Assignment #3 Report
Computer Vision

No	Name	ID
1	عبدالرحمن السيد جاد السيد	19015894
2	عبدالعزيز محمد عبدالعزيز محمد	19015941
3	عمر خيرت محمد ابو ضيف	19016063

Google colab link: here

Part 1: Block Matching:

1. Problem Statement:

Implement block matching for stereo vision using images I_l and I_r. Without rectification, match pixels from the left image to the right image along each scanline. Compute horizontal disparity using Sum of Absolute Differences (SAD) and Sum of Squared Differences (SSD) for window sizes w = 1,5 and 9. Generate six disparity maps (2 per window size) for analysis.

2. Code Explanation and process steps:

a. Libraries needed: (cv2, numpy, matplotlib)

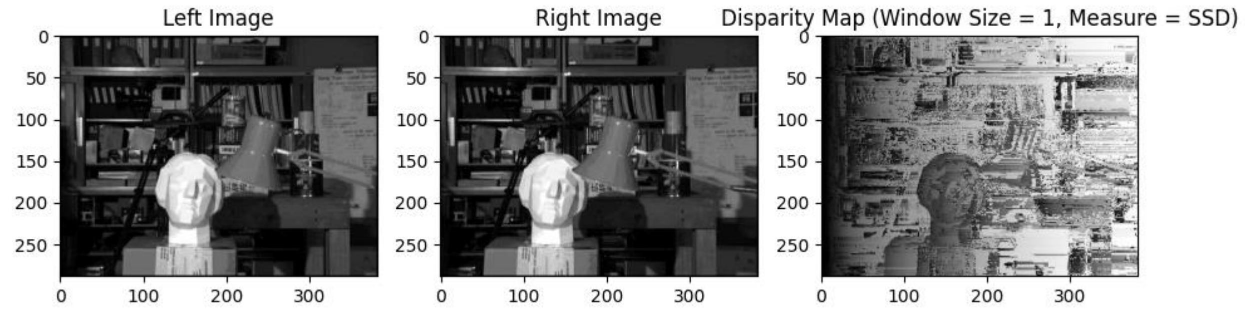
b. Block Matching Stereo Function:
   - The “Block_Matching_Stereo” function takes two images (“left_image” and “right_image”) and optional parameters for “block_size” (default is 1) and “cost_type” (“ssd” or “sad”).
   - It initializes an empty disparity map with the same dimensions as the left image.
   - It iterates over each pixel in the left image, defining a block around each pixel based on the specified block size.
   - For each block in the left image, it searches for the corresponding block in the right image within a certain range.
   - It calculates the cost using either Sum of Squared Differences (SSD) or Sum of Absolute Differences (SAD).
   - The disparity for the pixel is set to the best match found based on the minimum cost.

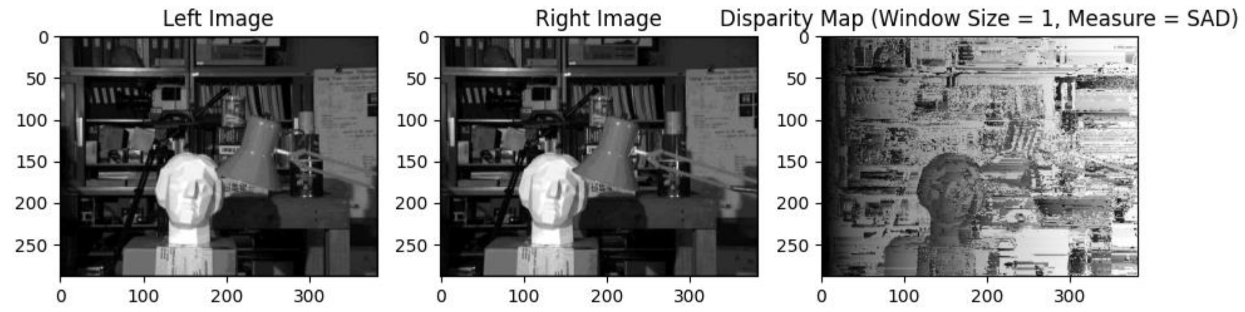
c. Image Loading and Processing:
   - Image paths are provided for left and right images in pairs.
   - The script loops over each pair, loads the images, and converts them to `float32` for processing.

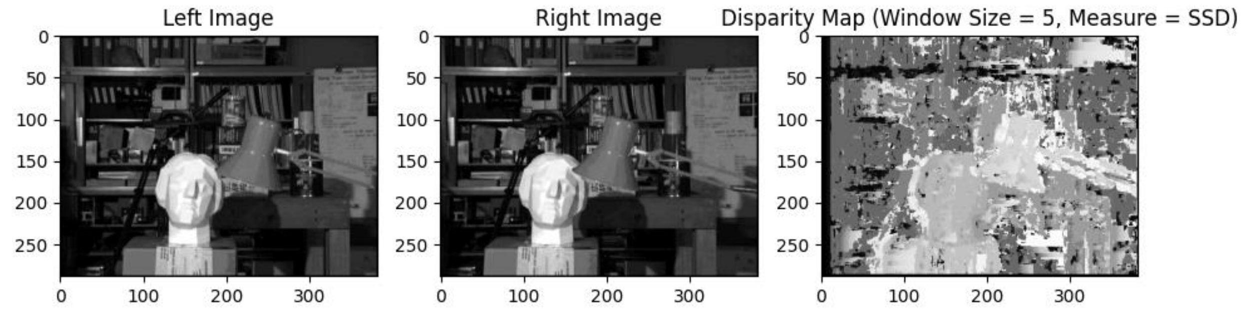
d. Stereo Matching for Different Window Sizes and Similarity Measures:
   - The code then loops over different window sizes (1, 5, and 9) and similarity measures (“ssd” and “sad”).
   - For each combination, it calls the “Block_Matching_Stereo” function to obtain the disparity map.
   - The disparity map is normalized for display and equalized for better visualization.

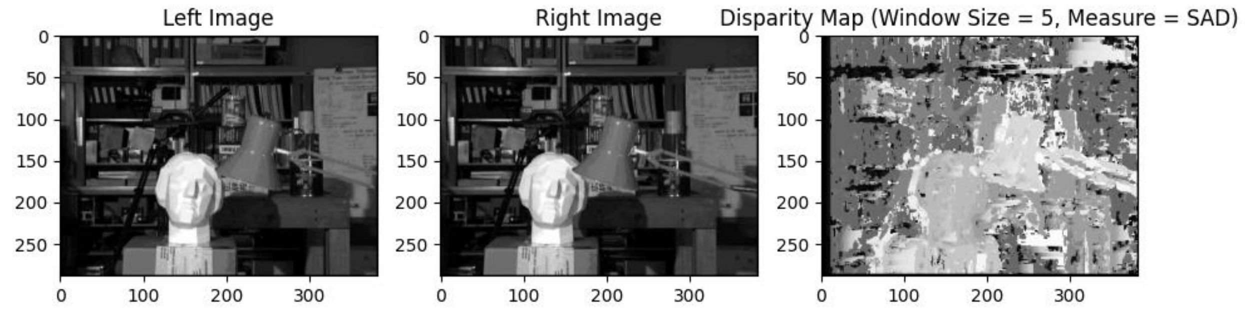
3. Output of test images:

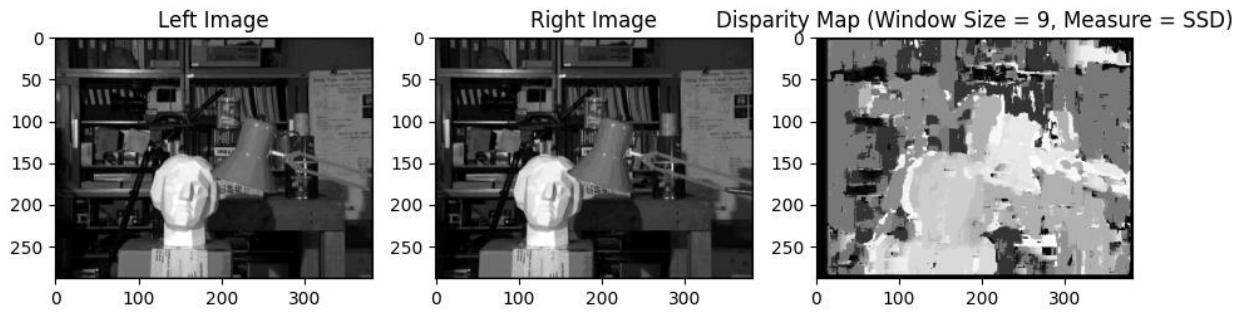
First image with window size (1, 5, 9) and measuring (SSD, SAD)

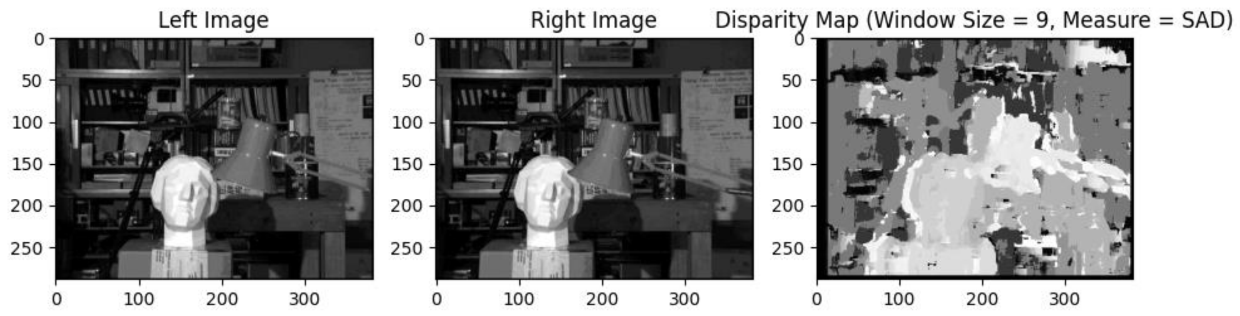




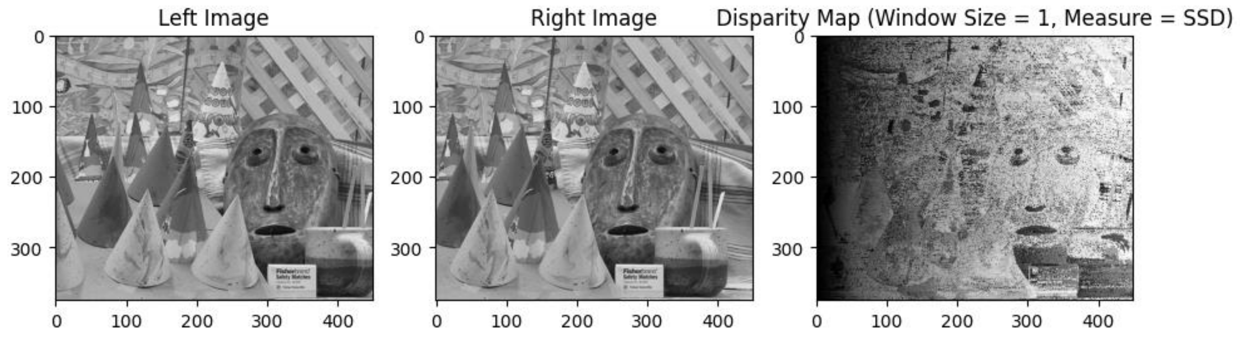


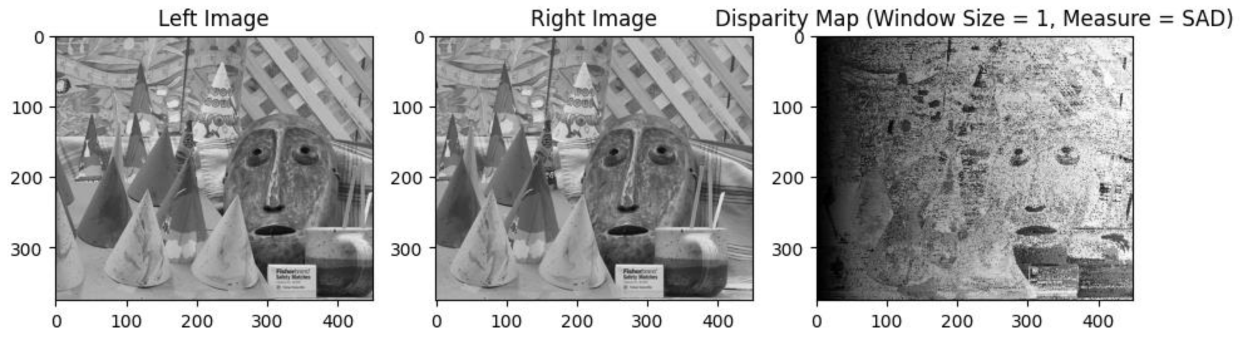


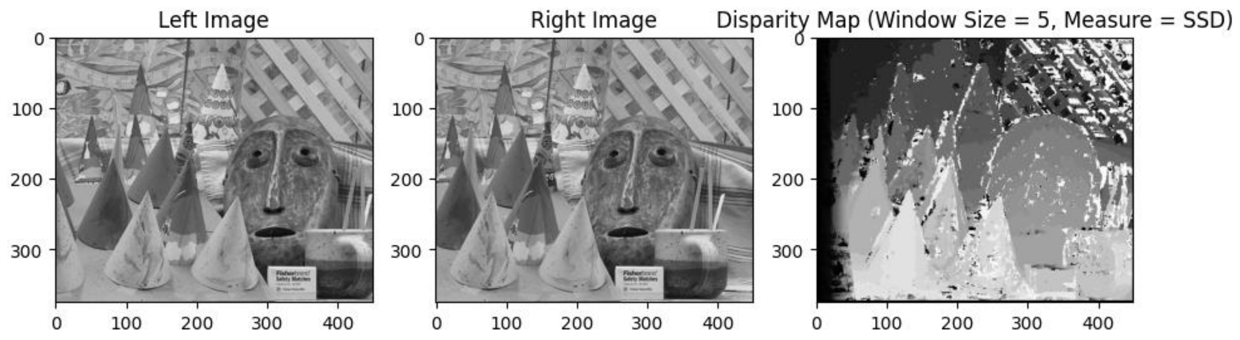


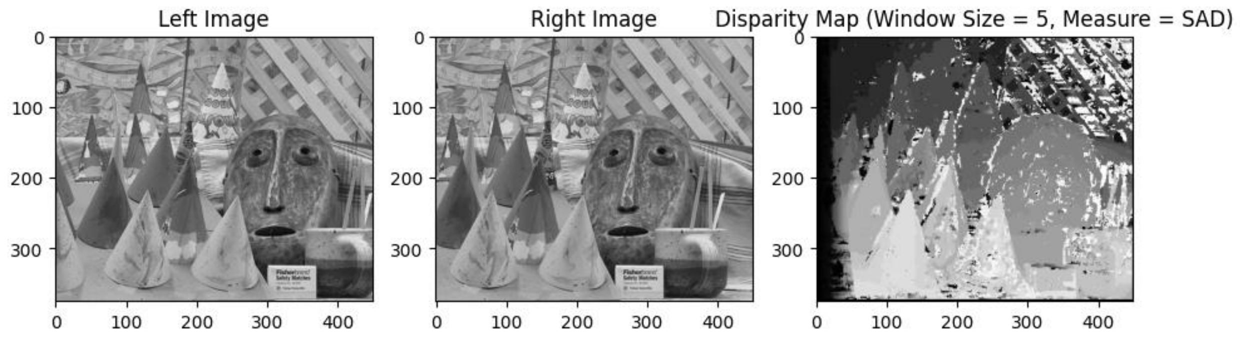


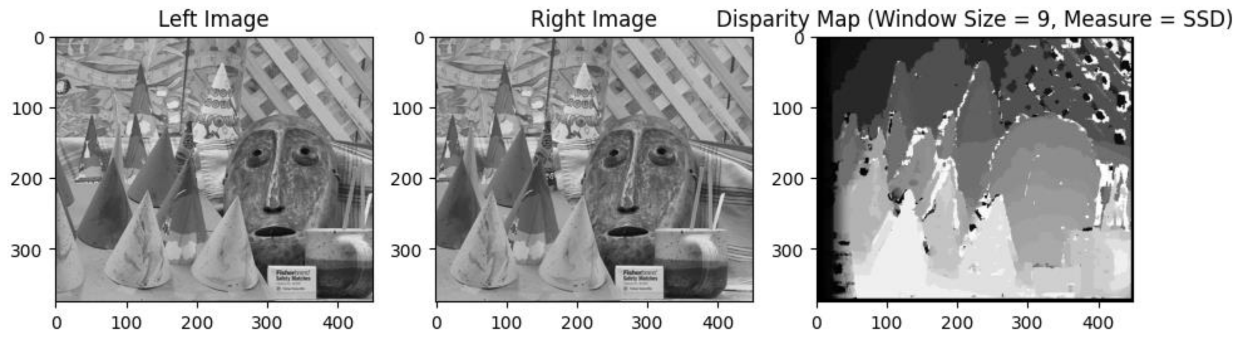
Second image with window size (1, 5, 9) and measuring (SSD, SAD)

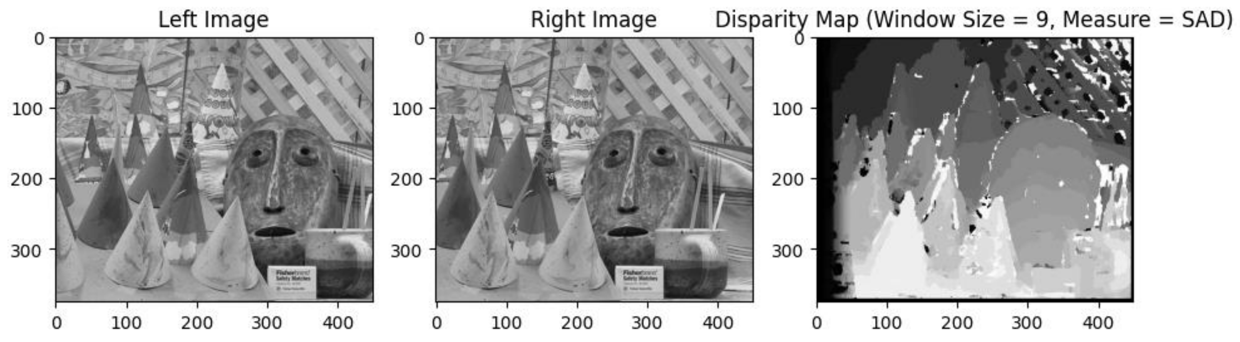




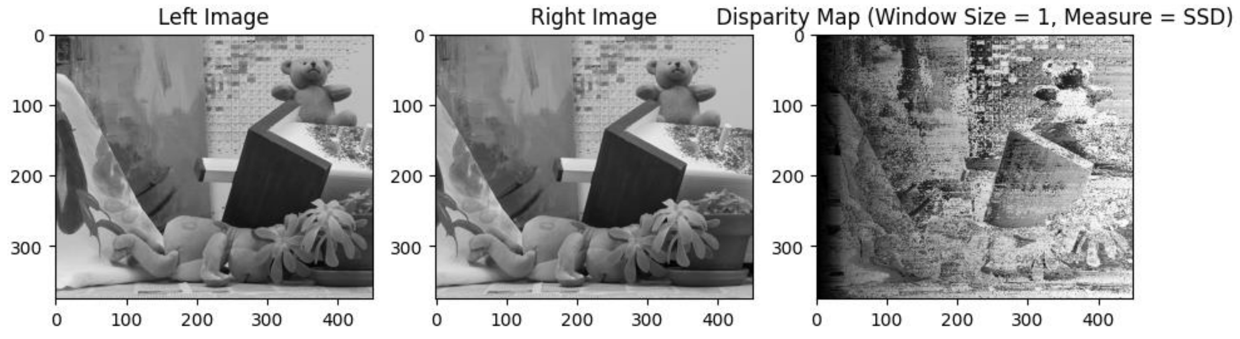

Left Image
Right Image
Disparity Map (Window Size = 5, Measure = SSD)

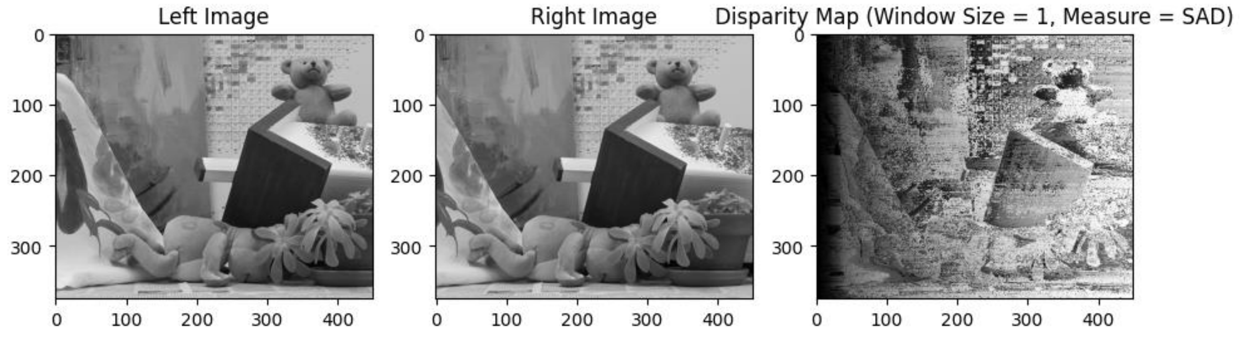

Left Image
Right Image
Disparity Map (Window Size = 5, Measure = SAD)

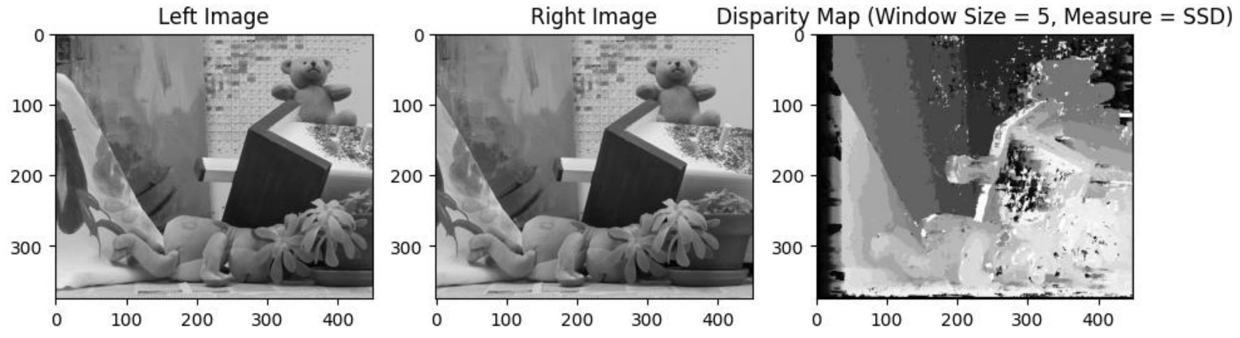

Left Image
Right Image
Disparity Map (Window Size = 9, Measure = SSD)

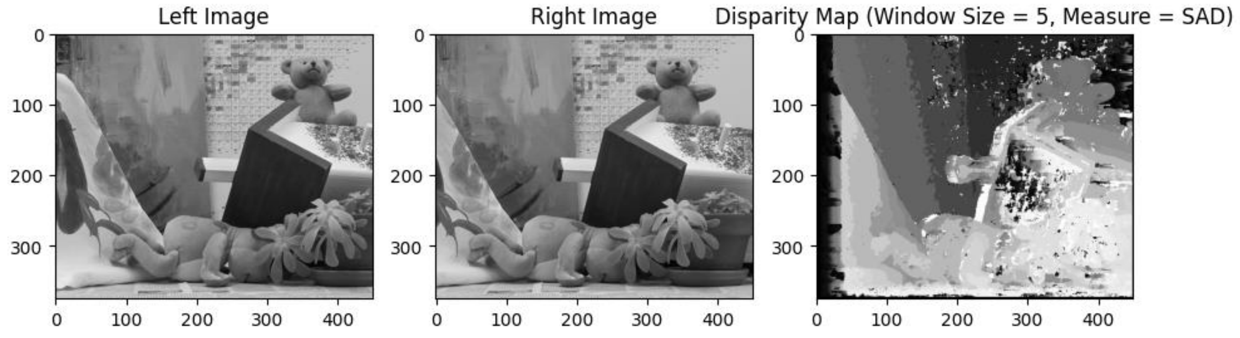

Left Image
Right Image
Disparity Map (Window Size = 9, Measure = SAD)

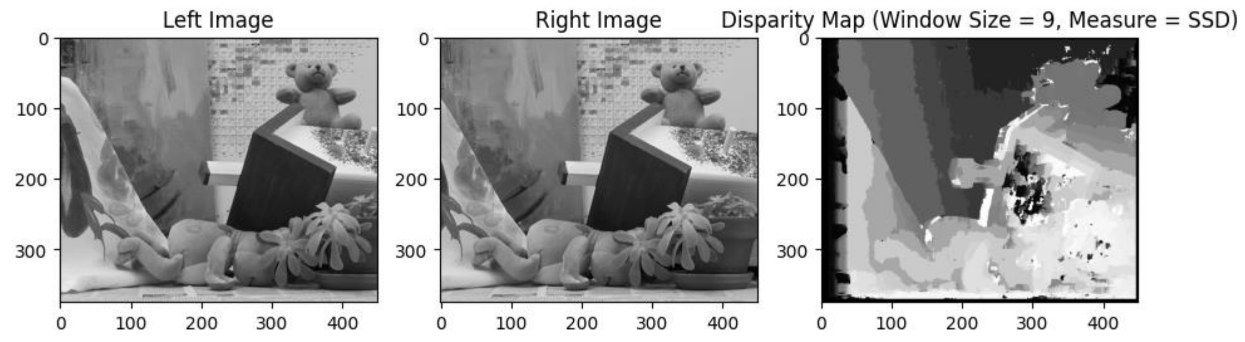
Third image with window size (1, 5, 9) and measuring (SSD, SAD)

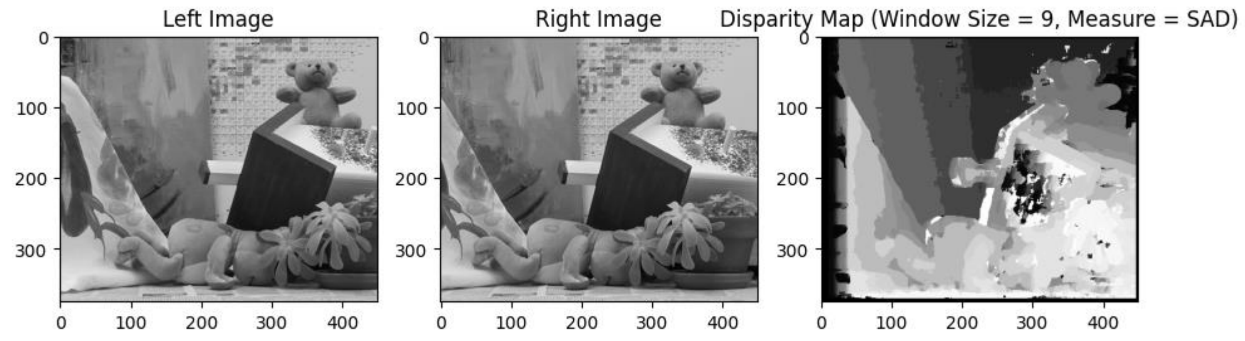









Left Image
Right Image
Disparity Map (Window Size = 9, Measure = SSD)
0
100
200
300
400


Left Image
Right Image
Disparity Map (Window Size = 9, Measure = SAD)
0
100
200
300
400

Part 2: Dynamic Programming:

1. Problem Statement:

In this section of the assignment, Implement dynamic programming for computing disparity between two scanlines I_l(i) and I_r(j). The cost of matching pixels is calculated using squared error, and skipping a pixel incurs a constant cost c_0. Utilize a recursive algorithm to compute the optimal alignment stored in matrix D. Backtrack to find the minimum cost alignment, assigning disparity values based on absolute differences. The approach accommodates left-to-right and right-to-left matching, producing disparity maps for both. Given constants σ = 2 and c_0 = 1, the algorithm is summarized by the relation:

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

Generate disparity maps considering pixel matching costs and occlusions.

2. Code Explanation and process steps:

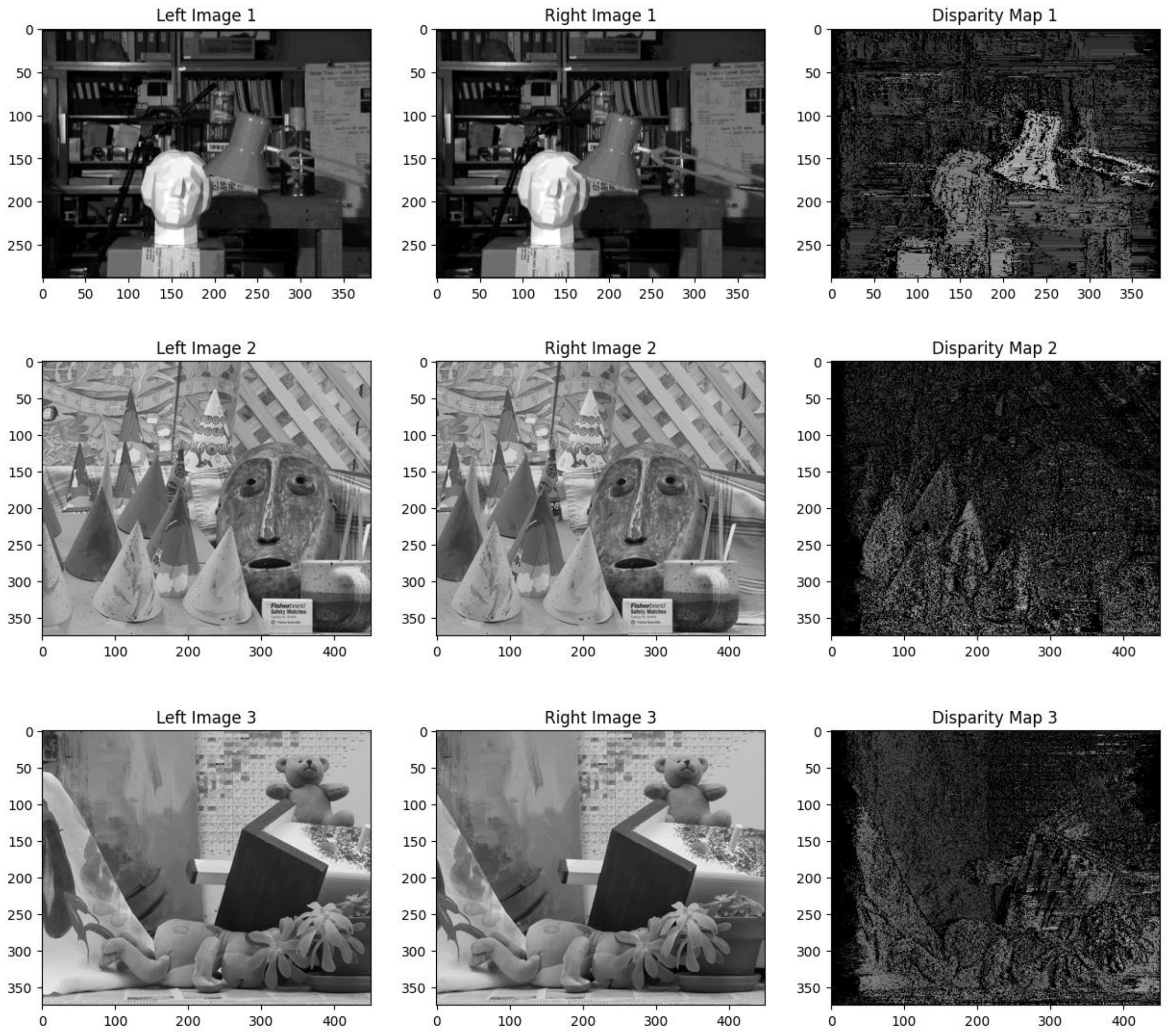
a. Libraries needed: (cv2, numpy, pandas, matplotlib)

b. “get_cost” Function:
- Calculates the cost of matching pixels I_l and I_r based on squared error, considering pixel noise σ and return the squared error.

c. “compute_disparity_dp” Function:
- Takes left and right images, along with parameters for pixel noise (σ) and constant cost (c_0).
- Initializes matrices for disparity and dynamic programming (dp).
- Iterates over each row of the left image:
  - Initializes the dynamic programming matrix.
  - Computes the optimal alignment using the recursive relation.
  - Backtracks to find the optimal alignment, updating the disparity map.
- Returns the normalized disparity map and binary disparity map.

d. Main process:
- Loads left and right images for each pair (l1, r1), (l2, r2), (l3, r3).
- Calls “compute_disparity_dp” for each pair, obtaining the disparity map and binary disparity map.

3. Output of test images:



Part 3: Bonus:

1. Problem Statement:

As a bonus, visualize the dynamic programming alignment for a single scan line by plotting a graph of I_l vs I_r. Start at D(N, N) and work backward to (1, 1). Represent the alignment with:

- Vertical lines for skipped pixels in I_l.
- Horizontal lines for skipped pixels in I_r.
- Diagonal lines for matched pixels.

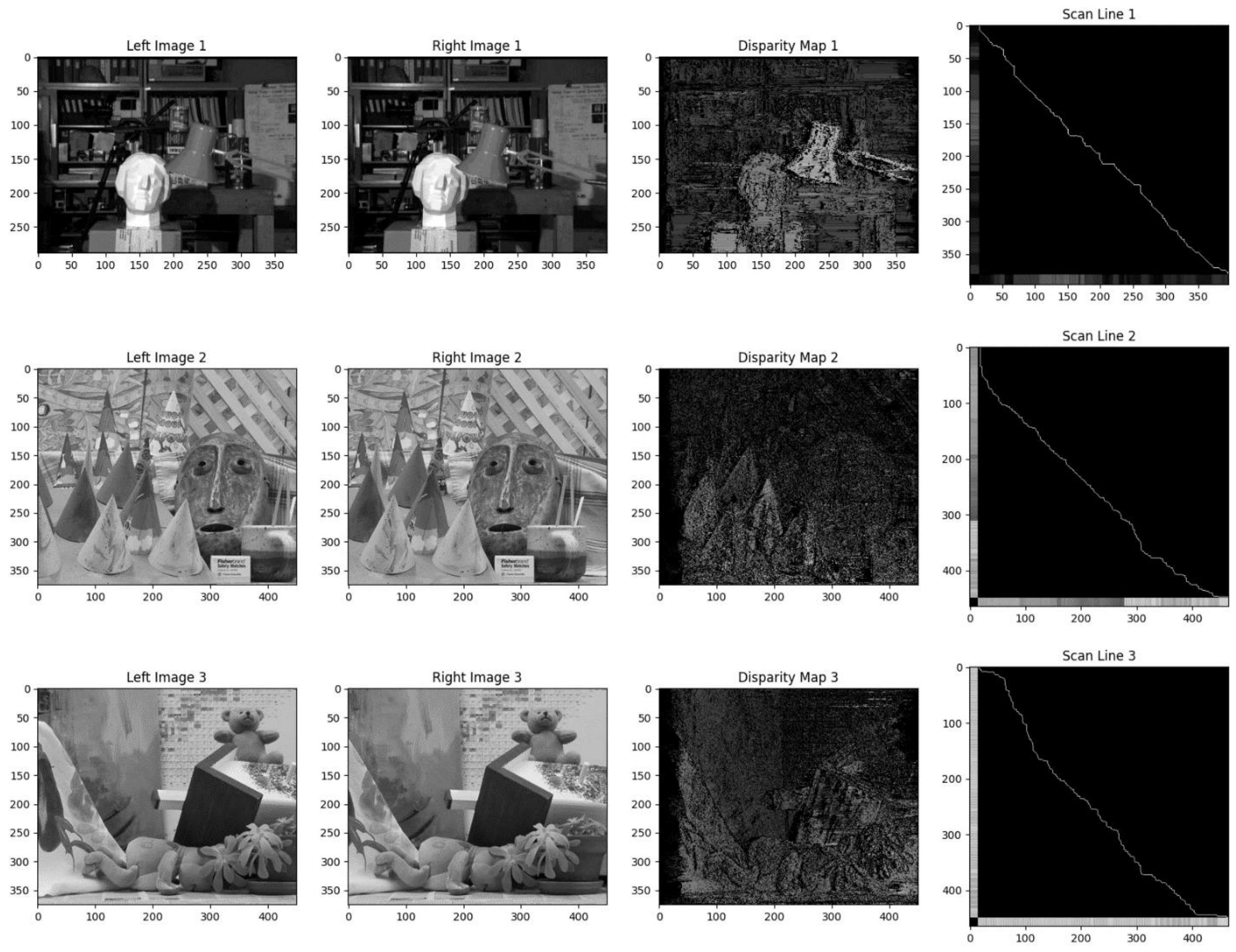
This graphical representation enhances the interpretation of the disparity calculation, providing a visual insight into the alignment decisions made during the dynamic programming process. The plot should conclude at (1, 1).

2. Code Explanation and process steps:

a. “get_scan_line” Function:
   - Takes the left image, right image, and disparity map as input.
   - Constructs a scan line for visualization by concatenating the last row of the left image and the disparity map on the left side, and zeros along with the last row of the right image on the right side.
   - Concatenates the left and right parts to form the complete scan line.

b. Main process:
   - Iterates over the three image pairs (l1, r1), (l2, r2), (l3, r3). For each pair, plots the left image, right image, and the corresponding disparity map, Additionally, plots the scan line obtained using the `get_scan_line` function.

3. Output of test images:


Left Image 1
Right Image 1
Disparity Map 1
Scan Line 1
Left Image 2
Right Image 2
Disparity Map 2
Scan Line 2
Left Image 3
Right Image 3
Disparity Map 3
Scan Line 3