**Pseudo Code for data filtering:**

**Pseudo code of file APP.py**

1. Import necessary libraries:

- cv2 for image processing

- numpy for numerical operations

- os for file operations

- NIR for converting RGB images to infrared

2. Define function to extract frames from a video:

- Initialize an empty list to store frames

- Open the video file

- While the video is open:

- Read each frame

- If frame is not read successfully, stop reading

- Add the frame to the list

- Close the video file

- Return the list of frames

3. Define function to convert a frame to an infrared image:

- Use the NIR module to convert the RGB frame to an infrared frame

- Return the infrared frame

4. Define function to extract the red color spectrum from a frame:

- If the frame is not a numpy array, convert it to one

- Extract the red channel (color) from the frame

- Compute the frequency spectrum of the red channel using FFT

- Shift the zero-frequency component to the center of the spectrum

- Compute the magnitude of the spectrum

- Return the magnitude spectrum

5. Define function to calculate frequency from the magnitude spectrum:

- Compute the average frequency from the magnitude spectrum

- Return the frequency

6. Define function to store frequencies in a file:

- Open a text file to write frequencies

- Write the header "frequency"

- For each frequency in the list:

- Write the frequency to the file

- Close the file

7. Define the main function to process the video:

- Extract frames from the video file

- Initialize an empty list to store frequencies

- Convert each frame to an infrared frame

- For each infrared frame:

- Extract the red spectrum

- Calculate the frequency

- Add the frequency to the list

- Store the list of frequencies in a file

8. Set the path to the video file

9. Call the main function with the video path

**Pseudo Code for file NIR.py**

1. Import necessary libraries:

- cv2 for image processing

- numpy for numerical operations

- os for file operations

- glob for file pattern matching

- time for time-related operations

- multiprocessing for parallel processing

2. Define function to compute homography matrix between two images:

- Convert both images to grayscale

- Detect features and compute descriptors in both images

- Match features using a feature matcher

- Apply a ratio test to filter good matches

- If there are enough good matches:

- Extract points from the matches

- Compute the homography matrix using RANSAC

- Else:

- Print a message about insufficient matches

- Set the homography matrix to None

- Return the homography matrix

3. Define function to align images using the homography matrix:

- For each pair of RGB and CIR images:

- Read the images

- Extract and convert channels to RGB

- Align the channels using the homography matrix

- Convert aligned channels to grayscale

- Adjust the size of the channels to be the same

- Stack the channels to create RGB and CIR images

- Resize and align the CIR image again

- Align the CIR and RGB images

- If aligned images are None, continue to the next pair

- Crop edges of the aligned images

- Save the aligned RGB and CIR images

4. Define function to align two images using a homography matrix:

- Compute the homography matrix between the images

- If the homography matrix is None, return None

- Compute the perspective transform

- Apply the perspective transform to the second image

- Return the transformed image

5. Define function to process lists in chunks:

- Loop over the list in chunks of a specified size

- Yield each chunk for processing

6. Define function to crop edges of an image:

- Crop a specified number of pixels from each side of the image

- Return the cropped image

7. Define function to convert RGB frames to NIR frames:

- Convert each RGB frame to an NIR frame

- Return the list of NIR frames

8. Define function to convert a frame to an infrared frame:

- If the frame is grayscale, convert it to BGR

- Extract the green channel and apply a colormap

- Return the NIR colored frame

9. If the script is run directly:

- Import and run the APP module