CS 556: Computer Vision

Lecture 17

Prof. Sinisa Todorovic

sinisa@eecs.oregonstate.edu

Oregon State University

Outline

- Edges, Shapes, Contours, Curves, Boundaries
- Shape descriptors

Image Features -- Edges

- Sharp changes in image brightness
- Interesting because:
    - Object boundaries often coincide with edges
    - Changes in surface orientation give rise to edges

Detecting Edges

- Find the gradient of pixel intensity values
- Find pixels whose gradient magnitude is large wrt neighbors

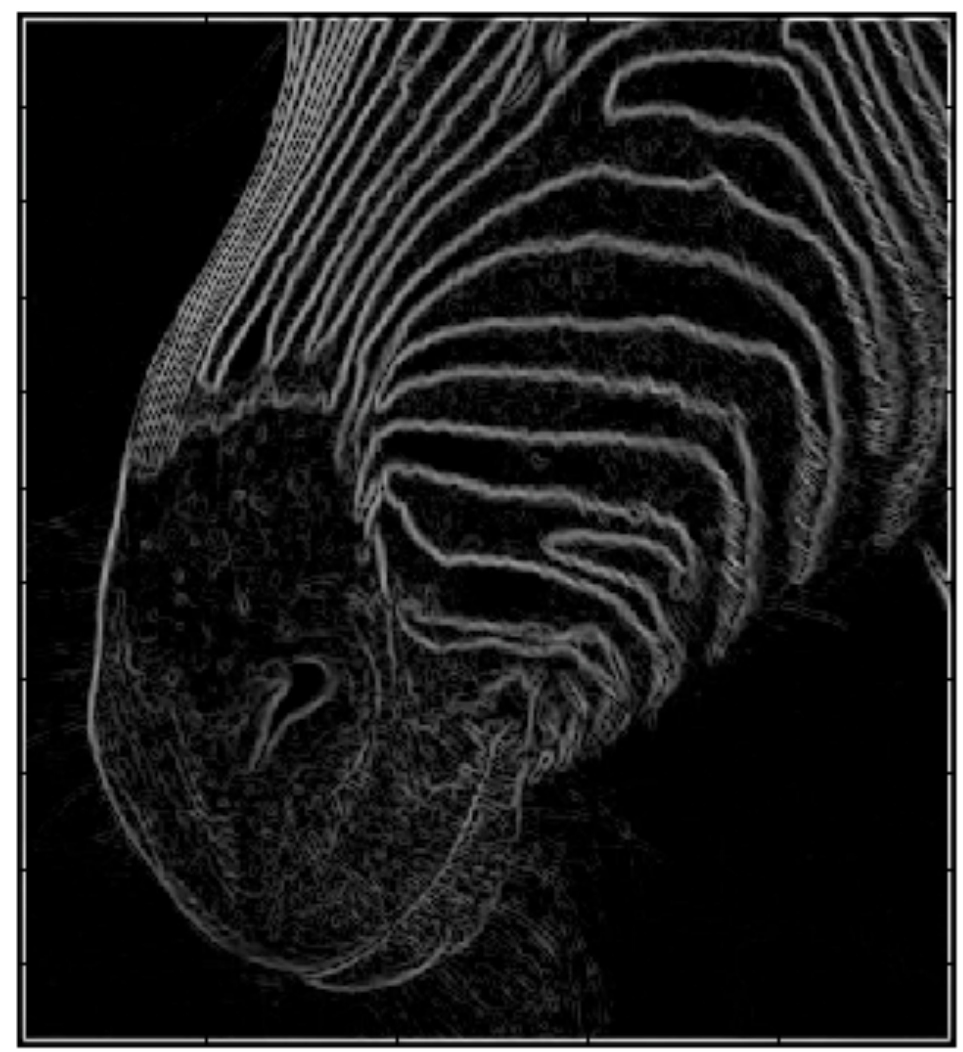


Image Gradient along Image Axes -- Example

$$\frac{\partial I(x,y)}{\partial x} \approx \frac{I(x+1,y) - I(x-1,y)}{2}$$

$$\approx \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix} * I(x,y)$$

$$\approx D_x * I(x,y)$$

filter

Differentiation

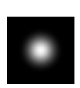
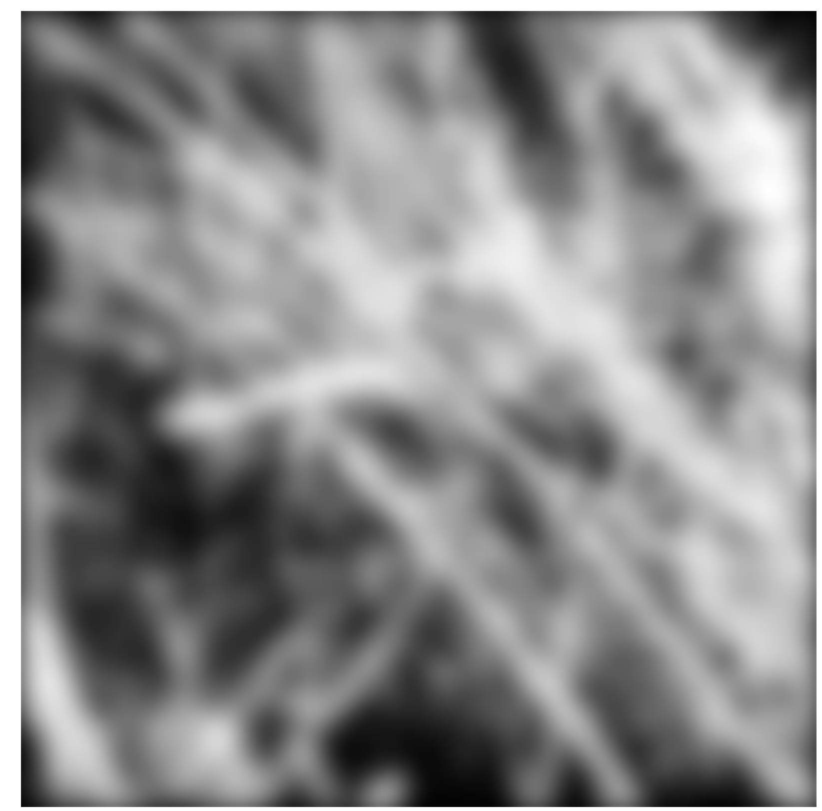
- Differentiation is filtering = Convolution

$$I_x \approx D_x * I$$

- Eliminate high frequencies $\Rightarrow$ Smooth before differentiation
  - First, differentiate the smoothing filter
  - Then, convolution with the differentiated smoothing filter

$$D_x * (G * I) = (D_x * G) * I = G_x * I$$

Example -- Smoothing with a Gaussian

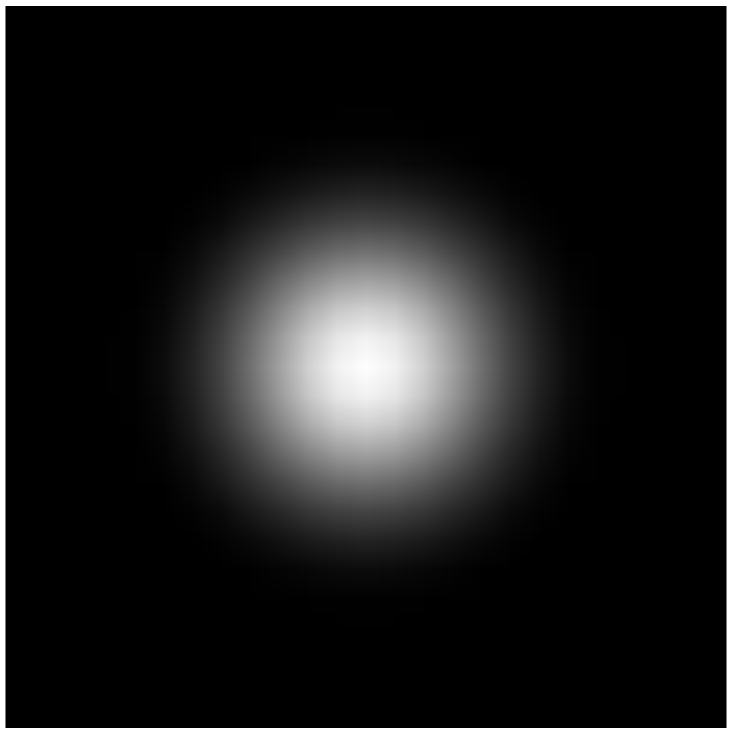
 *  = 

Gaussian filter

input

output

Gaussian Smoothing Filter



$$G_\sigma(x, y) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

- Sigma defines scale -- Spatial sensitivity
- Popular due to many convenient properties

$$G_\sigma(x, y) = G_\sigma(x) \cdot G_\sigma(y)$$

Detecting Edges -- Issues

- How to select a sufficiently large gradient magnitude?
- Gradient magnitude is not constant.
- How to link detected edge points into a curve?

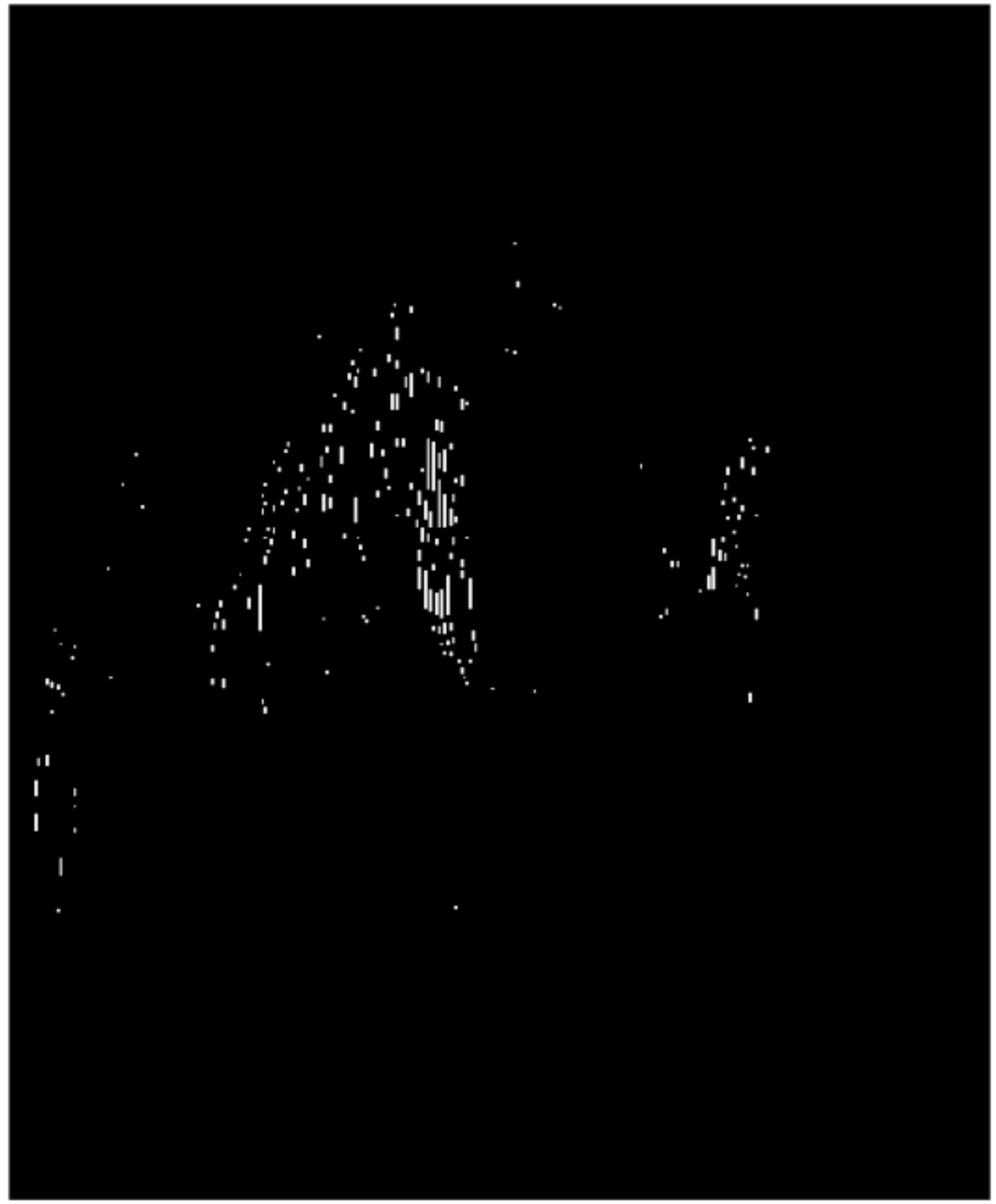
Canny Edge Detection

- After smoothing and differentiating
- Set the threshold high to initiate edge detection
- Lower the threshold to track the edges
- Hysteresis

http://www.cs.ucf.edu/~mikel/Research/Edge_Detection.htm

Canny Edge Detection

input

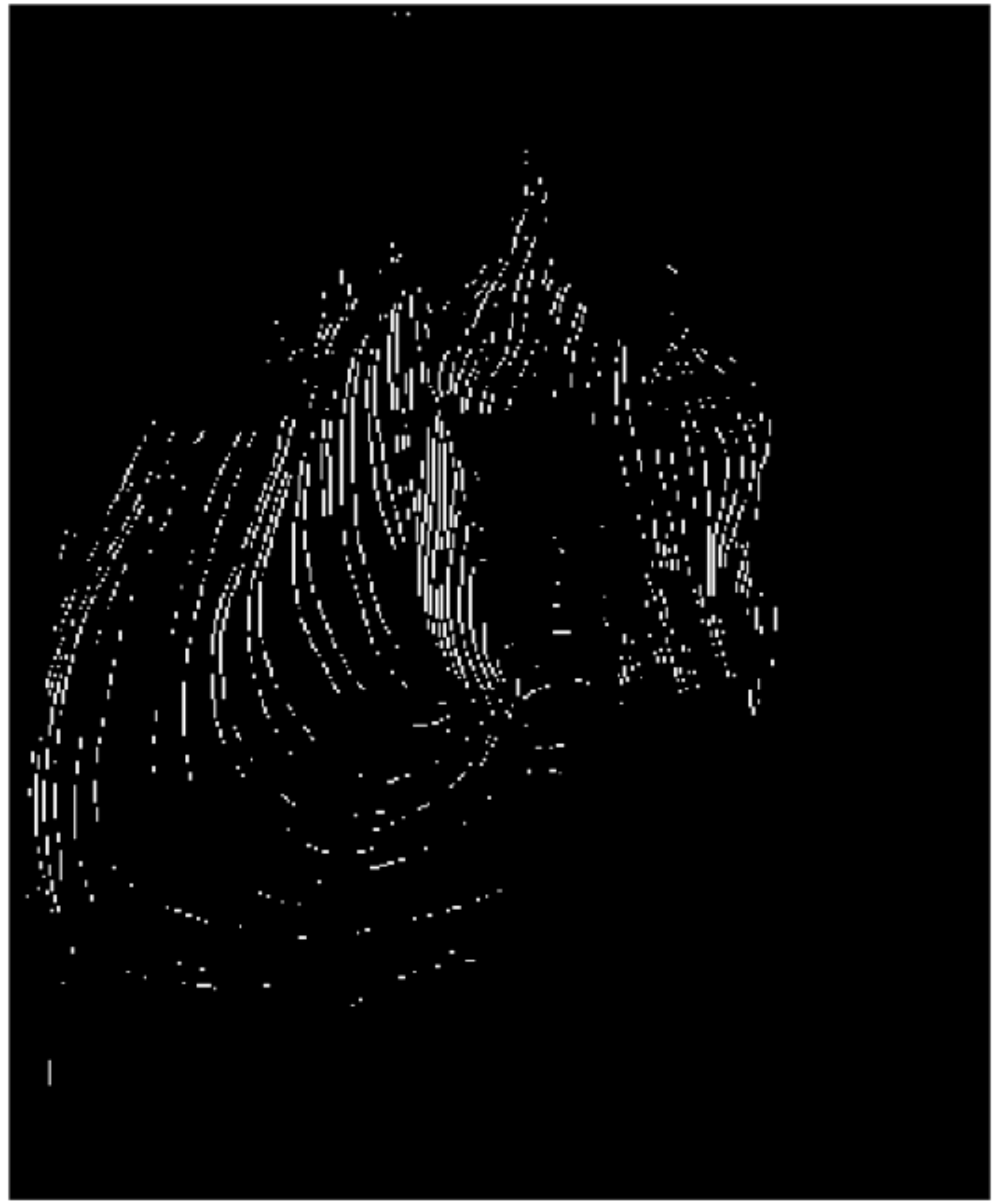


thresholding the image gradient

- Set the threshold high to initiate edge detection
- Lower the threshold to track the edges

Canny Edge Detection

input

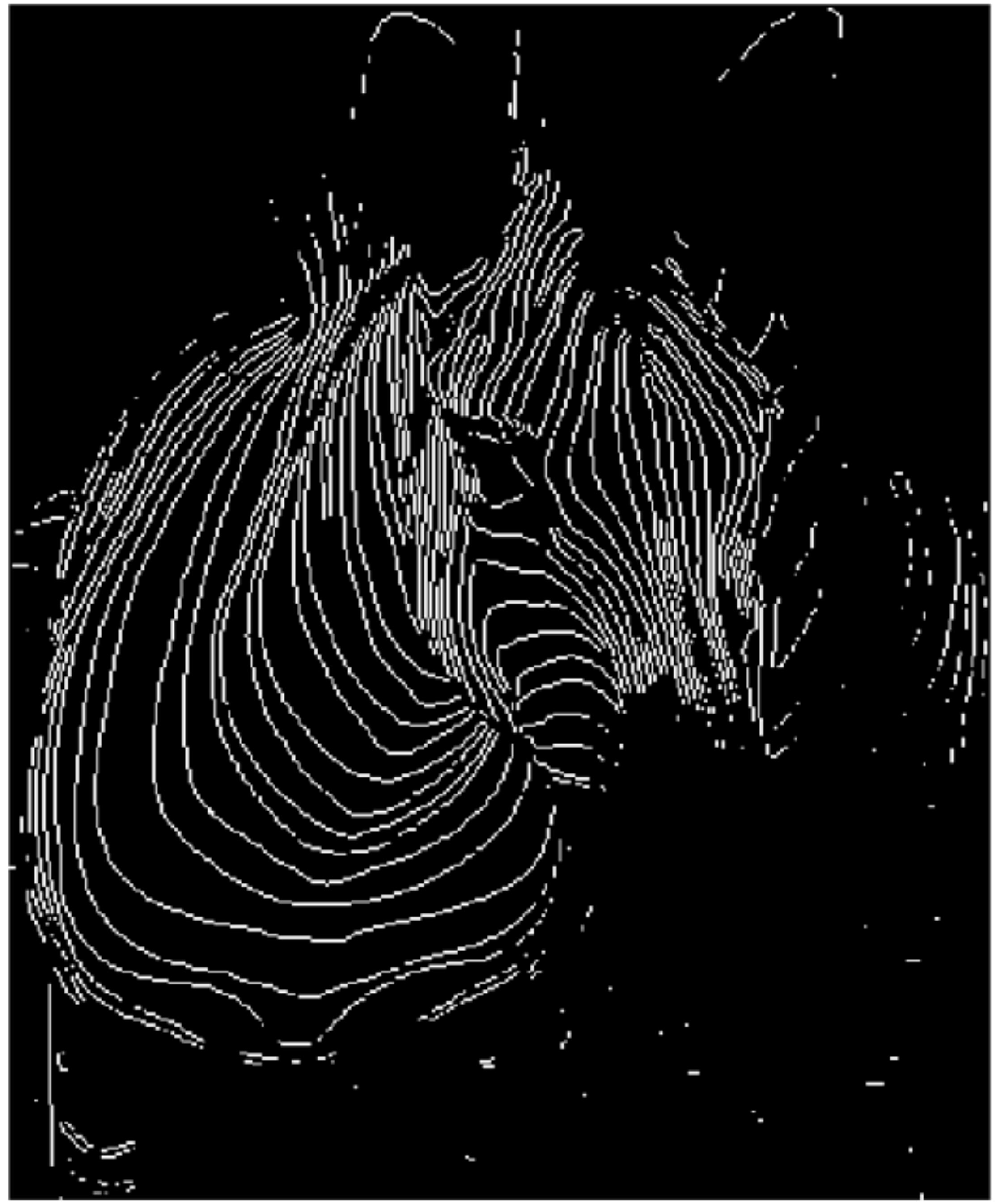


thresholding the image gradient

- Set the threshold high to initiate edge detection
- Lower the threshold to track the edges

Canny Edge Detection

input

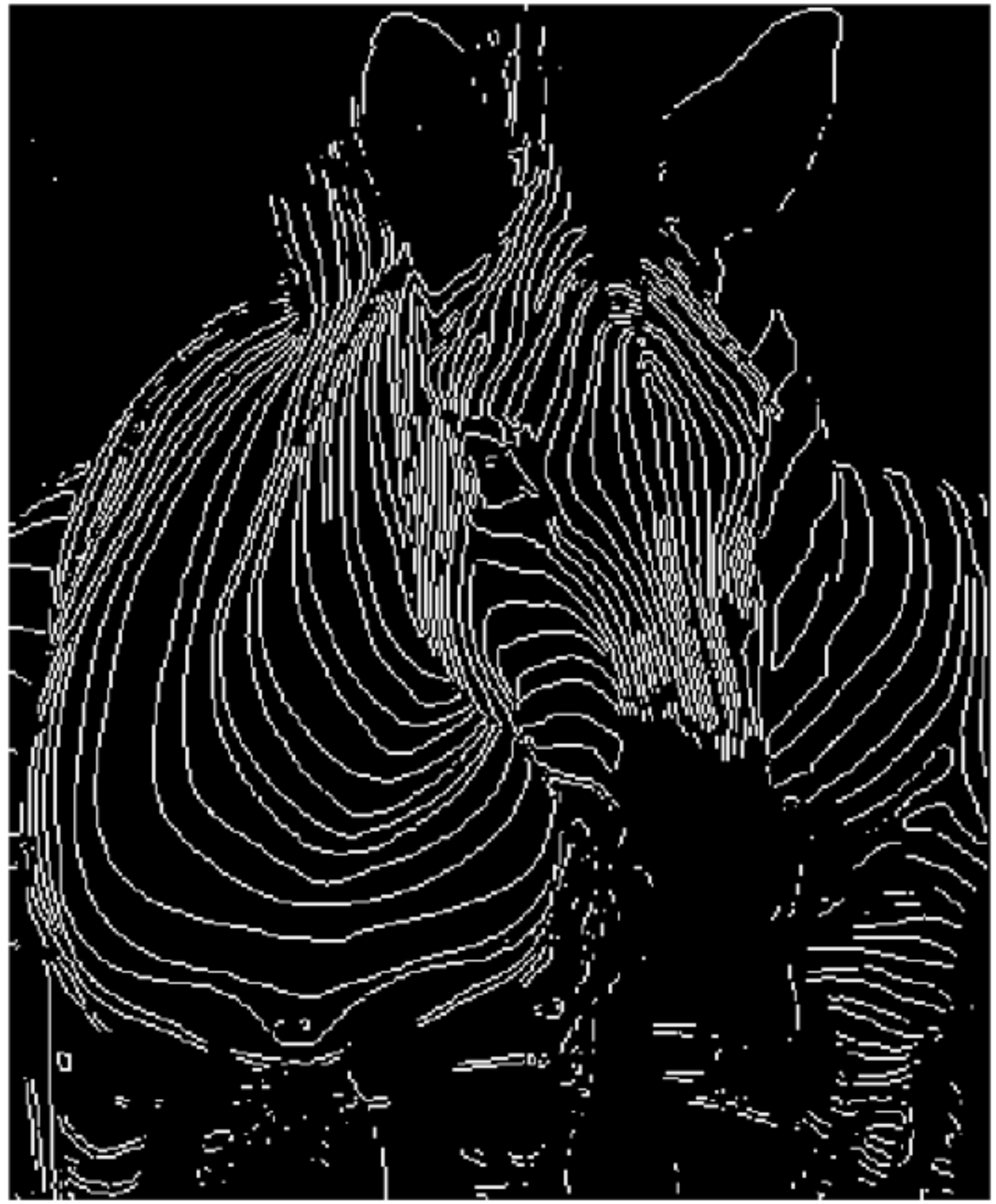


thresholding the image gradient

- Set the threshold high to initiate edge detection
- Lower the threshold to track the edges

Canny Edge Detection

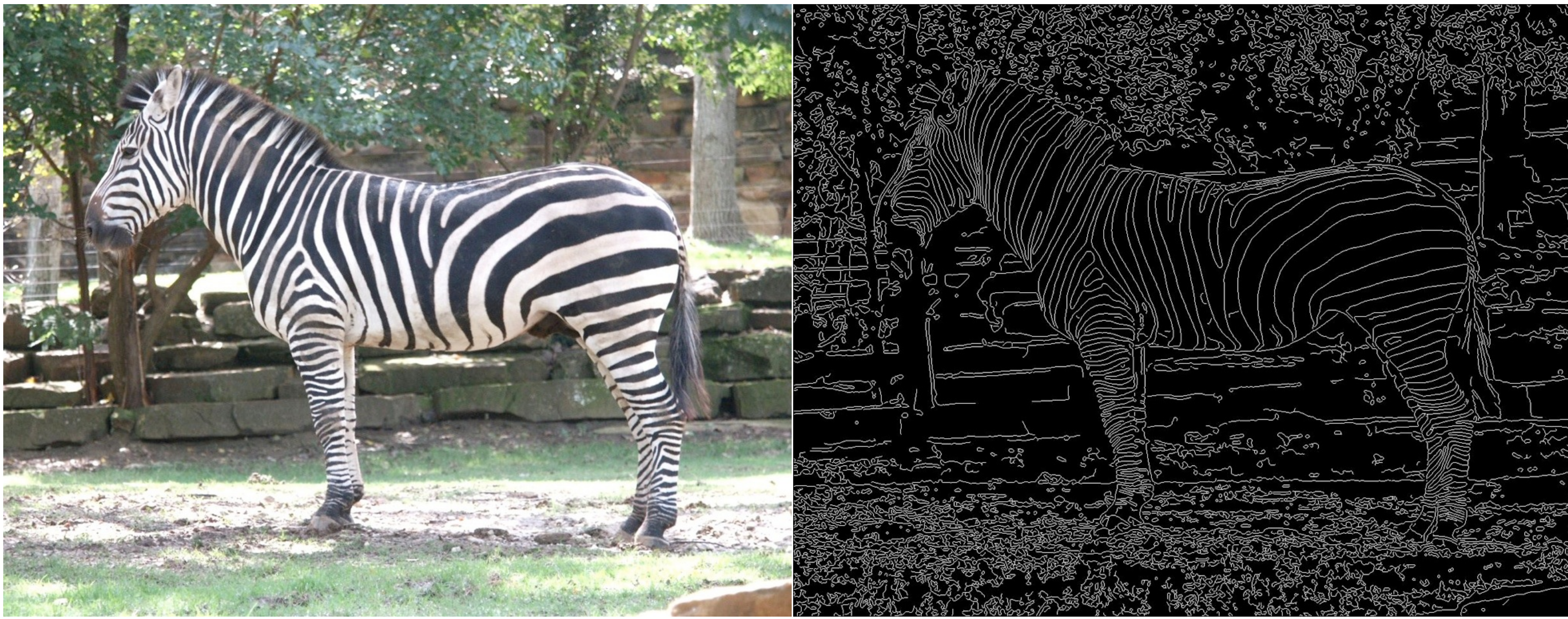
input



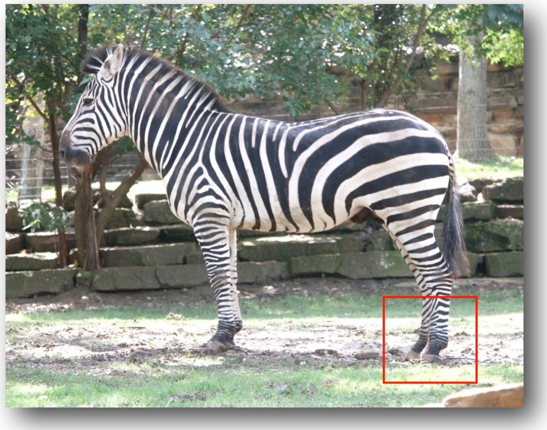
thresholding the image gradient

- Set the threshold high to initiate edge detection
- Lower the threshold to track the edges

Canny Edge Detector



Smoothing -- Bad for Detecting Corners



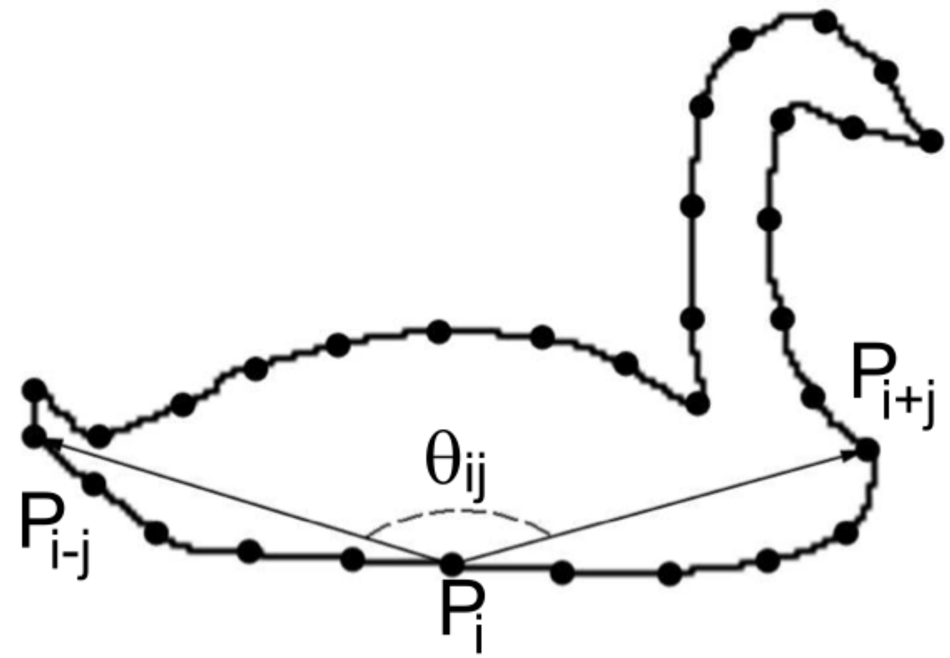
input

zoomed in image part

corresponding Canny edge detection

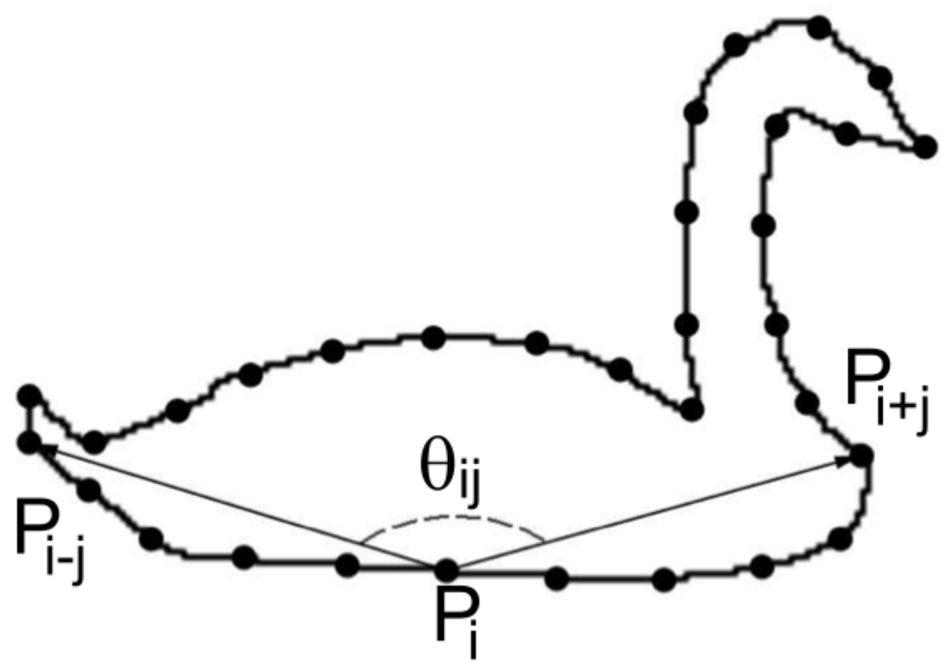
Edge Descriptors

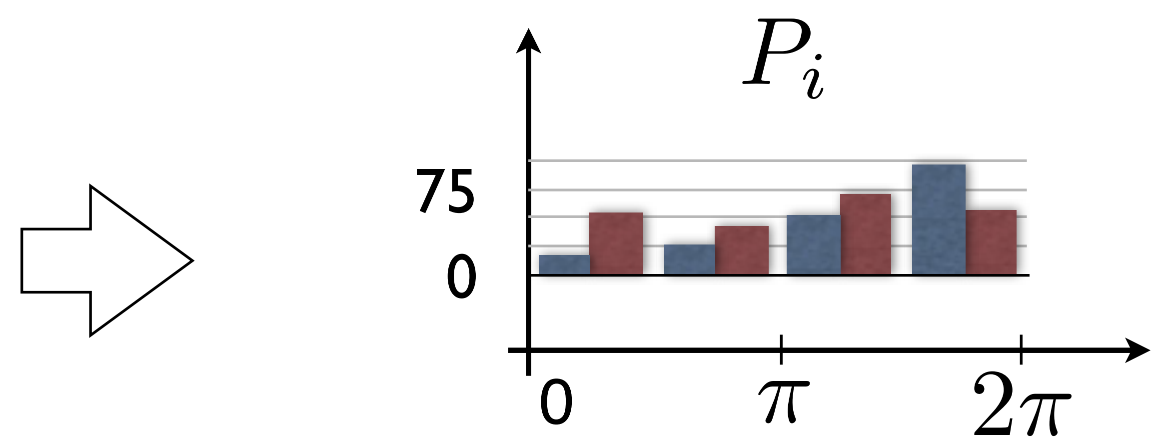
Shape as a Sequence of Points



- Shape = Sequence of points along the shape
  - Regular sampling
  - High-curvature points

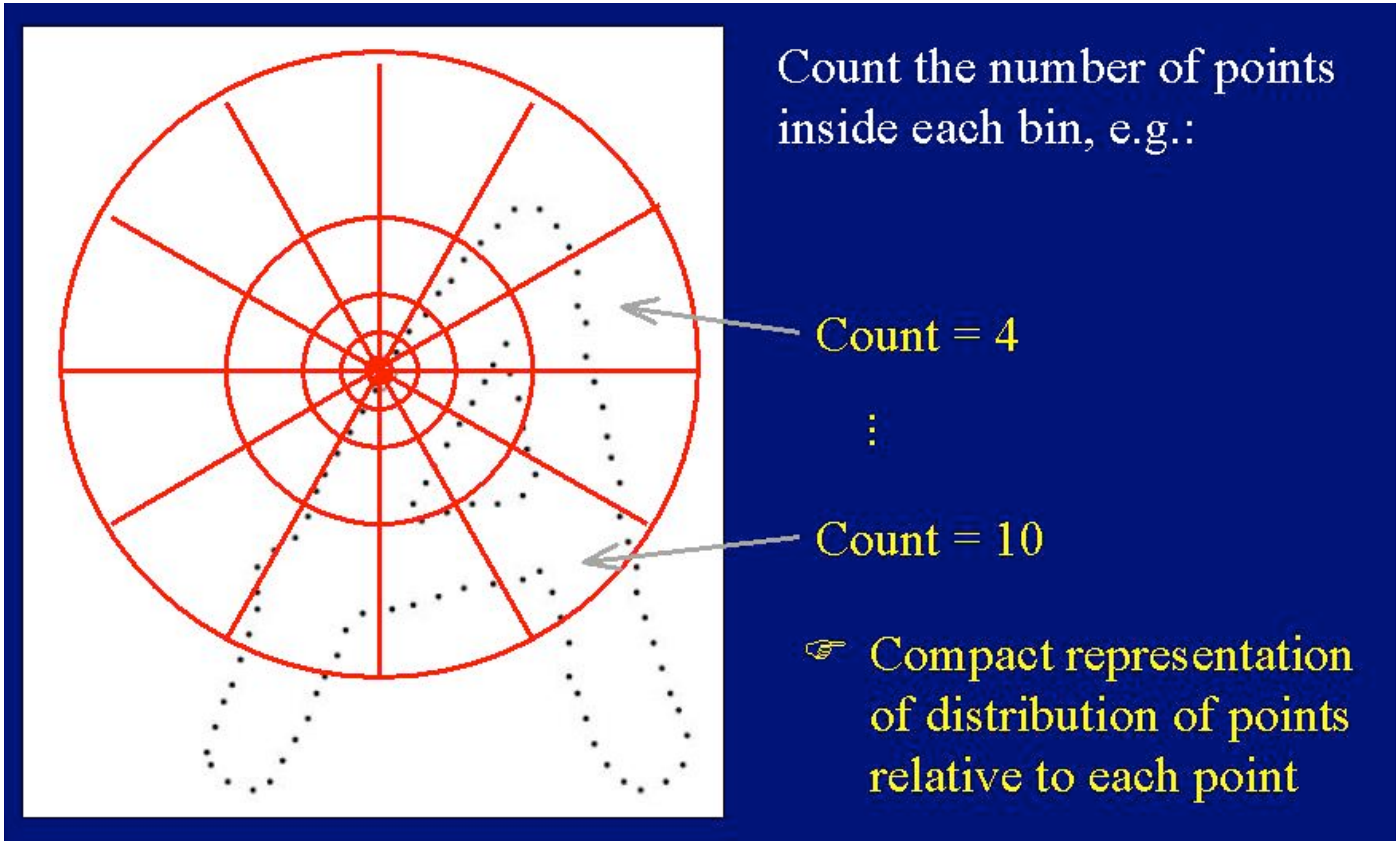
Beam-Angle Descriptor

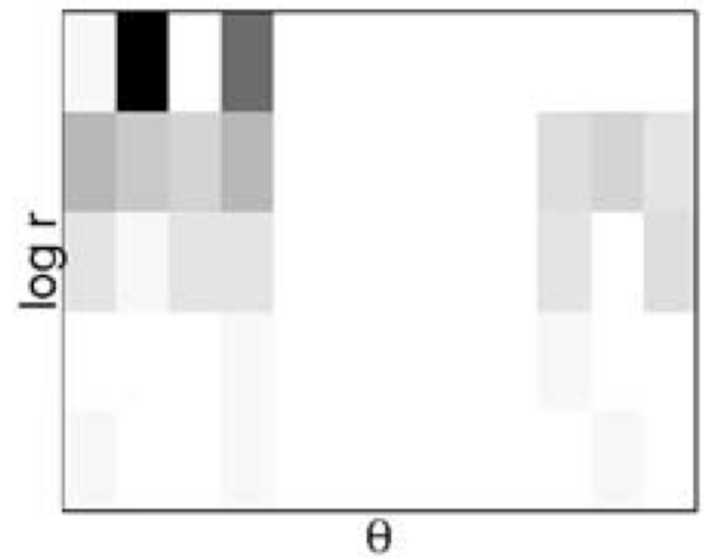




- Shape = Sequence of points along the shape
- Each point is characterized by a descriptor:
  - Beam-angle

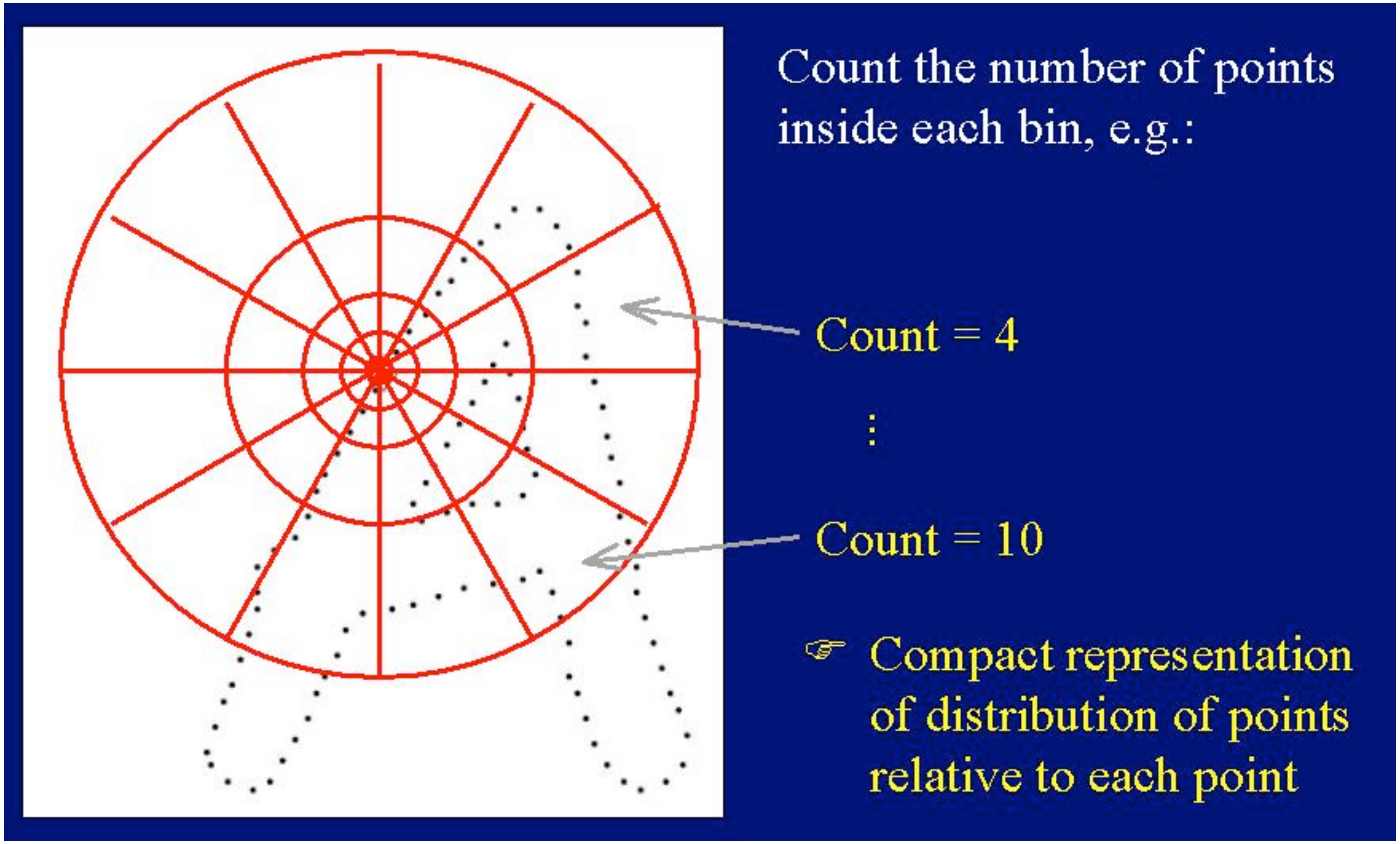
Edge Descriptor -- Shape Context

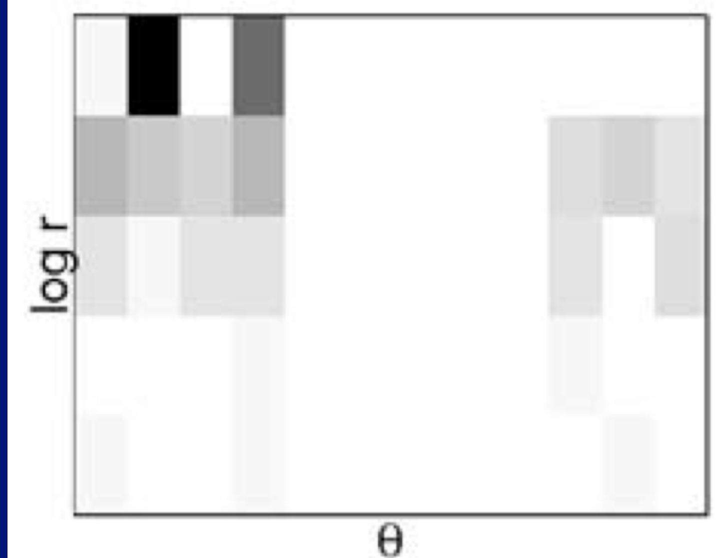




1. Shape = Sequence of points
2. For each point compute log-polar histogram of other points in the sequence

Shape Context





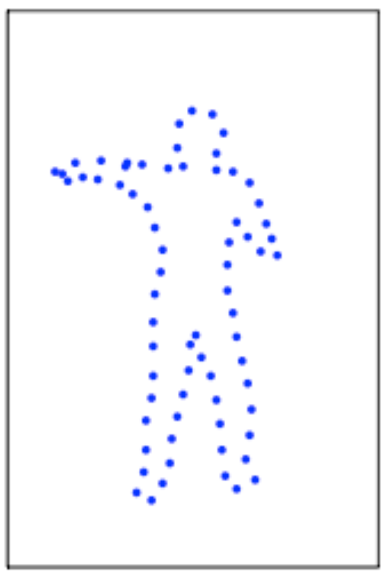
How to identify the reference direction?

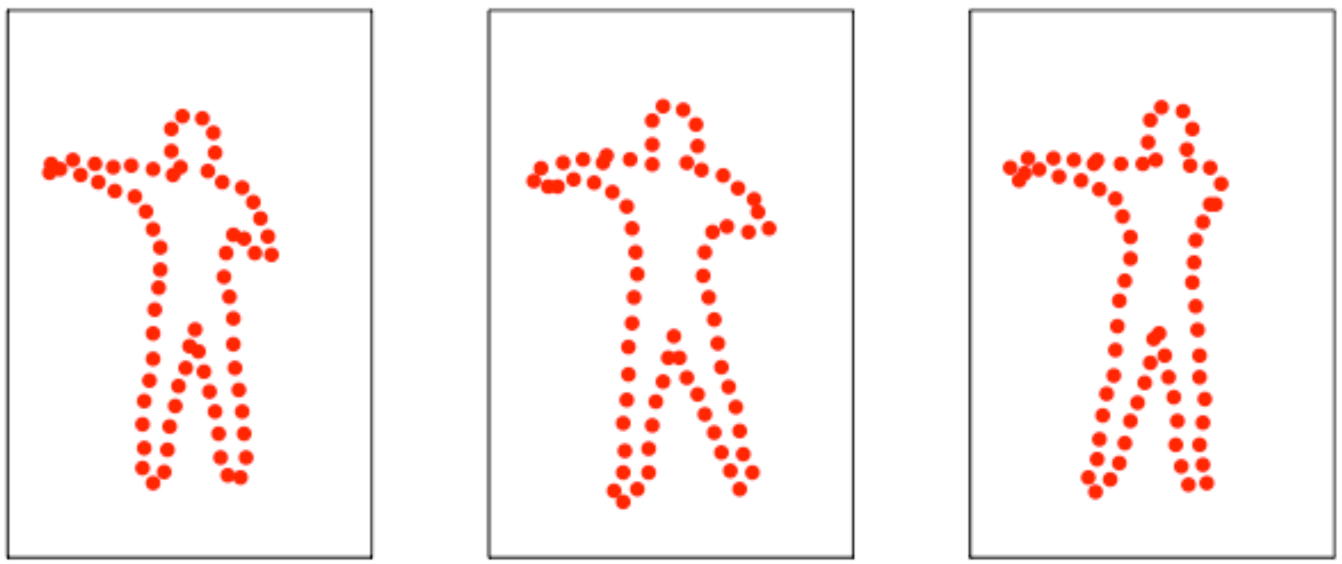
MATLAB

- bwboundaries
- bwtraceboundary
- edge
- hough

Shape Matching

Template Matching for Pose Recognition





query

retrieved exemplars from the dataset

Template Matching for Pose Recognition

Dataset of exemplars

Template Matching for Pose Recognition

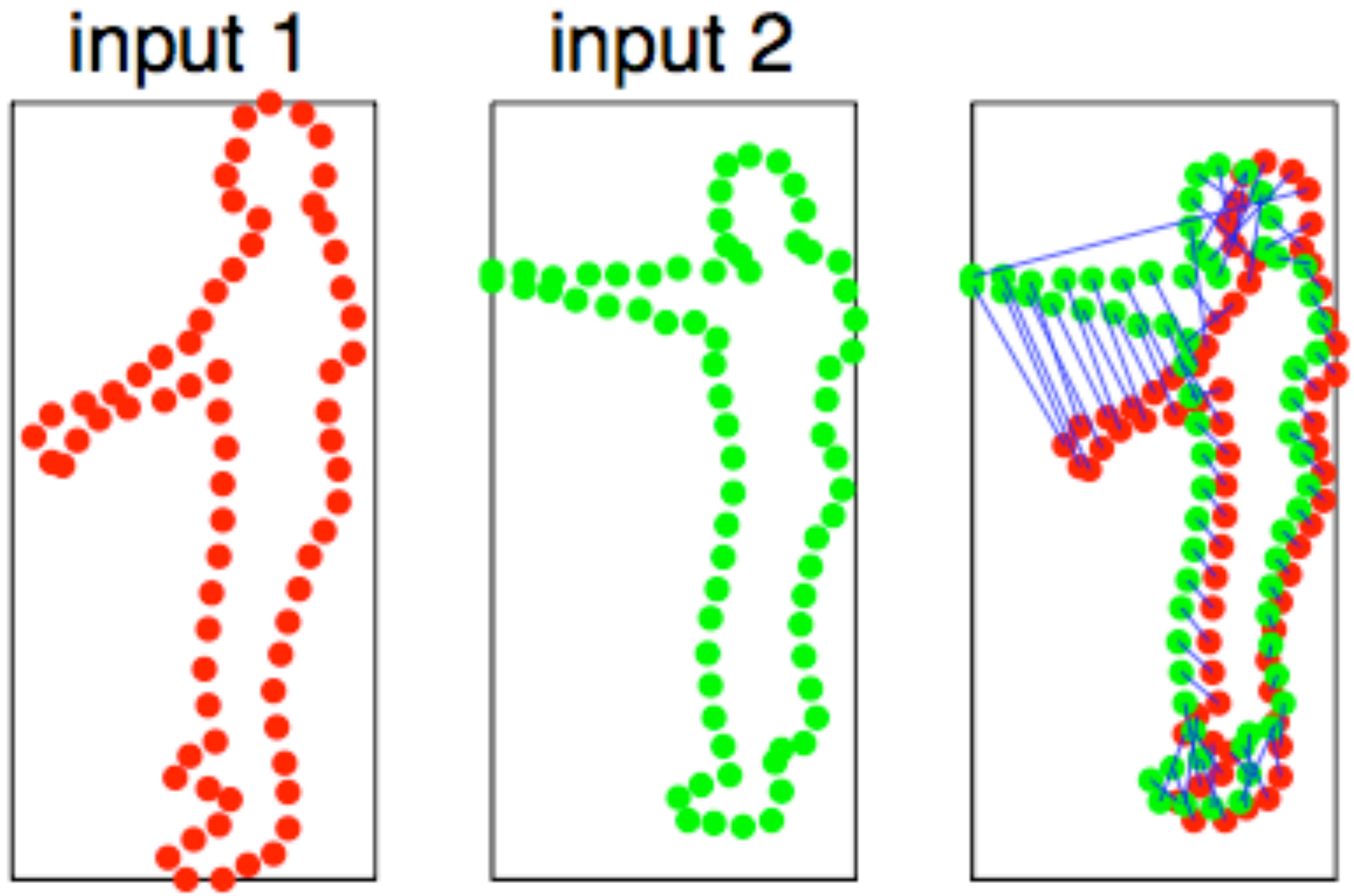
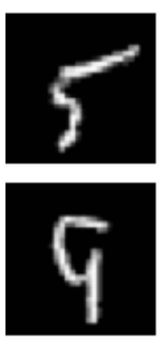


image exemplar result

Template Matching for Character Recognition

Template Matching for Character Recognition

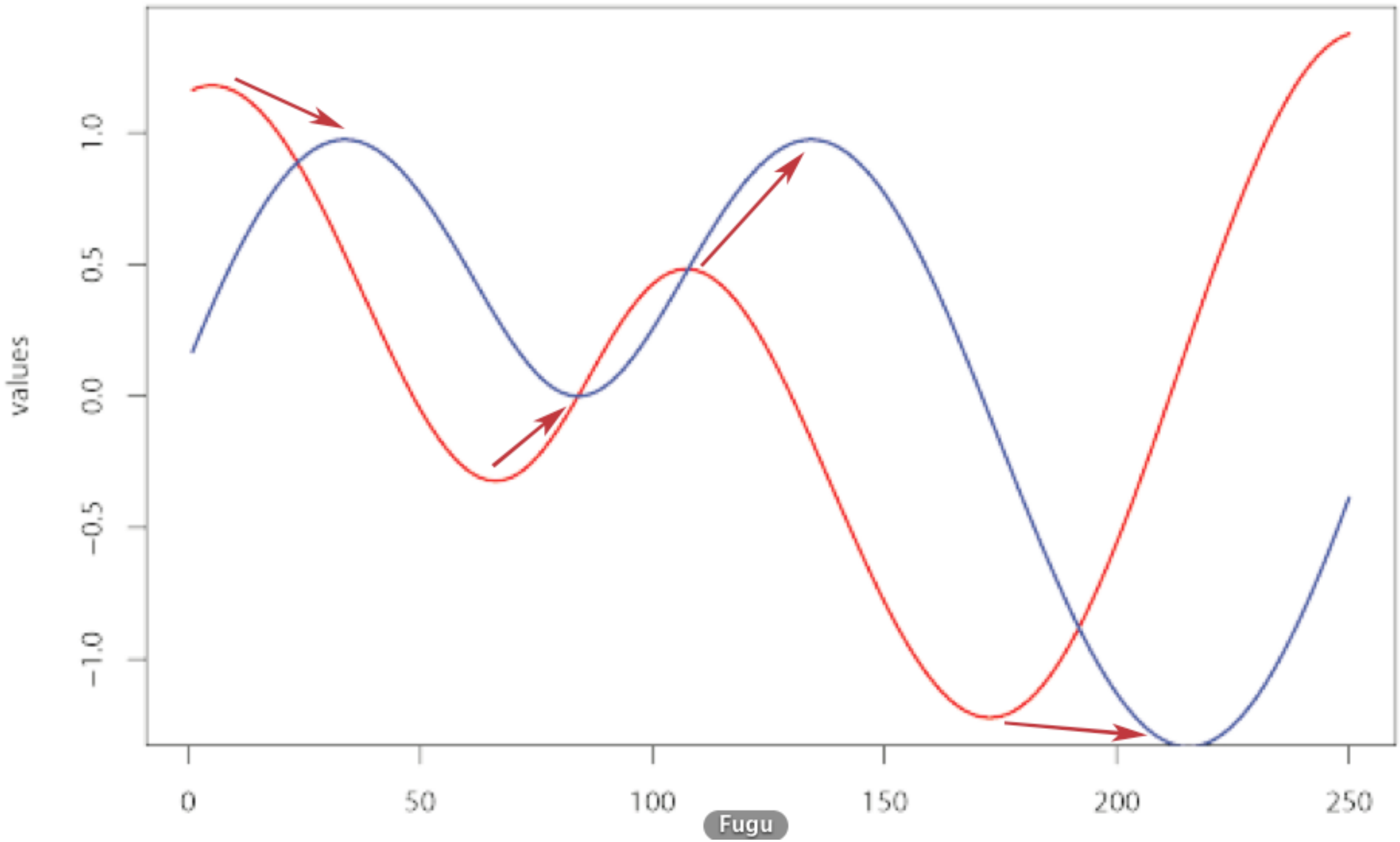


query

retrieved exemplars from the dataset

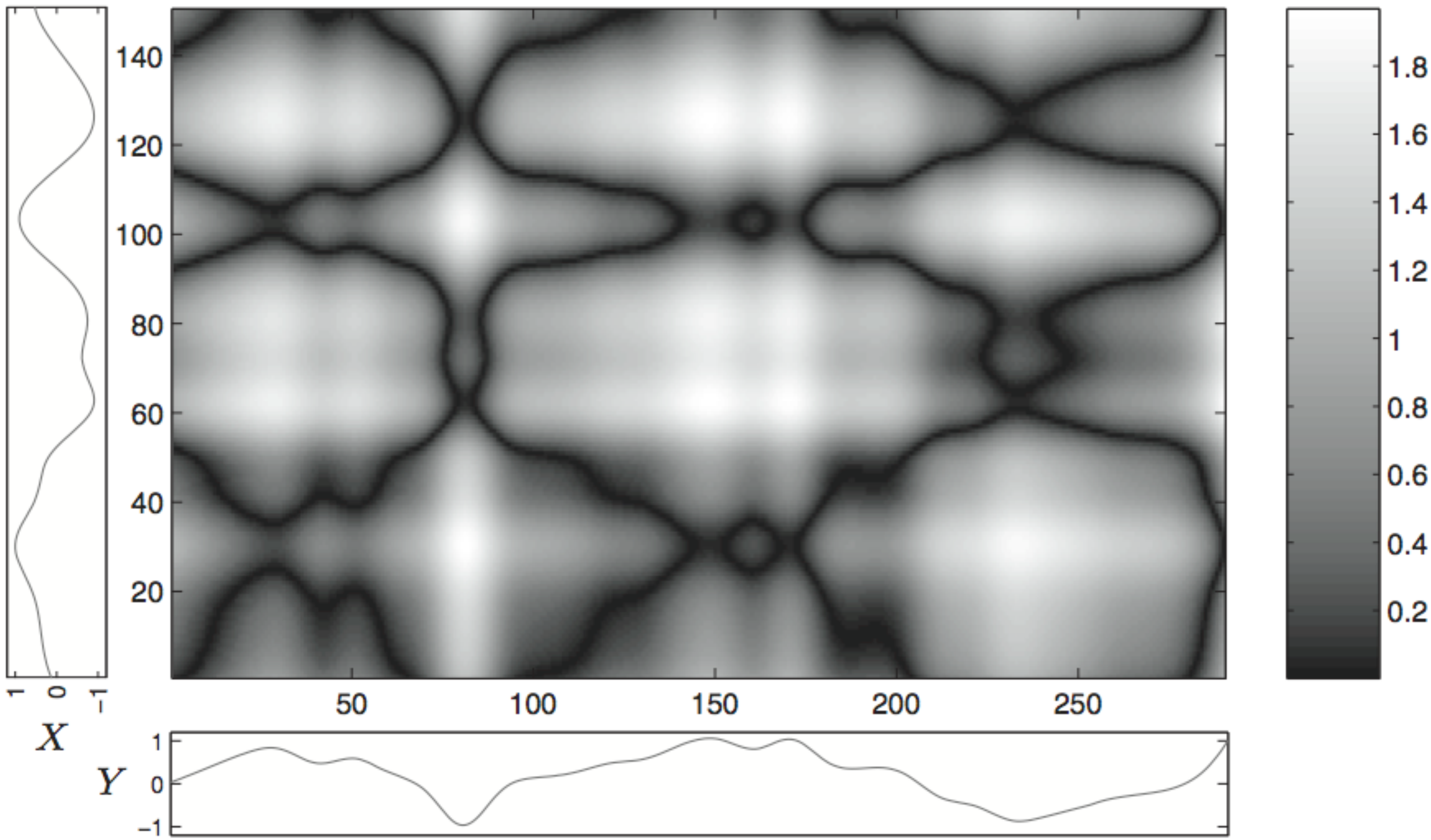
Dynamic Time Warping

Alignment of Two Sequences of Points



arrows show the points of alignment

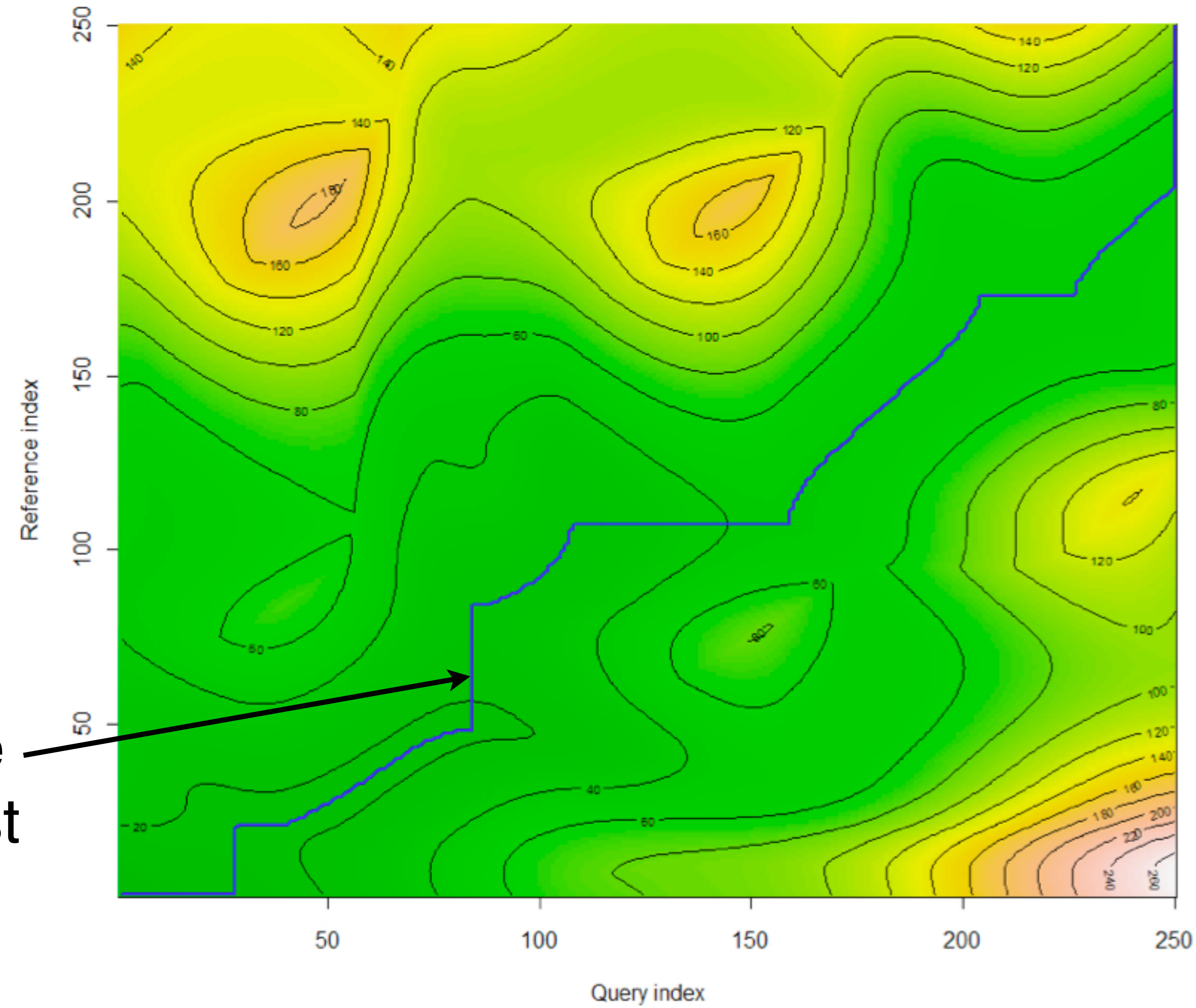
Alignment of Two Sequences of Points



Cost matrix

Building a Cost Matrix of Pairs of Points

$$c_{ij} = \|\boldsymbol{x}_i - \boldsymbol{y}_j\|, \;\; i = 1:N, \;\; j = 1:M$$

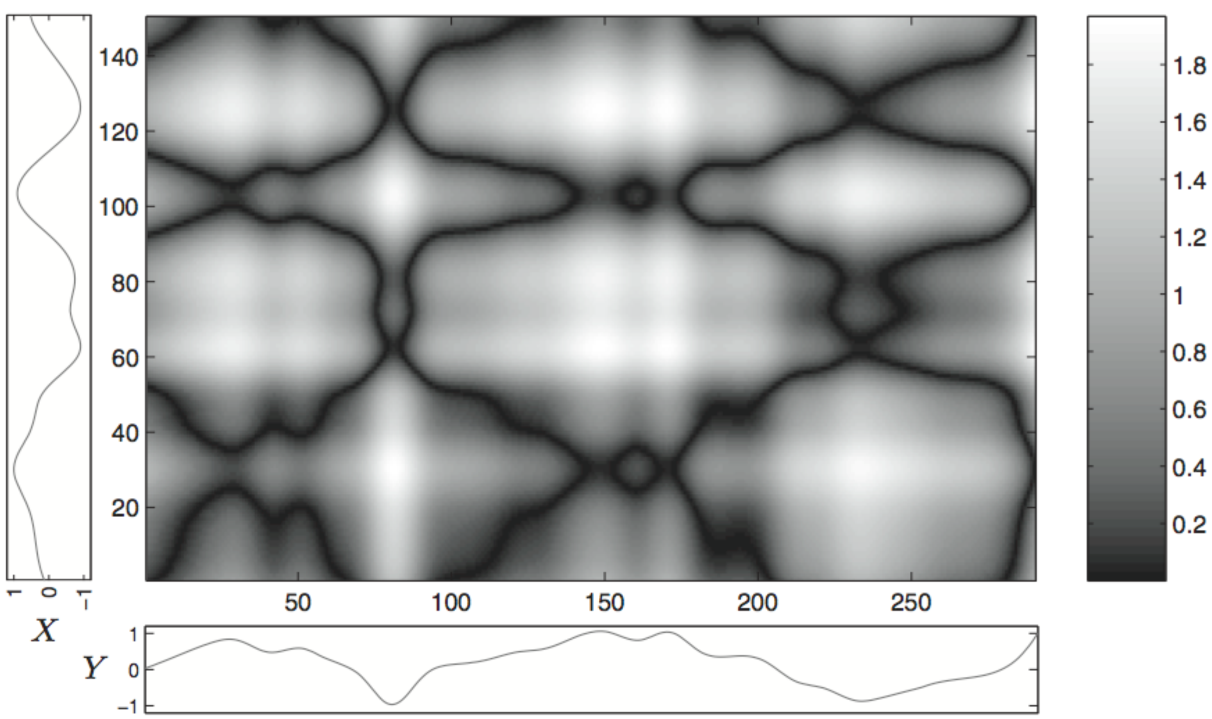


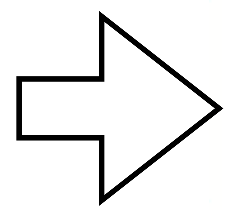
goal:
find the
min-cost
path

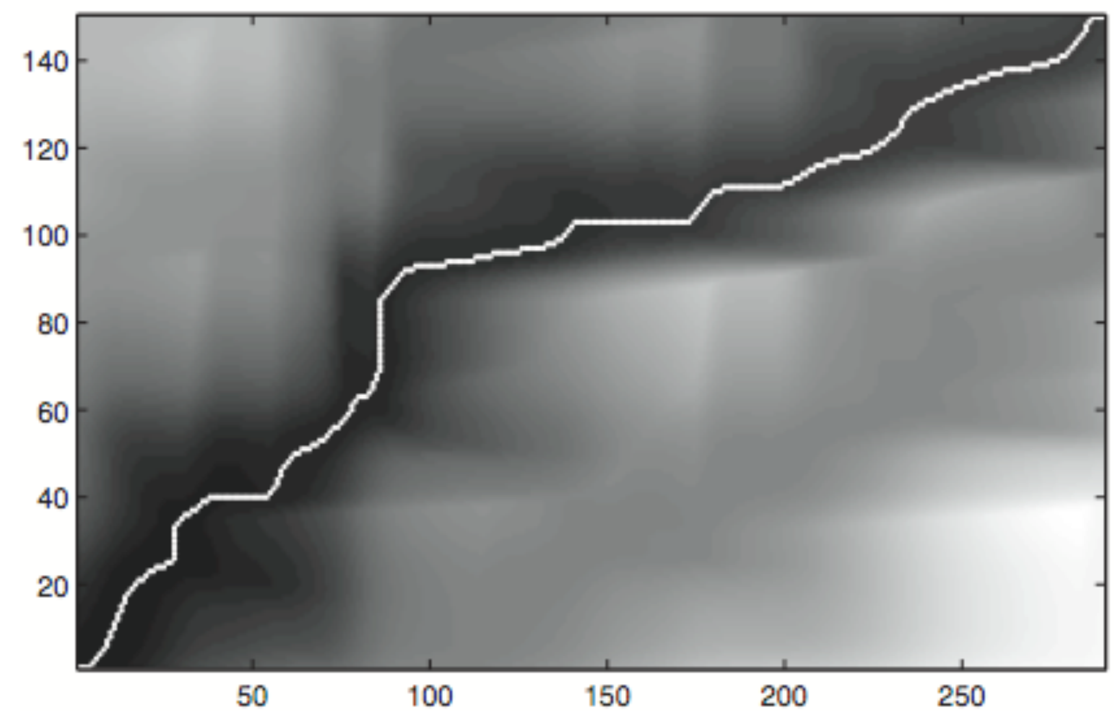
Building the Accumulated Cost Matrix

Cost matrix

Accumulated cost matrix







c_{ij}

element of
the cost matrix

$D(i,j)$

element of the
accumulated cost matrix

Building the Accumulated Cost Matrix

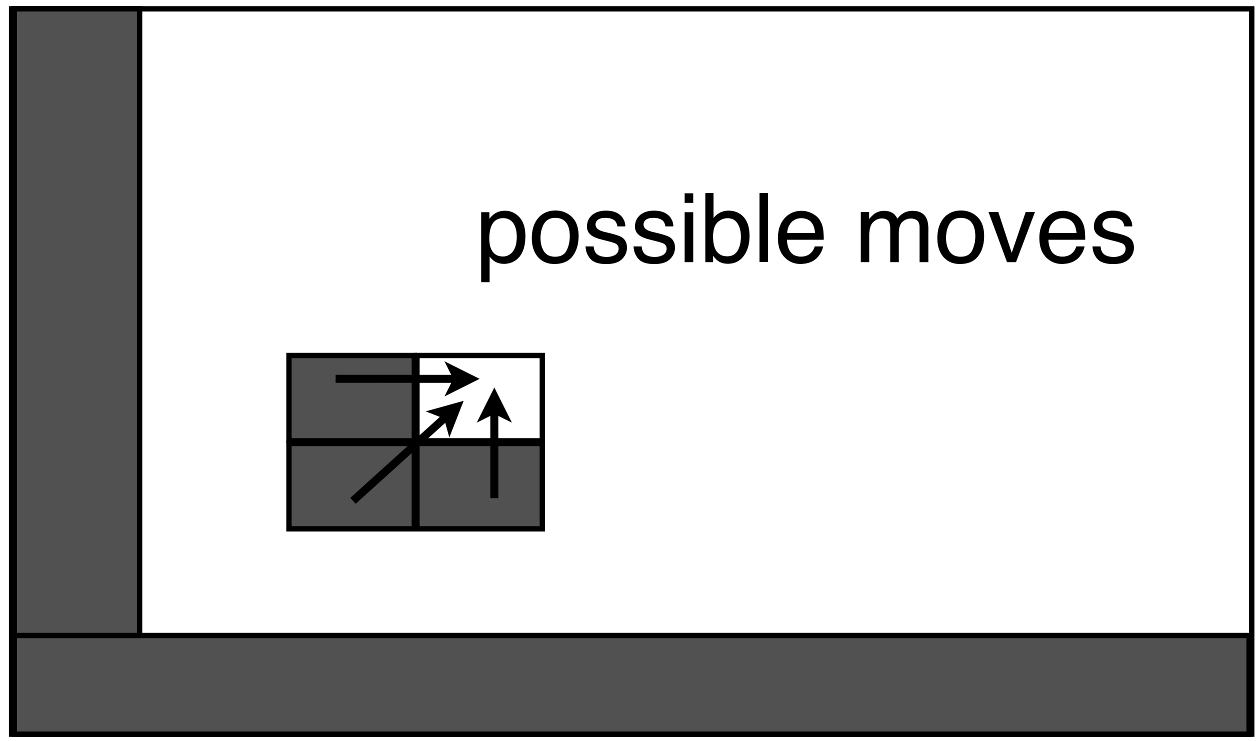
1. First row: $$D(1,j) = \sum_{k=1}^{j} c_{1k}, \quad j = 1 : M$$

Building the Accumulated Cost Matrix

1. First row: $$D(1,j) = \sum_{k=1}^{j} c_{1k}, \quad j = 1 : M$$

2. First column: $$D(i,1) = \sum_{k=1}^{i} c_{k1}, \quad i = 1 : N$$

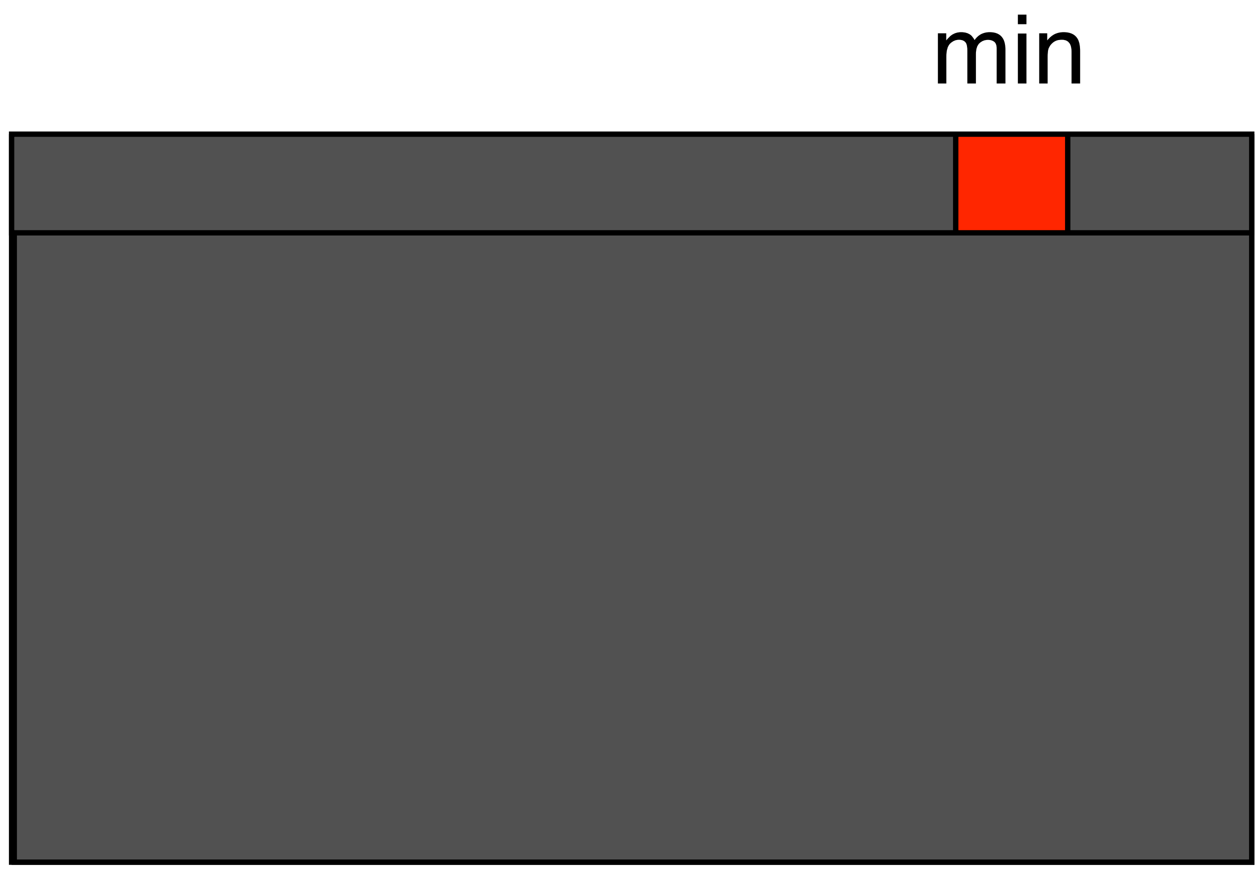
Building the Accumulated Cost Matrix



3. Compute:

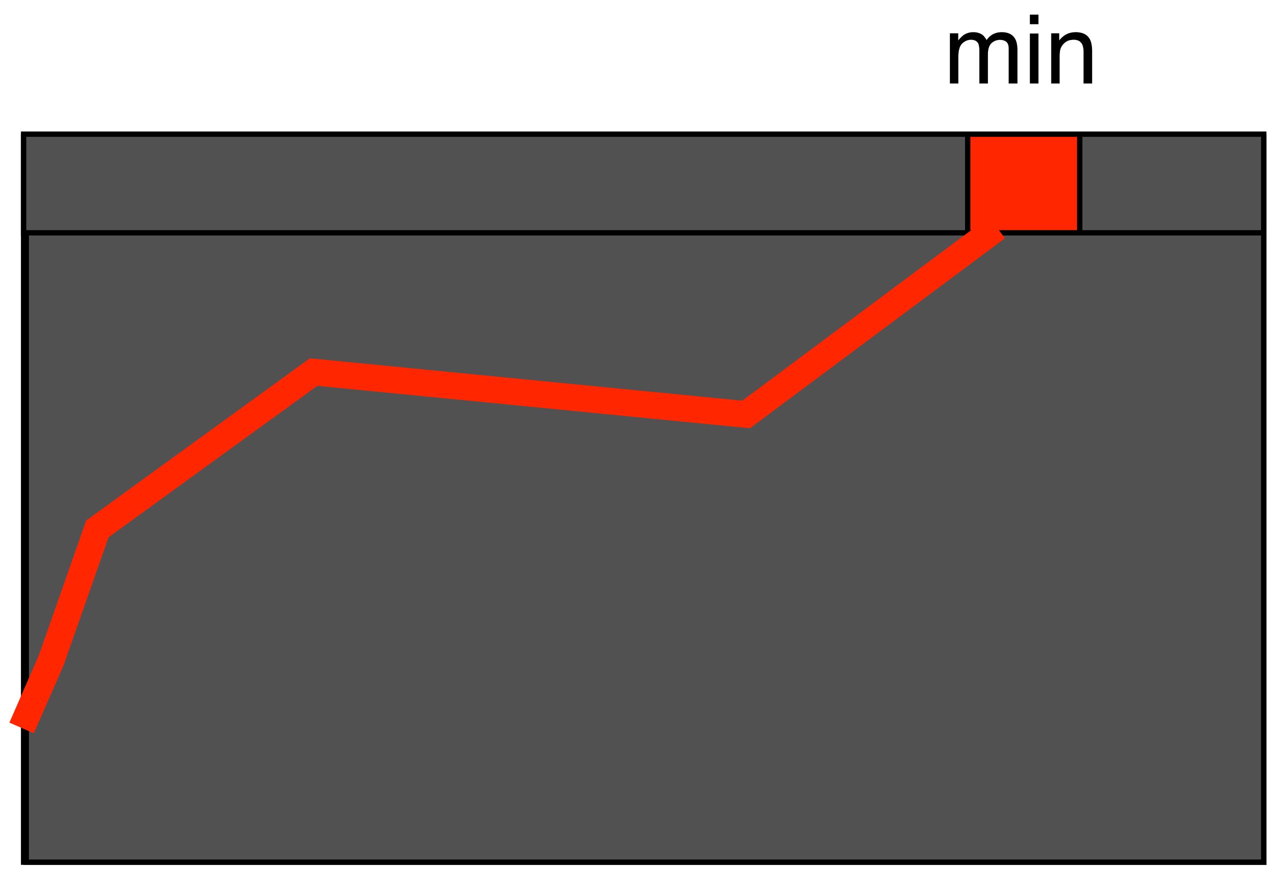
$$D(i,j) = c_{ij} + \min\{D(i-1,j-1), D(i,j-1), D(i-1,j)\}$$

Building the Accumulated Cost Matrix



4. Find min in the last row, and then back-track the path

Building the Accumulated Cost Matrix



4. Find min in the last row, and then back-track the path

Dynamic Time Warping

```
int DTWDistance(char s[1..n], char t[1..m]) {
    int DTW[0..n, 0..m]
    int i, j, cost

    for i := 1 to m
        DTW[0, i] := infinity;

    for i := 1 to n
        DTW[i, 0] := infinity;

    DTW[0, 0] := 0;

    for i := 1 to n

        for j := 1 to m

            cost:= d(s[i], t[j]);

            DTW[i, j] := cost + minimum(DTW[i-1, j  ],     // insertion
                                        DTW[i  , j-1],     // deletion
                                        DTW[i-1, j-1]);    // match


    return DTW[n, m];
}
```