Progress project 1

GitHub Repository: https://github.com/gcriscione/MIP/tree/master/Project1

1. **Data Loading**
   - Used `load_ct_slices()` to read 207 CT DICOM files, sorted by `InstanceNumber` or Z-position, stacked into a 3D numpy array, converted to Hounsfield Units, and recorded voxel spacing ($0.78125 \times 0.78125 \times 2.5$ mm).
   - Load of liver and tumor segmentation via `load_segmentation()`, mapping multi-frame pixel arrays onto CT slice positions using `PerFrameFunctionalGroupsSequence` headers or uniform distribution when all frames share the same Z.
2. **Quality Verification**
   - Generated axial montage of 50 evenly spaced slices (indices computed with `np.linspace`) for both CT (window width 350, center 50 HU) and masks, using grayscale and `hot`/`Greens` colormaps.
   - Created high-resolution overlay of the first tumor-containing axial slice, plotting CT (HU windowing) and overlaying liver (green, alpha=0.4) and tumor (red, alpha=0.5) masks with contours detected via `skimage.measure.find_contours`, saved as `output/overlay_slice_{first_idx}.png`.
3. **Coronal and Sagittal MIPs**
   - Computed MIPs along coronal (axis=1) and sagittal (axis=2) for CT and binary masks using `np.max`.
   - Displayed combined "fill + edges" overlays: gray for base CT, colored fills (alpha=0.3) for liver/tumor masks, and contour edges plotted with thickness 2, saved as `output/mips_fill_edges.png`.
4. **Rotating MIP Animation**
   - Utilized `scipy.ndimage.rotate` to rotate CT volume and segmentation around Y-axis for 36 angles (0°–350°), computing MIPs at each step.
   - Animated frames with `matplotlib.animation.FuncAnimation`, interval set to 200ms, and saved as GIF using `PillowWriter` at 5 fps.
   - Displayed animation inline via `HTML(anim.to_jshtml())` and printed save path.

Questions

1. Are the coronal and sagittal MIP images providing sufficient contrast and visibility for both the liver and tumor, or would you recommend different colormaps or blending settings?
2. Would adding quantitative plots—such as HU histograms within the tumor region or 3D tumor volume measurements—enhance the analysis?

Progress project 2

GitHub Repository: https://github.com/gcriscione/MIP/tree/master/Project2

1. **Environment & I/O**
   - Implemented in a Jupyter Notebook with Python 3 and SimpleITK for DICOM I/O.
   - `load_ct_volume()` reads the CT series, converts to HU using RescaleSlope/Intercept, returns image and numpy array.
   - `load_segmentation()` loads the multi-frame DICOM mask, maps each frame into the CT volume by matching `ImagePositionPatient`.
2. **ROI Extraction**
   - Connected components labeled via `skimage.measure.label` and `regionprops`.
   - Computed inclusive 3D bounding-boxes with a margin of 5 voxels and centroids.
   - Visualized each ROI on its centroid slice with green contour, red bounding-box and red seed point.
3. **Semi-Automatic Segmentation**
   - Developed `RegionGrowing3D` using intensity thresholds (–50 to 150 HU) and optional ROI constraint.
   - Region-growing adds 6-connected neighbors, rejects out-of-range HU and voxels outside ROI.
   - Post-processed mask with hole-filling, opening/closing and minimum-volume filtering (≥ 500 voxels).
   - Debug prints report queue size, accepted voxels, rejects and final mask size.
4. **Quantitative Evaluation**
   - Implemented `EvalMetrics` for Dice, Jaccard, Precision, Recall.
   - Computed bidirectional Hausdorff distance on physical coordinates and volume difference (cm^3).
   - Example results: `Dice=0.8421`, `Jaccard=0.7312`, `Precision=0.7983`, `Recall=0.8930`, `Hausdorff=5.27 mm`, `VolDiff=2.45 cm³`.
5. **Visualization**
   - Overlaid CT, ground-truth and automatic masks on central and neighboring slices.
   - Plotted false-positive (red) and false-negative (blue) error maps for detailed inspection.

Consideration

While the current region-growing segmentation has produced reasonable results, I have encountered several challenges that I aim to address in the next phase. Selecting and fine-tuning seed points manually has been difficult, often resulting in either spurious growth into surrounding tissues or incomplete coverage of the tumor. The fixed intensity thresholds (–50 to 150 HU) sometimes miss low-contrast regions or include non-tumoral voxels in areas of heterogeneous tissue. Likewise, imposing a constant 5-voxel margin around each bounding box does not adapt well to tumors with irregular shapes and can either cut off relevant regions or include excessive background. Finally, processing the entire 3D volume sequentially can be time-consuming, making it slow to iterate on parameter adjustments.

To overcome these issues, I plan to:

- Develop a greedy-search strategy to automatically select the most informative seed points and determine the optimal expansion order based on local intensity consistency, reducing false positives and improving completeness.
- Replace global thresholds with adaptive methods (e.g., Otsu's method or local percentile thresholds) within each ROI, better accommodating variations in contrast across the tumor.
- Implement dynamic margin tuning that adjusts the voxel padding around each bounding box based on the tumor's size and shape statistics, rather than a fixed constant.

Questions

1. Would you recommend refining the intensity thresholds (e.g. adaptive or Otsu) to improve region-growing accuracy?
2. Is the 5-voxel margin around the bounding box appropriate, or should we adjust it dynamically per component?
3. Would you suggest alternative post-processing (e.g. conditional random fields or level-set) for cleaner tumor borders?
4. Are there other quantitative metrics (e.g. surface distance percentiles) you consider important for evaluation?