Segmentation Segmentation is the process of assigning every pixel of an image to a region based on a predicate $P(h, w)$. The predicate is either true or false given the position of a pixel. For every created region, the predicate should be true.

1 Region uniformity

A region is uniform if the following predicate holds true for every pixel :math:‘(h, w)‘ in the region.

$$P(h,w) : \max_{(h,w)\in R} f(h,w) - \min_{(h,w)\in R} f(h,w) < \text{Threshold} \tag{1.1}$$

This condition is true when the difference between the maximum and minimum values of the in the region is less than a threshold. The threshold's value in this uniform predicate is constant for all regions in the image.

2 Region-grow

A region grow algorithm is seeded with some initial points provided by the user and grows every point into a region based on a predicate. The predicate in this exercise for region homogeneity is :ref:‘region uniformity<Region uniformity>‘. The resulting image has the pixel intestities replaced by the assigned region's mean.

- While true
  - For every pixel :math:‘f(h, w)‘ in the image, do
    - If pixel is a region's border
      - Check its 8 neighbourhood pixels.
      - If a neighbourhood pixel is not classified
        1. Compute the difference between the neighbouring pixel and the original pixel's region mean.
        2. If the difference is lower than the threshold
           (a) Add the neighbouring pixel to the region and to the region's border
           (b) Update the region's mean
      - Remove this pixel from the region's border
  - If no pixels have been classified in this loop
    - stop

Help The function that applies the region grow algorithm is `wrap_region_grow()`

- Tensor should be grayscaled
- Regions is an array of shape [frame, regionId, 3]. Last dimension holds: Number of pixels [NP], Sum of pixels in region [S], Mean of pixels in region [M] zero Id indicates unclassified pixel
- Initialize the regions array from the input points
- Loop through the algorithm while there are pixels assigned in the last iteration
- Return only the region ids [R] of the result tensor and the regions array
- Be sure that the sliced result tensor has the shape of a tensor

Help The function that applies the algorithm's loop is `region_grow_loop()`

- Result is a Tensor with two channels: (RegionId[R], BorderId[BR]). Zero Id indicates unclassified pixel
- Be sure to use absolute difference of intensities

3 Region-merge

This segmentation algorithm creates the region by merging every pixel to a prexisting region if the region stays region uniformity. At the start, the max number of available regions are given as input. The resulting image has the pixel intestities replaced by the assigned region's mean

- For every pixel $f(h, w)$ in the image, do
  - Check its 8 neighbourhood pixels.
  - If a neighbourhood pixel is classified
    - Find the neighbour whose region mean is the closest to the candidate pixel
    - If the difference is lower than the threshold
      - Add the candidate pixel to the region
      - Update the region's mean
  - If the candidate is not classified
    - Find an existing region whose mean is the closest to the candidate's intensity
    - If the difference is lower than the threshold
      - Add the candidate pixel to the region
      - Update the region's mean
  - If the candidate is not classified
    - Check if all available regions have been created.
    - If there is an unitialized region
      - Add the candidate pixel to the region
      - Update the region's mean

Help The function that applies the region grow algorithm is `wrap_region_merge()`

- Input tensor should be grayscaled
- Be sure to save the regions' mean values as floats
- Initialize the first region of the regions array with the first pixel
- Initialize the result of the regions array with the first pixel
- For every frame, height and width
- Save unclassified pixels to the 0-th region
- Return result tensor and regions array

4 Region-split

Region split algorithm, recursively splits the image in four sub images. The recursion stops when a homogenous sub-image is encountered. New regions will be created if needed. The resulting image has the pixel intestities replaced by the assigned region's mean

- Check if the image is uniform
- If it is not

  - Split the image in four sub-images
  - Rerun the algorithm
- Else
  - Create a new region and set its mean to be the contained pixels' mean.

Help The function that applies the region split algorithm is `wrap_region_split()`

- Create result tensor
- Create an nregs array for every frame and channel of input tensor
- Run the split for every pair of frame and channel
- Convert the result to uint8
- Return the result and the number of regions array

The recursive function that splits the frame is `region_split()`

- Be sure to continue the recursion if both height and width is geater than one
- Calculate halfs with integral division
- return the number of regions

5 Region-split-merge

This algorithm is a combination of splitting and merging algorithms.

- Check if the image is uniform
- If it is not
  - Split the image in four sub-images
  - Rerun the algorithm
- Else
  - Create a new region and set its mean to be the contained pixels' mean.
  - For all the neighbooring pixels outside this region, do.
    * If the neighbour is classified.
      · Find the neighbour whose region mean is the closest to this region's mean
      · If the difference is lower than the threshold
        1. Merge the two regions
        2. Update the merged region's mean

Help The function that applies the region split-merge algorithm to an input is `wrap_region_split_merge()`

- Create result tensor
- Create an nregs array for every frame and channel of input tensor
- Be sure to save the regions' mean values as floats
- Run the split for every pair of frame and channel
- Convert the result to uint8
- Return the result and the number of regions array

The recursive function of region split-merge algorithm is `region_split_merge()`

- Be sure to continue the recursion if both height and width is geater than one
- Calculate halfs with integral division
- Stop if buffer of regions is full
- Use region_split_merge_loop function for fast assignment
- Remember to check for available merging before exiting out of recursion
- Return the -1 if region buffer is full and the number of regions

The function combining two regions is `region_split_merge_c()`

- Calculate height and width edges with tensor's limits in mind
- Hold the merge label in a variable
- Run the search only if the edges are valid (eg. don't search outside of the tensor's limits)
- Be sure to avoid divisions by zero at mean value calculation
- Remember to decrement the number of regions if a merge occurs
- Return the number of regions and the region Id (0 if merge didn't occur)