CMSC 426 Homework 6 Due May 4 Spring 05

1. (25 points) Suppose we are trying to distinguish cats from dogs and the only measurement we have is the weight of the animal. Let the prior probabilities for the animal being a cat or a dog be P(CAT) = 0.3, P(DOG) = 0.7
   and let the class conditional density functions for (integer-valued) weight given class be uniformly distributed as follows:
   –P(Weight|CAT) is uniform on [20, 39]
   –P(Weight|DOG) is uniform on [30,59].

   Explain how Bayes' rule can be used to classify an unknown animal having weight 35 pounds given the prior and conditional probabilities defined above.

2. (60 points) A geometric transform is a vector function **T** that maps the pixel *(x, y)* to a new position *(x0, y0)*. **T** is defined by its two components

   $$\mathbf{x0} = \mathbf{T}_x(x, y) \qquad \mathbf{y0} = \mathbf{T}_y(x, y)$$

   Geometric transformations of images are usually implemented in two steps. First for a point (x0, y0) we find from the inverse transform $\mathbf{T}^{-1}$ the corresponding point (x, y). Second, since x and y are not integers, we need to estimate the brightness value at (x, y) by interpolation (from neighboring integer points).

   Explain why you think the transforms are implemented in this way.

   Develop programs for the following geometric transforms:
   (a) Rotation
   (b) Change of scale
   (c) Affine transform calculated from three pairs of corresponding points.
   Recall an affine transform is defined as

   $$x_0 = a_0 + a_1x + a_2y \qquad y_0 = b_0 + b_1x + b_2y$$

   For each of the above transforms, implement the following two brightness interpolation approaches:
   • Nearest-neighbor interpolation
   • Bi-linear interpolation (from four neighboring points).

   Run your algorithms on a picture of your choice. Print the code. Print your results for
   (a) Scaling by a factor 3
   (b) An affine transform with points A = (1, 0),B = (−1, 0),C = (0,p3) mapping to points A0 =(1.9, 0),B0 = (−0.5, 0),C0 = (0, 1) using bi-linear interpolation.

3. (65 points) Implement the Lukas-Kanade optical flow algorithm using the implementation described in class. The two images are posted online. Compute the normal flow using the following steps:
   a. The images are filtered with a spatial- Gaussian filter, with standard deviation _ = 1.5 and kernel size 11×11.
   b. The spatial derivatives are computed using the 5- point symmetric kernel (−1, 8, 0,−8, 1)/12
   c. Estimation of the normal flow at points with high spatial gradient (experiment with the threshold)
   d. Estimate the optical flow from the normal flow values with weighted least squares minimization. (solve using the SVD)
   Print your code. Plot the estimated normal flow and optical flow field. (Explore how you plot flow vectors in matlab)