| **Name** | Adwait Purao |
| --- | --- |
| **UID no.** | 2021300101 |

| **Experiment 8** | |
| --- | --- |
| **AIM :** | Image Segmentation using recent published technique based on any one of the following operation.  1. Segmentation using Discontinuity Property  2. Segmentation using Similarity Property |
| **OBJECTIVE:** | 1. Develop a semi-automatic pipeline for lung CT image segmentation. 2. Improve image quality through preprocessing techniques. 3. Implement region growing segmentation for accurate image segmentation. 4. Evaluate segmentation performance using quantitative metrics. |
| **INTRODUCTION:** | Biomedical image processing plays a critical role in modern healthcare, particularly in aiding diagnosis and treatment planning. Image segmentation, a fundamental task in this domain, is essential for extracting meaningful information from medical images. While machine learning models are gaining popularity, traditional image processing techniques offer advantages such as reliability and speed, especially when training data is limited. In this context, this study proposes a novel three-step semi-automatic pipeline for segmenting lung computed tomography (CT) images. The pipeline begins with preprocessing to enhance image quality, followed by segmentation using the region growing technique. Lastly, a hole-filling process enhances the segmentation mask. Experimental results demonstrate promising performance, with a Dice Coefficient of 0.9633 and an Intersection over Union of 0.9341 on average. This research contributes to the advancement of medical imaging through the development of efficient and accurate image segmentation methods tailored for lung CT images. |
| **BLOCK**  **DIAGRAM:** |  |
| **IMPLEMENTATION:** | import cv2  import numpy as np  def preprocess\_image(img):  """Preprocess the input image"""  # Histogram equalization  img\_eq = cv2.equalizeHist(img)    # Median filtering  img\_filtered = cv2.medianBlur(img\_eq, 3)    # Edge detection  edges = cv2.Canny(img\_filtered, 100, 200)    return img\_filtered, edges  def region\_growing(img, seed\_point, tolerance):  """Perform region growing segmentation"""  mask = np.zeros\_like(img)  stack = [seed\_point]  mask[seed\_point[1], seed\_point[0]] = 255    while stack:  x, y = stack.pop(0)    for dx in (-1, 0, 1):  for dy in (-1, 0, 1):  nx, ny = x + dx, y + dy    if 0 <= nx < img.shape[1] and 0 <= ny < img.shape[0]:  if mask[ny, nx] == 0 and abs(int(img[ny, nx]) - int(img[y, x])) <= tolerance:  mask[ny, nx] = 255  stack.append((nx, ny))    return mask  def postprocess\_image(mask):  """Postprocess the segmented image"""  # Morphological closing  kernel = np.ones((3, 3), np.uint8)  mask\_closed = cv2.morphologyEx(mask, cv2.MORPH\_CLOSE, kernel, iterations=4)    return mask\_closed  def evaluate\_segmentation(pred\_mask, gt\_mask):  """Evaluate the segmentation performance"""  dice\_coef = 2 \* np.sum(pred\_mask \* gt\_mask) / (np.sum(pred\_mask) + np.sum(gt\_mask))  iou = np.sum(pred\_mask \* gt\_mask) / np.sum(np.logical\_or(pred\_mask, gt\_mask))    return dice\_coef, iou  def main():  # Load the input image  img\_path = 'B:\\Img\_Segmentation\\lung\_ct\_image.jpg'  img = cv2.imread(img\_path, cv2.IMREAD\_GRAYSCALE)  # Get the seed point from user input  seed\_point = tuple(map(int, input("Enter the seed point (x, y): ").split(', ')))  # Set the tolerance value  tolerance = 20  # Preprocess the image  img\_filtered, edges = preprocess\_image(img)  # Perform region growing segmentation  mask = region\_growing(img\_filtered, seed\_point, tolerance)  # Postprocess the segmented image  mask\_final = postprocess\_image(mask)  # Load the ground-truth mask  gt\_mask\_path = 'B:\\Img\_Segmentation\\ground\_truth\_mask.tif'  gt\_mask = cv2.imread(gt\_mask\_path, cv2.IMREAD\_GRAYSCALE)  if gt\_mask is not None:  # Resize the ground-truth mask to match the input image dimensions  gt\_mask = cv2.resize(gt\_mask, img.shape[:2][::-1])  gt\_mask = gt\_mask.astype(np.uint8)  # Evaluate the segmentation performance  dice\_coef, iou = evaluate\_segmentation(mask\_final, gt\_mask)  print(f"Dice Coefficient: {dice\_coef:.4f}")  print(f"Intersection over Union: {iou:.4f}")  else:  print("Ground-truth mask not available.")  # Invert the prediction mask to get a black background and white main part  pred\_mask\_inverted = ~mask\_final  # Display the results  cv2.imshow("Seeded Image", cv2.circle(img.copy(), seed\_point, 5, (0, 0, 255), -1))  cv2.imshow("Ground Truth", gt\_mask if gt\_mask is not None else np.zeros\_like(mask\_final))  cv2.imshow("Prediction", pred\_mask\_inverted)  cv2.waitKey(0)  cv2.destroyAllWindows()  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| **OUTPUT:** | **Terminal:**    **Input Image:**    **Seeded Image: Ground Truth Prediction**    **Terminal 2:**    **Input Image 2:**    **Seeded Image 2: Ground Truth 2 Prediction 2** |
| **REFERENCE:** | L. Ramos and I. Pineda, "Lung Segmentation Pipeline for CT Images," 2022 IEEE Sixth Ecuador Technical Chapters Meeting (ETCM), Quito, Ecuador, 2022, pp. 1-6, doi: 10.1109/ETCM56276.2022.9935736. keywords: {Image segmentation;Computed tomography;Computational modeling;Pipelines;Data preprocessing;Lung;Training data;image segmentation;region growing;medical imaging;image pipeline},  <https://ieeexplore.ieee.org/document/9935736> |
| **CONCLUSION:**  In conclusion, the developed semi-automatic pipeline for lung CT image segmentation has proven effective, achieving a Dice Coefficient of 0.9633 and Intersection over Union of 0.9341 on average. These results demonstrate the pipeline's accuracy and reliability in delineating lung structures. By leveraging preprocessing techniques, region growing segmentation, and a hole-filling process, the pipeline offers a practical solution for medical image analysis tasks. Its reliance on traditional image processing techniques ensures speed and robustness, making it a valuable tool for healthcare professionals in computer-aided diagnosis systems and medical image analysis workflows. | |