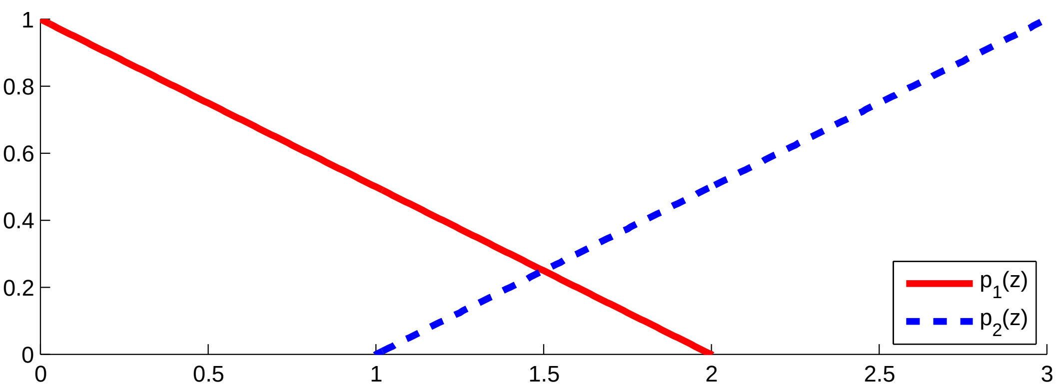
Homework #4
Due Th. 10/20

Be sure to show all your work for credit. You must turn in your code as well as output files (**code attached at the end of the report**).

Please generate a report that contains the code and output in a single readable format using Latex.

0. Getting Started
   - Download the homework images from the class website.
     `http://www.ee.unlv.edu/~b1morris/ecg782/hw/hw04`
1. (GW 10.6)
2. (GW 10.22)
3. (GW 10.36)
4. Thesholding

   Suppose an image has the gray-level pdf shown below. $p_1(z)$ corresponds to objects and $p_2(z)$ to background. Assume $P_1 = P_2$, find the optimal threshold between object and background pixels. Be sure to derive the optimal value mathematically to get the optimal value.



5. Canny Edge Detection
   (a) Give the convolution kernels for determining the gradient. You may examine the function `gradient.m` to help with the explanation. (It may be easiest to apply the `gradient` to an impulse and inspect the results).
   (b) Implement the simplified version of the Canny edge detector (no hysteresis thresholding). The syntax of the function should be

   `[E,M,A]=canny(I,sig,tau),`

   where `E` contains the detected edges, `M` the smoothed gradient magnitude, `A` contains the gradient angle, `I` is the input image, `sig` is the σ parameter for the smoothing filter, and `tau` is a single threshold.

(c) Apply your Canny detector on `wirebond_mask.tif` using $\tau = 0.8$ and 0.6 with the following values for $\sigma^2 = [0.5, 1, 3]$. Show your results in a (2,3) subplot. Invert the color, white for 0 and black for 1, to save ink. Discuss how the choice of σ affects the results.

(d) Apply your Canny detector on `city.jpg`. Adjust the σ and τ parameters as you see fit. Display the resulting edges and the parameter settings used. Also use the built-in Matlab function `edge.m` with default parameters on the `city.jpg` image. Compare.

(Bonus) Modify your Canny detector to implement hysteresis thresholding with `tau`$= [\tau_h, \tau_l]$.

6. Hough Transform

(a) Study the Matlab function `hough.m`. Compute the Hough transform of the `city.jpg` image. Display the Hough accumulator image and the original image with the top 5 lines as an overlay. Each overlay line should be a different color and a legend should be included.

(b) Write your own Hough transform implementation for circle detection. The function should take an image and radius as input. Test your function on the `quarters.bmp` image. Display the accumulator image and the original image with the top 3 circles as an overlay.

(Bonus) i) Use gradient magnitude accumulation instead individual count. ii) Upgrade the detector to find circles of different sizes. You may test results on `us_silver_coins.jpg`.