Assignment #3

(Due date: 11/18/2010 11/23/2010)

1. Epipolar Geometry [3pts]

Consider two cameras whose image planes are the z=1 plane, and whose focal points are at (-15, 0, 0) and (15, 0, 0). We’ll call a point in the first camera (x, y), and a point in the second camera (u, v). Points in each camera are relative to the camera center. So, for example if (x, y) = (0, 0), this is really the point (-15, 0, 1) in world coordinates, while if (u, v) = (0, 0) this is the point (15, 0, 1).

\[ \begin{align*}
(x, y) &= (0, 0) \\
(u, v) &= (0, 0)
\end{align*} \]

\[ z=1 \]

![Diagram](image.png)

- (a) Suppose the points (x, y) = (6, 6) is matched with disparity of 5 to the point (u, v) = (1, 6). What is the 3D location of this point?
- (b) Consider points that lie on the line \( x + z = 0, y = 0 \). Use the same stereo set up as before. Write an analytic expression giving the disparity of a point on this line after it projects onto the two images, as a function of its position in the right image. So your expression should only involve the variables \( u \) and \( d \) (for disparity). Your expression only needs to be valid for points on the line that are in front of the cameras, i.e. with \( z > 1 \).

2. Epipolar Rectification [3pts]

In stereo vision, image rectification is a common preprocessing step to simplify the problem of finding matching points between images. The goal is to warp image views such that the epipolar lines are horizontal scan lines of the input images. Suppose that we have captured two images \( I_A \) and \( I_B \) from identical calibrated cameras separated by a rigid transformation \( B T_A = R t \). From the lecture (this Tuesday), a rectifying transform for each image should map the epipole to the point infinitely far away in the horizontal direction \( H_A e_A = H_B e_B = [1, 0, 0]^T \). Consider the following special cases:
- (a) Pure horizontal translation \( t = [tx, 0, 0]^T \), \( R = I \)
- (b) Pure translation orthogonal to the optical axis \( t = [tx, ty, 0]^T \), \( R = I \)
- (c) Pure translation along the optical axis \( t = [0, 0, tz]^T \), \( R = I \)
- (d) Pure rotation \( t = [0, 0, 0]^T \), \( R \) is an arbitrary rotation matrix
For each of these cases, determine whether or not epipolar rectification is possible. Include the following information for each case
- The epipoles \( e_A \) and \( e_B \)
- The equation of the epipolar line \( l_B \) in \( I_B \) corresponding to the point \([x_A, y_A, 1]^T\) in \( I_A \) (if one exists)
- A plausible solution to the rectifying transforms \( H_A \) and \( H_B \) (if one exists) that attempts to minimize distortion (is as close as possible to a 2D rigid transformation). Note that the above 4 cases are special cases; a simple solution should become apparent by looking at the epipolar lines.
One or more of the above rigid transformations may be a degenerate case where rectification is not possible or epipolar geometry does not apply. If so, explain why.

3. Dense Stereo Matching with Dynamic Programming [10pts]
In this assignment you will implement an algorithm for binocular stereo vision. Your data will be pairs of stereo images that are available on the course website. You will hand in results for the three stereo pairs from that page. The images have been rectified so that the scan lines are epipolar lines. For each image pair, you will ultimately seek to output a disparity map for the left image indicating the disparity to the right image.

You can also test your algorithm on other stereo pairs available on the internet; the following websites contain suitable images,
http://vision.middlebury.edu/stereo/data/
http://vasc.ri.cmu.edu/idb/html/stereo/index.html

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, \( c_{occ} \).

In addition to the class lecture notes, 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 (available from the course web page). You should focus on section 2.1, which covers dynamic programming; the pseudo-code provided in figures 2 and 3 will provide a good starting point for your own code\(^1\). Please note that for consistency with that paper, all thresholds below are given assuming that image intensities fall in the range 0 to 1.

3.1. Part A: 1D Disparity Matching
Implement the dynamic programming stereo algorithm in 1D. You should write a function of the form

\[
d = \text{stereo}_1d(v1, v2, c_{occ})
\]

where \( v1 \) and \( v2 \) are vectors to be matched, and \( d \) contains the disparity for every pixel in \( v1 \). For this assignment, we will define disparity such that if \( v1(i) \) corresponds to \( v2(j) \), then the disparity is given by \( d = i - j \). If \( v1(i) \) has no corresponding pixel in \( v2 \), then this is considered an occluded pixel. You should write your code such that the “disparity” of any occluded pixels are set to \( NaN \). Test your function on the following input,

\(^1\) The rest of the paper may be useful, but you don’t need to implement or understand their maximum likelihood cost function or anything of that nature.
v1 = [71 5 9 4 3 0 8 0 2 6];
v2 = [71 0 5 0 9 4 3 8 2 6];
c_occ = .01;

What to turn in: Your code and the result of this test.

3.2. Part B: 2D Disparity Matching

Extend this algorithm to 2D. We assume the images are rectified, so that the epipolar lines are horizontal scan lines of the input images. Thus, you just need to run stereo 1d on each corresponding line in the two images. Your function should have the form: \( D = \text{stereo 2d}(II, I2, c\ occ) \). Test your algorithm on the three image pairs (DOTS, BARN1, TSUKUBA) using an occlusion penalty of \( c\ occ = 0.01 \) and display the resulting disparity maps as images. For display purposes, the disparity values should be mapped to the range \([0, 1]\) and, to distinguish valid disparities from occlusions, you should colorize the image so that occluded pixels are shown in color while the rest of the disparity map is shown in grayscale. The following code will scale the values appropriately and show occlusions in blue,

```matlab
function [d_color] = display_dmap(d)
% Map disparity into the range [0, 1]
max_d = max(d(:)); min_d = min(d(:));
d_scaled = (d-min_d) / (max_d-min_d);

%Colorize occluded pixels to be blue
d_color = repmat(d_scaled(:),[1 3]);
occ_inds = isnan(d(:));
d_color(oocc_inds,1) = 0;
d_color(oocc_inds,2) = 0;
d_color(oocc_inds,3) = 1;
d_color = reshape(d_color,[size(d) 3]);

% Display image
image(d_color);
```

What to turn in: Your code and an image showing the resulting disparity map for each input image pair (DOTS, BARN1, TSUKUBA).

3.3. Part C: Different Cost Metrics

Up to this point, we have used the squared difference in pixel value as our matching cost. In this section we will extend our definition of matching cost to include neighborhoods of pixels. We will also examine two different measures of matching cost: Sum of Squared Differences (SSD), and Normalized Sum of Squared Differences (NSSD). These cost metrics are defined in Table 1.

Modify your stereo algorithm to use each of these match costs. To illustrate the process, suppose we wish to compute the SSD match cost between pixel \((x_i, y)\) in image 1 and pixel \((x_j, y)\) in image 2. We first extract a window \(W_i\) centered at pixel \((x_i, y)\) from image 1. We then extract a window \(W_2\) centered at pixel \((x_j, y)\) from image 2. Plugging into the formula for the SSD cost we get 
\[
c = \sum_{i,j} \left| W_i(i,j) - W_2(i,j) \right|^2.
\]
Evaluate your algorithm using the SSD, and NSSD match costs with window sizes of 3x3 and 5x5 (4 disparity maps in total). The occlusion penalty you should use for each match cost is also provided in Table 1.

Since evaluating these cost functions can be computationally intensive, you may find it helpful to
optimize your implementation to get acceptable run-times. The following list of suggestions may be of help.

- Choose reasonable values for the minimum and maximum disparity (you should be able to roughly determine this by looking at the input images) and only evaluate the match cost for pixels in this disparity range.
- Think of ways to compute the match cost in batch (i.e., without using for loops).
- Take advantage of Matlab’s profiling tool (available under Desktop → Profiler in the menu). This will isolate the slowest parts of your code.
- Implement the algorithm in C (only if you are unable to get acceptable performance via other means – for comparison, the reference implementation takes about 3 minutes on the Tsukuba pair using NSSD and with a 5x5 window).

Implementing the first suggestion is probably sufficient to make your algorithm run in a reasonable amount of time. A third metric, Normalized Correlation (NCC) is also defined in the table. Show that maximizing the NCC is equivalent to minimizing the NSSD. Hint: Express $c_{\text{NCC}}$ and $c_{\text{NSSD}}$ in terms of the vectors $\tilde{w}_1 = \tilde{W}_1(:,::)$ and $\tilde{w}_2 = \tilde{W}_2(:,::)$. Thus, $c_{\text{NCC}} = \tilde{w}_1^T \tilde{w}_2$ and $c_{\text{NSSD}} = (\tilde{w}_1 - \tilde{w}_2)^T (\tilde{w}_1 - \tilde{w}_2)$

<table>
<thead>
<tr>
<th>Match Cost Function</th>
<th>Recommended $c_{\text{occ}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD $\sum_{i,j}</td>
<td>W_1(i,j) - W_2(i,j)</td>
</tr>
<tr>
<td>NSSD $\sum_{i,j}</td>
<td>\tilde{W}_1(i,j) - \tilde{W}_2(i,j)</td>
</tr>
<tr>
<td>NCC $\sum_{i,j} \tilde{W}_1(i,j) \cdot \tilde{W}_2(i,j)$</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: The different matching costs used in this assignment. $W$ refers to a window of pixels and $\tilde{W} = \frac{W - \bar{W}}{\sqrt{\sum_{k,l} (W(k,l) - \bar{W})^2}}$ is a mean-shifted and normalized version of the window. $N$ refers to the number of pixels in each window.

What to turn in:

- Disparity maps for DOTS, BARN1, and TSUKUBA using SSD, and NSSD with windows of 3x3 and 5x5. You can use the recommended occlusion penalty indicated in Table 1 (however, you may find that some tweaking of the occlusion penalty on the different image sets is helpful).
- Your proof showing equivalence of NSSD and NCC.

Figure 2. Random dot stereogram image pair and the depth map ground truth.
What to turn in:
a. A report with the items described above, any other figures, and comments/issues that you find relevant. Also, include a short summary of your conclusions, based on your experiences while completing this assignment.
b. Hardcopy of your code.
c. Email me (cutran@cs.ucsd.edu) a tar or zip file with the code and report. Please name the file in the following format <First name>_ <Last name>_CSE252_HW3. In the subject line, please put the string: CSE252HW3.