**Marco1b User Guide**

**Introduction**

*Marco1b* is designed to smooth out segmented images by applying different blurring methods, which reduces rough edges and noise left after segmentation. This is a critical step before further analysis in the Artificial Intelligence directed Voxel Extraction (AIVE) pipeline.

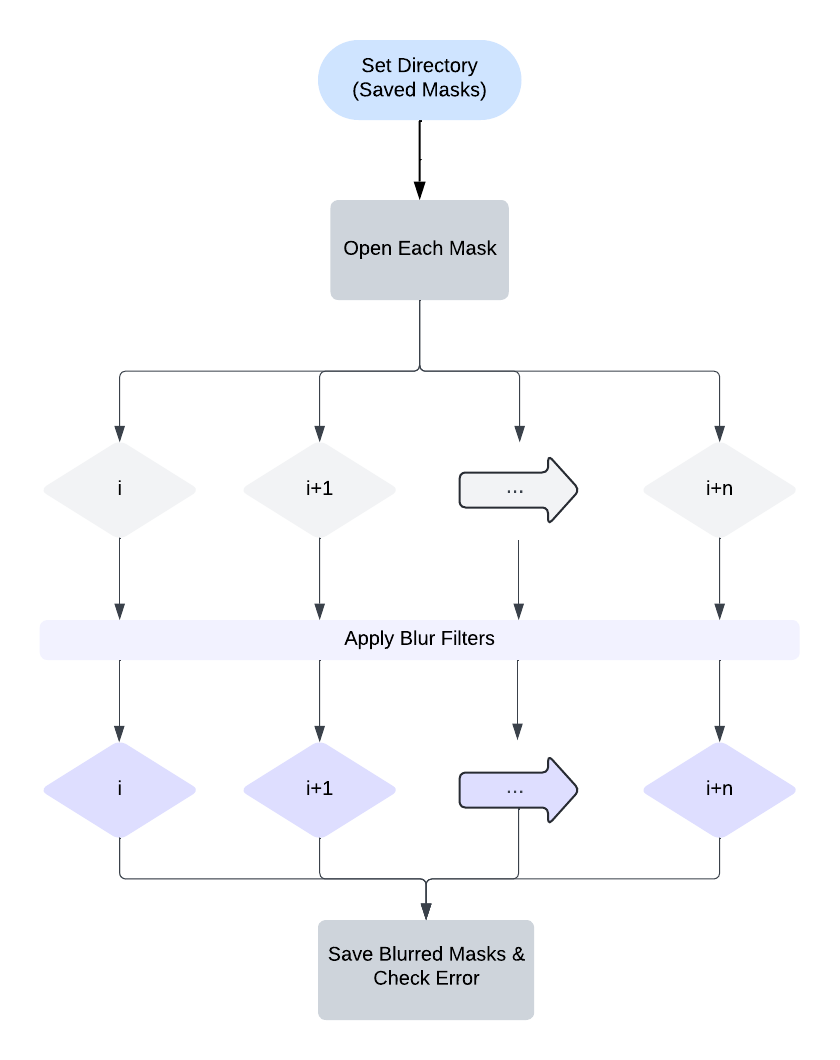
**Purpose**

This documentation will guide you through the setup and use of *Marco1b*, focusing on how to select the proper blur method and parameters to achieve the desired output.

**Who Should Use This Guide**

This guide is for users working with segmented image datasets who need to refine their images by blurring and smoothing them for further processing and analysis.

**Diagram**

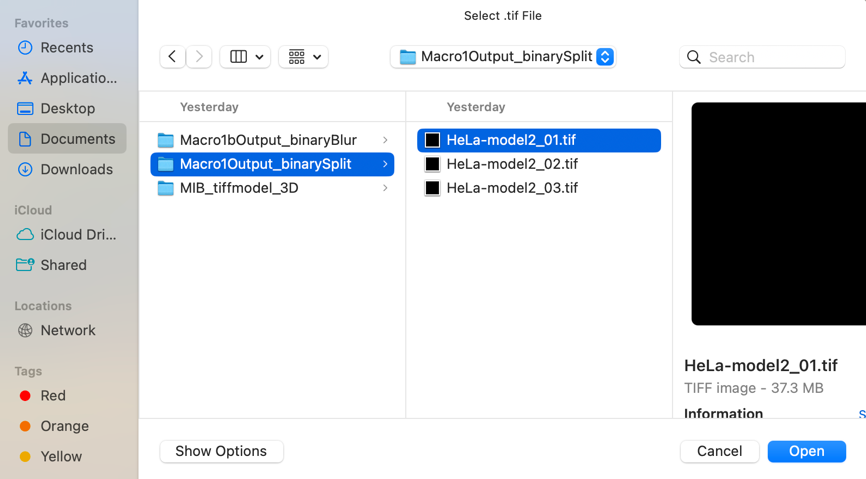


**System Requirements**

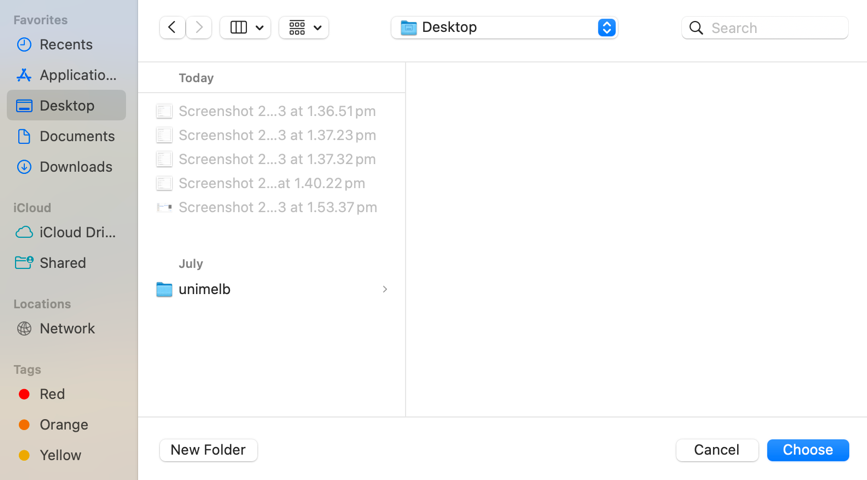
* **Software**:
  + *Python 3.x* (for Python users)
* **Supported Image Formats**:
  + TIFF (.tif) image stacks (3D)
* **Operating Systems**:
  + Windows, macOS, or Linux – In this document, I will demonstrate using the macOS interface.

**How to run it in Python**

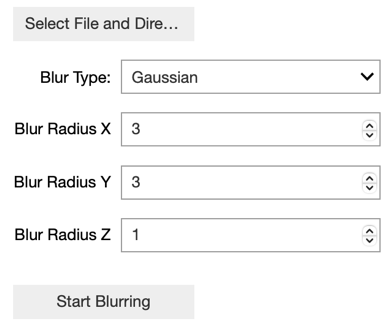
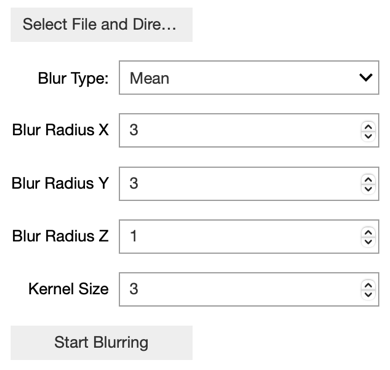
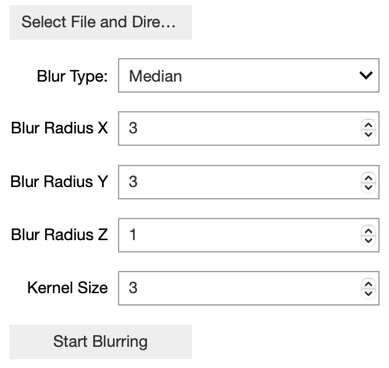
* 1. **Select input file:** Look the .tif file



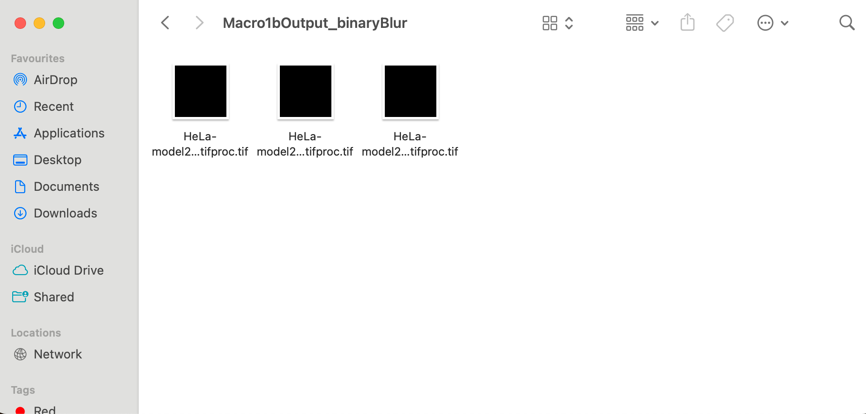
* 1. **Select the output directory:** Define the directory for saving the processed output



* 1. **Choose blur method:** Gaussian, Mean, or Median blur options are available

* **Different blur method**
  + **Gaussian blur**
    - **Key Idea**: Gaussian blur is based on **distance**, so the centre pixel is more important than the pixels farther away.
  + **Mean blur**
    - **Key Idea:** Mean blur is based on the **mean (average)** of all the pixel values in a selected area (kernel)
  + **Median blur**
    - **Key Idea:** Median blur replaces the centre pixel with the **median (middle value)** of the surrounding pixels.
  1. **Save the result:** The final blurred images are saved



**Why are Our Filters Uneven?**

When working with 3D images (like in bioimage analysis), each small part of the image is called a voxel. Think of a voxel like a 3D pixel. These voxels represent real-world measurements, like the size of a cell or tissue.

For example:

* Voxel Sizes: for our test image, each voxel has dimensions of
  + 3nm in the X direction (left-right)
  + 3nm in the Y direction (up-down)
  + 10nm in the Z direction (front-back)

Since these dimensions aren't the same (X and Y are 3nm, but Z is 10nm), we call these anisotropic voxels. If they were equal (like 3nm in all directions), we would call them isotropic voxels.

**Why Does This Matter?**

Because the voxel isn't a perfect cube, when we blur the image, we have to apply different amounts of blur in each direction to get a balanced result. If we used the same amount of blur in every direction, it wouldn’t look right because the voxel isn’t a perfect square.

**Choosing the Right Blur Size – using gaussian as exmaple**

Most blur methods in Python (like OpenCV’s blur or SciPy’s median) already know how to apply the blur—you just need to tell them how big the area (kernel) should be.

1. **For a Gaussian Blur**:  
   Gaussian blur works a bit differently because it doesn’t just look at nearby voxels—it gives more importance to voxels that are closer and less importance to voxels that are farther away.
2. **Calculating the Blur Radius**:
   * The blur radius is the number of voxels we want to include in our calculation.
   * For our **Z-axis**, we are lucky—it’s already **10nm**, so we just need to include **voxel on each side** of our middle voxel.
     + This gives us a blur radius of **1 voxel in Z**.
     + For example, if the Z coordinate of the middle voxel is 25, we will blur using voxels 24, 25, and 26.
   * For **X and Y**, the voxel size is smaller—only **3nm**. So, if we want to blur across **10nm**, we divide:
     + 10nm/3nm=3.33, which we round to **3 voxels** in both X and Y directions.
     + So, for X and Y, our blur radius is **3 voxels** on each side.
     + For example, if X = 100, we will use voxels 97, 98, 99, 100, 101, 102, and 103.

**The Final Blur Filter**

By combining these, we get a **3D kernel** that is **7x7x3**. This means:

* **7 voxels** in X (3 on each side of the middle voxel)
* **7 voxels** in Y (3 on each side)
* **3 voxels** in Z (1 on each side)

**Advanced Optimization**

* **Kernel Flexibility**: Allow switching between cubic and spherical kernels to control the shape of the blur in 3D space.
* **Performance Optimization**: Enable GPU acceleration for faster processing, especially with large datasets.