Exercise 5.1 Report

Exercise 5.1 Implementation Summary

Implemented Tasks (5.1–5.8)

All required parts are implemented in `code/selective_search.py`.

- **Task 5.1** Initial segmentation: `generate_segments()` uses Felzenszwalb and stores integer labels.
- **Task 5.2** Region extraction and similarity terms:
  - Color histogram and color similarity: `calc_colour_hist()`, `sim_colour()`.
  - Texture feature and texture similarity: `calc_texture_gradient()`, `calc_texture_hist()`, `sim_texture()`.
  - Size similarity: `sim_size()`.
  - Fill/shape compatibility: `sim_fill()`.
- **Task 5.3** Neighbor extraction: `extract_neighbours()`.
- **Task 5.4** Region merging: `merge_regions()`.
- **Task 5.5 & 5.6** Marking and removing old similarities in the main merge loop of `selective_search()`.
- **Task 5.7** Recomputing similarities for the newly merged region in the same loop.
- **Task 5.8** Final proposal generation (`rect`, `size`, `labels`) at the end of `selective_search()`.

Parameter Settings

Current parameter settings in `code/main.py`:

- `FAST_MODE = False`: `scale=200`, `min_size=40`, `max_merges=4000`, `min_rect_size=800`, `max_aspect=1.5`.
- `FAST_MODE = True`: `scale=150`, `min_size=80`, `max_merges=3500`, `min_rect_size=500`, `max_aspect=2.5`.
- `sigma` uses the default value from `selective_search(..., sigma=0.8)`.

Result Locations for the Three Domains

The script `code/main.py` processes all three folders and saves outputs to:

- `results/chrisarch/`
- `results/arthist/`
- `results/classarch/`

Q5.1 Answers

Q5.1.1

Felzenszwalb provides a single flat segmentation for one parameter setting. Object detection needs multi-scale object candidates. Selective Search starts from small segments and hierarchically merges them, generating many candidate boxes at different scales, which improves proposal recall.

Q5.1.2

Proposal filtering in `main.py` applies three criteria:

- remove duplicate rectangles,
- remove very small regions (`size < min_rect_size`),
- remove highly distorted boxes using aspect ratio threshold (`max_aspect`).

Effect: fewer noisy boxes and cleaner visualization, but possible loss of true positives (especially small or elongated objects). I agree with these criteria as a baseline. Additional useful criteria: score ranking, border-touch filtering, or NMS for heavy overlaps.

Q5.1.3

Selective Search merges arbitrary regions, but each region keeps a bounding box. The final box proposals are obtained by converting each region to its axis-aligned bounding rectangle (x, y, w, h), stored as `rect`.

Q5.1.4

Increasing the Felzenszwalb scale (or equivalently larger initial segment preference) generally creates larger initial regions and fewer proposals. This reduces runtime but may miss small objects. Increasing the number of proposals usually improves overlap/recall metrics first, then saturates, while runtime and false positives increase.