CS 484: Assignment 4, Stereo Vision:
Block Matching and Dynamic Programming
Due: 13:00, Tues. Mar. 26, 2013.

In this assignment you will implement and test some simple stereo algorithms discussed in class.

In each case you will take two images I_l and I_r (a *left* and a *right* image) and compute the horizontal *disparity* (ie., shift) of pixels along each scanline. This is the so-called *baseline stereo* case, where the images are taken with a forward-facing camera, and the translation between cameras is along the horizontal axis.

Your system will output a disparity image, where each pixel encodes the shift between the left and right images. In general one needs to know the interoccular distance, T, and the nodal distance, f, to reconstruct depth, Z, from disparity (ie., $d = \frac{fT}{Z}$). Here we will just focus on the computation of the disparity d.

You will be given two pairs of images. The first pair is a synthetic scene with images `synth1.tif` and `synth2.tif`. The second pair is a real scene with images `pm1.tif` and `pm2.tif`.

1 Block Matching

Write a `Matlab` program to compute the disparity (horizontal shift) of each scanline. In particular, your program should find the displacement d that minimizes the *mean-squared error* between an image patch at (y, x) in the left image (I_l) and a corresponding patch at $(y, x + d)$ in the right image (I_r):

$$\frac{1}{(2W+1)^2} \sum_{-W \le j \le W} \sum_{-W \le i \le W} (I_l(y+j, x+i) - I_r(y+j, x+i+d))^2$$

W is the half window width. For this assignment we will use $W = 3$, which gives a *7-by-7* window. The disparity, d, is restricted to be in range $-D \le d \le D$. For this assignment we will use $D = 8$.

Note: The summation above can be efficiently implemented in `Matlab` using matrix functions. Suppose matrices `A` and `B` are patches from the left and right images, respectively. We can compute the mean squared error using `sum(sum((A-B).^2))`.

1.1 Plain block matching

Perform the above block matching for both image pairs. To display the disparity, plot a greyscale image of the same size as the input images I_l and I_r. Each pixel of the output image denotes a disparity. Print out the disparity images to show the block matching results both the synthetic and real image pairs.

Note: You can use the Matlab function `imagesc(D,[-8,8])` to display a disparity matrix `D` as a greyscale image. Dark areas depict negative disparities, while bright areas depict positive disparities. Zero disparities should show up as gray areas in the image.

1.2 Censoring: removing areas of low texture

As with motion estimation, areas of low texture are hard to match reliably between images. In stereo, since we are attempting to determine the horizontal motion between frames we want to remove areas where the *horizontal* texture is low.

The average horizontal variation of an image patch centered at (y, x) is given by:

$$\sigma_h^2 = \frac{1}{(2W+1)^2} \sum_{-W \le j \le W} \sum_{-W \le i \le W} (I(y+j, x+i) - \bar{I}(y+j, x))^2$$

where y denotes a particular scanline of the image and

$$\bar{I}(y, x) = \frac{1}{(2W+1)} \sum_{-W \le i \le W} I(y, x+i)$$

denotes the mean value of the jth scanline of the patch centered at (y, x).

Use the above formula to censor image positions in the left image, I_l, which have low horizontal variation. Perform exactly the same matching as the previous section, except that when the horizontal variation in I_l is low, output a value of -8 to the disparity image instead of computing the disparity. Black areas of the resulting disparity maps will indicate places where there is no disparity information.

You should experiment with a few different values of σ*, but a good starting value is* $\sigma = 2$.

1.3 Consistency checking

Often a part of the scene may be visible in only one of the images. This will result in random or spurious matches, since a correct match does not exist.

Let $d_l(x)$ be the disparity estimate for a particular scanline (row) in the left image. (This was obtained above by matching every patch along the left scanline with a patch in the corresponding row of I_r.) Similarly let $d_r(x)$ be the disparity estimate for matching the right image to the left image.

For a given pixel at x_l in the left scanline the predicted position in right scanline is $x_r = x_l + d_l(x_l)$ (ie., the position in the left scanline plus the disparity). Projecting this position back to the left scanline, we have $x_l' = x_r + d_r(x_r)$. A match between the left and right scanlines is *consistent* iff $x_l' = x_l$.

Use the above consistency check to remove places where the match is inconsistent. At the places where the match is inconsistent, set the disparity value to -8.

Note: Implementing the consistency check requires that you do the block matching in both directions (to compute $d_l(x)$ and $d_r(x)$)!

Output the result for both the synthetic and real image pairs. You may use the block matcher from the plain block matching or the censored block matcher (your choice). However, please specify which one you used.

2 Dynamic programming

Another way to match scanlines is by dynamic programming.

Consider two scanlines $I_l(i)$ and $I_r(j)$, $1 \leq i, j \leq N$, where N is the number of pixels in each line (the process will be repeated for each row of the image). Pixels in each scanline may be matched, or skipped (considered to be occluded in either the left or right image).

Let $d(x_l, x_r, y)$ be the cost associated with matching pixel x_l in the left image with pixel x_r in the right image along scanline y. We use the same mean squared error measure as for block matching,

$$d(x_l, x_r, y) = \frac{1}{(2W+1)^2} \sum_{-W \leq j \leq W} \sum_{-W \leq i \leq W} (I_l(y+j, x_l+i) - I_r(y+j, x_r+i))^2$$

The cost of skipping a pixel (in either scanline) is given by a constant c_0. *You should experiment with a few values for* c_0*. Start with the value* $c_0 = 1$ *and go from there.*

Given these costs, we can compute the optimal (minimal cost) alignment of two scalines recursively as follows:

1. $D(0, 0) = 0$,
2. $D(i, 0) = i \cdot c_0$,
3. $D(0, j) = j \cdot c_0$,
4. $D(i + 1, j + 1) = \min(D(i, j) + d_{i+1,j+1}, D(i, j + 1) + c_0, D(i + 1, j) + c_0)$,

The intermediate values are stored in an N*-by-*N matrix, D. The total cost of matching two scanlines is $D(N, N)$. Note that this assumes the lines are matched at both ends (and hence have zero disparity there). This is a reasonable approximation provided the images are large relative to the disparity shift.

Given D we find the optimal alignment by backtracking. (This is called the *Viterbi algorithm.*) Starting at $(i, j) = (N, N)$, we choose the minimum value of D from $\{(i - 1, j - 1), (i - 1, j), (i, j - 1)\}$. Selecting $(i - 1, j)$ corresponds to skipping a pixel in I_l (a unit increase in disparity), while selecting $(i, j - 1)$ corresponds to skipping a pixel in I_r (a unit decrease in disparity). Selecting $(i - 1, j - 1)$ matches pixels (i, j), and therefore leaves disparity unchanged. Beginning with zero disparity, we can work backwards from (N, N), tallying the disparity until we reach $(1, 1)$.

Note: A good way to debug your program is to view the pixel matches d_{ij} and the match costs D_{ij} as matrices. For a given scan line you can plot these as either greyscale images or as surfaces (`mesh` command in Matlab).

2.1 Individual scanline matching

Implement the above method of scanline matching for both the synthetic and real image pairs.

A good way to interpret your solution is to plot the alignment found for single scan line(s). To display the alignment plot a graph of I_l (horizontal) vs I_r (vertical). Begin at $D(N, N)$ and work backwards to find the best path. If a pixel in I_l is skipped, draw a horizontal line. If a pixel in I_r is skipped, draw a vertical line. Otherwise, the pixels are matched, and you draw a diagonal line. The plot should end at $(0, 0)$.

Show the scanline matching for line 50 in both the synthetic and real pairs.

Optional: You can also show the values of $I_l(x)$ and $I_r(x)$ that are matched by the dynamic programming algorithm. Starting at $D(N, N)$, the path will either skip a pixel in I_l, skip a pixel in I_r, or match pixels in I_l and I_r. You will *only plot pixel values when the pixels are matched.* Otherwise the graph will be empty. The graph should be the same size as the scanline, just with missing values where no match was performed.

2.2 Disparity computation

Compute the disparity maps for both image pairs using the method described above. Output the disparity as a greyscale image.

Note: With dynamic programming (at least this simple implementation of the algorithm), the range of disparities is not limited. Before plotting, it is best to check the minimum and maximum value. If necessary, use these as parameters to the `imagesc` command to scale your images so they match those of the block matching algorithm.