# STEREO VISION

Sandra Sherif Wasfi **ID:** 3940

Rimon Adel Makram **ID:** 4441

Adel Atef Meleka **ID:** 3960

## INTRODUCTION

In this problem set you will implement and test some simple stereo algorithms discussed in class. In each case you will take two images *Il* and *Ir* (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.

# 1 Block Matching

- To get the disparity value at each point in the left image, you will search over a range disparities, and compare the windows using two different metrics mentioned in class: Sum of Absolute Differences (SAD) and Sum of Squared Differences. Do this for windows of size *w w* where w = 1, 5 and 9. The disparity, *d*, is restricted to be in range *DdD*. For this assignment we will use D = 8. You should try at least 2 pairs of images and for each show for each window size and each metric disparity image.

- To implement block matching algorithm, we do the following steps:

1. We implement a function that can compare a patch from left image with a full strip from right image.
2. We implemented also another function that iterates extracting each patch from a given strip from left matching image.
3. We iterate through all image strips of the left image.

# 2 Dynamic Programming

- Consider two scanlines *Il*(*i*) and *Ir*(*j*). Pixels in each scanline may be matched or skipped (considered to be occluded in either the left or right image). Let *dij* be the cost associated with matching pixel *Il*(*i*) with pixel *Ir*(*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 *σ* = 2 and *c*0 = 1. Given these costs, we can compute the optimal (minimal cost) alignment of two scanlines recursively as follows:

1. $D(1, 1) = d_{11}$

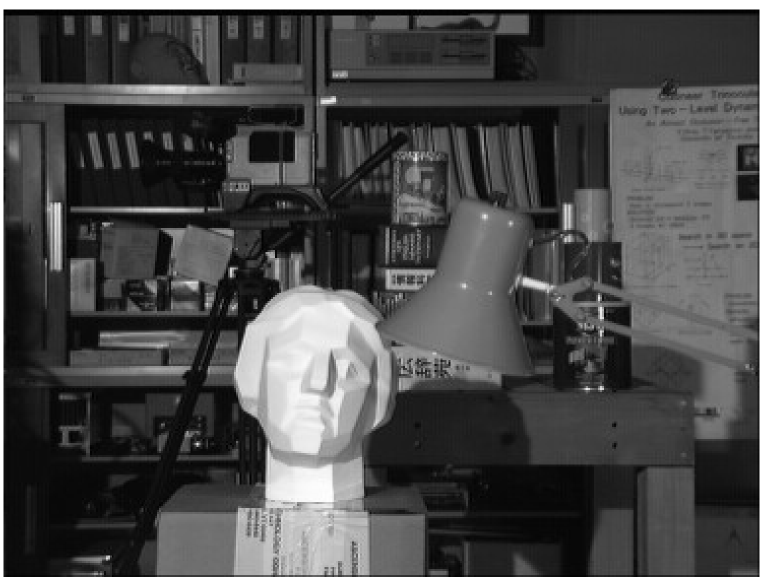
2. $D(i, j) = min(D(i - 1, j - 1) + d_{ij}, D(i - 1, j) + c_0, D(i, j - 1) + c_0)$

- 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. 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 Il (a unit increase in disparity), while selecting (i, j - 1) corresponds to skipping a pixel in Ir (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).

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

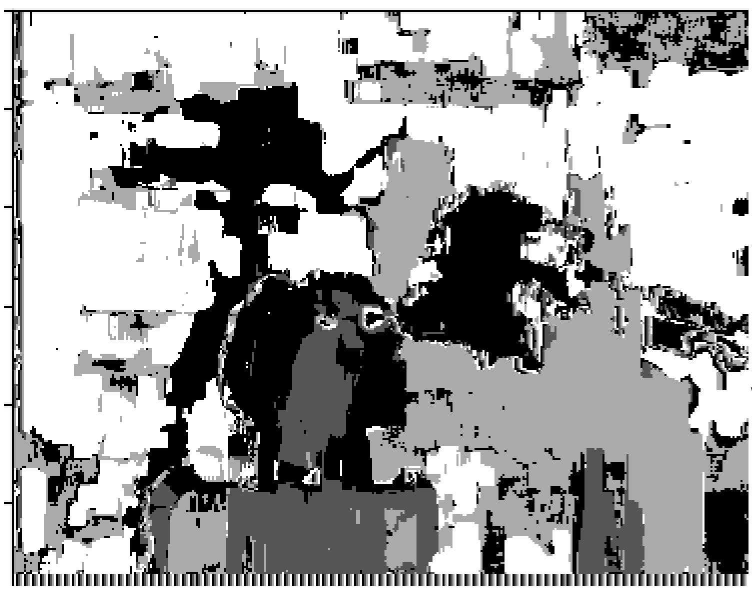
- We repeat the process explained above for each row of the image and compute the disparity maps for image pairs.

# 3 All Results

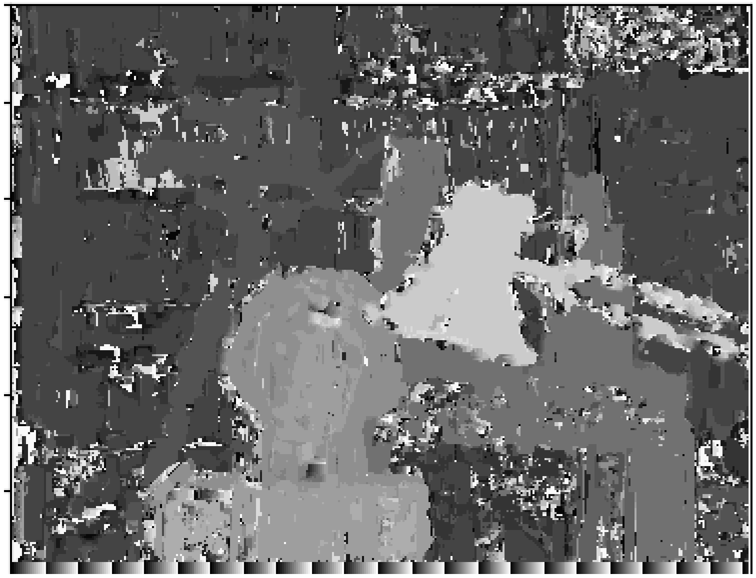


Block matching

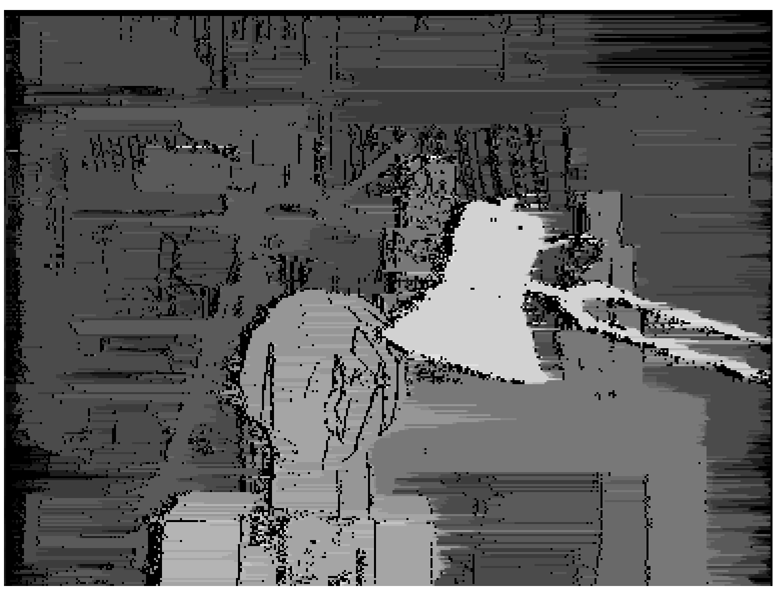
Window size = 1



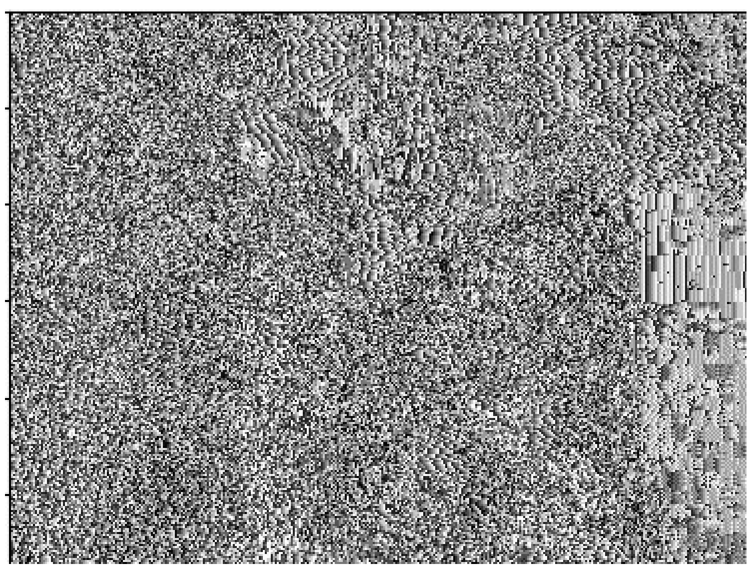
window size =5



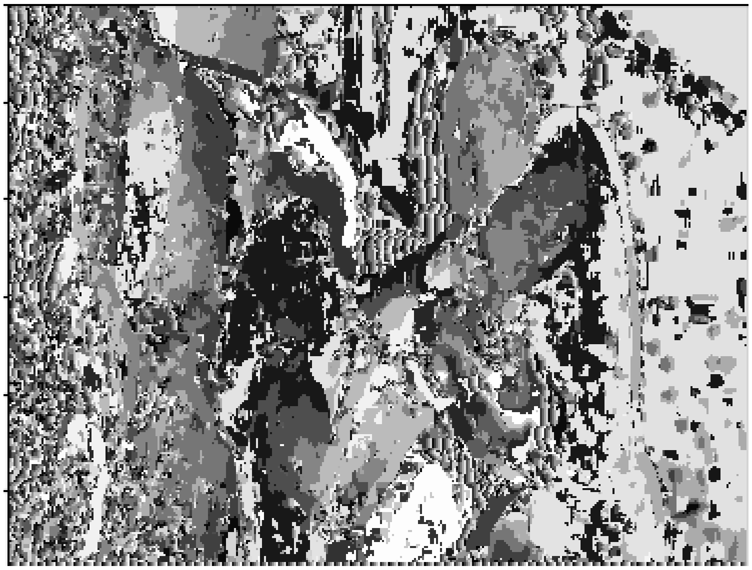
Dynamic programming



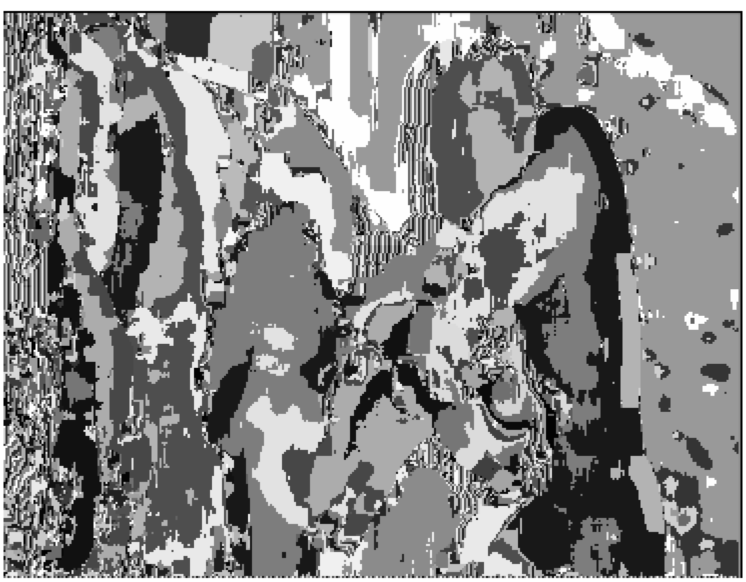
Block Matching



window =1



window =5



window = 9

Dynamic Programming

**THANK YOU**