Canny Edge Detection

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a computational theory of edge detection explaining why the technique works.

We provide the baseline code and 'Iguana.bmp' image to PLATO. And Get and use the functions associated with gaussconvolve2d that you used in the last HW02.

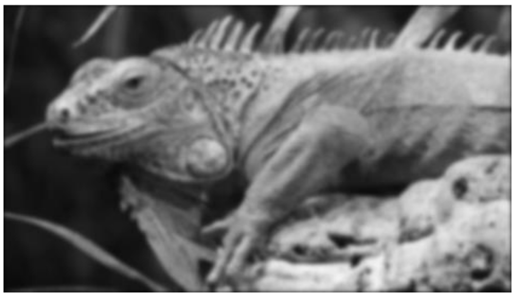
Hand in all parts of this assignment in PLATO (both the code and report PDF file as specified). To get full marks, your functions (i.e., *.py files) must not only work correctly, but also must be clearly documented with sufficient comments for others to easily use and understand the code. You will lose marks for insufficient or unclear comments. In this assignment, you also need to hand in scripts showing tests of your functions on all the cases specified as well as the images and other answers requested. The scripts and results (as screenshots or otherwise) should be pasted into a single PDF file and clearly labeled. Note that lack of submission of either the code or the PDF will also result in loss of points.

The assignment

1. **Noise reduction**

   (20 point) Load the image and switch to the grayscale image. It then blurs the image using the 'gaussconvolved2d(array,sigma)' function to eliminate noise. Use PIL to show both the original and filtered images.

   Original Image

   Filtered Image

   Note: The dtype of the image's numpy array must be used as np.float32. And use 1.6 for sigma value of gaussconvolved2d function.

2. **Finding the intensity gradient of the image**

   (20 point) Write a function 'sobel_filter(img)' that applies Sobel filter in the x,y direction and apply the convolve2d function to obtain the intensity x,y value.

+1	0	-1
+2	0	-2
+1	0	-1

X filter

-1	-2	-1
0	0	0
+1	+2	+1

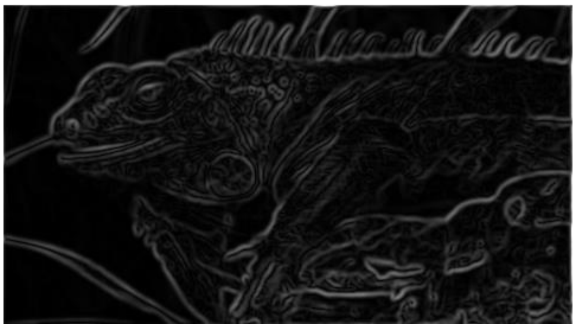
Y filter

X,Y Sobel filter

And apply the formula to obtain Gradient and Theta values. Note that gradient values should be mapped to values between 0-255. And the function must return both gradient and theta

$$G = \sqrt{I_x^2 + I_y^2} \ , \ \theta = \tan^{-1}\frac{I_y}{I_x}$$

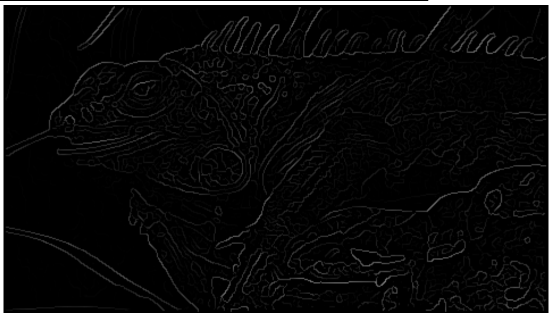
Show the results of applying sobel_filter Suppression.



HINT: You can easily get it by using the functions 'hypot' and 'arctan2' provided by Numpy.

3. Non-Maximum Suppression

(20 point) Write a function 'non_max_suppression(gradient, theta)' that extract the thin edge by applying Non-max-suppression according to the angle of the edge (0,45,90,135). Show the results of applying Non-Maximum Suppression.



Note: If the arctan2 function was used, the angle would be radians. And do it except for the edge part due to padding.

4. Double threshold

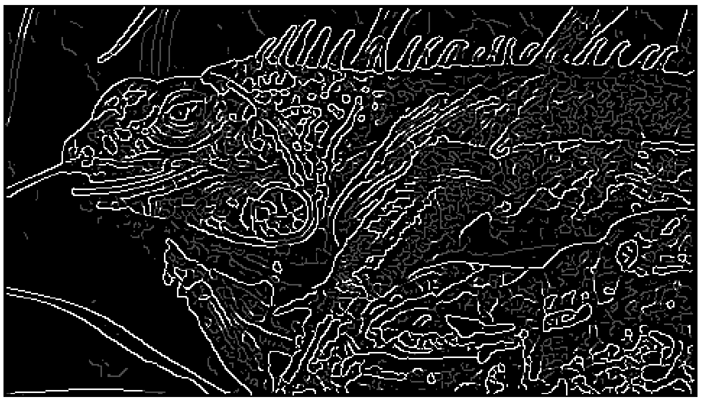
(20 point) Write a function 'double_thresholding(img)' that double threshold step aims at identifying 3 kinds of pixels: strong, weak, and non-relevant

1) Strong pixels are pixels that have an intensity so high that we are sure they contribute to the final edge.
2) Weak pixels are pixels that have an intensity value that is not enough to be considered as strong ones, but yet not small enough to be considered as non-relevant for the edge detection.
3) Other pixels are considered as non-relevant for the edge.

Use the following expressions to determine high and low threshold values.

$$diff = \max(image) - \min(image)$$
$$T_{high} = \min(image) + diff * 0.15$$
$$T_{low} = \min(image) + diff * 0.03$$

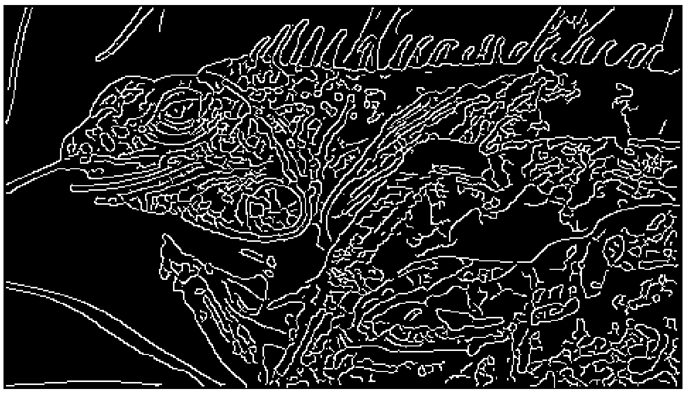
After classifying into three types, map non-related items to zero, map weak to 80, and map strong to 255. Show the results of applying Double threshold.



HINT: The np.where function makes it easy to calculate without using for loop.

5. Edge Tracking by hysteresis

(20 point) Write a function 'hysteresis(array)' that based on the threshold results, the hysteresis consists of transforming weak pixels into strong ones, if and only if at least one of the pixels around the one being processed is a strong one. Show the results of applying hysteresis.



HINT: Use DFS on all strong edges to obtain connected weak edges.

Note: If Hysteresis is performed without DFS, be careful because only the weak-edge in the progress direction is connected.

Deliverables

You will hand in your assignment in PLATO. You should hand in one zip file including two files, a file containing your code (i.e., *.py file). This file must have sufficient comments for others to easily use and understand the code. In addition, hand in a PDF document showing scripts (i.e., records of your interactions with the Python shell) showing the specified tests of your functions as well as the images and other answers requested. The PDF file has to be organized and easily readable / accessible.

Assignments are to be handed in before 11:59pm on their due date