Overview

In this assignment you will implement an algorithm 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: ARROYO and TSUKUBA. You can also find other stereo pairs at that site 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.

Stereo with Dynamic Programming

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. You should focus on section 2.1, which covers dynamic programming (the rest of the paper may be useful, but you don’t need to implement their maximum likelihood cost function or anything of that nature). 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 A: 1D Disparity Matching (7 points)

Implement the dynamic programming stereo algorithm in 1D. You should write a function of the form

\[ d = \text{stereo}_1d(v1, v2, \text{occlusion}\_\text{penalty}) \]

where v1 and v2 are vectors to be matched, and d contains the disparity for every pixel in v1. Test this for,
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];

The occlusion penalty may need to be adjusted for best results\(^1\), but a good starting point is to use

```
occlusion\_penalty = .01
```

Turn in your code and the result of this test.

**Part B: 2D Disparity Matching (3 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: \( D = \text{stereo}\_2d(V1, V2, \text{occlusion}\_\text{penalty}) \). Test your algorithm on the two pairs of images and 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 -5 to 40, you might set occlusions to be 80. 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.

**Part C: Different Cost Metrics (5 points)**

Three cost metrics mentioned in class were Sum of Absolute Differences (SAD), Sum of Squared Differences (SSD), and Normalized Correlation / Normalized SSD. Modify your stereo algorithm to use each of these metrics with window sizes of 5x5 and 9x9 (the metric used above is SSD with a window size of 1x1). To speed up your code, you can choose reasonable values for the minimum and maximum disparity (you should be able to roughly determine this by looking at the input images).

- Note 1: You may need to 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).
- Note 2: You may need to use different occlusion penalties for each cost function.

What to hand in: For each pair of images, each window size and each metric, show the resulting disparity image. (A total of 6 disparity maps).

**What to turn in**

1. Email Neil the code (nalldrin AT cs.ucsd.edu)
2. Hand in a report which contains
   (a) Hardcopy of code
   (b) Report which includes the output described above. Also, include a short description of your conclusions based on your experience.

\(^1\)If you use a different occlusion penalty, please note this in the writeup.