Here is a complete Python solution that implements both parts of your stereo-vision assignment: (1) block-matching with SAD and SSD for window sizes 1, 5, 9, and (2) scanline dynamic-programming stereo. Finally it plots the optimal alignment for one scanline.

**Summary of the approach**

* **Block Matching (Sec 1.1.1)**
  + For each pixel in the left image, we slide a w×w window along the same row in the right image over a disparity range and compute either the Sum of Absolute Differences (SAD) or the Sum of Squared Differences (SSD) cost.
  + We do this for w={1,5,9}, yielding 6 disparity maps.
* **Dynamic Programming (Sec 1.1.2)**
  + For each scanline (row), we build an N×N cost matrix D where D(i,j) = min of matching Il[i]–Ir[j] (squared error) or skipping a pixel (occlusion penalty c₀=1) [Cheriton School of Computer Science](https://cs.uwaterloo.ca/~mannr/cs787-w03/cs787-a3.pdf?utm_source=chatgpt.com).
  + We backtrack from D(N,N) to recover left and right disparity maps.
* **Visualization (Sec 1.2)**
  + We plot the alignment path on a 2D graph: vertical steps = skip in Il, horizontal = skip in Ir, diagonal = match [CMU School of Computer Science](https://www.cs.cmu.edu/~16385/s17/Slides/13.2_Stereo_Matching.pdf?utm_source=chatgpt.com).

 We slide a (2w+1)×(2w+1) window along the epipolar line and compute SAD or SSD cost [CMU School of Computer Science](https://www.cs.cmu.edu/~16385/s17/Slides/13.2_Stereo_Matching.pdf?utm_source=chatgpt.com).

 We repeat for window radii w={0,2,4} corresponding to sizes 1, 5, 9.

* + We follow the recurrence  
    D(i,j)=min{D(i−1,j−1)+dij, D(i−1,j)+c₀, D(i,j−1)+c₀} with dᵢⱼ=((Il(i)−Ir(j))²/σ²) and c₀=1
  + This draws vertical steps when Il is skipped, horizontal when Ir is skipped, and diagonal for matches