Programming Assignment 1: Hough Transform

In this assignment, we will perform a Hough Transform for line detection. In particular, you will add code to p3.py to do the following:

Todos

1. Read an Image and convert it into a floating point numpy array with values scaled to be between 0 and 1
2. Create a gaussian filter of size k x k and with standard deviation sigma
    a. Produces a k x k gaussian filter with standard deviation sigma
    b. Make sure the gaussian filter is **normalized** (filter should sum to one)
    c. Assume k is odd
3. Compute the image gradient
    a. Convert the image to grayscale by using the formula:
        i. Intensity = Y = 0.2125 R + 0.7154 G + 0.0721 B
    b. Convolve with a 5x5 Gaussian with a standard deviation of 1 to smooth out noise
        i. Perform “same” convolution, i.e., output image should be the same size as the input
        ii. Use `scipy.signal.convolve` for this task
    c. Convolve with [0.5, 0, -0.5] to get the X derivative on each channel and convolve with [[0.5],[0],[-0.5]] to get the Y derivative
        i. Again, perform “same” convolution
        ii. Use `scipy.signal.convolve`
    d. Return the gradient magnitude (remember to use `np.sqrt` to get the square root) and the gradient orientation (use `np.arctan2`)
4. Compute the distance between pixels (x, y) and a line with equation $x\,cos(\theta) + y\,sin(\theta) + c = 0$. Return a boolean array that is True for pixels whose distance is less than a given threshold
    a. Compute the distance using the formula:
        i. $d = |x\,cos(\theta) + y\,sin(\theta) + c = 0|$
5. Draw lines on the image
    a. Make a copy of the image
    b. Set pixels that are within threshold distance from a set of given lines to have RGB value (1,0,0)
6. Hough voting
    a. You get as input gradient magnitude and the gradient orientation arrays, as well as a set of possible theta values and a set of possible c values. If there are T entries in thetas and C entries in cs, the output should be a T x C array. Each pixel in the image should vote for (theta, c) if:

        i. Its gradient magnitude is greater than thresh1
        ii. Its distance from the (theta, c) line is less than thresh2, and
        iii. The difference between theta and the pixel's gradient orientation is less than thresh3
    b. You may want to filter by indices that meet condition (i) first to reduce the number of computations necessary
7. Find local maxima in the array of votes. A (theta, c) pair counts as a local maxima if 1. Its votes are greater than thresh, and
    a. Its value is the maximum in a nbhd x nbhd neighborhood in the votes array.
    b. Return a list of (theta, c) pairs

Once you are done, running p3-demo.ipynb should produce the output shown in p3-demo.html

Installation instructions

For this assignment, you will need to install a number of tools including Anaconda, python3, and Jupyter. Alternatively, you can also install python3, VSCode, and the Jupyter extension for VSCode.

OPTION 1: Anaconda installation

Anaconda will come with an installation of Python and many important packages that we will use in this class.
You may install Anaconda at the following links:

- macOS
- Windows
- Linux

Be sure to install a version of Anaconda that supports python3.

Jupyter installation

Now that you have conda, installation of Jupyter is fairly simple.
You will need Jupyter notebook, which you can install using `conda install -c conda-forge notebook`.
After installing, you can navigate to the directory in which the notebook is contained, and run the command `jupyter notebook`.

OPTION 2: VSCode Setup

Install Python, VSCode, and install the Python and Jupyter extensions in VSCode.

Next, create a virtual environment to install the necessary Python packages using `python3 -m venv p3-venv`, and activate it by typing `source p3-venv/bin/activate` (for Mac/Linux) or `p3-venv\Scripts\activate.bat` (for Windows) in the terminal

Install matplotlib, numpy, and Pillow by running `pip install matplotlib`. This will also install numpy and Pillow as dependencies.

Rules of the game

Only modify functions with a TODO above it. Submit p3.py. No additional imports are allowed.

Potentially Useful Functions

You are not required to use these functions, but you may find them helpful.

Basic numpy functions (sum, zeros, array, arange, etc)

numpy.unravel_index()

- np.unravel_index [indices] [shape] creates an array of shape [shape], with elements numbered in increasing order, and returns the coordinates of the index where the index value matches an element of [indices]
- E.g. np.unravel_index([2,4], (2,3)) → y index: array([0, 1], dtype=int64), x index: array([2, 1], dtype=int64)
    - [[0, 1, 2], [3, 4, 5]], where the indices of 2 and 4 are (0, 2) and (1, 1)

numpy.where()

- Produces the indices of a numpy array where a condition holds

zip()

- If you use zip() with n arguments, then the function will return an iterator that generates tuples of length n

```
>>> numbers = [1, 2, 3]
>>> letters = ['a', 'b', 'c']
>>> zipped = zip(numbers, letters)
>>> list(zipped)
[(1, 'a'), (2, 'b'), (3, 'c')]
```