Simone Mosco 2055298

Lab 1: Semi-Global Stereo Matching

Task 1: Compute Path Cost

The first task regards the extension of the `compute_path_cost()` function that, given a pixel p with current path index and direction, computes the path costs for all possible disparities in the range $[0, d]$. The proposed implementation is based on the following formulas

$$E(p_i, d) = E_{data}(p_i, d) + E_{smooth}(p_i, p_{i-1}) - min_{0 \leq \Delta \leq d_{max}} E(p_{i-1}, \Delta) \tag{1}$$

where

$$E_{smooth}(p, q) = min \begin{cases} E(q, f_q) & if f_p = f_q \\ E(q, f_q) + c_1 & if |f_p - f_q| = 1 \\ min_{0 \leq \Delta \leq d_{max}} E(q, \Delta) + c_2(p, q) & if |f_p - f_q| > 1 \end{cases} \tag{2}$$

In order to solve this task, an initial distinction is made in case the processed pixel is the first, assigning the default cost value (E_{data}) into the correspondent `path_cost_` index, without using the other two terms in equation 1.

In the other case we first loop through all disparities to find the `best_prev_cost` which corresponds to the third term in equation 1. Then we loop again for all disparities to calculate the E_{smooth} term and the final `path_cost_` regarding the current pixel and current path. Calculating the three terms used for E_{data} in equation 2, the first one corresponds to the `no_penalty_cost` that is simply the path cost value of the previous pixel at same disparity level while the third term is computed as the sum of `best_prev_cost` and penalty cost c_2. With regard to the second term, the implemented code makes a distinction if the current disparity corresponds to the first/last in the disparity range or to all the other values. In fact, in the first case, since we are at the borders, we consider respectively the following or previous disparity value in the path to retrieve the `path_cost` value of the previous pixel and add it with the penalty cost c_1 to calculate the `small_penalty_cost`. In the second case, we consider directly the minimum `path_cost` value between the previous or next value of disparity since all these terms belong to a general *min* function. Afterwards we take the three values computed in equation 2 and accept the minimum among them to compute the E_{smooth} term.

Finally the value of equation 1 is computed as the sum of the cost of the current pixel with a current disparity, the smooth term minus the `best_prev_cost` and is fed into the `path_cost_` 4-dimensional array.

Task 2: Aggregation

In the second task, two main operations are performed, the initialization of start, end, step variables and the actual aggregation that fill the `aggr_cost_` tensor.

The first sub-task is performed using a switch-case block over the `dir_x` and `dir_y` variables that assign the start, end and step values in both x and y directions. Based on the input directions, the current path will go from west to east or opposite for the x axis and from north to south or opposite for the y axis. Regarding the case where `dir_x` and/or `dir_y` are equals to 0, the values assigned for `start`, `end` and `step` are the same as in the case of direction equals to 1 with step size of 1.

The aggregation operation is obtained looping for each pixel, each disparity value between $[0, d]$, summing all the path cost for each direction. Since each path cost value contains the E_{data} of the relative pixel, this way we are summing it multiple times as the number of path directions. The proposed implementation avoid this issue, subtracting the term in the path direction loop and adding it only once at the end of the path for loop.

Results

Various tests were carried out, trying all three available input images, as shown in Table 1. In the next page are shown the disparity map images obtained with disparity range equals to 85.

Disparity	Aloe	Rocks	Cones
85	106.987	346.657	470.417

Table 1: Mean Squared Error values

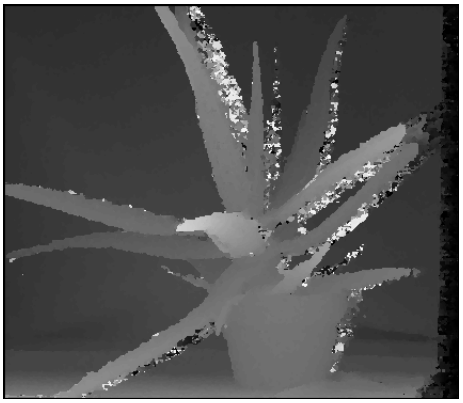

Figure 1: Aloe disparity map

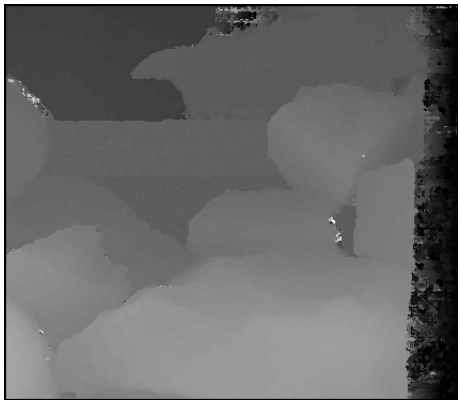

Figure 2: Rocks disparity map

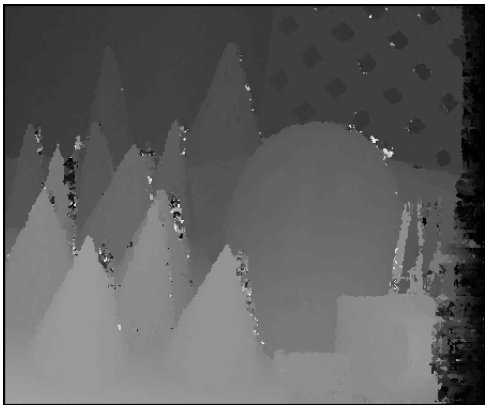

Figure 3: Cones disparity map