Advanced Techniques for 2D and 3D MRI Image Segmentation

2D segmentation

Aim: The objective of this study is to apply and evaluate various image segmentation techniques to a set of MRI slices. The images consist of 10 consecutive 'axial' cross-sections of a human brain, each with dimensions of 362x434 pixels, representing different tissue layers. The goal is to accurately segment out five specific tissue types: air, skin/scalp, skull, cerebrospinal fluid (CSF), gray matter, and white matter.

Methodology:

Data Preparation: MRI data were loaded from a MATLAB file **'Brain.mat'** using Python's SciPy library. For 2D segmentation, a combination of algorithms was explored including Thresholding, K-means clustering, watershed 2D, but finally the following 3 algorithm were selected

Otsu Thresholding: This method computes a threshold separating the tissue types based on intensity values, minimizing within-class variance. Otsu's method is effective for bimodal distributions but may be challenged by overlapping tissue intensities. $\boldsymbol{Threshold = argmax_t\ [\sigma_B^2(t)]}$ Where $\boldsymbol{\sigma_B^2(t)}$ is the between-class variance at threshold $\boldsymbol{t}$. Otsu's method was selected for its simplicity and effectiveness in scenarios with clear bimodality, a common feature in MRI datasets.

Active Contours (Snake): This technique uses energy-minimizing splines influenced by image forces and constraints to delineate object boundaries. The snake is modeled as: $\boldsymbol{v(s)=\alpha s''(s)+\beta s'''(s)-\nabla E_{image}(s)}$ Here, $\boldsymbol{v(s)}$ is the snake curve, $\boldsymbol{\alpha, \beta}$ control the spline's elasticity and rigidity, and $\boldsymbol{E_{image}}$ is the image-derived energy driving the snake towards edges and other notable features. This method was employed to capture the complex contours of anatomical structures that are not well-separated by intensity alone, as is often the case with MRI images.

Felzenszwalb Segmentation: This technique, not covered in our course, performs a graph-based segmentation that creates a pixel adjacency graph and partitions the image into components based on pre-defined criteria. It controls the scale of observation, allowing the detection of both small and large structures. The Felzenszwalb method stands out due to its flexibility and efficiency. It operates by optimizing a global criterion that measures the evidence for a boundary between two regions: $\boldsymbol{R=min_{C\in S}\,(min_{(i,j)\in R, i\neq j}\,[Int\,(C_i, C_j) - min\,(Int\,(C_i), Int(C_j)) + \tau(C)])}$ where S is the segmentation, $\boldsymbol{C_i}$ and $\boldsymbol{C_j}$ are components, Int(C) is the internal difference of component C, and τ(C) represents a threshold function. This balance allows for fine-tuned segmentation suitable for the complex structures within MRI images.

Selection of these algorithms was guided by their complementary strengths: Otsu for its simplicity and efficiency, active contours for their boundary adherence, and Felzenszwalb for its scalability and adaptability. This diversified approach aimed to address the inherent challenges in MRI segmentation, from varying tissue contrasts to irregular shapes and sizes. By applying these methods, the study intends to explore the most effective strategies for accurate delineation of anatomical structures within MRI images, thus catering to the wide spectrum of tissue characteristics observed. Each method was applied consistently across all slices. These algorithms were selected based on their suitability for the MRI data characteristics and their capacity to differentiate between subtle variations in tissue densities.

Comparison of the Three Algorithms: Each algorithm presents unique advantages for MRI segmentation. Otsu's method is highly efficient for data with clear global thresholds but may falter with overlapping tissue densities. In contrast, active contours offer precise localization of boundaries but require meticulous initial placement and parameter tuning. Meanwhile, Felzenszwalb's approach provides a versatile segmentation framework that balances detail preservation with computational efficiency, making it particularly suitable for heterogeneous MRI scenes.

Implementation and Output Analysis

The application of various segmentation techniques unveiled distinct outputs that shed light on the intricacies of MRI brain scans. Normalization and denoising were critical preliminary steps, ensuring image quality and clarity were optimized for segmentation. Gaussian smoothing, a denoising technique, was applied to reduce noise while preserving edges, essential for the accurate differentiation of brain structures. **Otsu's method** broadly delineated major brain areas but occasionally merged or oversimplified distinct regions due to its global thresholding nature. **Active Contours** provided more nuanced delineations, accurately tracing

complex tissue boundaries but exhibiting sensitivity to initial contour placement and parameter settings. **Felzenszwalb's** approach presented a finely segmented tapestry, demonstrating an impressive ability to differentiate tightly packed tissues and capture subtle distinctions missed by other methods. Its fine-grained segmentation illustrated the algorithm's potential in extracting meaningful patterns from intricate MRI data.

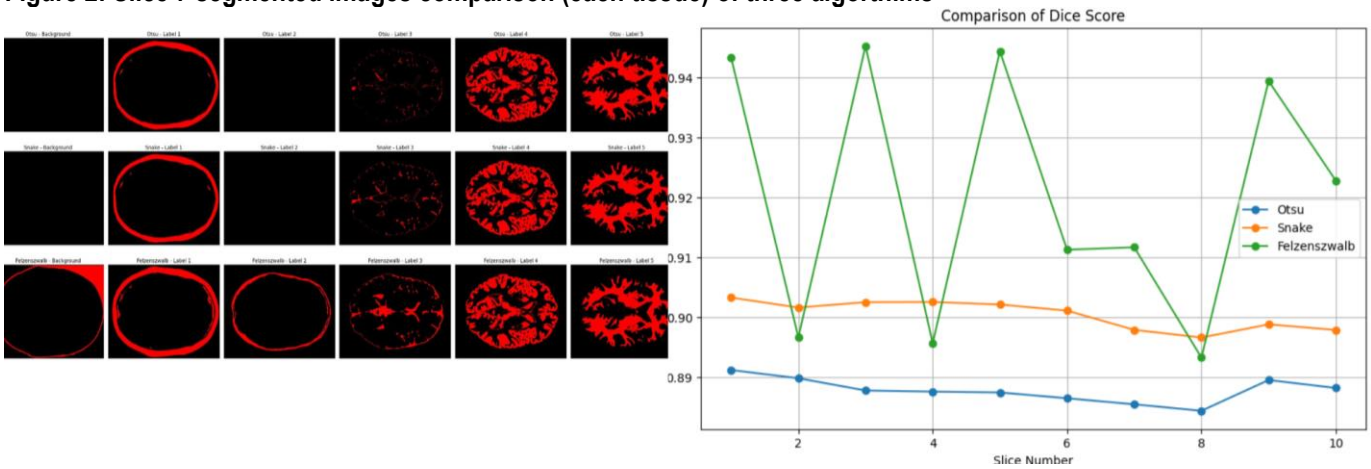
Segmentation Methodologies: A Critical Assessment

The strategic selection of evaluation metrics such as the Dice coefficient, F1 score, and Jaccard index was instrumental due to their significant relevance in medical image analysis. These metrics, combined with additional insights from a CSV file containing the results of evaluation metrics, that includes precision and recall values, provide a comprehensive framework for assessing the segmentation methods' efficacy. Specifically, Precision and Recall are pivotal in evaluating the segmentation's exactness and completeness, which is essential for accurately distinguishing between different brain tissues. **Jaccard Index**: Evaluates the intersection over union of label sets. **Precision and Recall:** Assess the exactness and completeness of the segmentation, respectively. **F1 Score:** The harmonic mean of precision and recall, providing a balance between them.

Figure 1 slice 7 segmented images comparison against ground truth table with evaluation metrics


```
Slice 7, Otsu: Dice=0.886, F1=0.886, Jaccard=0.795
Slice 7, Snake: Dice=0.898, F1=0.898, Jaccard=0.815
Slice 7, Felzenszwalb: Dice=0.912, F1=0.912, Jaccard=0.838
```

Figure 2: Slice 7 segmented images comparison (each tissue) of three algorthims



When reviewing the segmented images (refer to Figure 2), the Felzenszwalb algorithm's capacity to accurately detect diverse tissues, including elusive ones like air and the skull, is noticeable. This contrasts with the Otsu's and Snake methods, which show limited detection in certain tissues. The comparative performance (refer to Figure 1) echoes this finding, with the Felzenszwalb algorithm exhibiting consistently superior metric scores. The graph presents the Dice scores across ten slices, showing Felzenszwalb algorithm consistently achieving higher scores than Otsu's and Snake methods, indicating its more effective segmentation capability across different slices. Even though inconsistency is observed in the dice score of felzenszwalb algorithm with different slices, it works better than the other two methods comparatively.

From this limited research we can conclude that Felzenszwalb algorithm works better for MRI tissue segmentation. The conclusion that the Felzenszwalb algorithm is more suitable for MRI tissue segmentation is supported by comprehensive analysis. This includes reviewing segmented outputs across all slices, evaluating performance through metrics such as Dice score, precision, recall, F1 score, and Jaccard index, and employing tabular data alongside graphical representations of these results. Through this thorough examination, it is clear that the Felzenszwalb method provides a more nuanced and accurate segmentation across the various brain tissues compared to its counterparts.

Analysis of 3D Tissue Segmentation Techniques

In the realm of medical imaging, particularly MRI, the precision of segmentation directly influences the diagnosis and treatment planning. This section delves into the application of two 3D segmentation techniques: ***3D Otsu Thresholding and 3D Watershed Segmentation***, focusing on their methodology, implementation, and effectiveness.

Methodology and Justification: The **3D Otsu method** is an extension of the traditional Otsu's method, which computes a global threshold to segregate the image into foreground and background, simplifying the intensity distribution into two classes. Though effective for basic separation, its performance in complex tissues can be suboptimal. The **3D Watershed Segmentation**, conversely, is a more sophisticated approach that regards the image as a topographic surface, identifying 'catchment basins' and 'watershed ridge lines' to isolate different tissue types. The segmentation is initialized from markers obtained through local maxima, ensuring precise delineation of intricate structures.

Implementation Insight: The segmentation begins with preprocessing, using Gaussian smoothing to reduce noise, enhancing the algorithms' performance. For the Watershed, the local maxima guide the segmentation, crucial for distinguishing between adjacent tissues with similar intensities.

Result Analysis: Upon examining the segmented outputs against the ground truth (refer to Figure 3), the Watershed method notably outshines the 3D Otsu, particularly in distinguishing different brain tissues. This superior delineation is depicted through a clear separation of tissue layers, visible from the comparative segmentation outputs. Not only does the Watershed method accurately segment the major brain tissues, but it also better recognizes subtle features like the cerebrospinal fluid spaces, which Otsu's method fails to delineate accurately.

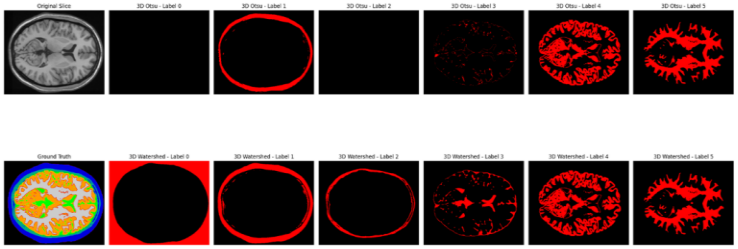


Figure 3: segmented image comparison

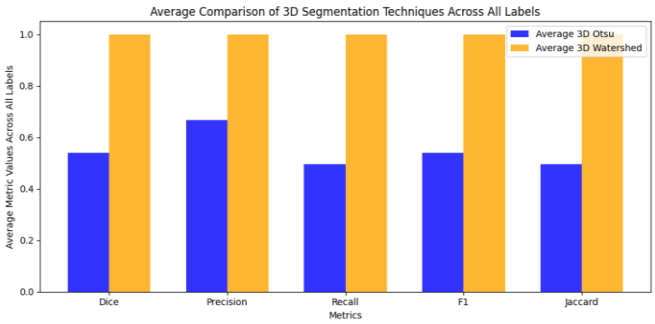


Figure 4: comparison of evaluation metrics

Moreover, the comparison graph (figure4) highlights the Watershed's consistent performance across different slices and labels. The algorithm maintains high scores across the Dice coefficient, Precision, Recall, F1 score, and Jaccard index, substantiating its reliability and robustness in handling 3D data. *On Analyzing why watershed algorithm works better*: Otsu's method, based on global thresholding, tends to group pixels into two categories, which can be limiting when dealing with complex structures like the brain where there are subtle intensity differences between different tissue types. In contrast, the watershed algorithm treats the image as a topographic surface and segments the image into regions based on the gradients. This approach is particularly beneficial for MRI images. Hence, the Watershed algorithm demonstrates remarkable capabilities in 3D segmentation tasks.

Conclusion

In summary, the comparative study distinctly highlights the superior performance of the 3D Watershed algorithm over both traditional 2D methods and other 3D techniques in segmenting MRI images from the 'Brain.mat' dataset. Despite the Felzenszwalb algorithm achieving commendable Dice scores around 0.9 in certain instances, its inconsistency across slices marked a limitation. Conversely, the 3D Watershed method demonstrated remarkable consistency and accuracy, showing nearly perfect results across all evaluation metrics, which underscores its reliability for complex brain tissue differentiation.

This analysis confirms the effectiveness of the 3D Watershed algorithm for MRI image segmentation, suggesting its potential for real-time clinical applications. However, future studies should explore other innovative algorithms, like deep learning models, which may offer enhanced performance and adaptability for real-time MRI segmentation tasks. Such advancements could significantly improve diagnostic processes and patient outcomes in medical imaging.

Extra section

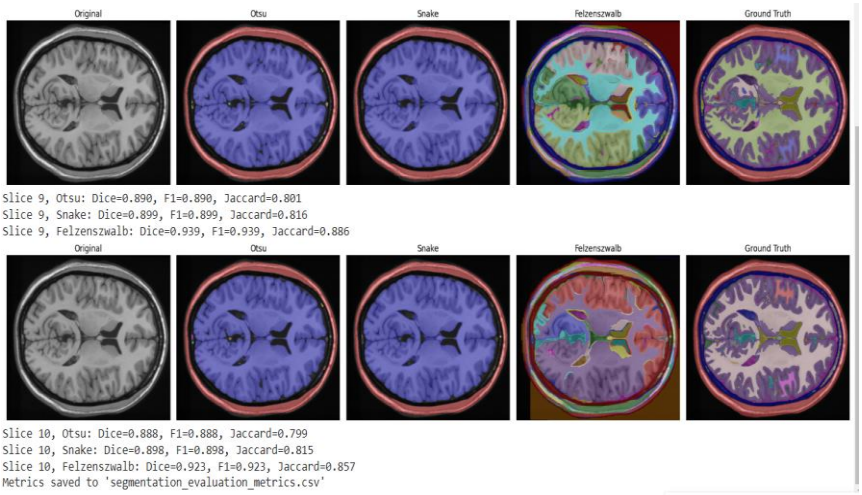
2D segmentation code:

```
#imported all necessary modules
# Load MRI data and ground truth labels
data = loadmat('Brain.mat')
T1 = data['T1']  # MRI images
gt_labels = data['label']  # Ground truth labels
# Define evaluation metrics
def dice_coefficient(y_true, y_pred):
    y_true = np.asarray(y_true).astype(bool)
    y_pred = np.asarray(y_pred).astype(bool)
    intersection = np.logical_and(y_true, y_pred).sum()
    return 2. * intersection / (y_true.sum() + y_pred.sum()) if (y_true.sum() + y_pred.sum()) != 0 else 1
def precision(y_true, y_pred):
    y_true = np.asarray(y_true).astype(bool)
    y_pred = np.asarray(y_pred).astype(bool)
    return np.logical_and(y_true, y_pred).sum() / y_pred.sum() if y_pred.sum() != 0 else 0
def recall(y_true, y_pred):
    y_true = np.asarray(y_true).astype(bool)
    y_pred = np.asarray(y_pred).astype(bool)
    return np.logical_and(y_true, y_pred).sum() / y_true.sum() if y_true.sum() != 0 else 0
def f1_measure(precision_val, recall_val):
    return 2 * (precision_val * recall_val) / (precision_val + recall_val) if (precision_val + recall_val) != 0 else 0
def jaccard_index(y_true, y_pred):
    y_true = np.asarray(y_true).astype(bool)
    y_pred = np.asarray(y_pred).astype(bool)
    intersection = np.logical_and(y_true, y_pred).sum()
    return intersection / np.logical_or(y_true, y_pred).sum() if np.logical_or(y_true, y_pred).sum() != 0 else 1
# Normalization function for your images
def normalize_image(img):
    img_min, img_max = img.min(), img.max()
```

```
    return (img - img_min) / (img_max - img_min)
# Initialize lists to store evaluation metrics for all slices and methods
metrics_data = []
# Process each slice
total_images = T1.shape[2]
for i in range(total_images):
    img_slice = T1[:, :, i]
    gt_slice = gt_labels[:, :, i]
    img_slice_normalized = normalize_image(img_slice)
    # Denoise the image using Gaussian smoothing
    smoothed_slice = gaussian(img_slice_normalized, sigma=1)
    # Apply segmentation algorithms
    thresh = threshold_otsu(smoothed_slice)
    otsu_segmentation = smoothed_slice > thresh
    snake_segmentation = chan_vese(smoothed_slice, mu=0.1, lambda1=1, lambda2=1, tol=1e-3)
    felzenszwalb_segmentation = felzenszwalb(smoothed_slice, scale=100, sigma=0.5, min_size=50)
    # Visualization
    fig, axes = plt.subplots(1, 5, figsize=(20, 4))
    axes[0].imshow(img_slice_normalized, cmap='gray')
    axes[0].set_title('Original')
    axes[1].imshow(label2rgb(label(otsu_segmentation), bg_label=0, image=img_slice_normalized))
    axes[1].set_title('Otsu')
#similarly visualize for other 2D alogorithm
    for ax in axes:
        ax.axis('off')
    plt.tight_layout()
    plt.show()
    # Evaluate and print metrics for each segmentation method
    for method_name, segmentation in zip(['Otsu', 'Snake', 'Felzenszwalb'], [otsu_segmentation, snake_segmentation,
felzenszwalb_segmentation]):
        prec = precision(gt_slice, segmentation)
        rec = recall(gt_slice, segmentation)
```

```
        f1 = f1_measure(prec, rec)
        dice = dice_coefficient(gt_slice, segmentation)
        jaccard = jaccard_index(gt_slice, segmentation)
        print(f'Slice {i + 1}, {method_name}: Dice={dice:.3f}, F1={f1:.3f}, Jaccard={jaccard:.3f}')
        metrics_data.append({'Slice': i + 1, 'Method': method_name, 'Dice': dice, 'F1': f1, 'Jaccard': jaccard})
# Convert evaluation metrics into a DataFrame and then to a CSV file
df_metrics = pd.DataFrame(metrics_data)
df_metrics.to_csv('segmentation_evaluation_metrics.csv', index=False)
print("Metrics saved to 'segmentation_evaluation_metrics.csv'")
```

Visualization of 2D algorithms



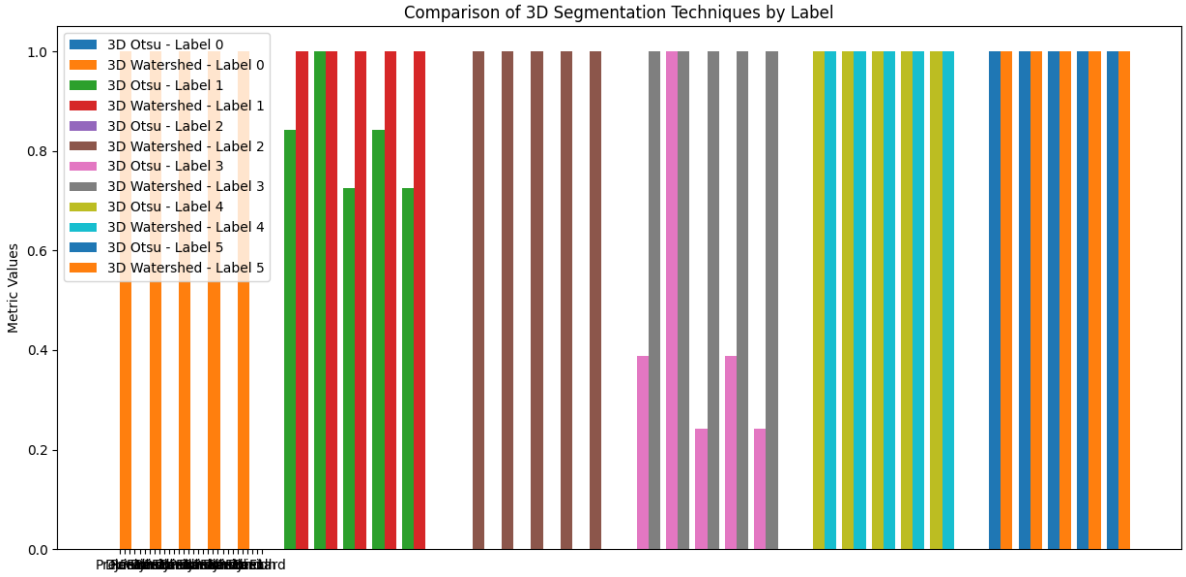
Slice	Method	Label	Dice	Precision	Recall	F1	Jaccard
1	Otsu	Air	0.786087219237705	0.6475648264812481	1.0	0.786087219237705	0.6475648264812481
1	Otsu	Skin/Scalp	0.8452663644431652	1.0	0.732001163225458	0.8452663644431653	0.732001163225458
1	Otsu	Skull	0.0	0.0	0.0	0.0	0.0
1	Otsu	CSF	0.0	0.0	0.0	0.0	0.0
1	Otsu	Gray Matter	0.00010562450488513335	0.16666666666666666	5.282899255111205e-05	0.00010562450488513335	5.281504172388296e-05
1	Otsu	White Matter	0.0	0.0	0.0	0.0	0.0
1	Snake	Air	0.8034663115434351	0.6714949351571069	1.0	0.8034663115434351	0.6714949351571069
1	Snake	Skin/Scalp	0.8679396134129709	1.0	0.7666902081342695	0.8679396134129709	0.7666902081342695
1	Snake	Skull	0.0	0.0	0.0	0.0	0.0
1	Snake	CSF	0.002016535591853196	0.07692307692307693	0.001021659174499387	0.002016535591853196	0.0010092854259184498
1	Snake	Gray Matter	0.0	0.0	0.0	0.0	0.0
1	Snake	White Matter	0.0	0.0	0.0	0.0	0.0
1	Felzenszwalb	Air	0.7994924589785737	1.0	0.6659620466010089	0.7994924589785738	0.6659620466010089
1	Felzenszwalb	Skin/Scalp	0.3338254399433093	0.5770133714402368	0.2348469112209713	0.3338254399433093	0.20035442140705298
1	Felzenszwalb	Skull	0.0	0.0	0.0	0.0	0.0
1	Felzenszwalb	CSF	0.0	0.0	0.0	0.0	0.0
1	Felzenszwalb	Gray Matter	0.0	0.0	0.0	0.0	0.0

3D image segmentation code:

```
#import required libraries
# Load 3D MRI data and ground truth labels
def apply_3d_watershed(volume): # Function for 3D Watershed Segmentation
```

```
    smoothed_volume = gaussian(volume, sigma=[1, 1, 1])
    distance = ndi.distance_transform_edt(smoothed_volume > 0.5)
    local_maxi_coords = peak_local_max(distance, footprint=np.ones((3, 3, 3)), labels=smoothed_volume > 0.5)
    markers = np.zeros_like(distance, dtype=np.int32)
    for i, coord in enumerate(local_maxi_coords, start=1):
        markers[tuple(coord)] = i
    segmented_volume = watershed(-distance, markers, mask=smoothed_volume > 0.5)
    return segmented_volume
# Apply segmentation algorithms
otsu_threshold = threshold_otsu(T1_volume)
otsu_segmentation = T1_volume > otsu_threshold
watershed_segmentation = apply_3d_watershed(T1_volume)
# Evaluation metrics function, here 'calculate_metrics'
# Visualization and metrics calculation for a specific slice
slice_index = 5
unique_labels = np.unique(gt_labels_volume)
metrics_names = ['Dice', 'Precision', 'Recall', 'F1', 'Jaccard']
metrics_results = {label: {method: [] for method in ['3D Otsu', '3D Watershed']} for label in unique_labels}
#plot the metrics
for i, label in enumerate(unique_labels, start=1):
    # Otsu segmentation mask for specific label
    otsu_mask = np.where(gt_labels_volume[slice_index] == label, otsu_segmentation[slice_index], 0)
    # Watershed segmentation mask for specific label
    watershed_mask = np.where(gt_labels_volume[slice_index] == label, watershed_segmentation[slice_index], 0)
     # Visualization
     # Metrics calculation
    for method, seg_mask in zip(['3D Otsu', '3D Watershed'], [otsu_mask, watershed_mask]):
        metrics = calculate_metrics(gt_labels_volume[slice_index] == label, seg_mask)
        for name, value in metrics.items():
            metrics_results[label][method].append(value)
for ax in axes.flatten():
    ax.axis('off')
```

```
plt.tight_layout()
plt.show()
# Print Metrics for each label
for label in unique_labels:
    print(f"\nMetrics for Label {label}:")
    for method in ['3D Otsu', '3D Watershed']:
        print(f"{method} Metrics:")
        for metric_name in metrics_names:
            print(f"{metric_name}: {np.mean(metrics_results[label][method][metrics_names.index(metric_name)]):.3f}")
# Plotting comparison graph
```



Label	Method	Dice	Precision	Recall	F1	Jaccard
0	3D Otsu	0.0	0.0	0.0	0.0	0.0
0	3D Watershed	1.0	1.0	1.0	1.0	1.0
1	3D Otsu	0.8413178809576536	1.0	0.7260989594393714	0.8413178809576536	0.7260989594393714
1	3D Watershed	1.0	1.0	1.0	1.0	1.0
2	3D Otsu	0.0	0.0	0.0	0.0	0.0
2	3D Watershed	1.0	1.0	1.0	1.0	1.0
3	3D Otsu	0.388822652757079	1.0	0.24132827675515678	0.388822652757079	0.24132827675515678
3	3D Watershed	1.0	1.0	1.0	1.0	1.0
4	3D Otsu	1.0	1.0	1.0	1.0	1.0
4	3D Watershed	1.0	1.0	1.0	1.0	1.0
5	3D Otsu	1.0	1.0	1.0	1.0	1.0
5	3D Watershed	1.0	1.0	1.0	1.0	1.0

Csv file with results of evaluation for each label of first 5 slides