# CS75 – Project requirements and Literatures V3.0

## Overview

This project "Interactive Bio-Image Annotation Tool Based on Large-Scale Basic Vision Models" (CS75) is mainly to develop an advanced bio-image annotation platform to improve the efficiency and accuracy of bio-image analysis.

## Main Contents:

### Project Goal (Mainly for literature review)

- The tool uses **Napari** for interactive visualization. **Napari needs to support multi-channel images (RGB, multi-channel support for medical images).**
    - What is Napari? Link to go
    - Main features:
        - Multi-dimensional support: It can efficiently process and visualize 2D, 3D and higher-dimensional image data.
        - Interactive operation: It provides an intuitive user interface and supports operations such as zooming, rotating, and panning of images, which facilitates users to explore data in depth.
        - Plug-in ecosystem: With a rich plug-in library, users can expand Napari's functions according to specific needs to meet the visualization and analysis needs of different fields.
    - Related literature and tutorials:
        - Napari Image Visualization Tutorial
        - Using Napari for interactive visualization
        - Napari Start Guide

- Combined with **Segment Anything Model (SAM)** for automated cell segmentation, based on large-scale vision models to accurately detect targets in multi-dimensional microscope images. **SAM is primarily used for cue-driven segmentation, but needs to be improved to accommodate 3D dimensional data (it currently does not handle 3D data efficiently). We need to find ways to improve on 3D data while limiting video data.**
    - What is SAM? Link to go
    - Main features:
        - Hint-driven segmentation: SAM accepts multiple forms of hints (such as points, boxes, text, or masks) to indicate the object to be segmented. Link to go
        - Zero-shot transfer capability: Without specialized training or fine-tuning for specific tasks or datasets, SAM can be directly applied to new image distributions and tasks, demonstrating strong generalization capabilities. Link to go
        - Large-scale dataset support: SAM is trained on a large-scale dataset (SA-1B) containing more than 110,000 images and 1 billion masks, ensuring the robustness and wide applicability of the model. Link to go No.1 / Link to go No.2 / Link to go No.3

    - Related Literature and tutorials:
        - Segment Anything User Manual (Interactive Data Annotation)
        - Segment Anything Model Basic Usage Learning Experience
        - General Image Segmentation Model: Segment Anything Model (SAM) Training Tutorial
        - LLM Large Model: Segment Anything Model Principle Explanation
        - Open Source Automatic Segmentation Segment-Anything Cutout AI Usage Tutorial

Researchers can manually adjust the segmentation results generated by AI to achieve an automated + manually corrected annotation method.

- Specific process:
    - Automated segmentation: Use deep learning models (such as U-Net) or other image segmentation algorithms to perform preliminary segmentation of medical images.
    - Manual correction: Experts review and make necessary modifications to the automatic segmentation results to correct possible errors in the model.
    - Iterative optimization: Feed the corrected results back to the model for retraining or fine-tuning to gradually improve the segmentation performance of the model.
- Related Literature and tutorials:
    - New breakthrough in medical image segmentation: 6 articles give you insight into the most cutting-edge medical AI technology
    - Research progress on automatic segmentation methods for liver cancer lesions based on electronic computed tomography images
    - CN109690554B - Method and system for medical image segmentation based on artificial intelligence

**Key Features（Group Proposal and Plan, can be changed later)**

1. **Load and visualize multi-channel biological images.**
    a. Implementation steps:
        i. **Data loading: reading multi-channel bioimaging data using suitable libraries (e.g. tifffile) that support reading of multi-layer segmentation masks, ensuring that the segmentation layers of each biological structure can be stored and edited independently.**
        ii. **Data processing: Pre-processing of loaded image data, such as normalisation or denoising, based on datasets provided by project stakeholders.**
        iii. **Data visualisation: Use Napari's Viewer object to add multi-channel images to the view and set different display parameters for each channel.**
        iv. **Multi-Channel Support: Supports Napari's ability to process medical images assigned to multiple channels and allows users to adjust the display of different channels, as well as a channel-mask binding mechanism that ensures that mask data is**

**switched in sync with channel data to avoid annotation confusion.**

b. Required tools:
    i. Napari: Used to visualize multi-dimensional imaging data.
    ii. tifffile: Used to read image files in TIFF format.

2. **Deep learning-based cell segmentation (automatic segmentation).**
    a. Implementation steps:
        i. **Model loading: use sam_model_registry to register and load pre-trained SAM models.**
        ii. **Image Processing: Call SamPredictor.set_image method to convert the input image into embedded representation that can be processed by the model, and support scrolling and cyclic loading of different images, add multi-layer mask processing, allow different layers of masks to be stored via JSON and loaded on different channels.**
        iii. **Prompt Input: Provide prompt information such as points and boxes to guide the model to segment a specific region according to the needs, and support different layers of prompt input to improve the fineness of automatic segmentation.**
        iv. **Mask Generation: Call predictor.predict method to generate the segmentation mask of the target region.**
    b. Tools required:
        i. Deep learning framework: Such as TensorFlow or PyTorch.
        ii. Training dataset: High-quality annotated biological image data from our project stakeholders.
        iii. **OpenCV: for image processing and preprocessing.**

3. **Interactively edit cell masks (manually adjust AI results).**
    a. Implementation steps:
        i. **Mask loading: Load automatically split generated masks into Napari, enabling multi-channel fusion on Napari and allowing the user to adjust the visualisation of different layers.**
        ii. **Interactive editing: manually modify masks to correct errors, using Napari's drawing tools, with support for the Eraser tool, allowing manual correction of AI-generated masks.**
        iii. **Undo/Redo: Supports the user to undo and redo actions to improve the editing experience.**
        iv. **Multi-Level Editing: Allows separate operations on different segmentation layers without affecting other layer data.**
        v. **Save Changes: Save the edited mask for further analysis or model retraining.**
    b. Required tools:
        i. Napari: Provides interactive editing capabilities.

4. **Export annotated data for further analysis.**
    a. Implementation steps:

i. Data format selection: Determine the format of the exported data (such as CSV, JSON, or TIFF).
ii. Data export: **Use the appropriate libraries to export the annotation data to the desired format and ensure that the JSON store supports multiple mask layers and is bound to the channel information for subsequent data analysis.**
iii. Data validation: Ensure that the exported data is complete and in the expected format. **Additional support for RGB files and PNG image formats for mask visualization.**

b. Required tools:
i. **Pandas: Processing tabular data.**
ii. **tifffile: Processing the export of image data.**

5. **The tool will be developed as a Napari plugin to ensure its compatibility with existing bio-image analysis pipelines.**
   a. Implementation steps:
      i. Plugin template usage: Create new plugin projects using the plugin templates provided by Napari.
      ii. Functional implementation: Implement the required functionality in the plugin, such as data loading, processing and visualisation.
      iii. Plugin Testing: **Test the plugin for compatibility and stability in different environments, optimise remote data loading support and ensure that it can be tested on cloud servers and not just local devices.**
      iv. **User Feedback Iteration: Supplement the plugin testing phase with user feedback mechanisms to help improve the UI interaction experience.**
      v. **Plugin Publishing: Publish the plugin to the Napari plugin repository for use by project stakeholders.**
   b. Required tools:
      i. Napari plugin template: Accelerate plugin development.
      ii. Python development environment: Used to write and test plugin code.

## Development Scope

1. **Implement segmentation models (based on SAM).**
   a. Implementation steps:
      i. Model loading: Use sam_model_registry to register and load the pre-trained SAM model.
      ii. Image processing: Call the SamPredictor.set_image method to convert the input image into an embedding representation that the model can handle. **It is necessary to optimize SAM or find other models to adapt to 3D dimensional data.**
      iii. Prompt input: Provide prompt information such as points and boxes as needed to guide the model to segment specific areas.
      iv. Mask generation: Call the predictor.predict method to generate the segmentation mask of the target area.

b. Tools required:
    i. Python: Used to write and run code.
    ii. PyTorch: Used to load and operate deep learning models.
    iii. OpenCV: Used for image processing and preprocessing.

2. **Develop UI interfaces to support annotation optimization.**
    a. Implementation steps:
        **i. Create plugin template: Initialize UI using Napari plugin API.**
        **ii. Implement interactive features: Enhance Napari interactive experience, such as dynamic data navigation, multi-channel visualization adjustment. Design UI to support image loading, segmentation and interactive operations.**
        **iii. Test and debug: Test UI stability in different systems.**
    b. Required tools:
        i. Napari: A platform for image visualization and plugin development.
        ii. Python: Used to write plugin code.
        iii. Related plugins: Such as napari-plugin-manager, which is used to manage and install other plugins.

3. **Support dataset export (for subsequent analysis).**
    a. Implementation steps:
        i. Data format selection: Determine the format of the exported data, such as TIFF, PNG or custom format. **Additional support for TIFF, PNG or other common formats avoids relying solely on .npy.**
        ii. Export function implementation: Add data export function in the plug-in to allow users to save processed images or annotation results locally.
        iii. Metadata preservation: Ensure that relevant metadata information (such as annotation time, operator, etc.) is also saved during the export process to facilitate subsequent analysis.
        iv. Compatibility testing: Verify whether the exported data can be correctly read and processed by other commonly used analysis tools.
    b. Required tools:
        i. Napari: Used for image processing and export function implementation.
        ii. Image processing library: Such as PIL or OpenCV, used for image format conversion and processing.

4. Possible future extensions:
    a. Fine-tune SAM to specific data
    b. Support multi-category annotation
    c. Integrate other deep learning visual models

## Expected deliverables

1. Python code base (implements interactive annotation tools).
    a. The code should be modularized to facilitate subsequent maintenance and expansion.

2. Technical report (shows the development process and example results).
    a. Need to show the performance comparison results of the tools on different datasets.
3. README document (guides users on how to use the tool).
    a. Need to include examples of using the plugin and provide tutorials for multi-channel data processing.

**Required technologies**

1. Deep learning
    a. SAM performs automatic segmentation, optimization, or finds new models for 3D adaptation.
2. Python & libraries List (Based on Jay CHEN's demo)

```
# Basic scientific computing library
numpy==1.26.4
scipy==1.14.1
pandas==2.2.2
matplotlib==3.9.2
seaborn==0.13.2

# Machine learning & deep learning
scikit-learn==1.5.1
torch
torchvision
torchaudio
tensorflow-intel
tensorboard==2.17.1

# FastAPI / Flask API service
fastapi==0.115.9
Flask==3.1.0
uvicorn==0.34.0

# Dependent tools
requests==2.31.0
tqdm==4.66.5
protobuf==4.25.3
pydantic==2.10.6
typing_extensions==4.12.2
PyYAML==6.0.2

# Other important tools
joblib==1.4.2
six==1.16.0
tabulate==0.9.0
```

3. Data visualization
    a. Implementation of interactive annotation, scrolling and loop browsing.

### Fields involved

1. Artificial intelligence
2. Data science/analysis
3. Bioinformatics/biomedicine

## Conclusion

The core of this project is to use **deep learning** and **interactive visualization tools** in the field of **biomedical image analysis** to create an efficient, intuitive, and adjustable cell segmentation and annotation platform suitable for researchers or biomedical engineers.

## **Appendix** (Main key points and updates from 1st stakeholders meeting)

✅ Multi-channel image support (optimized multi-channel display of RGB and medical images) Literature updated 1 Literature updated 2
✅ 3D data processing issues (SAM cannot process 3D efficiently at present, needs improvement) Literature updated 1 Literature updated 2
✅ Improved data storage format (support PNG and other easy-to-read formats, avoid using only NPY)
✅ Scrolling & looping (optimize user interaction and improve data loading experience)
✅ Optimize interactive annotation tools (reduce manual drawing time, support more annotation methods, such as points, boxes, automatic segmentation, etc.) Literature updated
✅ Real-time model fine-tuning (currently the tool lacks this function, needs improvement to improve model flexibility)
✅ Richer export options (ensure that the exported data can be easily used by non-programmers)
✅ Support server connection (improve the feasibility of remote data loading)

### AWS & EC2 Literature

AWS EC2 Official Guide
docs.aws.amazon.com/AWSEC2/latest/UserGuide/
AWS Free Training and Certification
aws.amazon.com/training/
AWS Whitepapers
aws.amazon.com/whitepapers/
AWS Developer Guide
docs.aws.amazon.com/sdk-for-java/latest/developer-guide/
Get Started with Amazon EC2
docs.aws.amazon.com/zh_cn/AWSEC2/latest/UserGuide/EC2_GetStarted.html
AWS Case Study: Oxford University Using EC2 for Image Recognition
aws.amazon.com/cn/solutions/case-studies/oxford-case-study/
AWS Case Study: Codeway Using EC2 G5 Examples
aws.amazon.com/cn/solutions/case-studies/codeway-case-study/

AWS Case Study: CICC Online uses EC2 to improve system reliability
aws.amazon.com/cn/solutions/case-studies/cnfol/
AWS Case Study: Zixun Technology's cloud practice
www.youtube.com/watch?v=-7PtIUCEPcA