Dynamic Programming:

To decrease runtime, we started using dynamic programming to calculate disparity by getting minimum cost of matching a whole row in the image.
We used top-down approach with memorization to keep track of pre calculated terms in our concerns.

1) We start by calling **minimum_cost_initialization ()** to initialize an empty matrix with size N x N (where N is columns count in image) for storing every cost calculated, and then iterating over image rows to return minimum cost.

2) Within same loop we calculate disparity for each scanline for both left and right stereo images and appending them to disparity matrix using **calc_disparity ()**

minimum_cost_initialization () :

$D(1, 1) = d11$ (base case)

$D(i, j) = \min(D(i - 1, j - 1) + dij , D(i - 1, j) + c0, D(i, j - 1) + c0)$ //Recursion formula

Note: special cases as pixels at image border are handled within implementation

calc_disparity ():

starting from I and j (N x N) through all the matrix

(i-1, j-1)	(i, j-1)
(i-1, j)	(i, j)

Getting minimum value around (i, j):

- Selecting (i - 1, j) corresponds to skipping a pixel in $I_{(left)}$, so the left disparity map of i is zero.
- Selecting (i, j - 1) corresponds to skipping a pixel in $I_{(right)}$, and the right disparity map of j is zero.
- Selecting (i -1, j - 1) matches pixels (i, j), and therefore both disparity maps at this position are set to the absolute difference between i and j.

Note: we initialized both left and right disparity with zeros for better run-time.

Bonus:
Plotting path taken by algorithm for matching a single line starting from N, N going through all matrix to reach 1,1

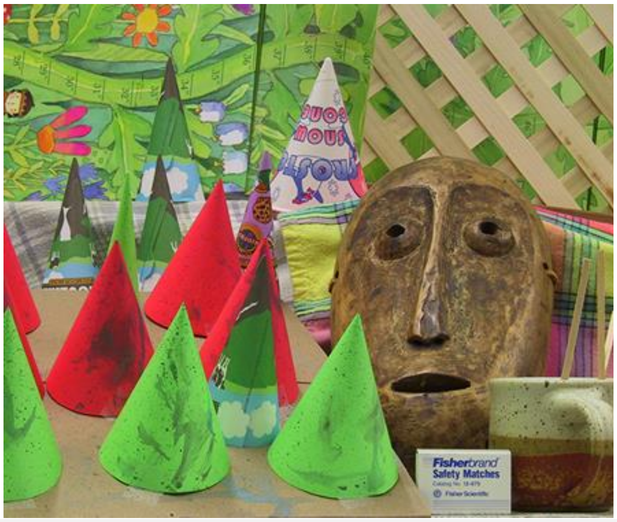
Within same loop of main algorithm:
First initializing a 2D list of zeros and then setting visited pixels by 255 to show path taken for a single scanline.

Note: we plot the path for last scanline by default

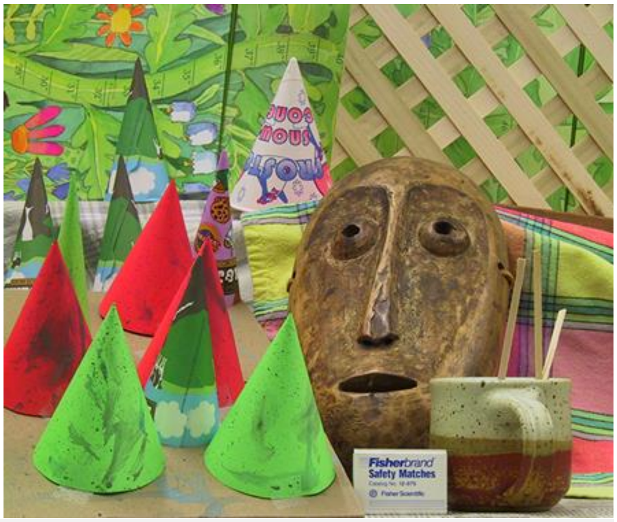
Test Cases:

1)

Inputs

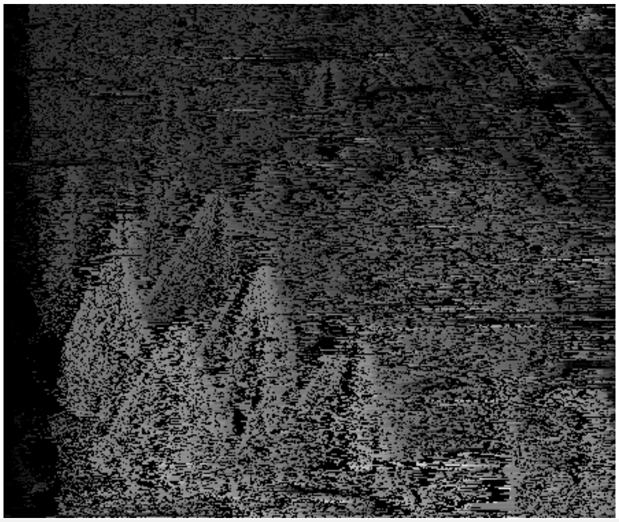


Left

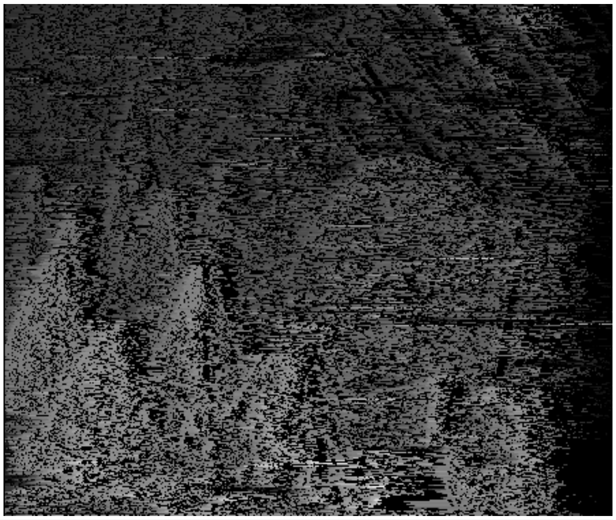


Right

Outputs

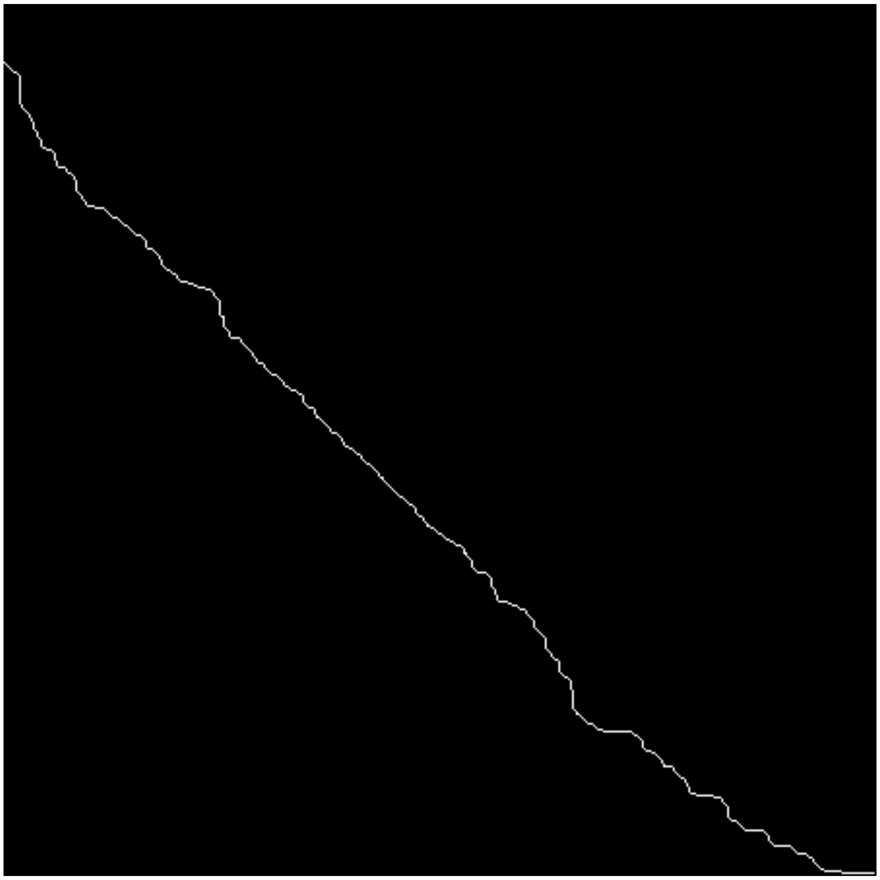


Left



Right

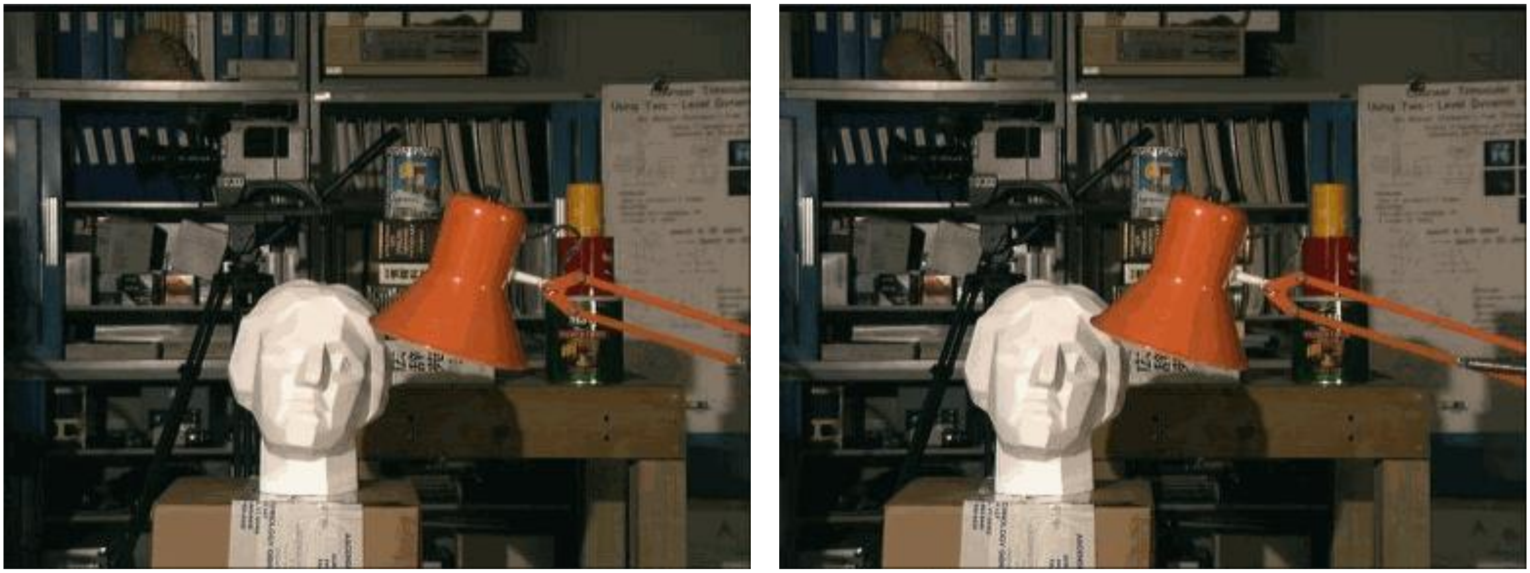
Route (Bonus)



Route of last scanline

2)

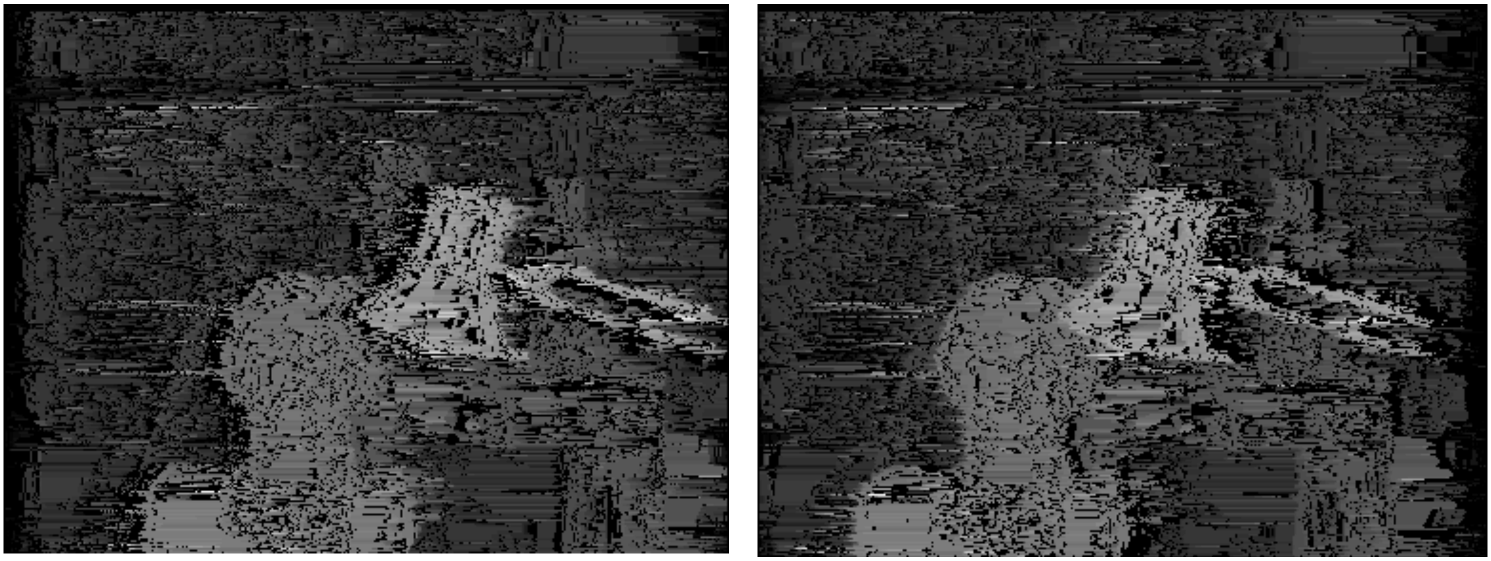
Inputs



Left

Right

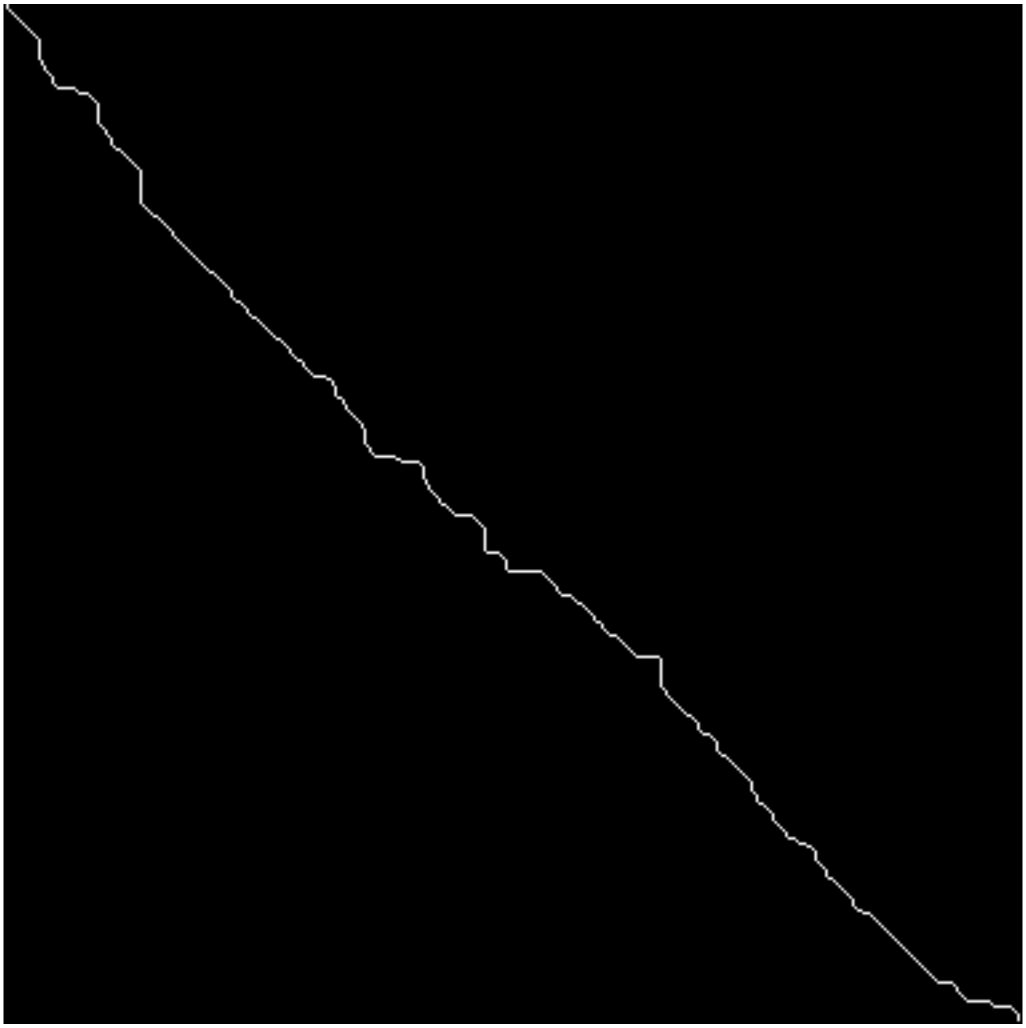
Outputs



Left

Right

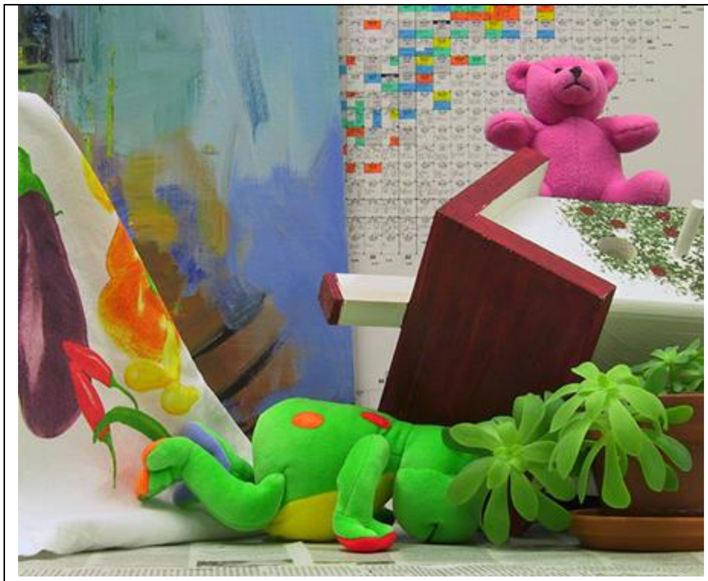
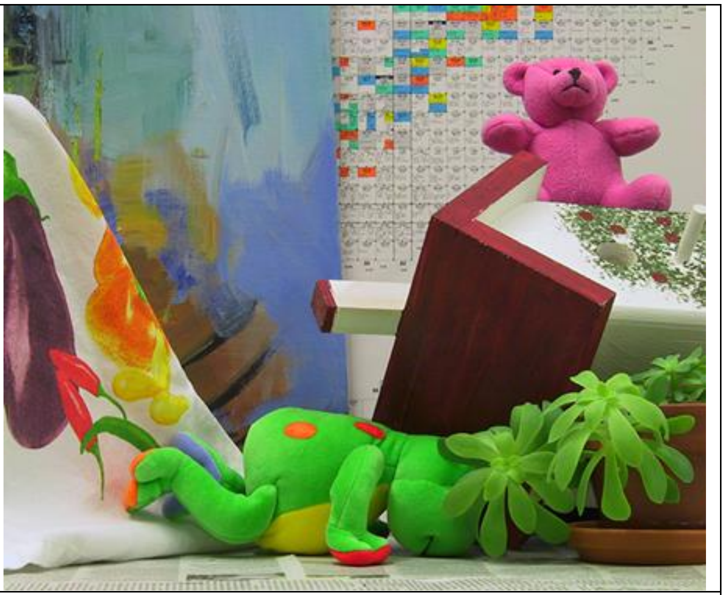
Route (Bonus)



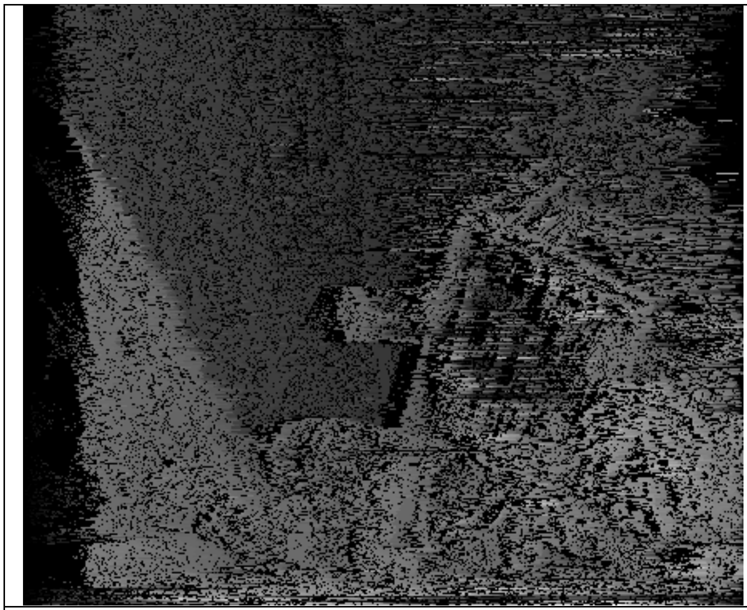
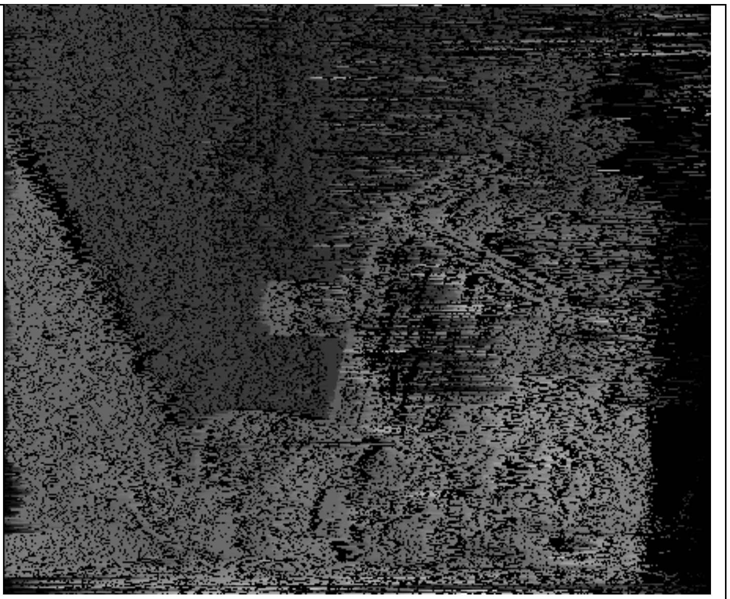
Route of last scanline

3)

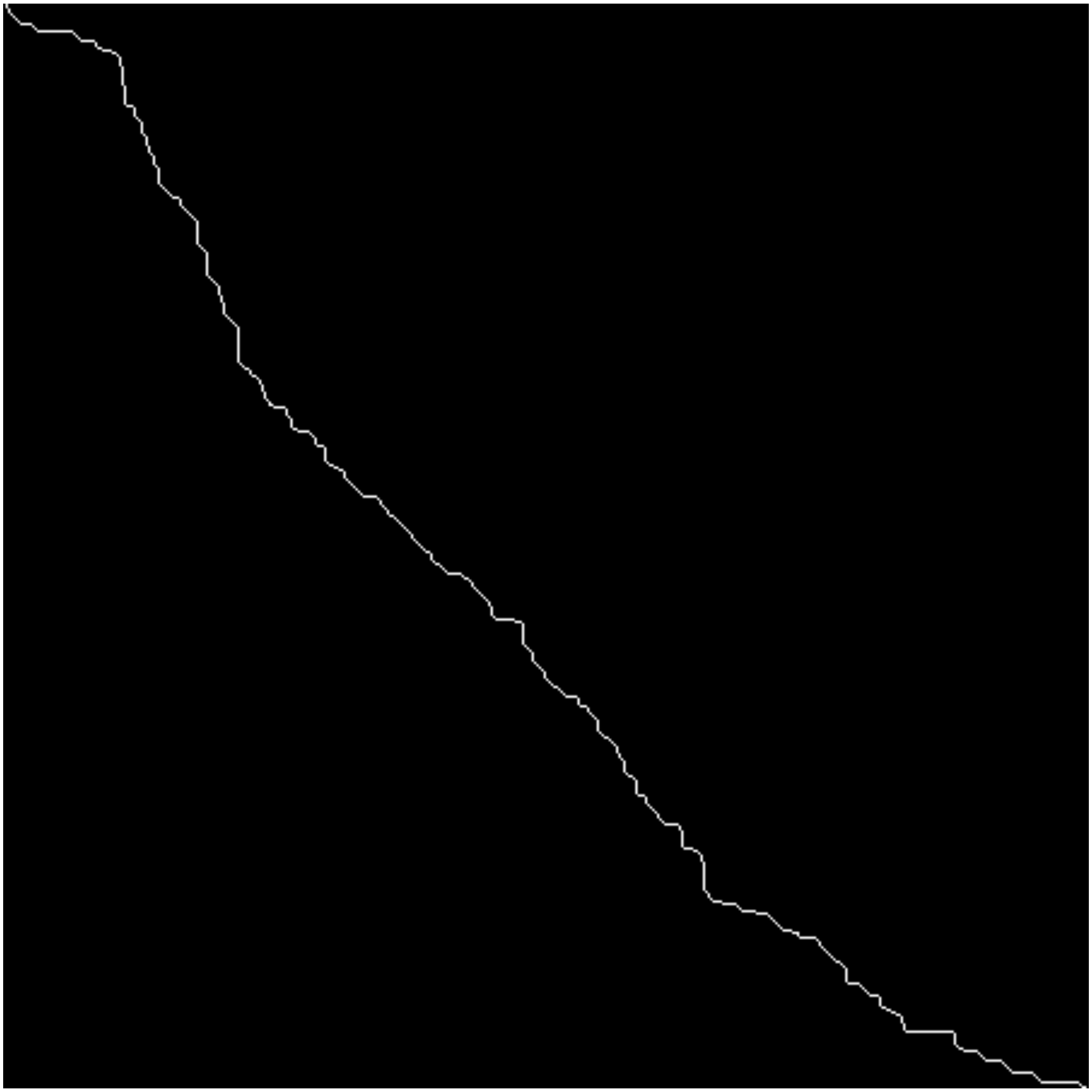
Inputs

	
Left	**Right**

Outputs

	
Left	**Right**

Route (Bonus)



Route of last scanline