CSE166 – Image Processing – Homework #7
Instructor: Prof. Serge Belongie
`http://www-cse.ucsd.edu/classes/fa06/cse166`
Due (in class) 11:00am Thursday Nov. 30, 2006.

General Homework Guidelines

- Use the Cover Sheet provided.
- Please attach all code that you use. Attach code at end of submission.
- In general try to keep you answers concise. Use as many words as you need and no more. Also work on your presentation skills. This means organize your plots and displays. Always use titles and add captions when appropriate. *Points will be awarded for clarity and presentation.*

Reading

- GW 10.2.2 and 11.4.
- GW Review Material Ch. 1, "A Brief Review of Matrices and Vectors."

Written exercises

1. GW, Problem 10.13.
2. GW, Problem 10.14.
3. GW, Problem 11.17.
4. GW, Problem 11.18.
5. Consider the 2×2 matrix
$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$
Show that the inverse is given by
$$A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

Matlab exercises

1. Hough Transform.
   (a) Implement the Hough Transform (HT) using the (ρ, θ) parameterization as described in GW Section 10.2.2. Use accumulator cells with a resolution of 1° in θ and 1 pixel in ρ.
   (b) Produce a simple 11×11 test image made up of zeros with 5 ones in it, arranged like the 5 points in GW Figure 10.20(a). Compute and display its HT; the result should look like GW Figure 10.20(b). Now threshold the HT to find the (ρ, θ)-coordinates of cells with more than 2 votes and plot the corresponding lines in (x, y)-space on top of the original image.
   (c) Load in the matchstick image in GW Figure 8.02(a) and shrink it to half its size using `I=imresize(I,0.5,'bil','crop');`. Compute and display its edges using the Sobel operator with default threshold settings, i.e. `BW=edge(I,'sobel');`. Now compute and display the HT of `BW`. As before, threshold the HT and plot the corresponding lines atop the original image; this time, use a threshold of 50% of the maximum accumulator count over the entire HT.

(d) Repeat the previous step for another image of your choice. The image can be from the textbook or elsewhere, but its size must be at least 128×128 and it should contain several extended straight lines.

Things to turn in:

- Code listing for part 1a.
- Code listing for generating results in parts 1b, 1c, and 1d.
- Printouts of program output for parts 1b, 1c, and 1d.

2. Lucas-Kanade optical flow.

(a) Implement the Lucas-Kanade algorithm for measuring optical flow, as described in class. Allow the user to specify the size of the window used in enforcing the smoothness constraint. Use the `quiver` function to display the optical flow vectors. In addition, have your program return the two eigenvalues of the windowed image second moment matrix at each pixel.

(b) Construct two frames of a simple motion sequence as follows. Make a 16×16 white square centered on a black background of size 32×32. Blur it with a Gaussian filter with $\sigma = 1$. This image represents $I(x, y, t)$. Produce the second image, representing $I(x, y, t+1)$, by displacing the first image down one pixel and to the right one pixel. Display each frame, as well as I_t and the two components of ∇I.

(c) Compute and display the optical flow for the above sequence using a window size of 5×5. Since you know the "ground truth" displacement (i.e. $u = 1, v = 1$), comment on the accuracy of your measured optical flow at various points throughout the image. Demonstrate how, by applying a threshold on the eigenvalues, you can suppress the flow vectors at pixels that suffer from the aperture problem.

(d) Construct a new sequence consisting of the original first frame and a second frame produced by rotating the first one by 5° (use `imrotate` with the `'bil'` and `'crop'` options). Now repeat step 2c using this sequence.

Things to turn in:

- Code listing for steps 2a, 2b, 2c, and 2d.
- Program output for steps 2b, 2c, and 2d.
- Written comments for steps 2c and 2d.