**SELECT MODULE：**

**Input:**

* Step 1: Image sequence **I(t)**
* Step 2: 3D feature point records **P\_3D(t)**

**Output:**

* Step 1: Region of Interest **ROI(t)**
* Step 2: Displacement time-history curve **avg\_displacement\_list**

1: Initialize avg\_displacement\_list as [ ]

2: For each frame t in the image sequence I(t):

3: # Step 1: Select region of interest ROI(t)

4: If selection\_type == "manual" then:

5: Display image I(t) to the user

6: Prompt the user to manually select the region of interest `ROI(t)`

# Here, the user can define the top-left and bottom-right coordinates of the ROI (assuming it is a rectangle)

7: User manually selects `ROI(t)`

8: Else if selection\_type = "automatic" then

9: gray\_image ← convert\_to\_grayscale(I(t))

10: blurred\_image ← apply\_gaussian\_blur(gray\_image, 5)

11: mean\_shift\_image ← apply\_meanshift(blurred\_image)

# Enhance regions with Mean Shift

12: thresholded\_image ← apply\_otsu\_threshold(mean\_shift\_image)

# Segment windows with Otsu's thresholding

13: windows ← detect\_windows(thresholded\_image)

# Detect window regions

14: ROI(t) ← select\_interlayer\_region(windows)

# Define interlayer region between windows

15: End If

16: Store `ROI(t)` for further processing (feature extraction / tracking)

17: # Step 2: Calculate the average displacement of all successfully tracked feature points in ROI(t)

18: Initialize displacement\_sum as 0

19: Initialize valid\_point\_count as 0

20: For each feature point point in Tracked\_Points(t):

21: If point is valid in ROI(t):

22: displacement = calculate\_displacement(point)

23: displacement\_sum += displacement

24: valid\_point\_count += 1

25: End If

26: End For

27: # Calculate the average displacement for the frame

28: If valid\_point\_count > 0 then:

29: avg\_displacement = displacement\_sum / valid\_point\_count

30: Else:

31: avg\_displacement = 0 # If no valid feature points, set displacement to 0

32: End If

33: # Store the average displacement in the list

34: Append avg\_displacement to avg\_displacement\_list

35: End For each frame t

36: Return avg\_displacement\_list # Return the displacement time-history curve for the entire video

**FEATURE PROCESSING ALGORITHM：**

**Input:** Region of Interest **ROI(t)**

**Output**: Matched feature points **M**

**Key Parameters:**

**p(t)**: The feature point positions extracted at time t, representing the 2D coordinates of the feature points in the ROI.

**d(t)**: Descriptors calculated based on the feature point positions p(t) at time t. Each descriptor corresponds to a specific feature point.

**f(t)**: Encoded feature descriptors generated from d(t) through an encoder. These are used as the basis for inter-frame matching.

**S(t, t+1)**: A pairwise score matrix between frames t and t+1, used to evaluate the matching score between feature points in consecutive frames.

1. Initialize frame count `t = 1`

2. H, W = Dimensions of ROI(t) image

3. Mid(t) ← VGG-like CNN(ROI(t)) # (dimensions H/8 \* W/8 \* 128)

4. p(t) ← Feature Decoder(Mid(t)) # (dimensions H \* W \* 1)

5. d(t) ← Descriptor Decoder(Mid(t)) # (dimensions H \* W \* 256)

6. Initialize the matched feature point list `M = []`

7. Sort `p(t)` by confidence

8. Store sorted `p(t)` in `M` and record corresponding `d(t)`

9. While True: (Continuously process next frame)

10. Compute `p(t+1)`, `d(t+1)`

11. f(t) ← Intra&Extra Attention(d(t), d(t+1))

12. S(t, t+1) ← Compute\_Score\_Matrix(f(t), f(t+1))

13. M(t) ← Sinkhorn\_Optimal\_Matching(S(t, t+1))

14. p(t), d(t) ← p(t+1), d(t+1)

15. Store matching points from `p(t)` and `p(t+1)` in `M`

16. End While

**TRACKING ALGORITHM:**

**Input:** Feature point matches **M(t)**, Depth Map **Depth\_Map(t)**

**Output:** 3D feature point records **P\_3D(t)**, written to a CSV file

**Key Parameters:**

* **Feature\_Point\_Record** data structure includes:
  + **ID**: Unique identifier of the feature point
  + **2D\_Coordinates**: 2D coordinates of the feature point
  + **3D\_Coordinates**: 3D coordinates derived from the depth map
  + **Frame\_ID**: The frame number where the point was last detected
  + **Is\_Active**: Indicator of whether the feature point is active

1: Initialize `Feature\_Points\_List` as [ ]

2: Initialize `frame\_counter` as 0

3: while True do:

4: if `frame\_counter == 0` then:

5: For each feature point pair in `M(t)`:

6: Extract the 2D coordinates and corresponding 3D coordinates from `Depth\_Map(t)`

7: Create a `Feature\_Point\_Record` to store these values

8: Append the `Feature\_Point\_Record` to `Feature\_Points\_List`

9: Save the current feature point list as `last\_Feature\_Points\_List`

10: else:

11: For each feature point match in `M(t)`:

12: Retrieve the corresponding `last\_Feature\_Point\_Record` and current `Feature\_Points\_2D`

13: If the current feature point matches the `last\_Feature\_Point\_Record`, then:

14: Update the 2D and 3D coordinates in `last\_Feature\_Point\_Record`

15: Update the `Frame\_ID` to the current `frame\_counter`

16: Mark `last\_Feature\_Point\_Record` as active

17: else:

18: Mark `last\_Feature\_Point\_Record` as inactive

19: End If

20: End For

21: For each feature record in `last\_Feature\_Points\_List`:

22: If the feature record is not active:

23: Remove the feature record from `Feature\_Points\_List`

24: End If

25: End For

26: For each `current\_Feature\_Point` in `M(t)`:

27: If `current\_Feature\_Point` is not in `last\_Feature\_Points\_List`, then:

28: Create a new `Feature\_Point\_Record`, storing the 2D and 3D coordinates

29: Append the new `Feature\_Point\_Record` to `Feature\_Points\_List`

30: End If

31: End For

32: End If

33: Increment `frame\_counter` by 1

34: If there is no next frame, then:

35: Break

36: End If

37: End While

38: For each feature record in `Feature\_Points\_List`:

39: Write the feature record to a CSV file

40: Close the CSV file