

**Computer Vision
CSE 252A, Winter 2005
David Kriegman**

**Assignment 4
Stereo**

(Due date: Thursday, 3/10/05)

In this assignment, you will implement three different algorithms for binocular stereo vision. Your data will be pairs of stereo images that you can download from http://www.ijrr.org/contents/20_07/abstract/banks/1196-1.html

You will hand in results for two stereo pairs from that page: CORRIDOR and TREES. You can also find other stereo pairs at that sight to test out your algorithm.

Both pairs of images have been rectified so that the scan lines are epipolar lines. For each of these image pairs, you will ultimately seek to output a disparity map for the left image indicating the disparity to the right image.

Part I: Window Based Stereo

(30 pts.)

Part 1.A: Matching using differing window sizes and metrics

(30 pts.)

To get the disparity value at each point in the left image, you will search over a range of disparities, and compare the windows using two different metrics mentioned in class: Sum of Absolute Differences (SAD), Sum of Squared Differences, and Normalized Correlation. Pick two out of these three. Do this for windows of size w by w where $w=1, 5$ and 9 . (Note, for normalized correlation, you can't just use a window size of 1, so use $w=3$ for this case). Restrict the possible disparity values to the integers and choose reasonable values for the minimum (possibly negative) and maximum disparity from an examination of the images and your results.

Note: When coded as four nested loops in Matlab, this can be painfully slow. Be creative about how you code this. If you can't get it to run quick enough to get results in the end, you can choose to recode in another language like C (It's simple enough, but it's up to you to figure out how to read the images in).

What to hand in: For each pairs of images, each window size and each metric, show the disparity image. (You should show 12 disparity maps).

Part I.B: Using Filtered Images

(10 pts.)

Prior to matching, filter the two images with a Laplacian of Gaussians (LoG) filter. See text for this filter kernel. You can use `conv2` for this. Based on your results in part from Part, you will need to choose values for the Gaussian variance for the LoG filter, the window size in matching, and the match metric.

Show your best result for the two test images (using the same parameters for both images), and describe any issues in choosing the parameters.

Part I.c: Sub-pixel matching

(10 pts.)

For one of the two images and for a good setting for parameters (you choose them), implement a method for performing sub-pixel matching at a resolution of $\frac{1}{4}$ pixel. If your code runs too slowly to try this on a full image, you may select an interesting sub-section of the image.

In your report, show results for the two image pairs.

Part II: Stereo with Dynamic Programming

(This part of the assignment is a variant of one from David Jacobs at Maryland).

Here, you will implement a stereo algorithm that uses dynamic programming. This algorithm enforces the ordering constraint (or else you couldn't use dynamic programming), and matches individual pixels. Every pixel in one image can either match one pixel in another image, or be marked as occluded. The penalty for matching two pixels is the square of the difference in their intensity. The penalty for being occluded is a fixed value, *occlusion_penalty*. You should consult the paper: "A Maximum Likelihood Stereo Algorithm", by Cox, Hingorani, Rao, and Maggs, from the journal *Computer Vision and Image Understanding*, 63, 3, pp. 542-567. This is available from the course web page. This part closely follows the method in that paper. ***Please note that for consistency with that paper, all thresholds below are given assuming that image intensities fall in the range 0 to 1.***

Part II.A: **(30 points)** Implement the dynamic programming stereo algorithm in 1D. You should write a function of the form `d = stereo_1d(v1, v2, oc)`, where `v1` and `v2` are vectors to be matched, and `d` contains the disparity for every pixel in `v1`. Test this for, `oc = .01`,

```
v1 = [1 0 1 1 0 1 1 0 0 1 1 0 1 1 1];  
v2 = [1 0 1 0 1 0 1 1 0 0 1 1 1 1 1];
```

Turn in your code and the result of this test.

Part II.B: (20 points) Extend this algorithm to 2D. We assume the images are rectified, so that the epipolar lines are horizontal lines. You just need to run `stereo_1d` on each

corresponding line in the two images. This will have the form: $D = \text{stereo_2d}(V1, V2, oc)$. Test your algorithm on the two pairs of images with $oc = .01$. Display the resulting disparity map as an image, with `imagesc(D)` or a similar function. To do this, you must be sure that occlusions in the disparity map are encoded as numbers that will display in a way that is distinct from any valid disparity. For example, if you expect disparities to range from -20 to 20 , you might set occlusions to be -40 . You may need to play with this a little, for example, multiplying D by some constant, so the individual disparities display clearly (You can also use a different color to display occlusions).

Turn in your code and an image showing the disparity map.

What to turn in

1. Email me the code
2. Hand in a report which contains
 - a. Code listing
 - b. Report which includes the output described above. Also, include a short description of your conclusions based on your experience.