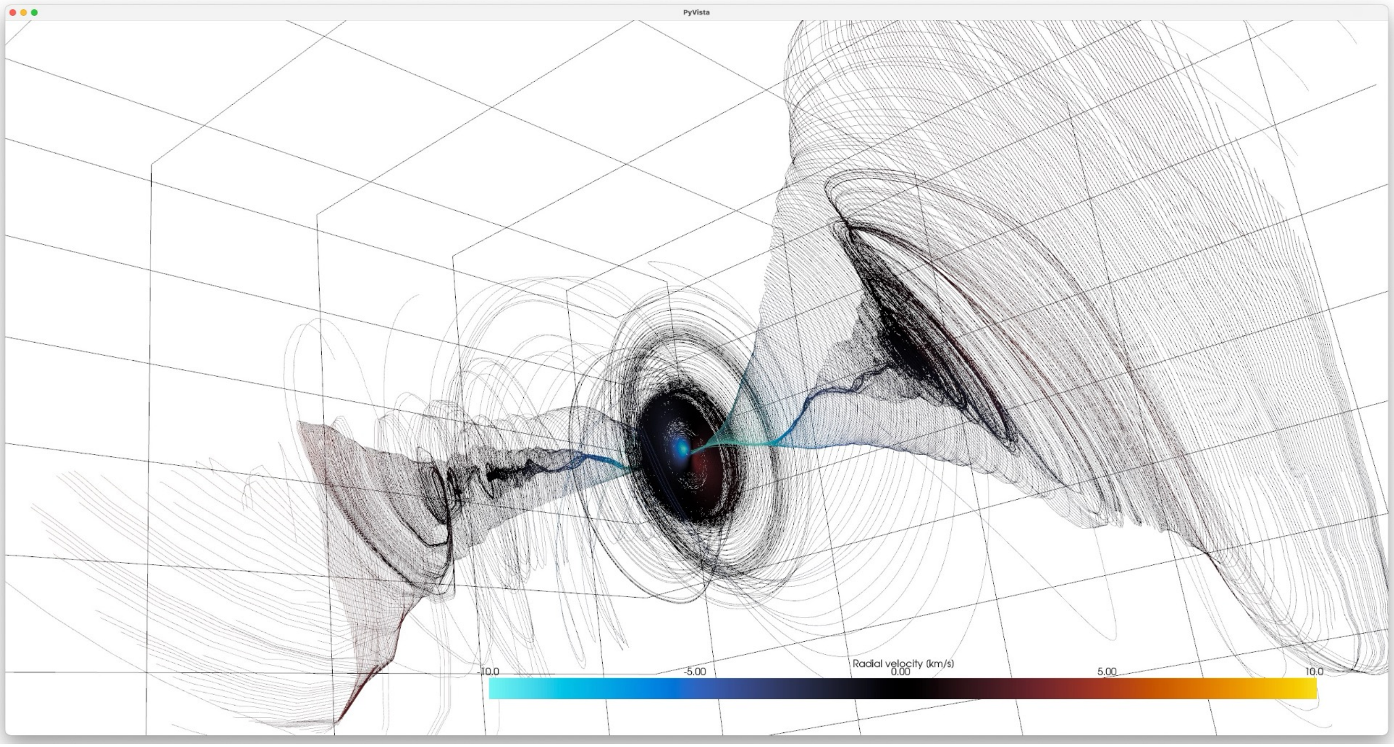
**Final publication-ready render**
**Ahmad+ 2025b**



# III – Streamlines

- Vector field instead of scalar field
- Requires quite a bit of fine-tuning to find good starting positions

**Static streamlines of velocity vector field in outflow**

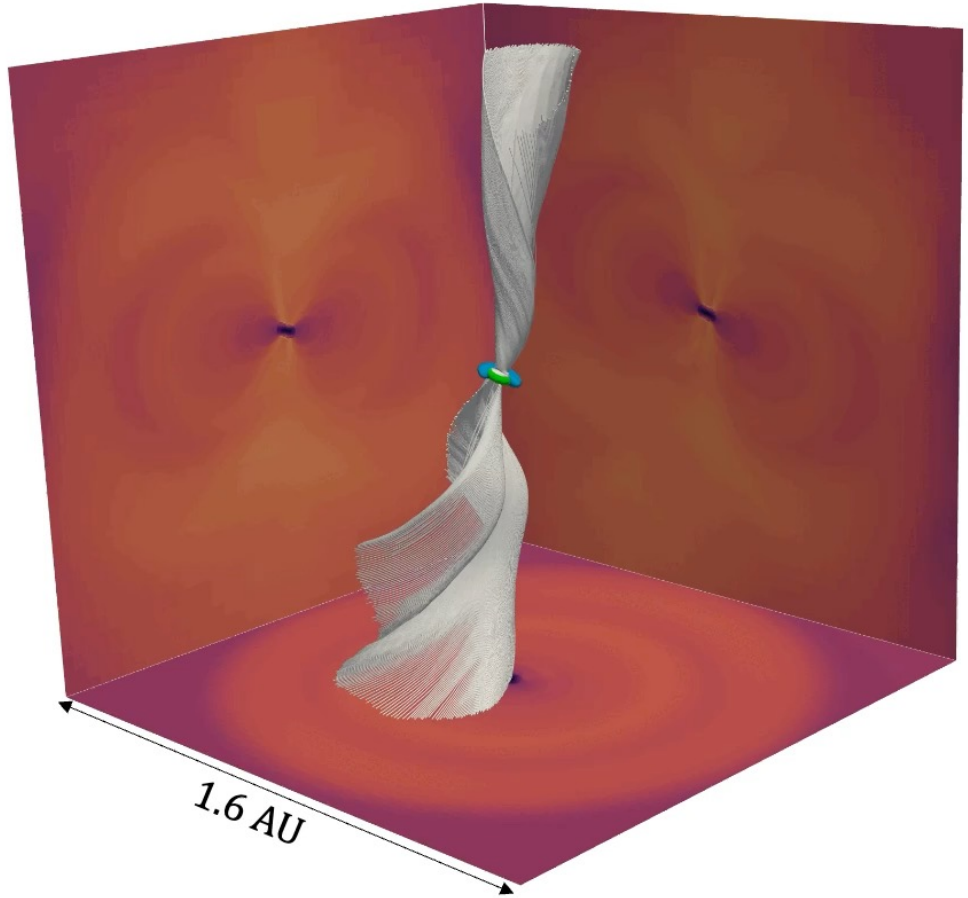


# Some examples of combinations

**Birth of a protostar and disk**

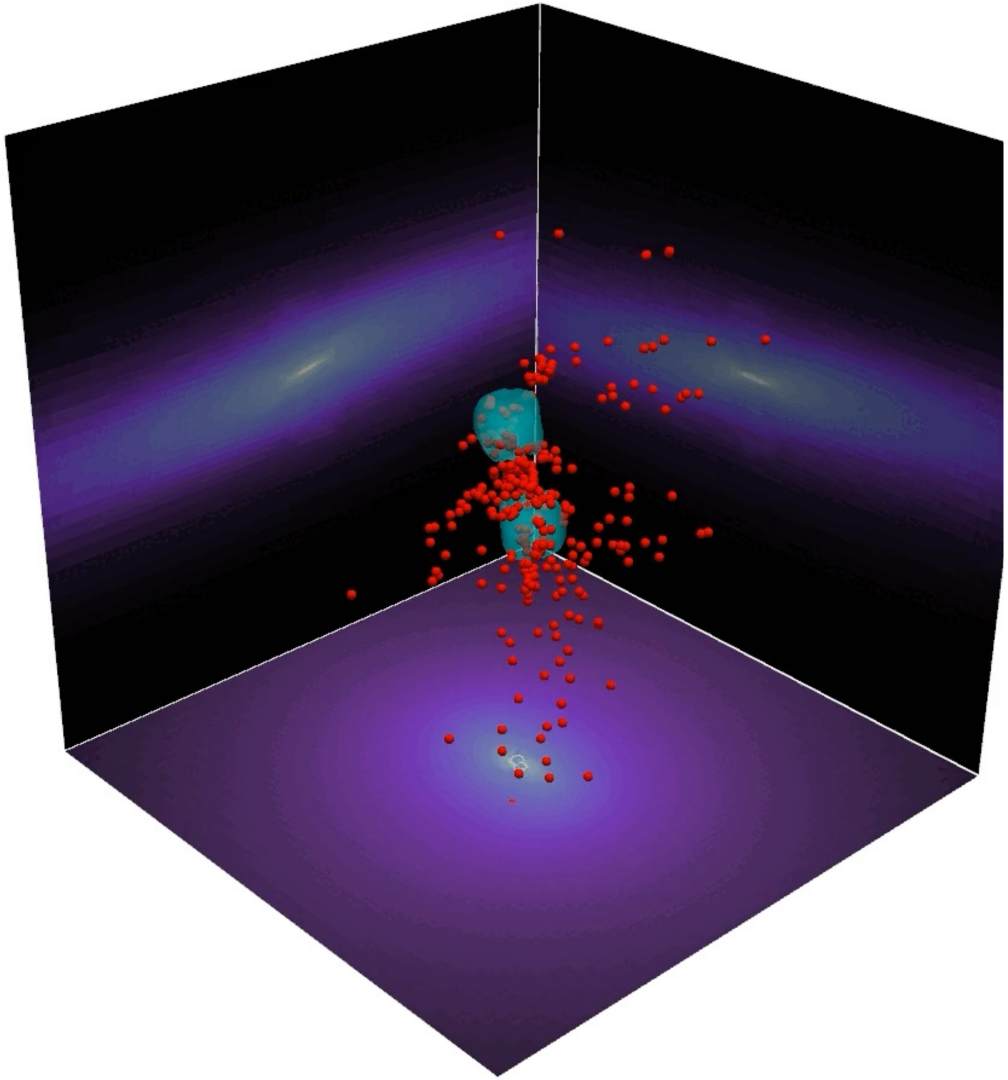
t = -0.02 (yr)

**Ahmad+ 2024**



# Some examples of combinations

**Tracer particles in outflow cavity**



# Summary

- Lots of techniques available for 3D visulizations
- Play with low-resolution data locally
- Deploy pipelines on clusters