Filtering A Frame is a either a 3 dimensional numpy array `[height, width, channel]` if it has more than 1 channel (BGR format). If it is grayscaled, it is a 2 dimensional array `[height, width]`

1 Prewitt filter

The Prewitt filtering algorithm is consisted of two filters:

$$f_1 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad f_2 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{1.1}$$

Each filter is applied to the source image and the absolute values are added together.
The function `prewitt()` applies the prewitt filtering algorithm to the input tensor.

Help

- Be sure to increate the size of the type
- Create a result tennsor
- For all frames, height, width and channel apply the filter
- Convert to result type to uint8

2 Sobel filter

The Sobel filtering algorithm is consisted of two filters:

$$f_1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad f_2 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.1}$$

Each filter is applied to the source image and the absolute values are added together.
The function `sobel()` applies the sobel filtering algorithm to the input tensor.

Help

- Be sure to increate the size of the type
- Create a result tennsor
- For all frames, height, width and channel apply the filter
- Convert to result type to uint8

3 Kirsch filter

The Kirsch filtering algorithm is consisted of four filters:

$$f_1 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} f_2 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} f_3 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} f_4 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix} \tag{3.1}$$

Each filter is applied to the each pixel of the source image. The result saved to the target pixel is the maximum absolute value of the four filters.
The function `kirsch()` applies the sobel filtering algorithm to the input tensor.

Help

- Be sure to increate the size of the type
- Create a result tennsor
- For all frames, height, width and channel apply the filter
- Convert to result type to uint8

4 Laplace filter

The Laplace filtering algorithm is computed as:

$$\nabla^2 f(h,w) \simeq f(h,w) - \frac{1}{4}\left[f(h-1,w) + f(h+1,w) + f(h,w-1) + f(h,w+1)\right] \tag{4.1}$$

It is the difference between the pixel's intensity and the mean of the 4 neighbourhood of the pixel.

The function `laplace()` applies the sobel filtering algorithm to the input tensor.

Help

- Be sure to increate the size of the type
- Create a result tennsor
- For all frames, height, width and channel apply the filter
- Convert to result type to uint8

5 Local Thresholding

The application of the previous filters in an image results in a grayscale output, where each pixel magnitude holds information about the existence of a pixel. If the magnitude is greater than a value, then we decide that an edge exists. So, a thresholding operation is required. In this exercise, a local thresholding method is applied, where the threshold value is local for each pixel. The computation of a threshold is:

$$T(h,w) = \bar{e}(h,w)(1+p) \tag{5.1}$$

$$\bar{e}(h,w) = \frac{1}{2M+1}\sum_{i=h-M}^{h+M}\sum_{j=w-M}^{w+M} e(i,j) \tag{5.2}$$

The function `local_threshold()` applies the local thresholding algorithm in an input tensor.

Help

- Create a result tennsor
- For all frames, height, width and channel
- Be sure to calculate local mean as float
- Convert to result type to uint8