Alexandria Univeristy
Faculty of Engineering
Computer and Systems
Engineering Department

CSE: Computer Vision
Assigned: Sat, Dec 03, 2022
Due: Sat, Dec 10, 2022

Assignment 3

1 Part One

1.1 Stereo Vision

In this part of the assignment you will implement and test some simple stereo algorithms. 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. We will calculate the disparity using two ways.

1.1.1 Block Matching

The first way to get the disparity is by matching each pixel in the left image to a pixel in the right image. Since there is no rectification needed, you will only need to match the row in the left image with its equivalent in the right image. you will calculate the disparity in two ways, once using the cost as the Sum of Absolute Differences (SAD) and another time using Sum of Squared Differences. Do this for windows of size w where w = 1, 5 and 9. You will produce 6 maps: 2 maps for each window size, once using SAD and the other using SSD.

1.1.2 Dynamic programming

The second way to calculate disparity is to use a dynamic programming algorithm to get the minimum cost of matching a whole row in the image. Consider two scanlines $I_l(i)$ and $I_r(j)$. 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 $I_l(i)$ with pixel $I_r(j)$. Here we consider a squared error measure between pixels given by:

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

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 $c_0 = 1$. Given these costs, we can compute the optimal (minimal cost) alignment of two scanlines recursively as follows:

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

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. You can find the disparity map of matching the left image to the right or vice versa at the same time or only calculate one of them. Given the cost matrix 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_l, so the left disparity map of i is zero. Selecting (i, j - 1) corresponds to skipping a pixel in I_r, and the right disparity map of j is zero. Selecting (i - 1, j - 1) matches pixels (i, j), and therefore both disparity maps at this position are set to the absolute difference between i and j.

1.2 Bonus

A good way to interpret your solution is to plot the alignment found for single scan line. Display the alignment by plotting a graph of I_l (vertical) vs I_r (horizontal). Begin at D(N, N) and work backwards to find the best path. If a pixel in I_l is skipped, draw a vertical line. If a pixel in I_r is skipped, draw a horizontal line. Otherwise, the pixels are matched, and you draw a diagonal line. The plot should end at (1, 1).

1.3 Notes

You are required to deliver the following:

- Your code.
- Output for a test image.
- Report including explanation of your code and representative results on sample test images.

You can work in groups of 3.