Supplementary Material — Topology-guided patch extraction (TDA)

1. Overview

This document describes the topology-driven steps used to select tumor-centered patches from raw MRI intensities. The pipeline:

1. constructs a **cubical complex** from volumetric intensities,
2. induces an **intensity filtration** (sublevel or superlevel),
3. computes **persistent homology** to obtain a persistence diagram (PD) with spatial representatives,
4. converts the PD into a **persistence image (PI)** or voxelwise lifetime map,
5. thresholds & refines the PI to obtain candidate regions,
6. applies **connected component analysis (CCA)** and ranks components, and
7. uses component centroids to define patch coordinates (with fallbacks).

This file documents the mathematical choices, the main parameters, recommended defaults, and pseudocode plus implementation notes.

2. Key implementation choices & recommended defaults

1. **Filtration type**
   - Use **sublevel** filtration when lesions appear darker than background; use **superlevel** if lesions appear brighter (FLAIR, T1ce usually bright → superlevel recommended).
   - Default for this work: **superlevel filtration** on FLAIR and T1ce modalities (so large intensities appear earlier).
2. **Cubical complex construction**
   - Use a cubical complex API (e.g., GUDHI's `CubicalComplex`, Cubical Ripser or equivalent) built directly from the voxel intensity array.
   - For memory saving, you may downsample or compute PH on a smoothed/denoised version (e.g., Gaussian sigma 1–2).
3. **Smoothing / prefilter**
   - Gaussian smoothing (sigma = 1–3) often helps remove spurious small features.
   - We used `sigma = 1.0` as default for 3D arrays; increase if noisy.
4. **Persistence computation**
   - Compute H_0, H_1, H_2 as available; usually H_0, H_2 are most informative in Brain MRI (connected components and cavities).
   - Limit to features with lifetime above a small epsilon to remove numerical artifacts (e.g., lifetime > 1e-4 in normalized intensity units).
5. **Max lifetime clipping (for visualization/PI)**

    - Clip lifetimes at a quantile (e.g., 0.8) to avoid domination by a few extreme features. Default: `max_life = quantile(lifetimes, 0.8)`.
6. **Persistence-to-spatial mapping**
    - If PH output includes spatial representatives, build a voxelwise **lifetime map** $L(x, y, z)$ where each representative adds its (clipped) lifetime to the voxel(s) indicating the feature location. If PH representatives are single coordinates, set $\mathrm{L(coord)} = \max(\mathrm{L(coord)}, \mathrm{lifetime})$
    - Alternative (no representatives available): compute a persistence image (PI) from PD and then **project PI back** to voxels by placing Gaussian kernels centered on candidate voxel positions found by intensity-based saliency (hybrid approach). This is less precise but usable.
7. **Saliency thresholding**
    - Two recommended strategies:
        - **Percentile thresholding:** select voxels with L above the 90th (or 95th) percentile.
        - **Otsu / Yen threshold** on the L distribution.
    - Default: `saliency_threshold = percentile(L, 95)`.
8. **CCA and component ranking parameters**
    - Minimum component volume (`min_voxels`) to ignore tiny noise components: default `min_voxels = 100` (adjust for resolution).
    - Ranking score: `score = lifetime_mean * volume` or `score = max_lifetime * volume^(0.5)`. Default uses `score = mean_lifetime * volume`.
9. **Centroid computation**
    - Use persistence-weighted centroid: centroid $c = \frac{\sum_i w_i x_i}{\sum_i w_i}$ where $w_i = L(x_i)$ within component.
    - This reduces sensitivity to irregular shapes.
10. **Fallbacks**

- If no components survive thresholding, fall back to center-crop or intensity-based maximum.

3. Data formats and convention used in the provided `vec` function

The `vec` function in your code expects an array `ph` whose rows have the format:

```
[dim, birth, death, x, y, z]
```

- `dim`: homology dimension (0,1,2)
- `birth`, `death`: filtration values (float)
- `x,y,z`: integer voxel coordinates indicating a spatial representative

Documented expectation: PH computation must provide representative coordinates for each persistent feature (e.g., location of birth or a representative simplex). If your PH library outputs

only `(dim, birth, death)` without spatial coordinates, you must obtain representatives explicitly (if the library supports them).

4. Pseudocode

```
INPUTS:
  Volumes = {V_m} for m in modalities (e.g., FLAIR, T1ce)
  patch_size = (Px,Py,Pz)  # e.g., (128,128,128)
  k = number of patches per subject (default 1)
  prefilter_sigma = 1.0
  max_life_quantile = 0.8
  saliency_percentile = 95
  min_component_voxels = 100

FOR each subject:
  # 1. Preprocess volumes
  for each modality m:
    Vm = gaussian_smooth(Volumes[m], sigma=prefilter_sigma)
    if modality indicates bright lesion: use superlevel filtration else
sublevel

    # 2. Build cubical complex (from Vm)
    K = CubicalComplex(top_dimensional_cells=Vm)

    # 3. Compute persistent homology with spatial representatives
    PD_with_repr = compute_persistent_homology(K, dims=[0,1,2],
return_representatives=True)
      # PD_with_repr is list of tuples (dim,birth,death,x,y,z)

    # 4. Convert PD to a modality-level voxel lifetime map Lm(x,y,z)
    max_life = quantile( PD_lifetimes, max_life_quantile )
    initialize Lm = zeros_like(Vm)
    for each feature f in PD_with_repr:
      dim, b, d, x, y, z = f
      life = min(d - b, max_life)
      Lm[x,y,z] = max(Lm[x,y,z], life)     # take max to represent strongest
feature

  # 5. Fuse modality-level lifetimes into S(x,y,z)
  S = fuse( {Lm} )   # e.g., S = sum(Lm over modalities) or pointwise max

  # 6. Threshold S to binary saliency_mask
  T = percentile(S, saliency_percentile)
  saliency_mask = S > T

  # 7. Connected-component analysis (CCA) on saliency_mask
  components = connected_components(saliency_mask)
  filtered_components = [c for c in components if volume(c) >=
min_component_voxels]

  if filtered_components empty:
    centers = fallback_centers(Volumes)  # center-crop or global max-
intensity centroid
  else:
    # 8. Rank components
```

```
    scored = []
    for comp in filtered_components:
      comp_lifetimes = S[comp_voxels]
      comp_volume = len(comp_voxels)
      score = mean(comp_lifetimes) * comp_volume
      centroid = weighted_centroid(comp_voxels, weights=comp_lifetimes)
      scored.append( (score, centroid, comp) )

    sort scored descending by score
    centers = [centroid for top k scored entries]

  # 9. Extract patches at centers across all modalities
  patches = [ crop_volume_at_center(Volumes, c, patch_size) for c in centers
]

  # 10. For training: crop corresponding segmentation masks at same centers -
> label patches
  # For inference: proceed with patches only (no masks)

  # 11. Save patches, centers, and metadata (scores, component volumes,
thresholds)
```

5. Recommendations for libraries & representative extraction

- **Cubical Ripser :** supports `CubicalComplex` and `persistence()` and also give spatial representatives.
- **GUDHI**: supports `CubicalComplex` and `persistence()`; recent versions allow retrieving representatives (check library docs).
- **Dionysus / Ripser / Dipha**: popular PH libraries; some require additional work to extract spatial representatives.
- If representatives are not available in your chosen library, options are:
    - compute PH on *localized regions* (sliding windows) and assign lifetime to center voxel,
    - compute PI from PD and combine with intensity-based candidate voxels to project PI back,
    - use morphological maxima on smoothed intensity as proxy seeds and weight by local persistence (hybrid method).

6. Sensitivity & robustness guidelines

- **Centroid sensitivity:** use persistence-weighted centroids to reduce sensitivity to irregular shapes. Test with synthetic shifts (±5 voxels) to estimate effect on downstream Dice.
- **Threshold selection:** report results for a small grid of percentile thresholds (90, 95, 99) in ablation to show robustness.
- **Component volume min:** choose `min_voxels` relative to voxel spacing (e.g., at 1 mm^3, 100 voxels ~ 0.1 cc).
- **Multi-focal cases:** allow `k>1` and extract top-k centroids for multi-focal coverage.