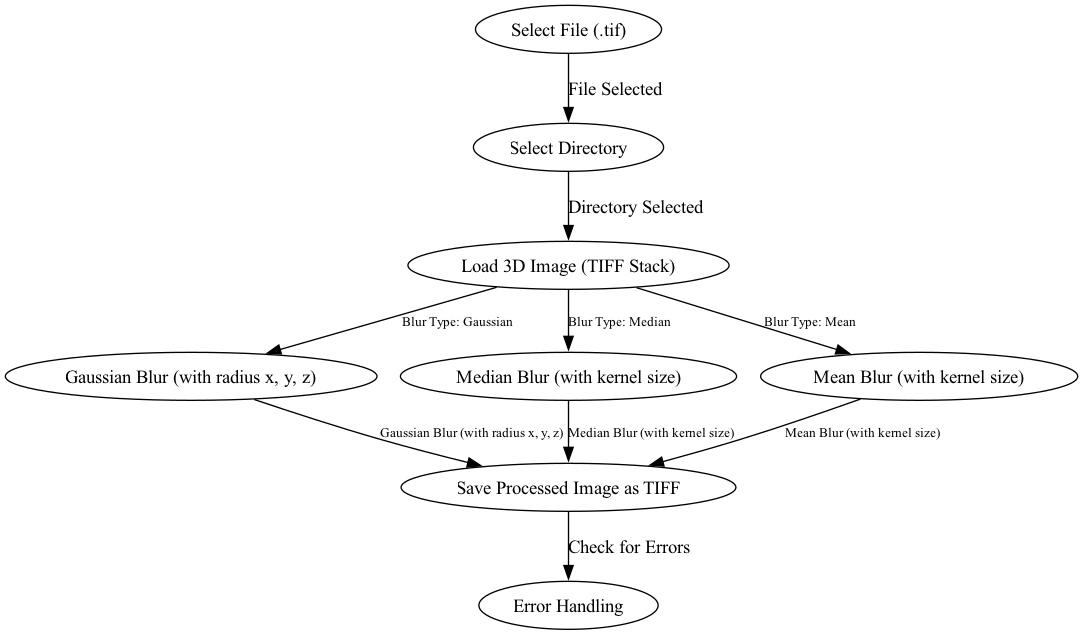
**Marco1b**

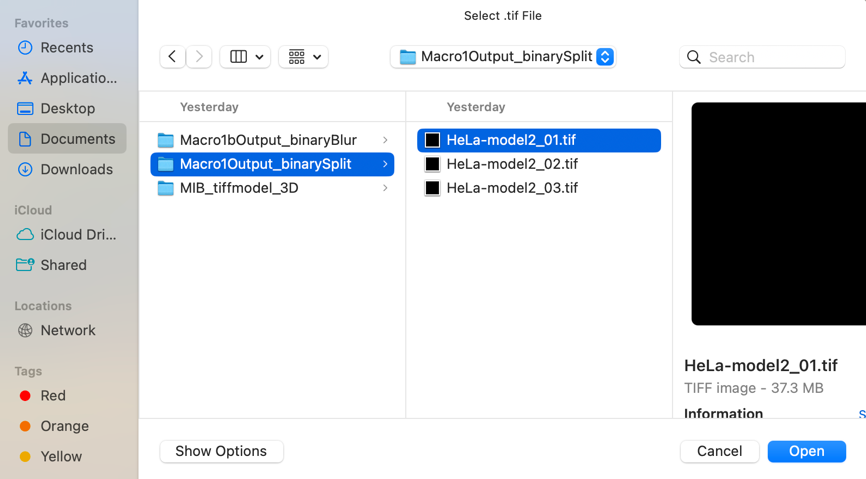
1. What’s Marco1b use?

Marco1b is used to blur the images and smooth out the segmented structures, reducing rough edges or noise that may remain after segmentation. This process prepares the images for further processing in the AIVE pipeline, ensuring cleaner and more accurate segmented parts for subsequent steps, such as voxel extraction or detailed analysis. The blurring is performed by manually choosing the appropriate parameters to achieve the desired level of smoothing, using one of three methods: Gaussian, Mean, or Median. Additionally, it requires selecting different blur radii and kernel sizes to fine-tune the effect based on the specific image characteristics.

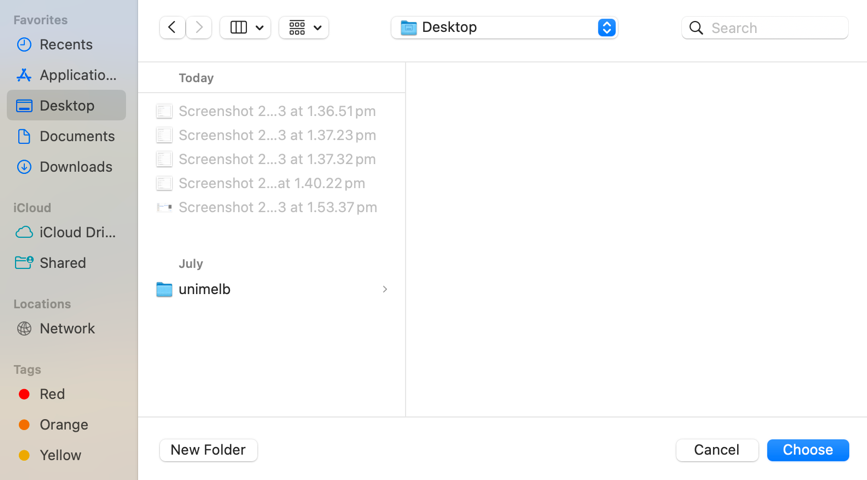
1. Process diagram



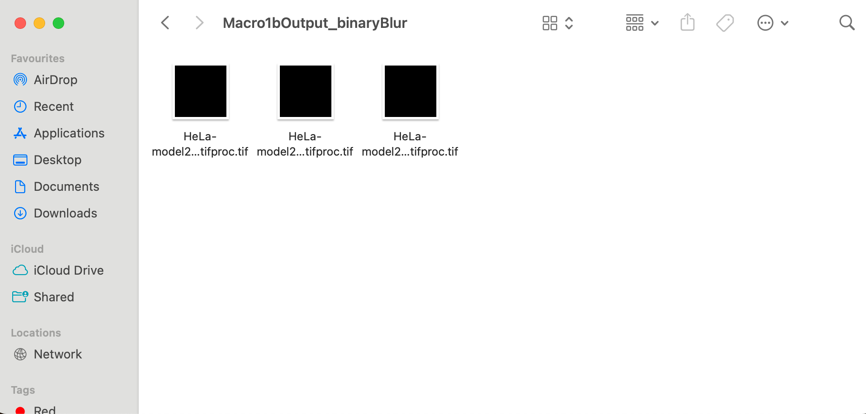
1. **Select File (.tif)**: User selects the .tif file (marco1 output) to process.
2. **Select Directory**: The output directory is chosen where the processed file will be saved.
3. **Load 3D Image (TIFF Stack)**: The selected TIFF stack is loaded.
4. **Choose Blur Type**: The user selects between three blurring methods:
   1. **Gaussian Blur (radius x, y, z)**
   2. **Median Blur (kernel size)**
   3. **Mean Blur (kernel size)**
5. **Save Processed Image as TIFF**: Once the blur is applied, the processed image is saved.
6. **Check for Errors & Error Handling**: Any potential issues during processing are checked and handled.
7. How to use it in Fiji and Python
8. Fiji (Based on JAVA language)
   1. **Select input file:** Look the .tif file



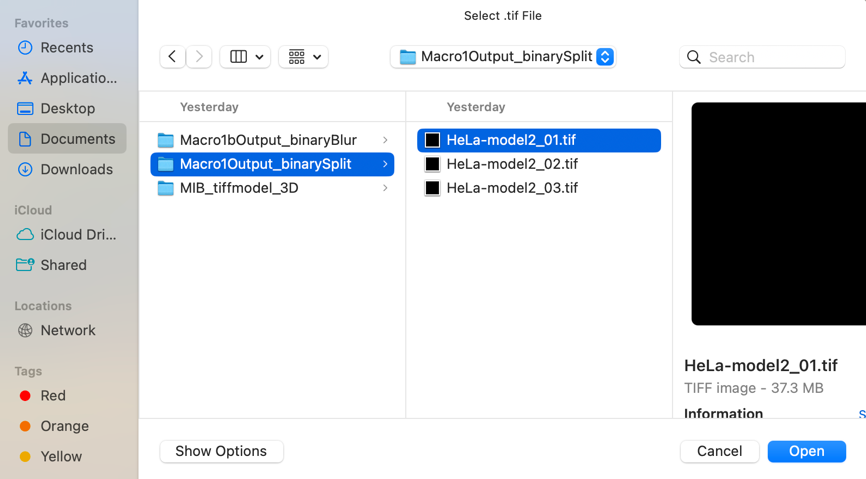
* 1. **Select the output directory:** Define the directory for saving the processed output



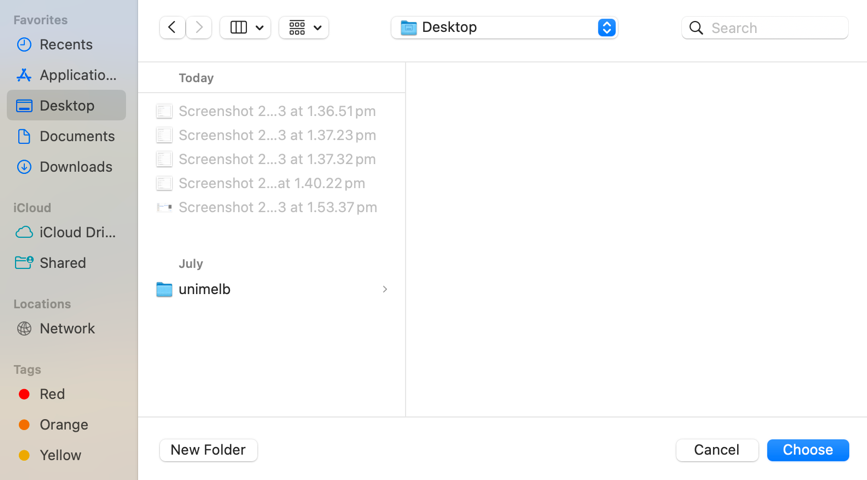
* 1. **Save the result:** The final blurred images are saved



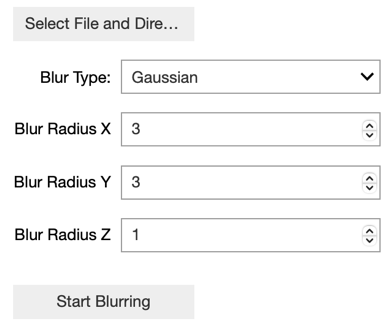
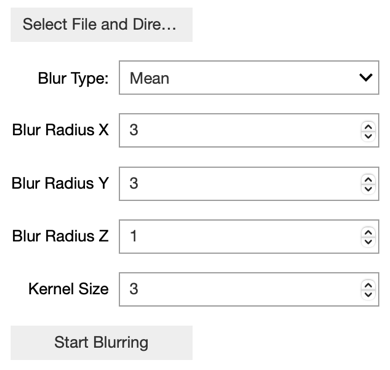
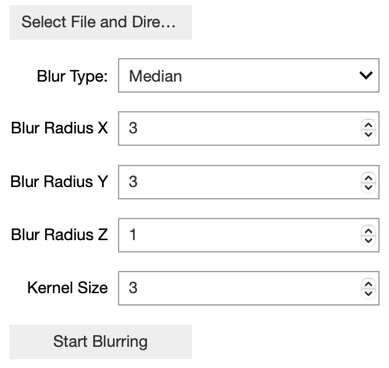
1. Python
   1. **Select input file:** Look the .tif file



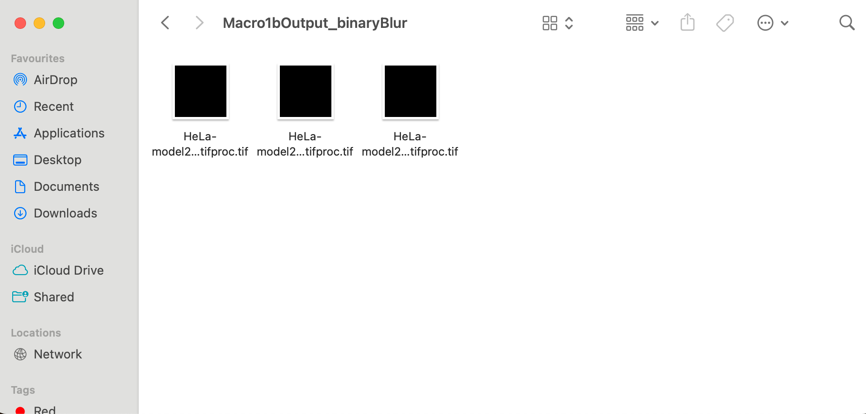
* 1. **Select the output directory:** Define the directory for saving the processed output



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

* + 1. Gaussian Blur:
       1. **How it works:** Gaussian blur applies a Gaussian function to smooth the image. Pixels closer to the center of the kernel are given more weight, while those farther away contribute less, resulting in a smooth blur that gradually reduces the influence of distant pixels.
       2. **Kernel Size:** The kernel size determines the window or area over which the blur is applied. A larger kernel size results in stronger blur but requires more computational resources. In 3D images, the kernel size can be adjusted independently in the x, y, and z directions.
       3. **Sigma (Standard Deviation):** Sigma controls the spread of the Gaussian function, determining how far the influence of each pixel spreads. A larger sigma results in a more diffuse, softer blur. Sigma can also be adjusted separately in the x, y, and z directions.
       4. **Strengths:** Gaussian blur is great for reducing noise while maintaining the overall structure of the image. It’s ideal for producing a natural, smooth blur and preserving gradients. Fine-tuning the blur allows for effective noise reduction with minimal impact on edges.
    2. Mean Blur:
       1. **How it works:** Mean blur (also known as averaging) calculates the average pixel value within the kernel and replaces the central pixel with that average. This results in a uniform blur across the entire image.
       2. **Kernel Size:** Similar to Gaussian blur, the kernel size controls how much area is averaged. A larger kernel increases the degree of smoothing. In 3D, the kernel size can be independently adjusted in the x, y, and z projections.
       3. **Strengths:** Mean blur is computationally efficient and effective at reducing noise by averaging out pixel values. However, it can lead to a loss of detail, especially in areas with sharp edges or fine textures.
    3. Median Blur:
       1. **How it works:** Median blurring replaces the central pixel in the kernel with the median value of the surrounding pixels. This method is especially effective at removing salt-and-pepper noise (isolated pixel noise) without blurring edges.
       2. **Kernel Size:** The size of the kernel defines the neighborhood of pixels considered when calculating the median. A larger kernel increases the effectiveness of noise removal but may blur out finer details.
       3. **Strengths:** Median blur excels at preserving edges while removing isolated noise. It is particularly useful in situations where maintaining sharpness at boundaries is crucial, such as medical or technical imaging.
  1. **Save the result:** The final blurred images are saved



1. Limitation
2. **Different result:** Even when the same algorithm and parameters (such as blur radius and kernel size) are selected, the results can sometimes vary. This inconsistency may be due to factors like image quality, noise, or slight variations in the segmented structures. Differences in image resolution or the inherent randomness in certain algorithms can also contribute to varying outcomes, making it challenging to achieve identical results across different images or processing runs.
3. **High time complexity:** The computational time required for blurring, especially when processing large 3D images or using more complex algorithms like Gaussian, can be significant. This is particularly evident when larger kernel sizes or higher resolutions are used, leading to longer processing times, which may affect the efficiency and scalability of the image processing pipeline, especially in large datasets or real-time applications.
4. How to improve?
5. Kernel Flexibility: Allow users to toggle between cubic and spherical kernels.
6. Performance Optimization: Implement parallel processing and GPU acceleration to handle large image datasets efficiently.
7. Parameter Optimization: Provide smart defaults or auto-tuning to help users find the best blur settings.