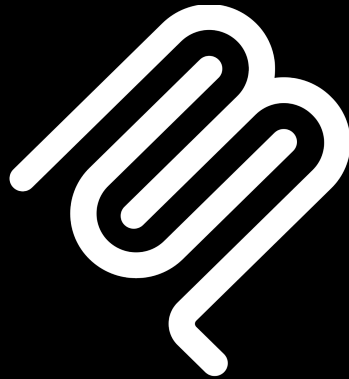




Fiji-MCP: AI-assisted Image Analysis in Fiji / ImageJ

Suraj Sahu¹ · ssahu2@ucmerced.edu

¹Department of Physics, University of California Merced, CA, USA

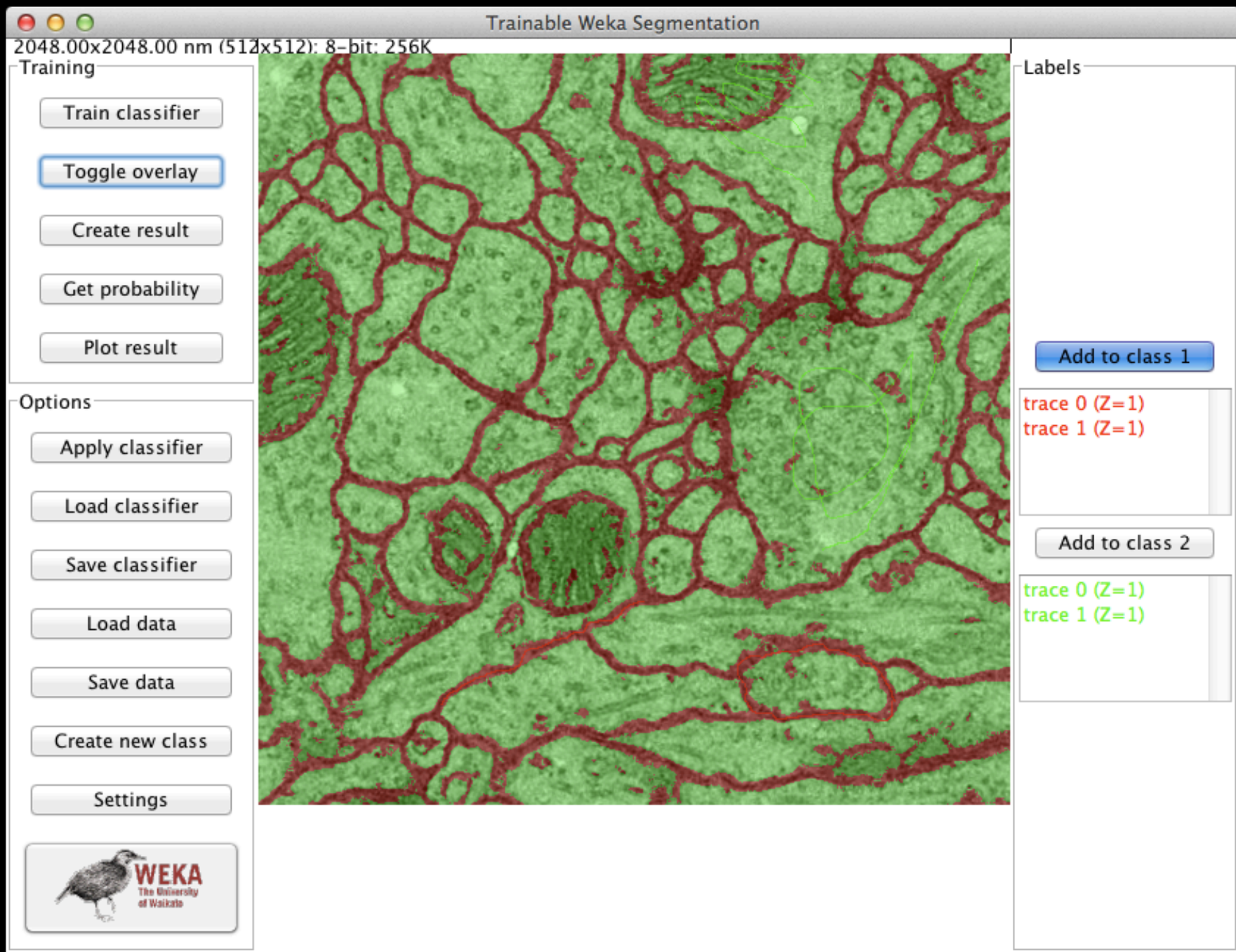
Abstract

Fiji-MCP connects an AI assistant to a local Fiji / ImageJ installation through the Model Context Protocol (MCP). Nine tools provide image input and output, command discovery, command and script execution, measurement tables, and visual inspection. FastMCP and PyImageJ expose installed Fiji capabilities through a shared interface. An assistant can inspect the current image, select a supported command, run an analysis, and return measurements alongside before-and-after images. ImageJ macros and Groovy scripts support workflows beyond the structured command interface. Researchers select appropriate methods and check the resulting images and measurements.

Fiji / ImageJ

Fiji is an ImageJ distribution for scientific image analysis [1]. Its installed commands and plugins provide the analysis operations used by this server. Fiji-MCP connects to these existing capabilities.

Workflows can combine image processing, thresholding, regions of interest, and measurements through supported commands or scripts.



Fiji's Trainable Weka Segmentation interface on a TEM image. ImageJ Wiki documentation example [5].

The available operations depend on the local installation. Plugins that require dialogs or other unscriptable interaction may not work in headless mode.

MCP (Model Context Protocol)

MCP provides a shared interface through which an AI application can discover and call tools [2].

The assistant translates a user's request into calls to the Fiji server, then uses returned image metadata, measurements, and screenshots to inform subsequent steps.

Communication uses JSON-RPC over standard input and output. FastMCP handles the tool interface; a serialized bridge uses PyImageJ [4] to access Fiji. Analysis operations run one at a time against the same Fiji session.

AI client → MCP → FastMCP → PyImageJ → Fiji

Fiji-MCP

The public interface contains nine tools [3]. Commands are discovered from Fiji's live SciJava and ImageJ1 registries, so the assistant can inspect accepted inputs and the invocation route before execution.

Image access

`get_state` inspects the session and open images; `open_image` loads a local image; `save_image` writes the active image.

Commands and scripts

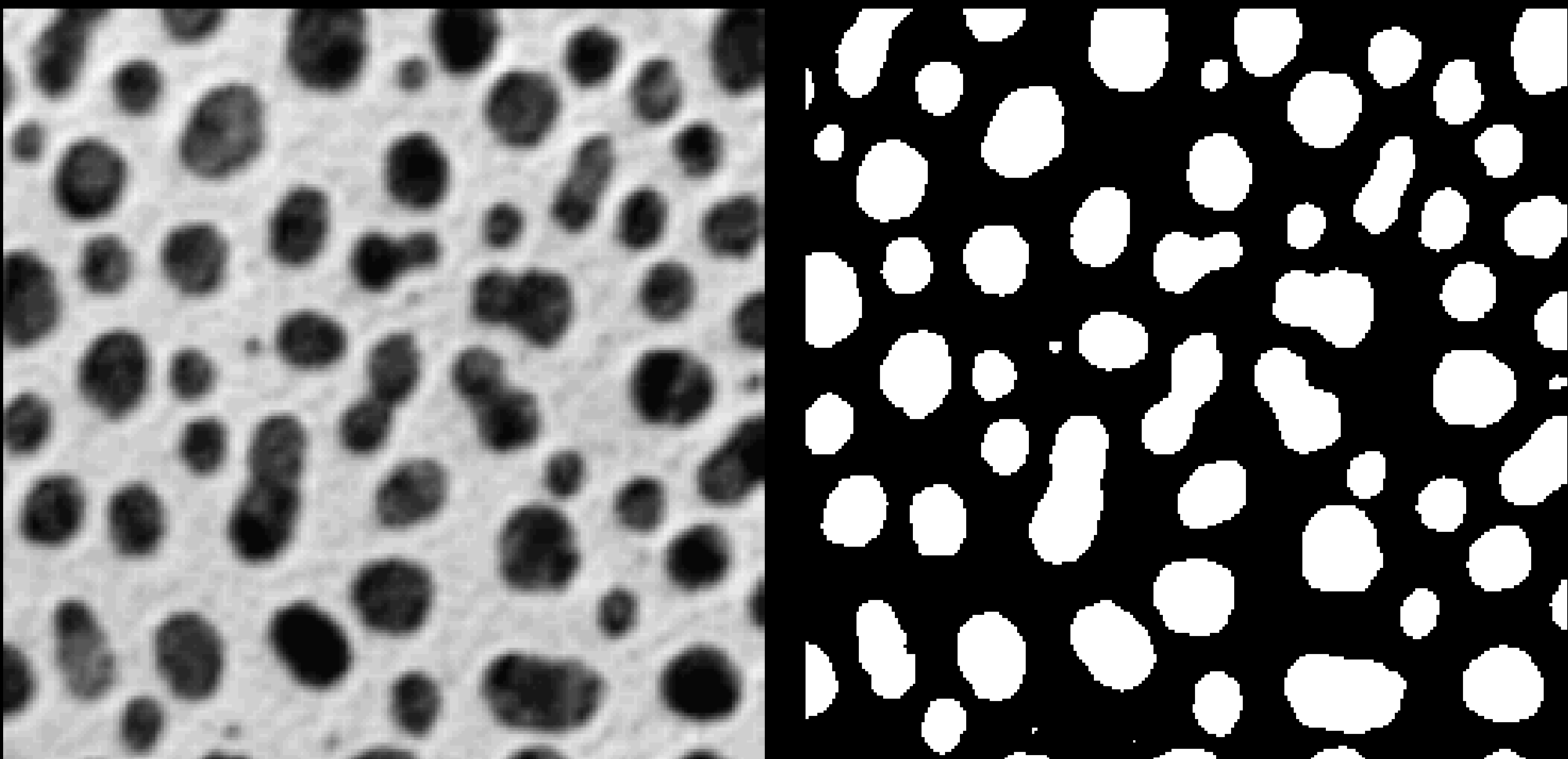
`search_commands` finds installed commands; `run_command` executes a supported command; `run_script` runs trusted ImageJ Macro or Groovy code.

Results and inspection

`get_results` reads the live Results table in pages; `screenshot` renders the current image plane or Results; `compare_screenshots` places saved images side by side and compares matching dimensions.

Example

User prompt: “Open blobs.gif, apply Otsu thresholding, measure particles of at least 50 pixels², exclude edge particles, and return the mask and Results table.”



Input - blobs.gif
ImageJ sample image [6] and the example thresholding output. The mask shows all thresholded objects; edge particles are excluded only from measurements.

Sample workflow parameters: Otsu threshold; minimum particle area 50 pixels²; particles touching the image edge excluded.

Check the mask against the input before interpreting measurements. These settings illustrate one workflow and must be chosen for the image under analysis.

Results and visual verification

Measurements and images provide complementary checks. `get_results` preserves column order and returns rows in bounded pages.

Particle	Area (px ²)	X (px)	Y (px)	Perim. (px)
1	285.00	197.71	16.26	62.18
2	81.00	137.52	22.28	32.38
3	278.00	219.29	28.29	63.94
4	231.00	45.08	26.59	55.70
5	501.00	174.44	33.69	82.67

First five of 43 measured particles. Otsu threshold; area ≥ 50 px²; edge particles excluded.

An image screenshot includes the current channel, slice, and frame, with the displayed lookup table, visible overlay, and region of interest.

For equal-sized images, screenshot comparison also reports mean absolute error, root mean square error, and the fraction of changed pixels. These describe differences between rendered images; they do not measure segmentation accuracy.

After an operation with an uncertain outcome, inspect the current state or image before repeating it.

Advantages

- Reuses the installed Fiji command and plugin ecosystem.
- Combines image handling, measurements, and inspection in one session.
- Supports both structured command inputs and existing macros or scripts.
- Returns images and measurement tables to the assistant for review.
- Keeps command discovery tied to the local Fiji installation.

Conclusions

Fiji-MCP makes established Fiji operations available within an assistant-led workflow. Its small tool interface links command discovery, execution, measurement, and visual review.

The quality of an analysis depends on its method, parameters, and validation. Screenshots help confirm what changed; scientific conclusions require checks appropriate to the images and research question.



Official ImageJ logo.

Features

9 tools Public MCP interface	IJM / Groovy Trusted script execution
Python 3.10+ Python runtime	BSD-3-Clause Source license

Headless mode is the default for an IDE-hosted MCP server. A local Fiji installation supplies the commands and plugins.

Scripts execute trusted arbitrary local code. The server is intended for a trusted local client and is not a sandbox.

Local setup, from the repository:

```
python -m pip install .
export FIJI_PATH="/path/to/Fiji"
export FIJI_MODE=headless
fiji-mcp-server
```

`FIJI_PATH` points to the Fiji root containing `jars/` and `plugins/`. Configure the MCP client to launch the server with these environment variables.

REFERENCES

- [1] Schindelin, J. et al. Fiji: an open-source platform for biological-image analysis. *Nature Methods* 9, 676–682 (2012). doi:10.1038/nmeth.2019
- [2] Model Context Protocol. modelcontextprotocol.io
- [3] Fiji-MCP source and tool reference. github.com/surajinacademia/Fiji_imageJ_mcp
- [4] Rueden, C. T. et al. PyImageJ: A library for integrating ImageJ and Python. *Nature Methods* 19, 1326–1327 (2022). doi:10.1038/s41592-022-01655-4

IMAGE SOURCES

- [5] Trainable Weka Segmentation documentation, ImageJ Wiki. imagej.net/plugins/tws. CC BY 3.0.
- [6] ImageJ blobs sample. imagej.net/ij/images/blobs.gif.
- Fiji and ImageJ logos: ImageJ project. MCP logo: Model Context Protocol.

PROJECT AND CONTACT

Source, setup, and tool documentation
github.com/surajinacademia/Fiji_imageJ_mcp
Suraj Sahu
ssahu2@ucmerced.edu