Assignment 3 CV

Name	Id
Hossam Osama Ahmed	21010451
Yahia Ibrahim AlKaranshawy	21011554
Mohamed MM Abdel-Moneim	21011213

1.1 Block Matching Algorithm

For each pixel in the **left image**, the algorithm:

1. Extracts a local window.
2. Slides the window horizontally on the right image.
3. Computes similarity using SAD or SSD.
4. Selects the displacement (d) with the minimum cost.
5. Stores d as the **disparity** at that pixel.

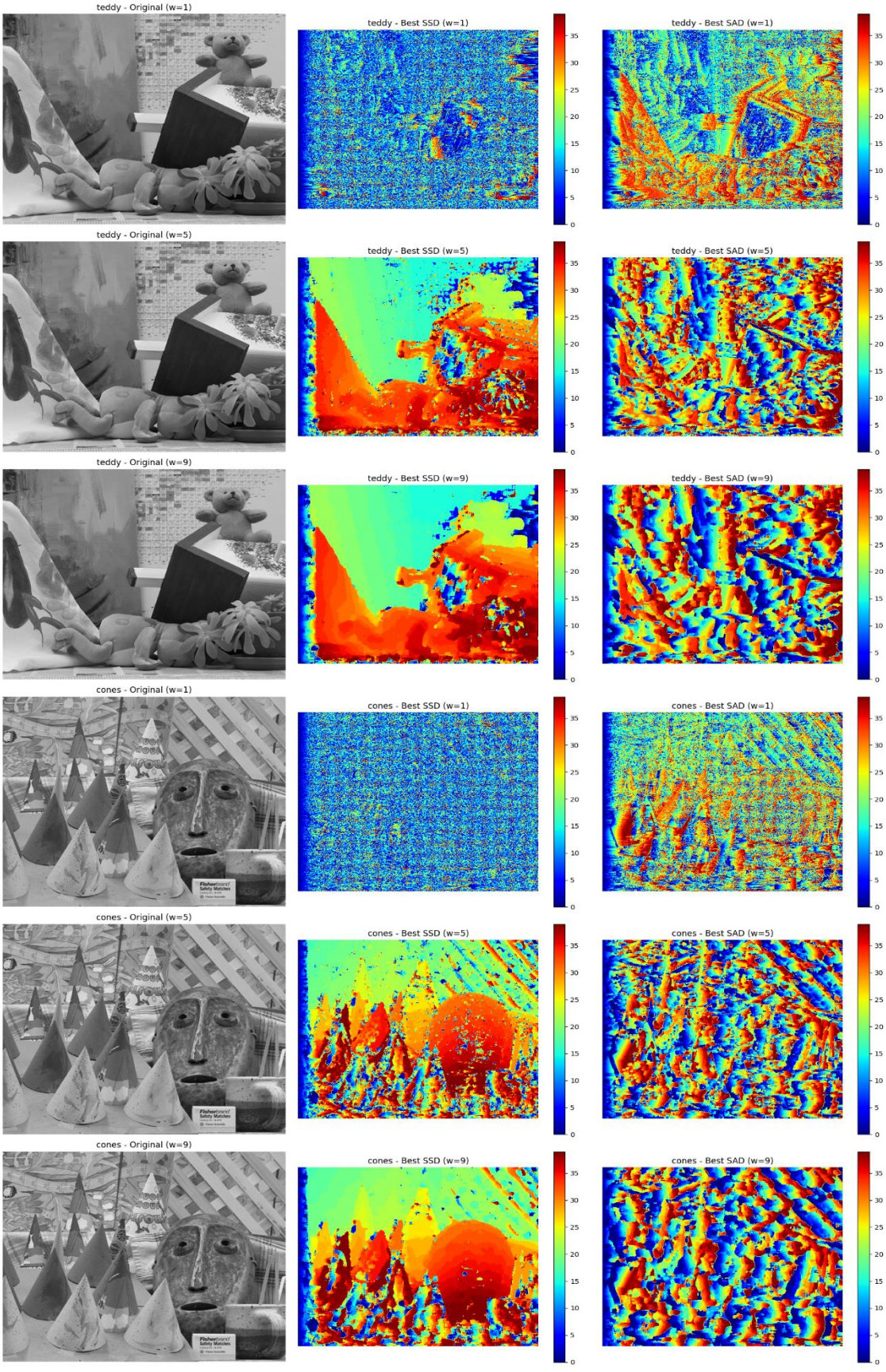
1.2 Automatic Selection of max_disp

Since optimal disparity range differs between datasets, four candidates are tested:

[40, 80, 120, 160]

For each window size:

1. Compute disparity for each max_disp candidate.
2. Compute a quality score using:
   total_cost = sum(disparity_map)
   Lower cost = less noise + more consistent matching.
3. Select the max_disp that yields the smallest total_cost.


teddy - Original (w=1)
teddy - Best SSD (w=1)
teddy - Best SAD (w=1)
teddy - Original (w=5)
teddy - Best SSD (w=5)
teddy - Best SAD (w=5)
teddy - Original (w=9)
teddy - Best SSD (w=9)
teddy - Best SAD (w=9)
cones - Original (w=1)
cones - Best SSD (w=1)
cones - Best SAD (w=1)
cones - Original (w=5)
cones - Best SSD (w=5)
cones - Best SAD (w=5)
cones - Original (w=9)
cones - Best SSD (w=9)
cones - Best SAD (w=9)
0
5
10
15
20
25
30
35

Part 1.2: Dynamic Programming

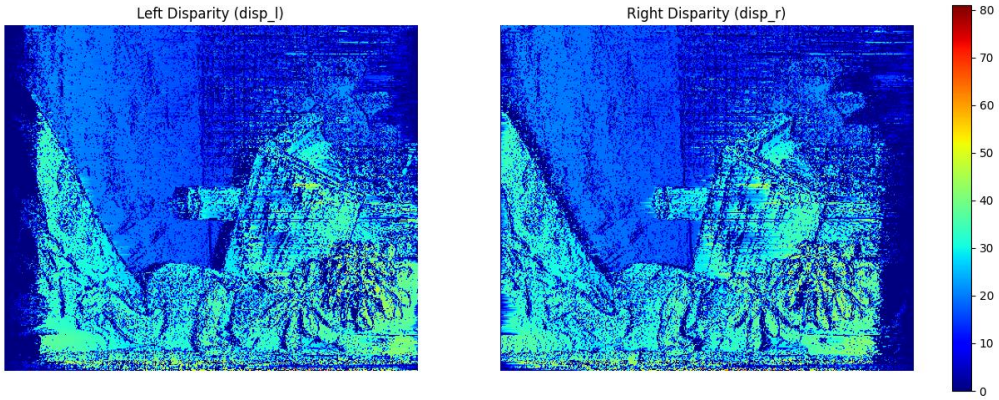
We compute disparity by aligning each left/right image row using dynamic programming.

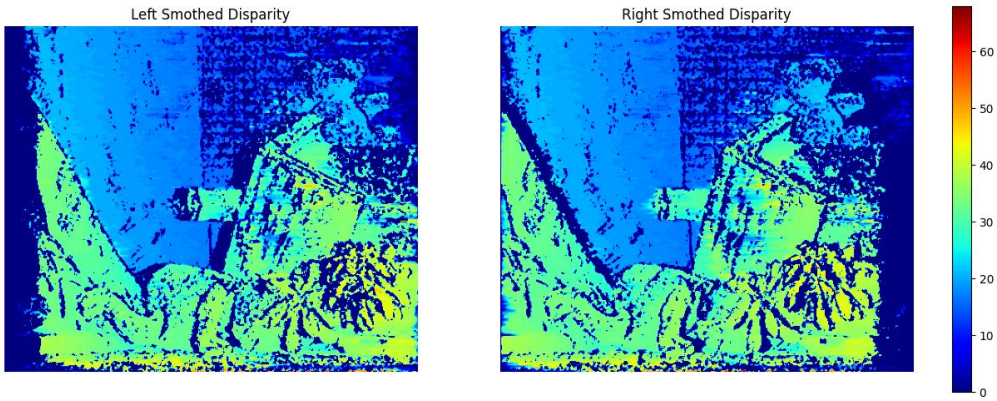
- **Data structures:**
  - `dp` (W+1 × W+1): DP cost matrix for one scanline.
  - `disp_left`, `disp_right` (H × W): output disparity maps.
- **Cost function:**
  Matching cost: $(Il[i]-Ir[j])2/\sigma 2$
  Occlusion cost: constant c0 .
- **DP step:**
  For each row, `dp` is filled using:
  - match: `dp[i-1][j-1] + d`
  - skip-left: `dp[i-1][j] + c0`
  - skip-right: `dp[i][j-1] + c0`
- **Backtracking:**
  Starting from `(W, W)`, we follow the minimum-cost predecessor.
  - diagonal → pixels match → disparity = |i − j|
  - vertical → left pixel skipped → left disparity = 0
  - horizontal → right pixel skipped → right disparity = 0
- **Output:**
  The algorithm produces full left and right disparity maps by repeating this process for every scanline.

Results

1- 

Result:

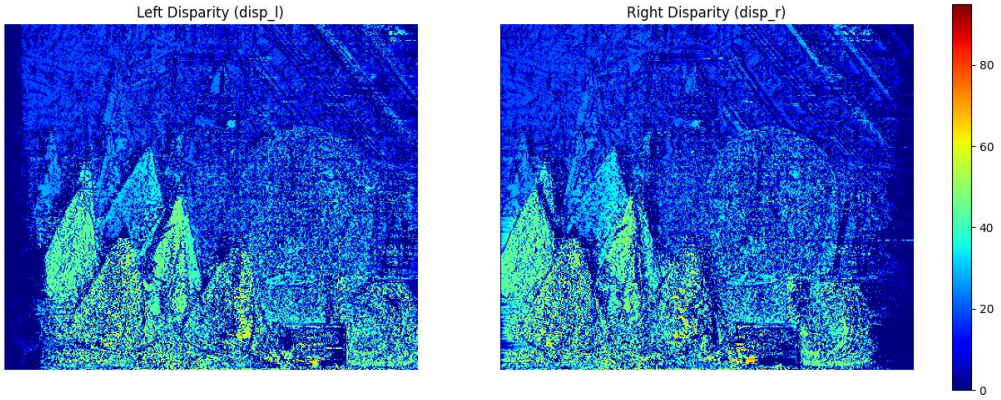


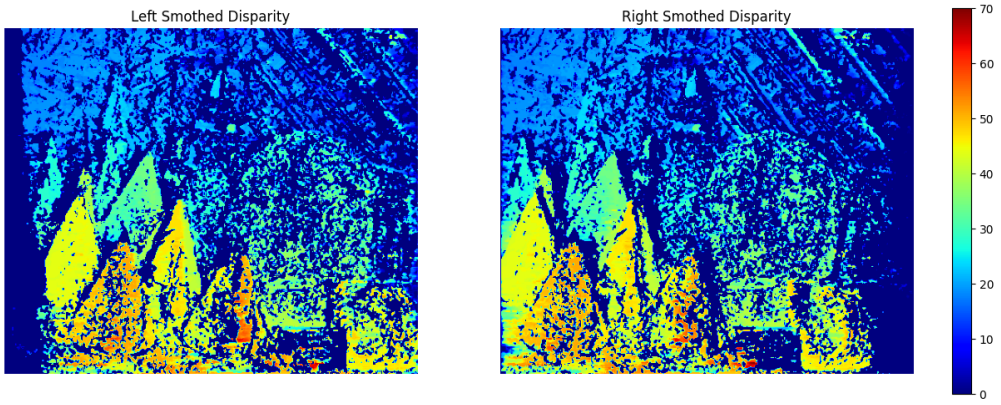


Smoothed version

2-


Fisherbrand
Safety Matches


Left Disparity (disp_l)
Right Disparity (disp_r)
80
60
40
20
0


Left Smothed Disparity
Right Smothed Disparity
70
60
50
40
30
20
10
0

Bonus

We keep track of the path we take as we backtrack by maintaining two arrays path_x and path_y to store the indices i and j along the way, we then print these arrays against each other.

Example plot