# CS6640 – Project 5 Hough Transform for Line Detection
Assigned Wed Nov. 17, 2010
Due Wed Dec. 1 (Just before midnight)
Instructor: Guido Gerig
TA: Miaomiao Zhang

Required Readings: Book Gonzales Woods Chapter 10.2
Course notes GG
Use of Hough Transformation to detect Lines ..., Duda and Hart

## I. Theoretical Problems

### Problem 1: Choice of parametrization

We will use the Hough Transform to recognize straight lines in contour images. Explain why the polar coordidnate representation of lines, $\rho = x\cos\theta + y\sin\theta$, is more suitable for line detection than the standard line representation $y = ax + b$.

### Problem 2: HT as a point to curve transformation

Show that points that lie on the same curve $\rho = x_0\cos\theta + y_0\sin\theta$ in parameter space with axes $(\rho, \theta)$ all represent straight lines in image space which intersect at a specific image point $(x_0, y_0)$.

### Problem 3: Edge direction information

Explain why a priory information about edge directions may increase the speed of Hough transform-based image segmentation. Hint: Best is if you make a numerical example where N points in image space are transformed into N curves in parameter space, sampled as curves with M sample points. Estimate the number of operations with and without edge direction information, and make a sketch to explain.

### Problem 4: Compensation for errors

Given a straight line $g_0$ passing through two opposite corner points of a quadratic image (upper right to lower left). The quantization of cells in parameter space is given by $\Delta\rho$ and $\Delta\theta$. We would like to get a feeling for the error region in image space (region for all points in image space) which contributes to the cell centered at $(\rho_0, \theta_0)$ representing the straight line $g_0$. You can think of a cell in parameter space centered at $(\rho_0, \theta_0)$ with widths $\Delta\rho$ and $\Delta\theta$, i.e. widths $\pm\frac{\Delta}{2}$ for $\rho$ and $\theta$. You may remember (look at the Java demos) that the four corners of this cell represent four lines in image space defining the boundaries of an "error" region.

a) Do the calculations b) and c) for putting the image origin (0,0) lower left and also for putting the image origin at the center of the image. Discuss the differences.

b) Points in the *error* region contribute to our parameter cell, not just the central line $g_0$. Which is the longest straight line piece in the image orthogonal to $g_0$ that could *erroneously* contribute with its full length to the parameter cell centered at $(\rho_0, \theta_0)$? Hint: Calculate intersections of the bounding lines of the *error* region with the image boundary, sketch the situation, and use this to find your solution.

c) Calculate a numerical example for an image size of 512x512 pixels and two parameter space quantizations with $(\Delta\theta = 1^o, \Delta\rho = 2$ pixels), and $(\Delta\theta = 5^o, \Delta\rho = 5$ pixels).

# I. Practical Problem

## Problem 3: Hough Transform for straight lines

Write a program that calculates the Hough Transform for straight lines using the parametric form $\rho = x\cos\theta + y\sin\theta$. (Hint: Use the result of Theoretical Problem 2 to decide where to put the origin most efficiently).

Each point in image space generates a cos-curve which is accumulated in parameter space. The increment can be chosen as 1 for each point, but could also be proportional to the edge-strength which is the brightness attribute for each contour point (see contour images described below), so that the accumulation reflects the *importance of edge points*.

You might use the test images provided with this project, the simulated scene containing edges and lines "edges-lines" and the airport scene.

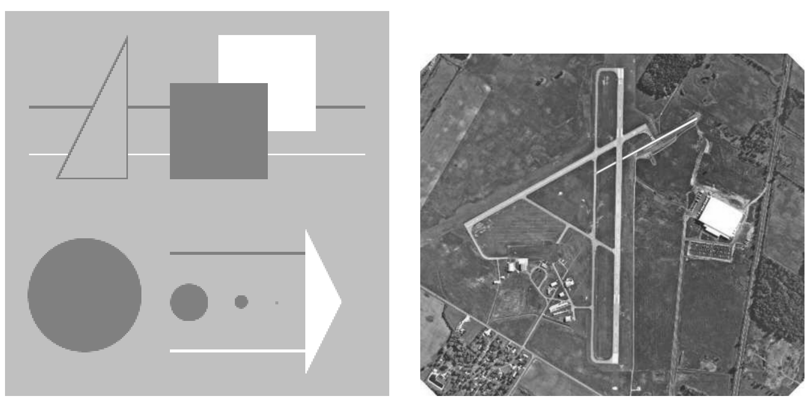

Figure 1: Example of images presenting straight line features.

a) Images need to be preprocessed with edge detection, thresholding and non-maximum suppression to get a small set of thin edge candidate pixels.

b) Implement the Hough-Transform and create an accumulator that corresponds to a 2-D image array with axes $\theta$ and $\rho$. Choose the cell size $\Delta\rho$ and $\Delta\theta$ as a parameter in your program for doing your own tests.

c) Detect the N largest maxima and write the pairs $(\rho_j, \theta_j)$ to a file for control.

d) Calculate the corresponding N straight lines and draw these lines back into the image space for comparison with the original image.

## Instructions, Requirements, and Restrictions

1. Please use your name **"NAME"** in all reportts and submitted materials to uniquely identify your projects.

2. Write your project code in a single directory, called `project1-NAME`.

3. For Matlab each individual function (including functions you define) should be a ".m" file, and your functions should call one another as necessary.

4. We **do not allow to use Matlab toolbox functions** (e.g. Imaging Toolbox) or other existing image processing libraries in order to give all students the same conditions for code development [1].

5. You should have in your report a short description of each algorithm you used and documentation on how your code is organized. Failure to do this will result in a loss of points. Please remmember to **add your name** to the report title.

6. **You can use submit your report as html or pdf document** (created from Word, LaTeX or other word processing system of your choice.

   - Using html: Your project report will be in the form of an html file called `index.html`, contained in that directory. All links from that file must be relative and all other files necessary to read your report must be in that directory (or subdirectories).

7. You should use examples of images in your report. They should be viewable in the browser when we open your html file.

8. You will submit a single tar file created from from your project directory with the unix command *tar -czf project1-NAME.tgz./project1-NAME*.

9. Please remember or look-up the honor code and requirement to provide your own solution as discussed in the syllabus.

10. **Please look up the late policy as defined in the syllabus**

[1] The core MATLAB package comes with several rudimentary functions that can be used to load, save, and perform custom functions on images. Taken from wikibooks