CS 498/698: Assignment 3, Stereo Vision: Block Matching and Dynamic Programming
Due: 1pm, Thurs. Mar. 16, 2000.

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.pgm` and `synth2.pgm`. The second pair is a real scene with images `pm1.pgm` and `pm2.pgm`.

1 Block Matching

Write a Matlab or C 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):

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

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 the MSE function. In particular, `mse(A,B)` returns the mean-squared-error between two matrices A and B.

1.1 Plain block matching

Compute the above block matching for both image pairs. To display the disparity, output a greyscale (PGM) image of the same size as the input images I_l and I_r. Each pixel of the output image denotes a disparity. A maximum disparity (D) will be depicted by 255, while the minimum disparity ($-D$) will be depicted by a value of 0.

You should 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 `showIm(D,[-8,8])` (in the course software directory) to display a disparity matrix `D` as a greyscale image with brightness scaled to a given range.

1.2 Censoring: removing areas of low texture

As with motion estimation, areas of low texture cause 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:

$$\sigma^2 = \frac{1}{4W^2} \sum_{-W \le j \le W} \sum_{-W \le i \le W} (I(y+j, x+i) - \bar{I}(y+j, x))^2$$

where y denotes a particular scanline of the image and

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

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 disparity value of 0 to the disparity image instead of computing (and scaling) the disparity. Black areas of the resulting disparity maps will indicate places where there is no disparity information.

You may experiment with 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 row in the left image. (This was obtained above by matching every patch of I_l 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. **Note:** This requires that you do the block matching in both directions!

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 $I_1(i)$ and $I_2(j)$, $1 \leq i, j \leq N$, 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_{ij} be the cost associated with matching pixel $I_1(i)$ with pixel $I_2(j)$. Here we consider a squared error measure between pixels given by:

$$d_{ij} = \frac{(I_1(i) - I_2(j))^2}{\sigma^2}$$

where σ is some measure of pixel noise. The cost of skipping a pixel (in either scanline) is given by a constant c_0. For the experiments here we will use $\sigma = 2$ and $c_0 = 1$ (although you may experiment if you wish).

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

1. $D(1,1) = d_{11}$,
2. $D(i,j) = \min(D(i-1,j-1) + d_{ij}, D(i-1,j) + c_0, D(i,j-1) + c_0)$,
3. $D(N,N) = d_{NN}$.

This assumes that the beginning and end of the both scanlines are taken to be matched (ie., unoccluded pixels).

Given D we find the optimal alignment by backtracking. In particular, 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_1 (a unit increase in disparity), while selecting $(i,j-1)$ corresponds to skipping a pixel in I_2 (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 $(1,1)$.

2.1 Individual scanline matching

Implement the above method of scanline matching. For scanline 50 in both the synthetic and real pairs, show the alignment found. That is, show the scanline for both the left and right images.

A good way to display this is to plot a graph as you backtrack through the D matrix above. A pixel will be plotted only when it is matched in the matrix. (Use the matlab "hold on" command to plot multiple points on the same graph).

2.2 Disparity computation

Compute the disparity maps for both image pairs using the method described above. Output the disparity image as a greyscale image. Again, you can use the Matlab function `showIm` to automatically scale the 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 (eg., `min(min(M))` and `max(max(M)) will give the minimum and maximum values of a matrix in Matlab), and use these as parameters to the showIm command.`