UC1 RETROCOVER - user guide

Author: Šimon Saloň

This guide describes the RETROCOVER application in this package (version 1.0).

RETROCOVER supports the reconstruction of historical land cover from georeferenced satellite imagery. Users prepare rasters at suitable resolutions, segment them into candidate polygons, then assign classes and refine boundaries through expert review. The application records review status, processing parameters and result lineage, and produces land-cover maps, class-area statistics and derived vector exports. Saved versions support review and comparison, while reliable assessment of landscape change requires compatible maps from different acquisition dates.

Contents

1. What the application does 2

2. Install and start 2

3. Create or open a project 2

4. Follow the main workflow 3

5. Choose a notebook 4

6. Files, interpretation, and troubleshooting 4

Review-state compatibility and verification 5

Release 1.0 setup and catalog paths 5

1. What the application does

RETROCOVER supports object based land cover mapping from georeferenced raster imagery. A typical session imports a GeoTIFF, optionally creates coarser working rasters, segments the image into candidate polygons, reviews and classifies those polygons, and produces statistics or derived vector exports. Segmentation does **not** assign a verified land cover class. A person must review the polygons and their categories.

The package includes a sample classification catalog (`catalog.json`) and four English notebooks. Large source imagery is not included. Add your own GeoTIFF to a project for real analysis; the standalone EOSC notebook can run a synthetic technical demonstration without one.

2. Install and start

Run the commands from the folder containing `app.py` and `pyproject.toml`.

Windows / local Conda

```
conda env create -f environment.yml
conda activate flandscape
python -m pip install --no-build-isolation -e .
panel serve app.py --show
```

The package also contains `launch_uc1_retrocover_app.cmd` for a configured standalone Windows installation. That script expects a Python environment at `..\venv\Scripts\python.exe` relative to the package folder and serves the app on port 5029. If that environment is absent, use the Conda commands above.

EOSC / JupyterLab

```
conda env create -f environment-eosc.yml
conda activate flandscape-eosc
python -m pip install --no-build-isolation -e .
python eosc_smoke_test.py
```

Open a notebook with the `flandscape-eosc` kernel, or start the web interface from the extracted package root:

```
panel serve app.py --address 0.0.0.0 --port 5006 --allow-websocket-origin="*"
```

Open the interface through the address or proxy supplied by the EOSC service. Port exposure depends on that service. The smoke test writes `outputs/eosc_smoke_report.json`; a successful run reports `Result: OK`.

3. Create or open a project

The first screen is **Project management**. Choose a recent or discovered project, enter its folder path, or locate its `project.json` with the file browser, then select **Open selected project**. The application folder itself is not a user project.

To make a project, enter both a Slovak and an English project name, choose the parent folder, leave **Create a basic land cover catalog** selected if you want the starter categories, and select **Create and open project**. The target folder must be empty. The application creates `project.json`, a project catalog at `data/catalog/catalog.json`, and separate input, output, export, and log folders. You can later edit the two project names without changing its identity.

The example configuration bundled with this package is `config/project.json`; it points to the package root `catalog.json`. Newly created projects use their own catalog. Keep each project's `project.json` and data folders together when moving it to another computer, because the configured paths are relative to the project root.

4. Follow the main workflow

The left rail contains **Data**, **Preparation**, **Segmentation**, **Review**, **Editing**, and **Display**. The right rail contains the project overview, legend, polygon table, statistics, reclassification, methodology, catalog, and diagnostics. Select a rail item to open its panel; selecting the same item again closes it.

Step 1 - Add and display a raster

In **Data**, add a `.tif` or `.tiff` by entering a full path to a file accessible on the application computer, or upload a GeoTIFF through the browser. The local path option copies the source into the project and avoids browser upload limits. Refresh the raster list if needed. Add the raster to the working selection, choose the **Input raster for new segmentation**, and select **Display input raster**. The working selection can show two images with an opacity control for the second image.

Check that the raster appears in the map and that its resolution and location are as expected before processing. The original input raster is kept separate from derived rasters.

Step 2 - Prepare a working resolution (optional)

In **Preparation**, select the source image and enter target resolutions in metres per pixel. Select **Create raster pyramid**. This produces lower resolution derived rasters while preserving the original. Return to **Data** and choose the derived raster as the segmentation input when you want to work at that scale.

Step 3 - Segment the image

In **Segmentation** → **New segmentation**, choose a method and its parameters, then select **Run segmentation**. The interface exposes marker watershed, Felzenszwalb, and SLIC parameters; applicable fields depend on the selected method. **Minimum area** is in square metres. For a first run, use **Whole current raster**. The selected parent polygon scope is for a confirmed polygon when refining an existing result.

Watch the status and run monitor. When the run completes, use **Segmentation** → **Select segmentation** to refresh the list, select the saved result, and open it. Each run is stored separately with its source and parameters. The result manager can assign an alias or archive a run; permanent deletion has its own confirmation control.

Step 4 - Review and classify polygons

In **Review**, select polygons in the map or polygon table. Ctrl + left click adds polygons to the selection; right click opens the review menu. For spatial selection, choose **Select by line** or **Select by polygon**, draw on the map, then use **Finish / cancel** to confirm or discard the geometry. **Add to existing selection** controls whether the new hits replace or extend the selection. **Skip verified polygons with a category** excludes already completed objects from the spatial selection.

Assign a land cover category and a review state to the selected polygons. The three stored review states are **Unverified**, **Suitable**, and **Requires re-segmentation**. A methodologically verified polygon needs both **Suitable** and an active leaf category. A review state alone does not make an unclassified polygon part of verified area statistics. When changing several already processed polygons, read the overwrite confirmation carefully and confirm the exact selection.

Use **Editing** to merge, split, or clip selected polygons where needed. Select **Save review version** to preserve a named reviewed result. **Save project** in the header saves the current project state. Reopen the saved vector layer when you need to confirm persisted attributes.

Step 5 - Inspect and export results

Use **Display** to switch map presentation, including land cover, review, or analytic views. The active display mode controls polygon colours; a selection highlight is separate from category styling. The right side **Legend** reflects the display mode. **Polygons** provides a filterable table and a way to locate a selected polygon on the map.

In **Statistics**, choose the category hierarchy level and refresh the summaries, charts, and table. The project overview shows the lineage of input and derived rasters, segmentation runs, working drafts, reviewed versions, and derived outputs. It can compare two vector layers by category, area, proportion, or polygon count; the spatial transition view uses actual overlap between the two layers.

In **Reclassification and export**, choose a scheme, prepare the mapping table, review the target groups and export options, then create a **derived** vector layer. Options include restricting input to polygons marked **Suitable**, dissolving equal target classes, and exporting GeoJSON or Shapefile. Reclassification leaves the detailed source layer intact.

5. Choose a notebook

Open notebooks from the package root so their relative paths resolve. Run cells in order. Saved outputs were cleared in the release copies, so results appear after execution.

Task	English notebook
Edit the classification catalog	`01_catalog_editor_en.ipynb`
Prepare rasters	`02_raster_preparation_en.ipynb`
Explore the interface and analysis functions	`03_interface_demo_en.ipynb`
Run the standalone EOSC workflow	`04_eosc_workflow_en.ipynb`

The catalog editor writes the legacy source-name field (`nazov`); English names are displayed from `nazov_en` where present. Older catalogs may retain Slovak values in `nazov`. The standalone workflow can create a small synthetic raster when `data/raw/` has no GeoTIFF. Treat synthetic outputs as a software demonstration, not a land cover interpretation. For real imagery, set the input raster path in the notebook's settings cell.

6. Files, interpretation, and troubleshooting

- **Inputs:** project imagery belongs in `data/raw/`; the project catalog is selected by `project.json`.
- **Working data:** derived rasters go under `data/processed/`; vector layers and review results are stored under the paths in the project configuration.
- **Results:** inspect `outputs/` and `exports/` in the selected project. GeoPackage layers and companion metadata record processing provenance; retain these when sharing results.
- **No raster in the list:** check that the import or upload completed, then refresh the list. Confirm the file is a GeoTIFF and that the chosen project is correct.
- **Segmentation cannot start:** select an input raster in **Data** and check the segmentation scope and parameters. For a new image, start with the whole raster.

- **Statistics look incomplete:** distinguish all segments from reviewed polygons, and check that reviewed polygons have both a category and **Suitable** state.
- **Interface or processing failure:** open **Diagnostics** for timings and logs. The standalone launcher writes `runtime-startup-<run-id>.log` beside the script; application logs are stored in the project's configured log location.

For method definitions and their limitations, open **Methodology** in the right rail. For EOSC setup details, see `README_EOSC_EN.md`; for the release's recorded test scope, see `QA_REPORT.md`. The recorded synthetic and automated checks do not establish classification accuracy on real imagery. Review the resulting polygons and classifications before using them for scientific conclusions.

Review-state compatibility and verification

Review states use stable stored identifiers: `neovereny`, `vyhovuje`, and `nova_segmentacia`. English presentation labels are **Unverified**, **Suitable**, and **Requires re-segmentation**. Stored IDs must remain consistent across vectorization, review actions, geometric edits, validation, and saved layers.

The November 2026 runtime check restored these identifiers in the affected writers after a presentation translation had also changed technical values. The smoke test and standalone EOSC workflow then completed successfully on synthetic data. Run the focused regression checks in the application environment:

```
python -m unittest discover -s tests -v
```

Release 1.0 setup and catalog paths

The catalog editor reads the catalog path from the project configuration in both catalog-loading cells. The bundled project uses `catalog.json`; new portable projects normally use `data/catalog/catalog.json`. Keep the configured path when opening, editing, or saving a project catalog.

Both environment definitions include `ipyvuetify`, which supplies the editor's color picker. After updating an existing EOSC environment, run `conda env update -f environment-eosc.yml`, activate `flandscape-eosc`, and restart the notebook kernel.

The application VERSION file, Python package metadata, Windows launcher, notebook release metadata, and current documentation identify release `1.0`.