Lab Assignment #5
Due Monday, May 9 by 11:59pm
(upload to CatCourses)

Title: Hough transform

In this lab, you will implement the Hough transform to detect the most prominent line in an image.

1) Carefully review the section on "Global processing using the Hough transform" in chapter 10 of the text. In particular, make sure you understand figure 10.32 in the 3rd edition or figure 10.29 in the 4th edtition. You should also review my course notes for the Hough transform from April 25 (they are available on CatCourses).

2) Write the function `hough_transform` that takes as input an edge image and determines the most prominent line using the method described in the text and in my notes. This function should output the parameters of the line in normal form: theta (θ), the angle the line makes with the vertical axis, and rho (ρ), the length of the perpendicular bisector of the line. It should also output the accumulator matrix.

   Your `hough_transform` function should have the following calling syntax:

   ```
   [theta_out, rho_out, accumulator] = hough_transform(i_edge)
   ```

   `i_edge` is an edge image which has value 0 for non-edge points and 255 for edge points.

   Here is a sketch of what your `hough_tranform` should do:

   a) Create an empty accumulator matrix. This matrix should have one column for each degree that theta can take for a line: -89 to 90. It should have one row for each value rho can take. Rho, the length of the perpendicular bisector, can range between –D and D where D is the diagonal size of the image. Thus, your accumulator matrix should have 2*D+1 rows and 180 columns.

   b) For every edge point in `i_edge`, plot the corresponding sinusoid in the accumulator matrix. That is, an edge point determines an x,y pair which in turn determines the relationship between rho and theta via the formula: x * cos(theta) + y * sin(theta) = rho. Let theta range over all possible values (-89 to 90) and compute rho. This determines which accumulator cell to add another "vote" to. Notes and warnings: we are working in

degrees so make sure you are using the correct cos and sin functions in Matlab. Also, remember the ranges of the indices of your accumulator matrix when indexing using the theta and rho computed above.

c) Once you have visited every edge point in your edge image, determine the cell with the most votes in your accumulator matrix. This should give you the theta and rho of the most prominent line in the image (again, remember the ranges of the indices).

3) Debugging your `hough_transform` function will be tricky so I'm providing some test scripts and results to help you. There are four scripts:

- test_horizontal_line.m
- test_vertical_line.m
- test_pos_diagonal_line.m
- test_neg_diagonal_line.m

I have provided my results from running these scripts with my `hough_transform` function on pages 3-6. You should make sure your results are similar. Note that my estimated parameters are not always exactly equal to the true values. But, your theta should be within 1-2 degrees and your rho should be within 1-2 pixels.

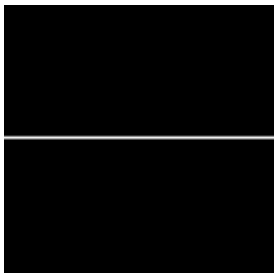
4) To prove to me that your `hough_transform` function works correctly, run the script test_random_line.m THREE TIMES and provide the results in a similar format to what I provided for the test scripts above. Page 7 of this lab contains an example of running it once with my `hough_transform` function. You should have three pages like this with different lines. Your theta should be within 1-2 degrees and your rho should be within 1-2 pixels of the true values.

5) Finally, run the script test_real_image.m to apply your `hough_transform` to the real image runway_image.tif. My results for this are shown on page 8. You don't need to turn in your results. I just want you to see that the technique actually works for finding lines in real images (assuming your function works correctly). Note that it took 10 or more seconds for test_real_image.m to run for me.

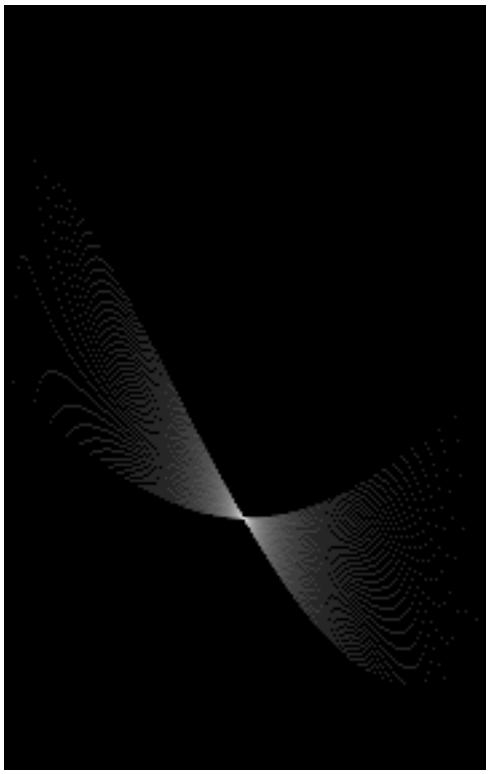
6) Your results section should discuss what you found to be the most difficult part of implementing the Hough transform algorithm. Your report should contain the three results of running the test_random_line.m script (formatted similar to mine). The only code you need to turn in is your `hough_transform` function.

My results of running test_horizontal_line.m

```
>> test_horizontal_line
True theta = 0, true rho = 50
Estimated theta = 0, estimated rho = 50
```



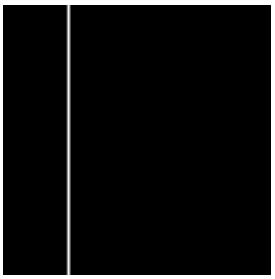
horizontal_line.tif



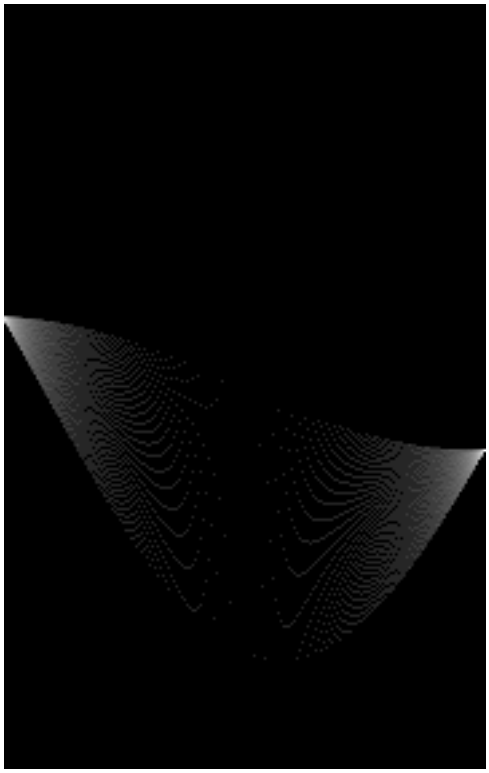
horizontal_line_accumulator.tif

My results of running test_vertical_line.m

```
>> test_vertical_line
True theta = 90, true rho = 25
Estimated theta = 90, estimated rho = 25
```



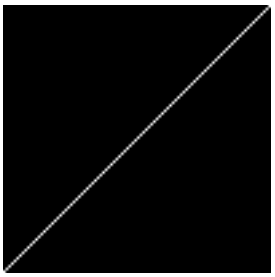
vertical_line.tif



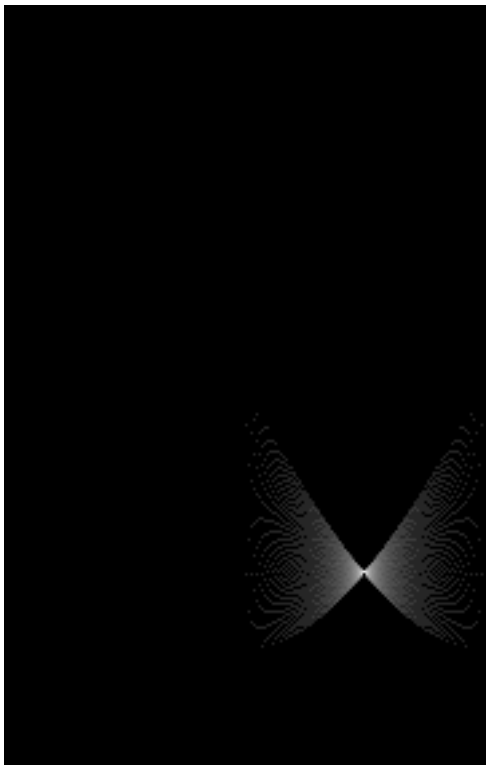
vertical_line_accumulator.tif

My results of running test_pos_diagonal_line.m

```
>> test_pos_diagonal_line
True theta = 45, true rho = 71
Estimated theta = 45, estimated rho = 71
```



pos_diagonal_line.tif



pos_diagonal_line_accumulator.tif

My results of running test_neg_diagonal_line.m

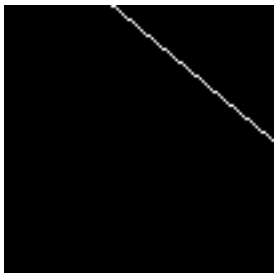
```
>> test_neg_diagonal_line
True theta = -45, true rho = -35
Estimated theta = -45, estimated rho = -36
```

neg_diagonal_line.tif

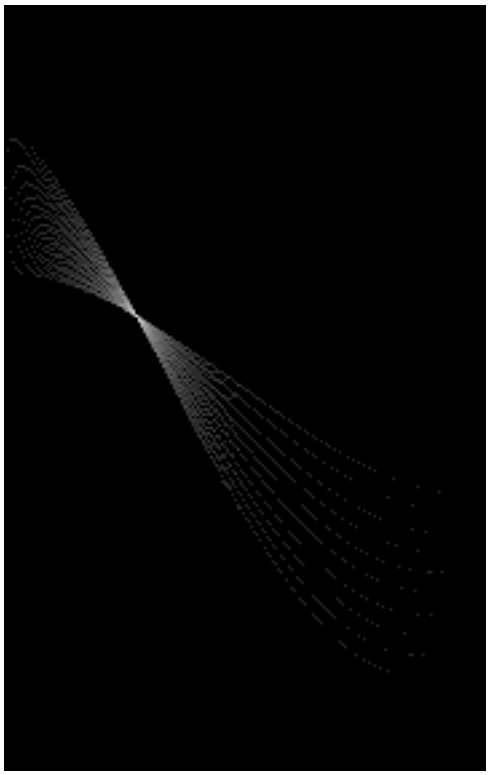
neg_diagonal_line_accumulator.tif

My results of running test_random_line.m

```
>> test_random_line
True theta = -40, true rho = -25
Estimated theta = -40, estimated rho = -26
```



random_line1.tif

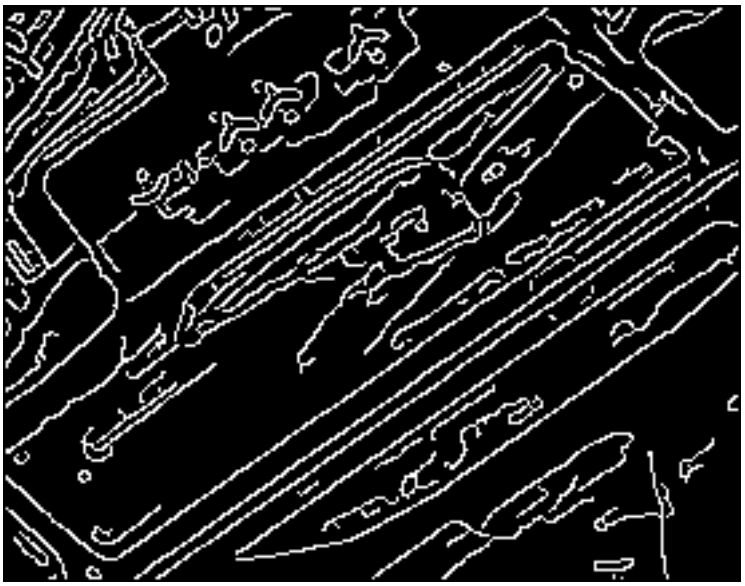


random_line1_accumulator.tif

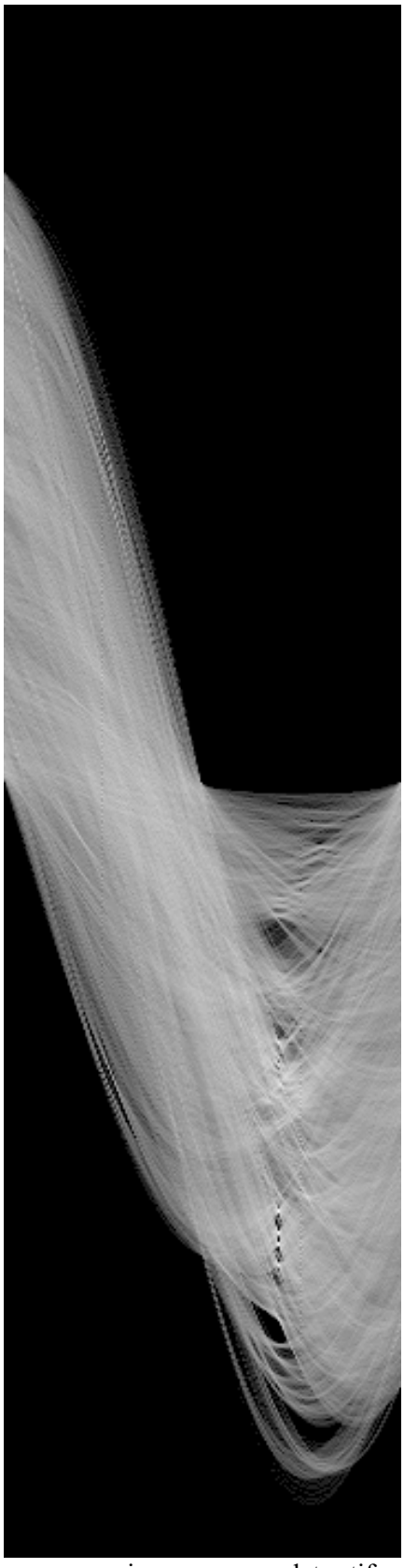
My results of running test_real_image.m

```
>> test_real_image
Estimated theta = 35, estimated rho = 220
```

runway_image.tif



runway_image_edge.tif



runway_image_accumulator.tif

runway_image_with_line.tif