Project 2 (one more time - without no match)

- The dynamic programming algorithm needs the following definitions:

  I_L is the left row; I_R the right row, with n pixels per row.

  $$SAD(x, x') = \sum_{i=-r}^{r} \sum_{j=-r}^{r} \left| I_L(x) - I_R(x') \right|$$

  If we let k be the radius of the search interval we will use around the input estimate of disparity, D_{in}, then we can define the match score matrix, M, as a (2k+1) x (n-2r) matrix with rows indexed from -k to k:

  M(offset, column)

  - maxint if D_{in}(column) + offset < 0
  - SAD(column, D_{in}(column) + offset) otherwise
  - Define Conjugate(offset,column) = column+ D_{in}(column) + offset

Last time around

- A disparity map, D, is a mapping from (r, n-r) - pixel positions - to (0, maxdis+1) - disparities. D must satisfy
  - j < j' implies j + D(j) <= j' +D(j')
- The Cost of a disparity map is defined as

  $$Cost(D) = \sum_{j=r}^{n-r} M(D(j) - D_{in}(j), j)$$

- Goal is to find the disparity map, D*, that has minimal cost

Home stretch

- Define **Best(offset,col)** to be the cost of the minimal cost disparity map for the first col columns of I_L constrained to assign disparity D_{in}(col) + offset to col.
- The cost of the best disparity map will then be:

$$\min_{-k \le offset \le k} Best(offset, n-r)$$

The finish

- Now we need a recursive definition of Best, and we are done.

$$Best(offset, r) = M(offset, r)$$
$$Best(offset, col) = M(offset, col) + \min_{offset' \ni conjugate(offset', col-1) \le conjugate(offset, col)} \{Best(offset', col-1)\}$$

The Delivery

- Automatic grading of the dynamic programming algorithm:
  - Given: Match matrix
  - Output: Best disparity map and its score
  - Due: May 17
  - Value: 35%
- Report
  - Describe implementation
  - Compare output disparity map to ground truth
  - Extra credit - implement no-match algorithm
  - Due: May 21 - no late submissions
  - Value: 65% + 25%