%% Step 1: Initialization and Setup

% Define the number of subjects and regions

n\_subjects = 10; % Number of subjects

n\_regions = 90; % Number of brain regions

% Initialize a cell array to hold the datasets for each subject

tractography\_data = cell(1, n\_subjects);

% Directory path for the data (adjust if necessary)

data\_dir = './';

% Initialize a 3D matrix to store all subjects' connection matrices

all\_subjects\_data = zeros(n\_regions, n\_regions, n\_subjects);

%% Step 2: Read the Data from CSV Files

for i = 1:n\_subjects

% Construct the filename for each subject

filename = fullfile(data\_dir, sprintf('subject\_%d\_tractography.csv', i));

% Read the CSV file into a table

data\_table = readtable(filename, 'ReadVariableNames', true);

% Convert the table to a matrix (ignore the region labels)

data\_matrix = table2array(data\_table(:, 2:end)); % skip the region name column

% Store the data matrix in the cell array and 3D matrix

tractography\_data{i} = data\_matrix;

all\_subjects\_data(:, :, i) = data\_matrix;

% Print a message indicating successful data loading

fprintf('Loaded data for subject %d\n', i);

end

%% Step 3: Compute Average Connection Matrix

% Compute the mean connection matrix across all subjects

mean\_matrix = mean(all\_subjects\_data, 3);

% Display the average connection matrix

figure;

imagesc(mean\_matrix);

colorbar;

title('Mean Connection Matrix Across 10 Subjects');

xlabel('Brain Region');

ylabel('Brain Region');

axis square;

%% Step 4: Compute Standard Deviation and Variability

% Compute the standard deviation across subjects for each connection

std\_matrix = std(all\_subjects\_data, 0, 3);

% Display the standard deviation matrix

figure;

imagesc(std\_matrix);

colorbar;

title('Standard Deviation of Connections Across Subjects');

xlabel('Brain Region');

ylabel('Brain Region');

axis square;

%% Step 5: Thresholding the Mean Matrix

% Apply a threshold to the mean matrix (retain only connections > threshold)

threshold = 0.5; % Example threshold

thresholded\_matrix = mean\_matrix;

thresholded\_matrix(mean\_matrix < threshold) = 0;

% Display the thresholded matrix

figure;

imagesc(thresholded\_matrix);

colorbar;

title(sprintf('Thresholded Mean Connection Matrix (Threshold = %.2f)', threshold));

xlabel('Brain Region');

ylabel('Brain Region');

axis square;

%% Step 6: Identify Strongest Connections

% Identify the top N strongest connections in the mean matrix

N = 10; % Number of top connections to identify

[sorted\_vals, sorted\_idx] = sort(mean\_matrix(:), 'descend');

% Get the row and column indices of the top N connections

[top\_rows, top\_cols] = ind2sub([n\_regions, n\_regions], sorted\_idx(1:N));

% Display the top N connections and their values

fprintf('Top %d Strongest Connections:\n', N);

for i = 1:N

fprintf('Connection between Region %d and Region %d: %.4f\n', top\_rows(i), top\_cols(i), sorted\_vals(i));

end

%% Step 7: Compute and Display Correlations Between Subjects

% Initialize a matrix to hold correlations between subjects

subject\_correlations = zeros(n\_subjects, n\_subjects);

% Loop through each pair of subjects and compute correlations

for i = 1:n\_subjects

for j = 1:n\_subjects

% Flatten the connection matrices for correlation calculation

subj1\_data = all\_subjects\_data(:, :, i);

subj2\_data = all\_subjects\_data(:, :, j);

subj1\_flat = subj1\_data(:);

subj2\_flat = subj2\_data(:);

% Compute the correlation coefficient between subjects i and j

subject\_correlations(i, j) = corr(subj1\_flat, subj2\_flat);

end

end

% Display the correlation matrix

figure;

imagesc(subject\_correlations);

colorbar;

title('Correlation Between Subjects');

xlabel('Subject');

ylabel('Subject');

axis square;

%% Step 8: Statistical Comparison of Connections Between Groups

% Suppose we want to compare two groups of subjects (e.g., first 5 vs last 5)

% Group 1: Subjects 1 to 5

group1\_data = all\_subjects\_data(:, :, 1:5);

% Group 2: Subjects 6 to 10

group2\_data = all\_subjects\_data(:, :, 6:10);

% Compute the mean connection matrices for both groups

group1\_mean = mean(group1\_data, 3);

group2\_mean = mean(group2\_data, 3);

% Perform a t-test for each connection between the two groups

p\_values = zeros(n\_regions, n\_regions);

for r1 = 1:n\_regions

for r2 = 1:n\_regions

% Extract connection values for both groups

group1\_conn = squeeze(group1\_data(r1, r2, :));

group2\_conn = squeeze(group2\_data(r1, r2, :));

% Perform a two-sample t-test

[~, p] = ttest2(group1\_conn, group2\_conn);

p\_values(r1, r2) = p;

end

end

% Display the p-value matrix

figure;

imagesc(p\_values);

colorbar;

title('P-Values for Group Comparison (T-Test)');

xlabel('Brain Region');

ylabel('Brain Region');

axis square;

%% Step 9: Visualization of Significant Differences

% Threshold the p-value matrix to retain significant differences

alpha = 0.05; % Significance level

significant\_diff\_matrix = p\_values < alpha;

% Display the significant differences

figure;

imagesc(significant\_diff\_matrix);

colorbar;

title(sprintf('Significant Differences (p < %.2f)', alpha));

xlabel('Brain Region');

ylabel('Brain Region');

axis square;

%% Step 10: Save Results for Further Analysis

% Save the mean connection matrix to a CSV file

csvwrite('mean\_connection\_matrix.csv', mean\_matrix);

% Save the thresholded matrix

csvwrite('thresholded\_connection\_matrix.csv', thresholded\_matrix);

% Save the standard deviation matrix

csvwrite('std\_connection\_matrix.csv', std\_matrix);

% Save the p-value matrix for group comparison

csvwrite('p\_value\_matrix.csv', p\_values);

% Save the significant differences matrix

csvwrite('significant\_differences.csv', significant\_diff\_matrix);

fprintf('Analysis completed and results saved.\n');

## ROI MASK

%% Initialize and load subject 1 data from 'fiber\_tracking\_dataset'

disp('Loading DWI data, b-values, b-vectors, and ROI mask for subject 1');

% Directory containing the dataset files

dataDir = 'fiber\_tracking\_dataset/';

% Load DWI data for subject 1

dwi\_filename = [dataDir 'subject\_1\_dwi.nii.gz'];

dwi\_nii = niftiread(dwi\_filename);

dwi\_info = niftiinfo(dwi\_filename);

% Load b-values and b-vectors

bvals\_filename = [dataDir 'subject\_1\_bvals.txt'];

bvecs\_filename = [dataDir 'subject\_1\_bvecs.txt'];

bvals = load(bvals\_filename);

bvecs = load(bvecs\_filename);

% Load ROI mask for subject 1

roi\_filename = [dataDir 'subject\_1\_roi\_mask.nii.gz'];

roi\_mask = niftiread(roi\_filename);

roi\_info = niftiinfo(roi\_filename);

%% Display basic information about the loaded data

disp('DWI Data Information:');

disp(dwi\_info);

disp('Size of DWI data:');

disp(size(dwi\_nii));

disp('B-values:');

disp(bvals);

disp('B-vectors:');

disp(bvecs);

disp('ROI Mask Information:');

disp(roi\_info);

%% Visualize the DWI data (first gradient direction) and ROI mask

disp('Visualizing the first gradient direction of DWI data and ROI mask');

figure;

% Display the middle slice of the first gradient direction of DWI data

subplot(1, 2, 1);

middle\_slice = round(size(dwi\_nii, 3) / 2); % Select the middle slice

imshow(dwi\_nii(:, :, middle\_slice, 1), []); % Display the first gradient volume

title('DWI (Gradient 1) - Middle Slice');

% Display the corresponding slice of the ROI mask

subplot(1, 2, 2);

imshow(roi\_mask(:, :, middle\_slice), []); % Display the same middle slice for ROI

title('ROI Mask - Middle Slice');

%% Prepare data for fiber tracking (example)

% Here, we will organize the data in a way that can be used for fiber tracking.

% Number of gradients

num\_gradients = size(dwi\_nii, 4);

% Normalize the b-vectors to ensure unit vectors

bvecs\_normalized = bvecs ./ vecnorm(bvecs);

% Display the first 5 b-values and b-vectors as a check

disp('First 5 b-values:');

disp(bvals(1:5));

disp('First 5 normalized b-vectors:');

disp(bvecs\_normalized(:, 1:5));

%% Save the preprocessed data (optional)

% If necessary, you can save the loaded and preprocessed data into MATLAB .mat files

save([dataDir 'subject\_1\_preprocessed\_data.mat'], 'dwi\_nii', 'bvals', 'bvecs\_normalized', 'roi\_mask');

disp('Preprocessed data saved as subject\_1\_preprocessed\_data.mat');

%% Further Steps

% This section outlines the next steps for performing fiber tracking. You can use

% the data loaded and preprocessed here with external tools (e.g., FSL probtrackx2)

% or other diffusion MRI analysis toolboxes in MATLAB.

% You may now prepare to use a fiber tracking algorithm. Depending on the specific

% tracking algorithm you plan to use, you will need the following:

% - The preprocessed DWI data (4D volume)

% - The b-values and b-vectors (gradient directions)

% - The ROI mask that defines the region for fiber tracking

disp('Data ready for fiber tracking.');

Fiber Tracking

%% Load Preprocessed Data (DWI, b-values, b-vectors, and ROI mask)

% Assuming you've already preprocessed the data and saved it as a .mat file

load('fiber\_tracking\_dataset/subject\_1\_preprocessed\_data.mat', 'dwi\_nii', 'bvals', 'bvecs\_normalized', 'roi\_mask');

%% Set Parameters for Fiber Tracking

step\_size = 1; % Step size for tractography (in voxel units)

max\_angle = 30; % Maximum angle between steps (in degrees)

num\_fibers = 1000; % Number of fibers to track

% Get the size of the DWI data

[dim\_x, dim\_y, dim\_z, num\_gradients] = size(dwi\_nii);

%% Fit the Diffusion Tensor Model (DTI)

% To perform deterministic fiber tracking, we need to fit a tensor model

disp('Fitting the diffusion tensor model to the DWI data');

% Calculate the diffusion tensor for each voxel

% The tensor fitting requires solving a linear equation for each voxel

D = zeros(dim\_x, dim\_y, dim\_z, 3, 3); % 3x3 tensor for each voxel

for x = 1:dim\_x

for y = 1:dim\_y

for z = 1:dim\_z

% Extract the signal for each gradient direction in this voxel

S = squeeze(dwi\_nii(x, y, z, :));

% Build the design matrix for DTI (simplified version)

G = [-bvecs\_normalized(1, :).\*bvals; -bvecs\_normalized(2, :).\*bvals; -bvecs\_normalized(3, :).\*bvals]';

% Solve the linear equation: ln(S/S0) = G \* D

if any(S > 0)

logS = log(S / S(1));

D\_vec = pinv(G) \* logS; % Pseudo-inverse to solve for tensor elements

D(x, y, z, 1, 1) = D\_vec(1);

D(x, y, z, 2, 2) = D\_vec(2);

D(x, y, z, 3, 3) = D\_vec(3);

D(x, y, z, 1, 2) = D\_vec(4);

D(x, y, z, 2, 1) = D\_vec(4);

D(x, y, z, 1, 3) = D\_vec(5);

D(x, y, z, 3, 1) = D\_vec(5);

D(x, y, z, 2, 3) = D\_vec(6);

D(x, y, z, 3, 2) = D\_vec(6);

end

end

end

end

%% Calculate Principal Diffusion Directions

% The principal direction is the eigenvector corresponding to the largest eigenvalue of the tensor

disp('Calculating principal diffusion directions for each voxel');

principal\_directions = zeros(dim\_x, dim\_y, dim\_z, 3); % Store the primary eigenvector for each voxel

for x = 1:dim\_x

for y = 1:dim\_y

for z = 1:dim\_z

% Get the 3x3 diffusion tensor for this voxel

tensor = squeeze(D(x, y, z, :, :));

if any(tensor(:)) % Check if the tensor is non-zero

% Perform eigen decomposition

[V, E] = eig(tensor); % V: eigenvectors, E: eigenvalues

% Find the eigenvector corresponding to the largest eigenvalue

[~, max\_idx] = max(diag(E)); % Get the index of the largest eigenvalue

principal\_directions(x, y, z, :) = V(:, max\_idx); % Save the principal diffusion direction

end

end

end

end

%% Fiber Tracking Using Deterministic Algorithm

disp('Performing deterministic fiber tracking');

% Seed points (starting points for fibers) are chosen randomly from the ROI mask

seed\_points = find(roi\_mask);

seed\_points = seed\_points(randperm(length(seed\_points), num\_fibers)); % Select random seed points

% Initialize array to store fibers

fibers = cell(num\_fibers, 1);

for fiber\_idx = 1:num\_fibers

disp(['Tracking fiber ' num2str(fiber\_idx) '/' num2str(num\_fibers)]);

% Get the initial seed point (voxel coordinates)

[x, y, z] = ind2sub(size(roi\_mask), seed\_points(fiber\_idx));

% Initialize the current position and direction

current\_pos = [x, y, z];

current\_dir = squeeze(principal\_directions(x, y, z, :));

% Initialize the fiber with the seed point

fiber = current\_pos;

% Track the fiber until it leaves the brain or ROI mask

while all(current\_pos > 1 & current\_pos < [dim\_x, dim\_y, dim\_z]) && roi\_mask(current\_pos(1), current\_pos(2), current\_pos(3)) == 1

% Move along the current direction by the step size

next\_pos = current\_pos + current\_dir' \* step\_size;

% Round to nearest voxel

next\_pos = round(next\_pos);

% Check if the angle between the current and next direction exceeds the threshold

next\_dir = squeeze(principal\_directions(next\_pos(1), next\_pos(2), next\_pos(3), :));

angle\_diff = acosd(dot(current\_dir, next\_dir) / (norm(current\_dir) \* norm(next\_dir)));

if angle\_diff > max\_angle

break; % Stop tracking if the direction changes too abruptly

end

% Add the new position to the fiber and update the current position and direction

fiber = [fiber; next\_pos];

current\_pos = next\_pos;

current\_dir = next\_dir;

end

% Store the fiber trajectory

fibers{fiber\_idx} = fiber;

end

%% Visualize the Fibers

disp('Visualizing tracked fibers');

figure;

hold on;

for i = 1:num\_fibers

fiber = fibers{i};

plot3(fiber(:, 1), fiber(:, 2), fiber(:, 3), 'LineWidth', 1);

end

xlabel('X');

ylabel('Y');

zlabel('Z');

title('Deterministic Fiber Tracking Results');

grid on;

view(3);

disp('Fiber tracking completed.');