Gesture Recognition

CSE 6367 – Computer Vision
Vassilis Athitsos
University of Texas at Arlington

Gesture Recognition

- What is a gesture?

Gesture Recognition

- What is a gesture?
  - Body motion used for communication.

Gesture Recognition

- What is a gesture?
  - Body motion used for communication.
- There are different types of gestures.

Gesture Recognition

- What is a gesture?
  - Body motion used for communication.
- There are different types of gestures.
  - Hand gestures (e.g., waving goodbye).
  - Head gestures (e.g., nodding).
  - Body gestures (e.g., kicking).

Gesture Recognition

- What is a gesture?
  - Body motion used for communication.
- There are different types of gestures.
  - Hand gestures (e.g., waving goodbye).
  - Head gestures (e.g., nodding).
  - Body gestures (e.g., kicking).
- Example applications:

Gesture Recognition

- What is a gesture?
  - Body motion used for communication.
- There are different types of gestures.
  - Hand gestures (e.g., waving goodbye).
  - Head gestures (e.g., nodding).
  - Body gestures (e.g., kicking).
- Example applications:
  - Human-computer interaction.
    - Controlling robots, appliances, via gestures.
  - Sign language recognition.

Dynamic Gestures

- What gesture did the user perform?

Class “8”

Gesture Types:10 Digits

Gesture Recognition Example

- Recognize 10 simple gestures performed by the user.
- Each gesture corresponds to a number, from 0, to 9.
- Only the *trajectory* of the hand matters, not the handshape.
  - This is just a choice we make for this example application. Many systems need to use handshape as well.

Decomposing Gesture Recognition

- We need modules for:

Decomposing Gesture Recognition

- We need modules for:
  - Computing *how the person moved*.
    - Person detection/tracking.
    - Hand detection/tracking.
    - Articulated tracking (tracking each body part).
    - Handshape recognition.
  - Recognizing *what the motion means.*

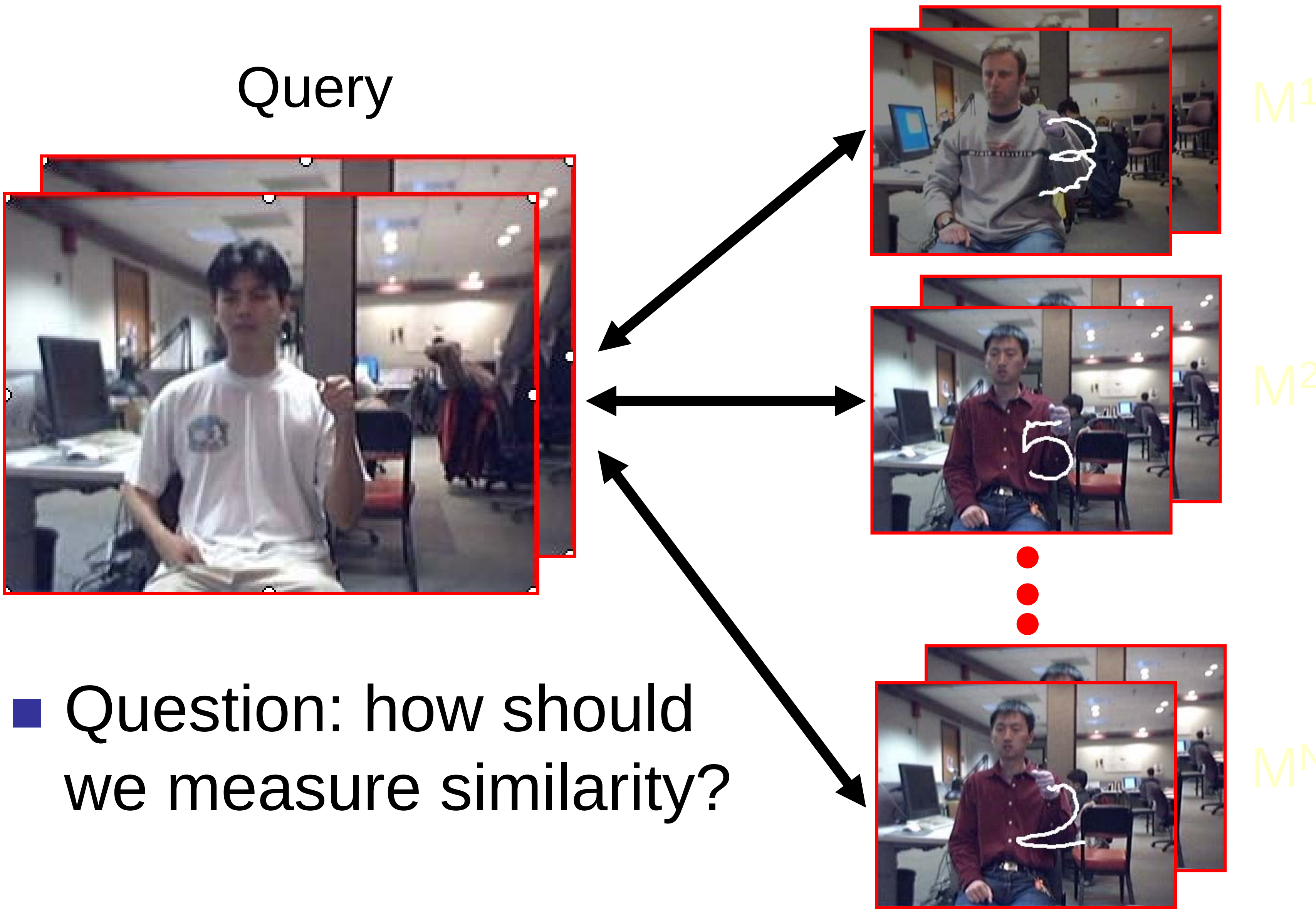
Decomposing Gesture Recognition

- We need modules for:
  - Computing *how the person moved*.
    - Person detection/tracking.
    - Hand detection/tracking.
    - Articulated tracking (tracking each body part).
    - Handshape recognition.
  - Recognizing *what the motion means.*
- Motion estimation and recognition are quite different tasks.

Decomposing Gesture Recognition

- We need modules for:
  - Computing *how the person moved*.
    - Person detection/tracking.
    - Hand detection/tracking.
    - Articulated tracking (tracking each body part).
    - Handshape recognition.
  - Recognizing *what the motion means.*
- Motion estimation and recognition are quite different tasks.
  - When we see someone signing in ASL, we know how they move, but not what the motion means.

Nearest-Neighbor Recognition

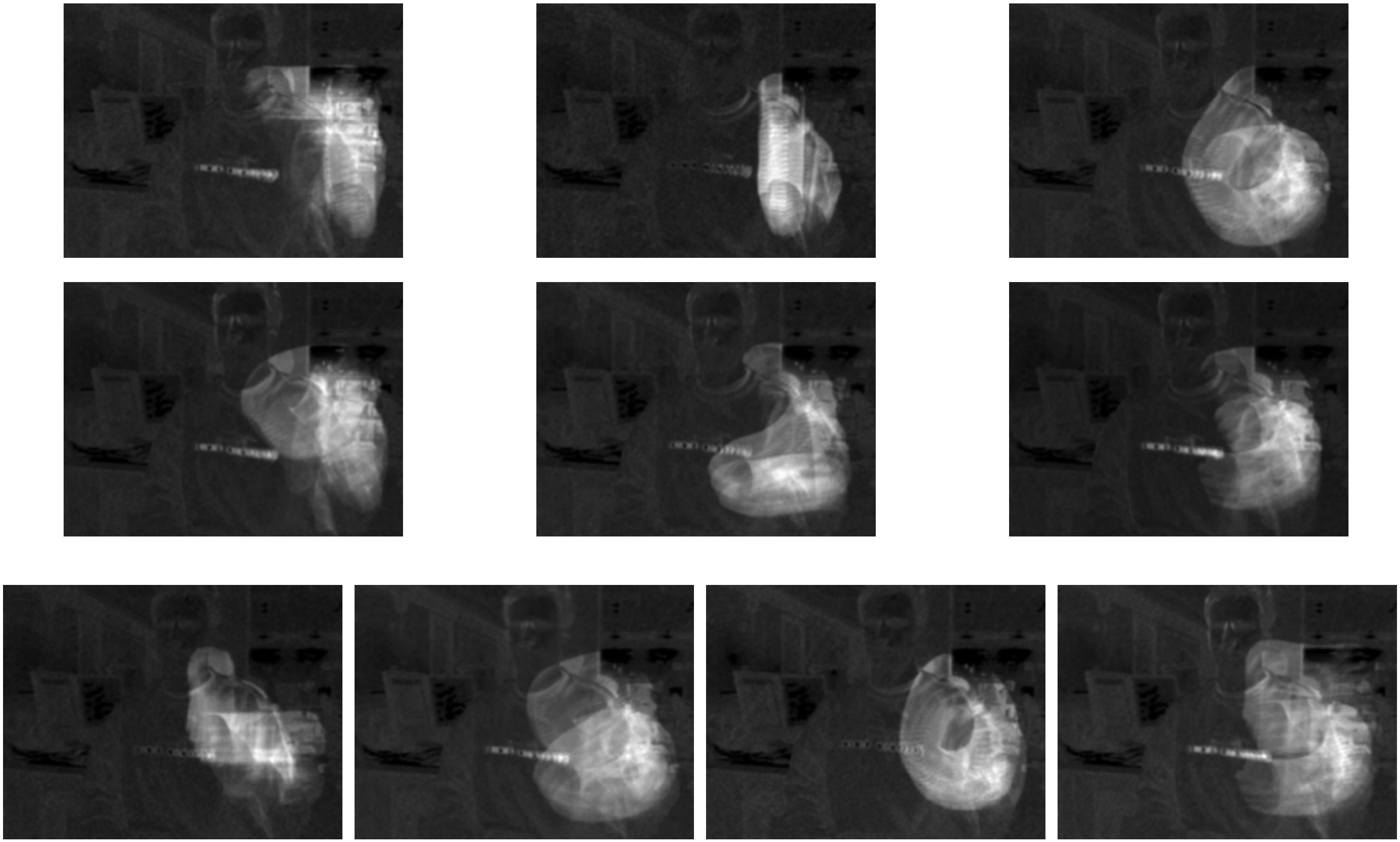


- Question: how should we measure similarity?

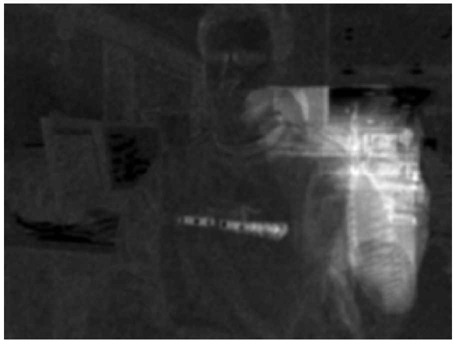
Motion Energy Images

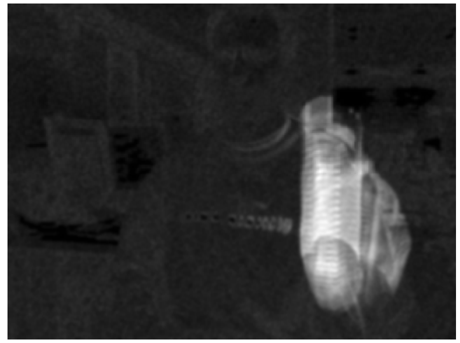
- A simple approach.
- Representing a gesture:
  - Sum of all the motion occurring in the video sequence.

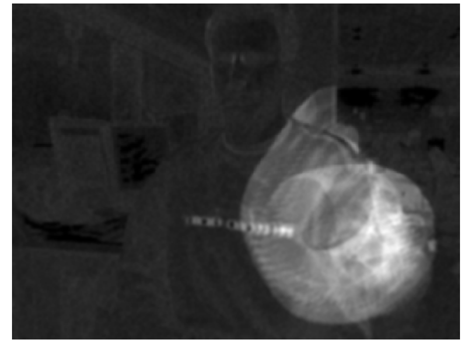
Motion Energy Images



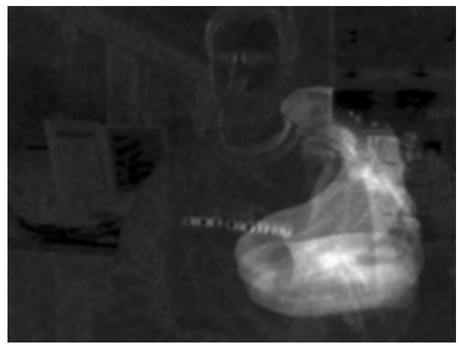
Motion Energy Images

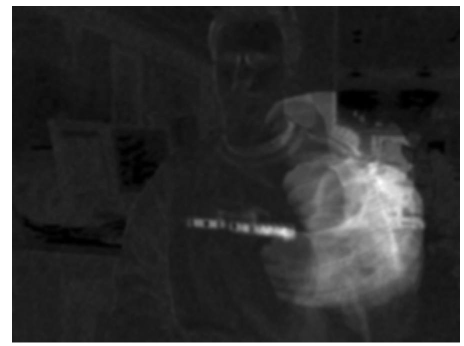
7


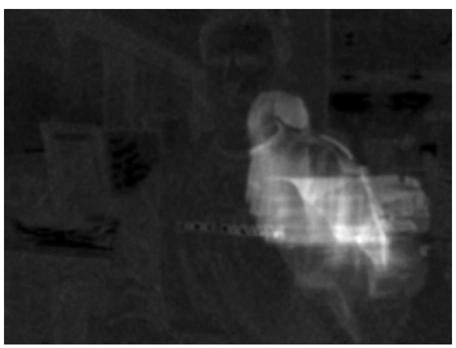
1


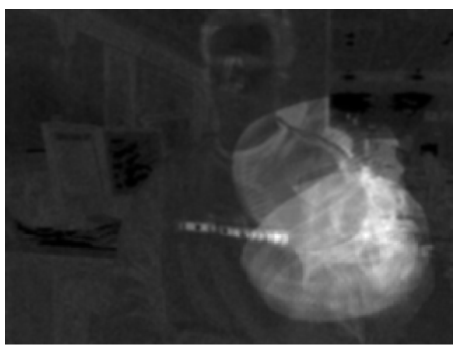
6


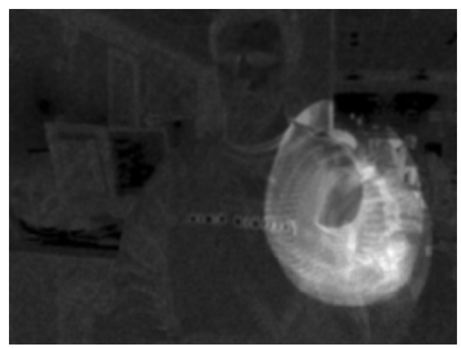
9

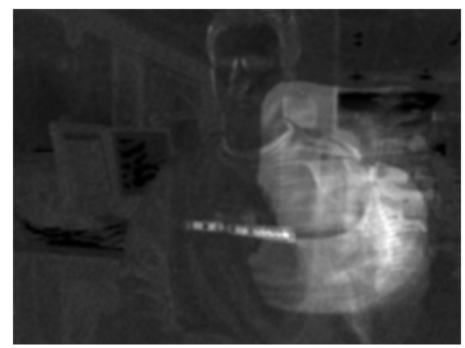

2


3



4


8


0


5

Motion Energy Images

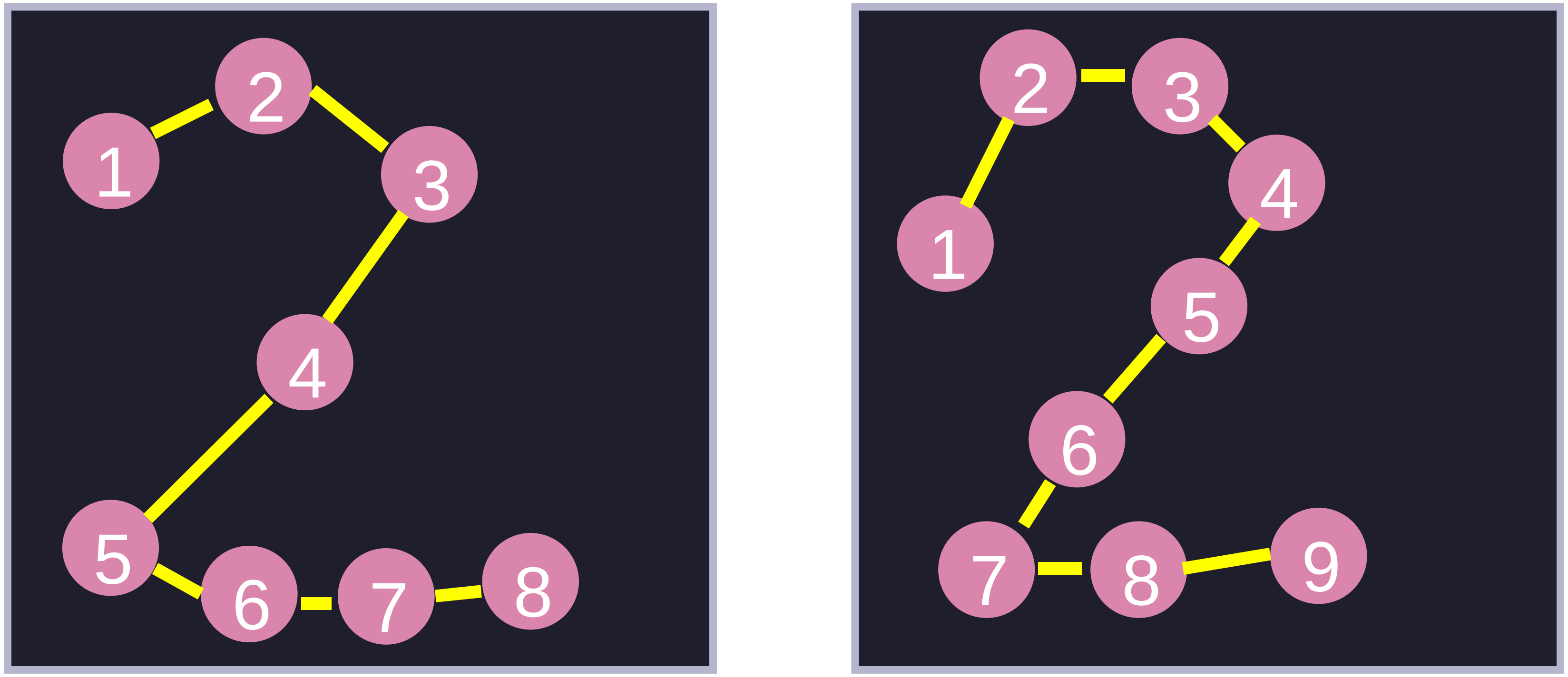
- Assumptions/Limitations:
  - No clutter.
  - We know the times when the gesture starts and ends.

If Hand Location Is Known:

Example database gesture

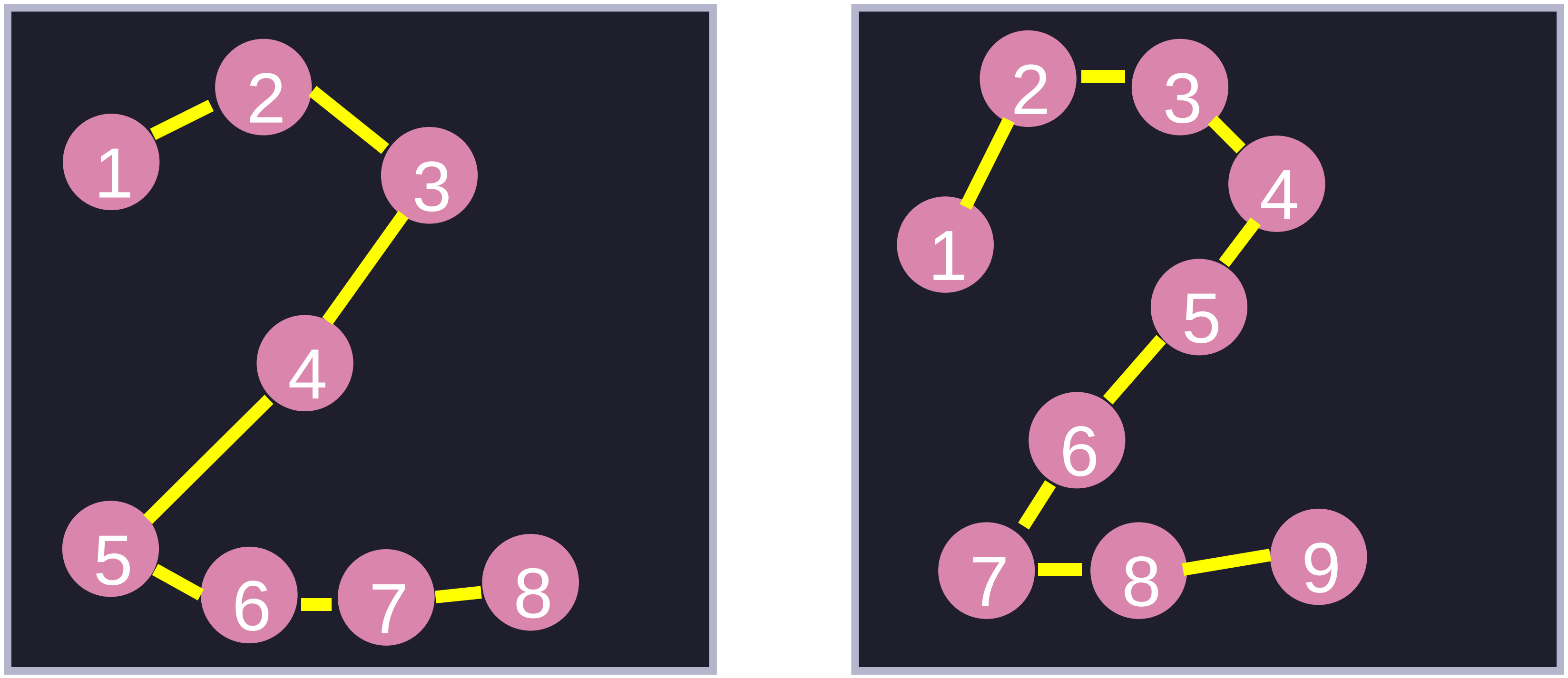
- Assumption: hand location is known in all frames of the database gestures.
  - Database is built offline.
  - In worst case, manual annotation.
  - Online user experience is not affected.

Comparing Trajectories



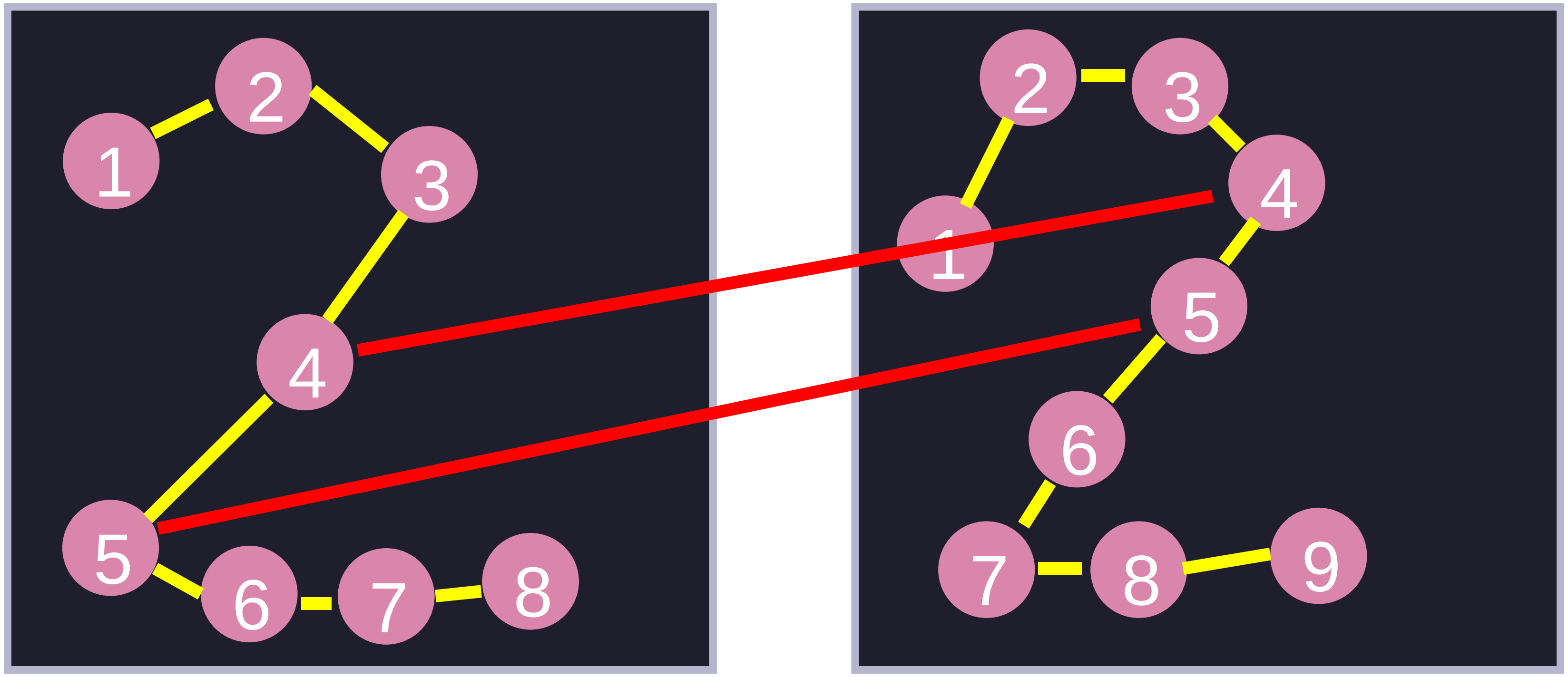
- We can make a trajectory based on the location of the hand at each frame.

Comparing Trajectories



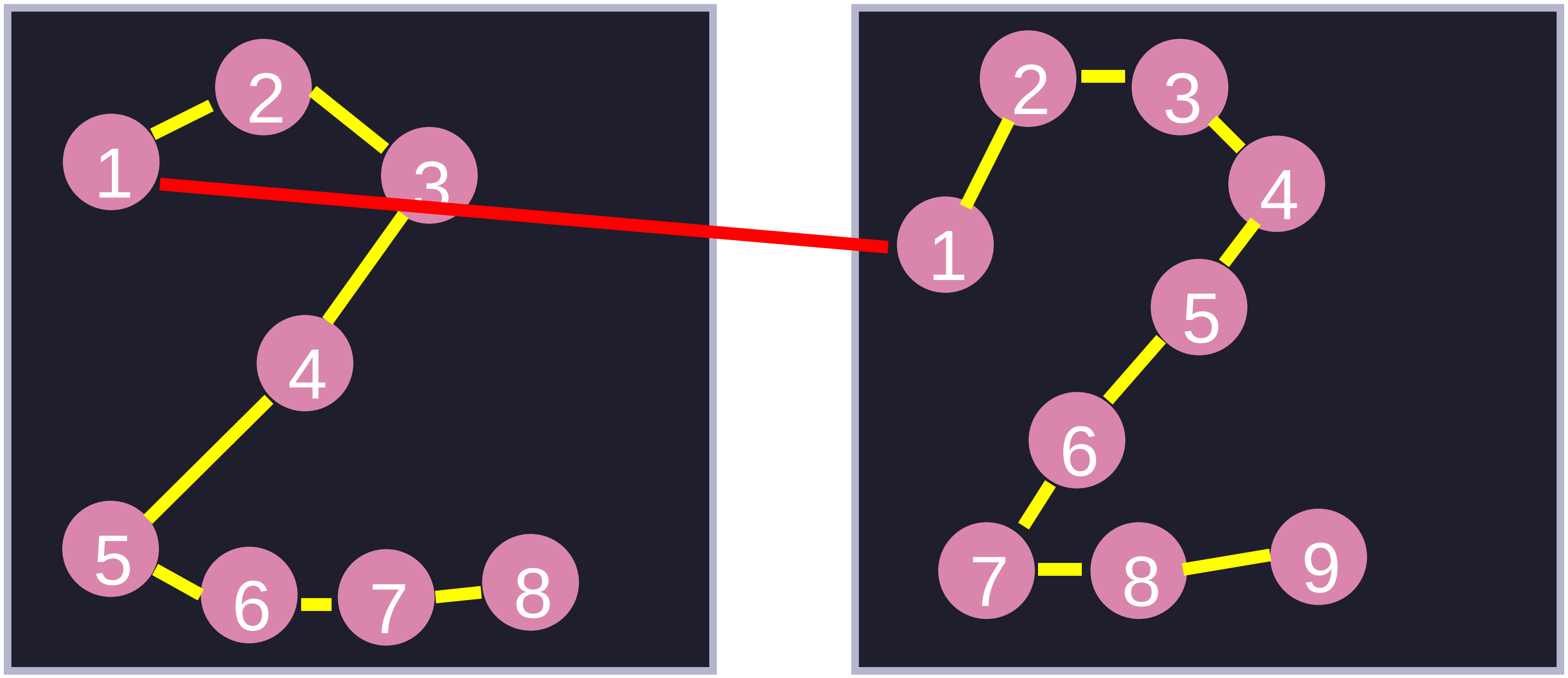
- How do we compare trajectories?

Matching Trajectories



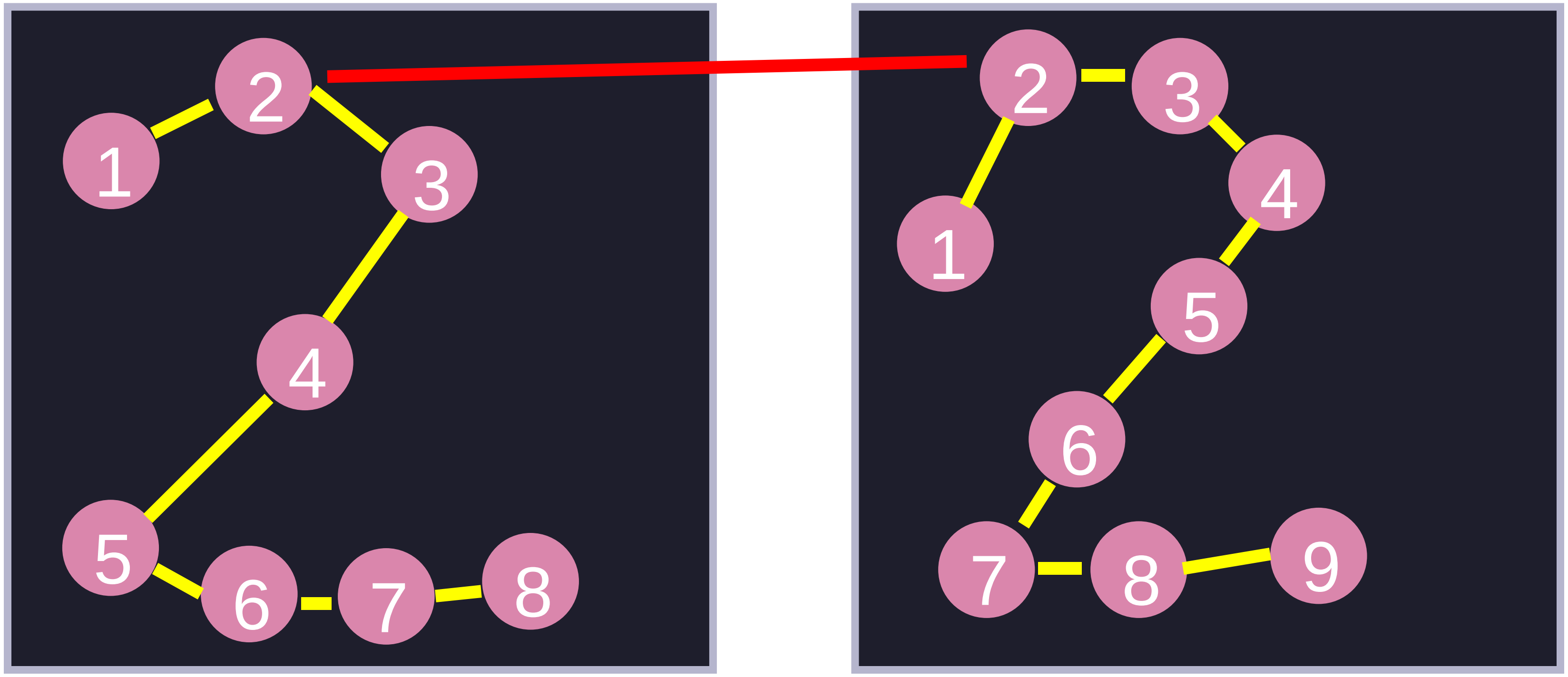
- Comparing i-th frame to i-th frame is problematic.
  - What do we do with frame 9?

Matching Trajectories



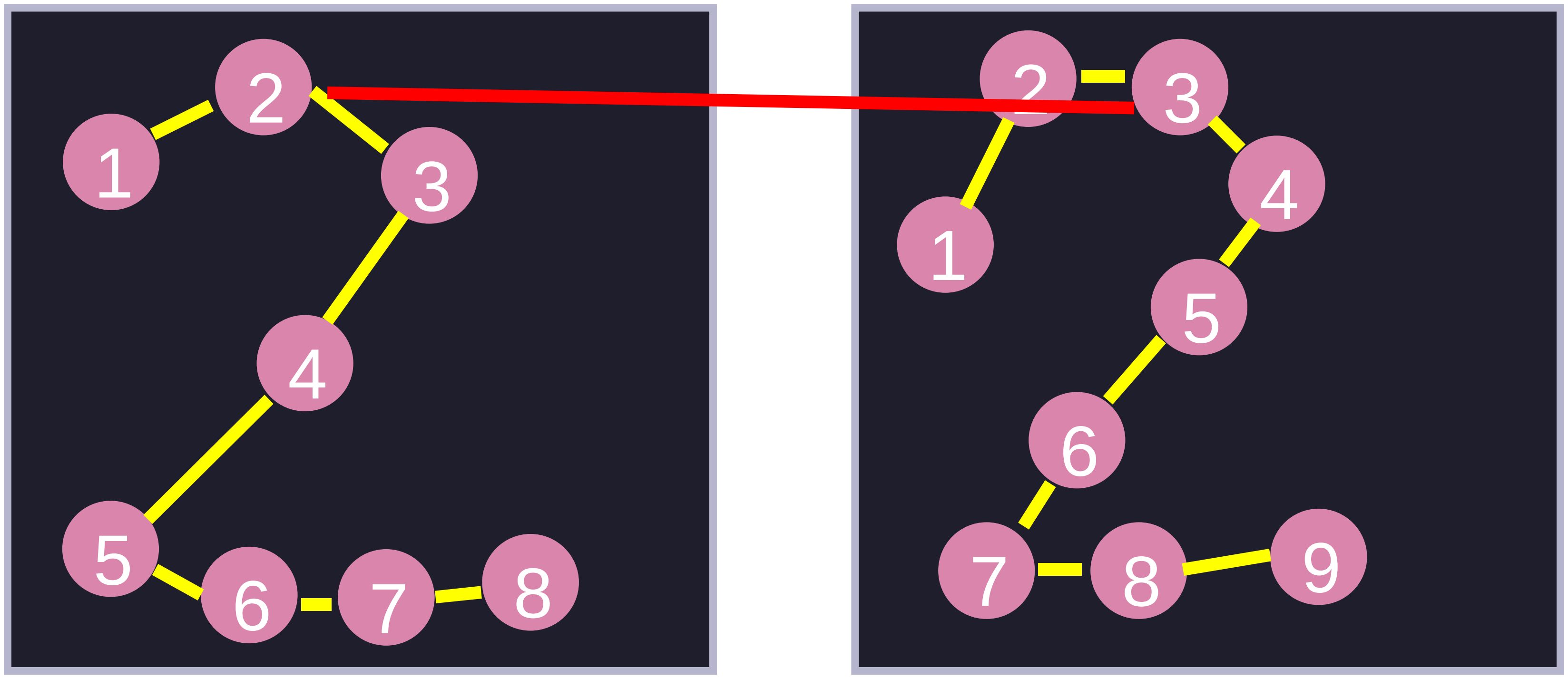
- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$

Matching Trajectories



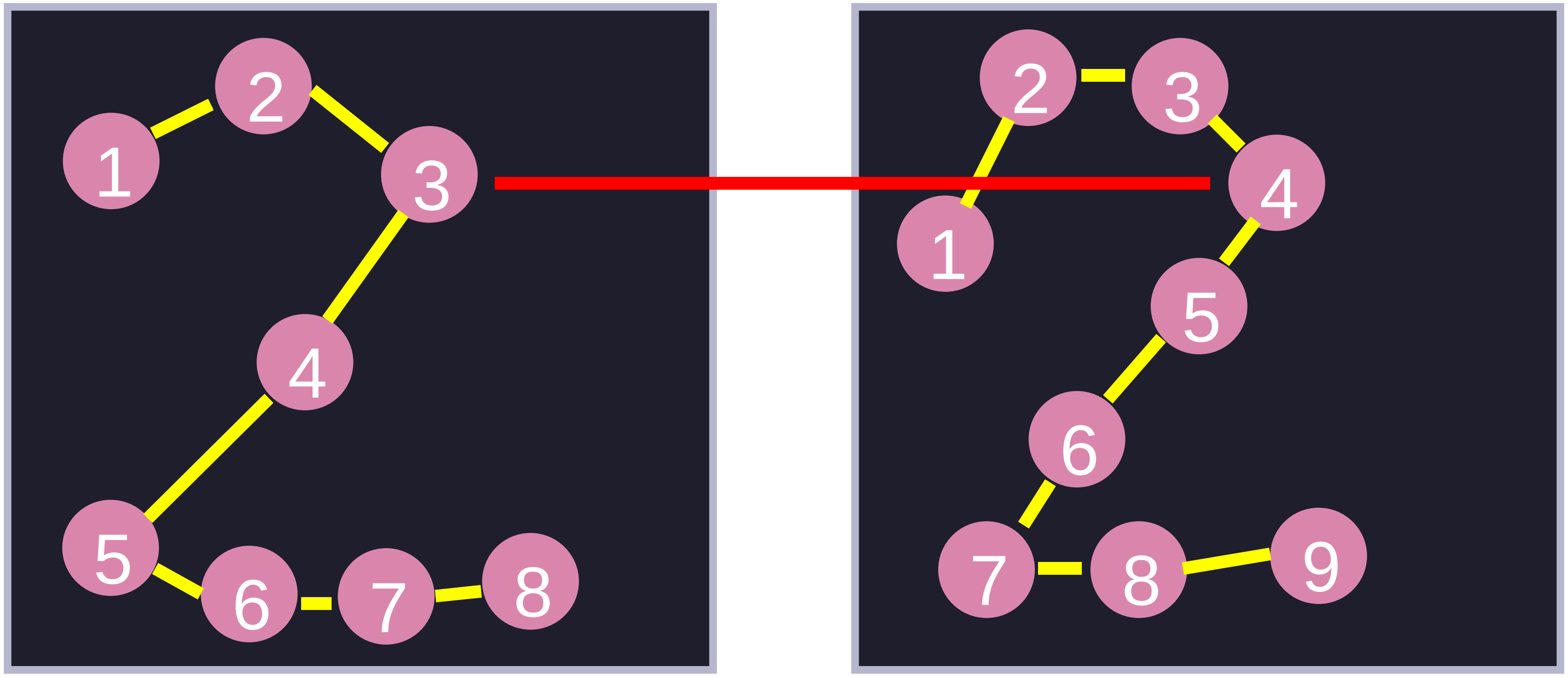
- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$

Matching Trajectories



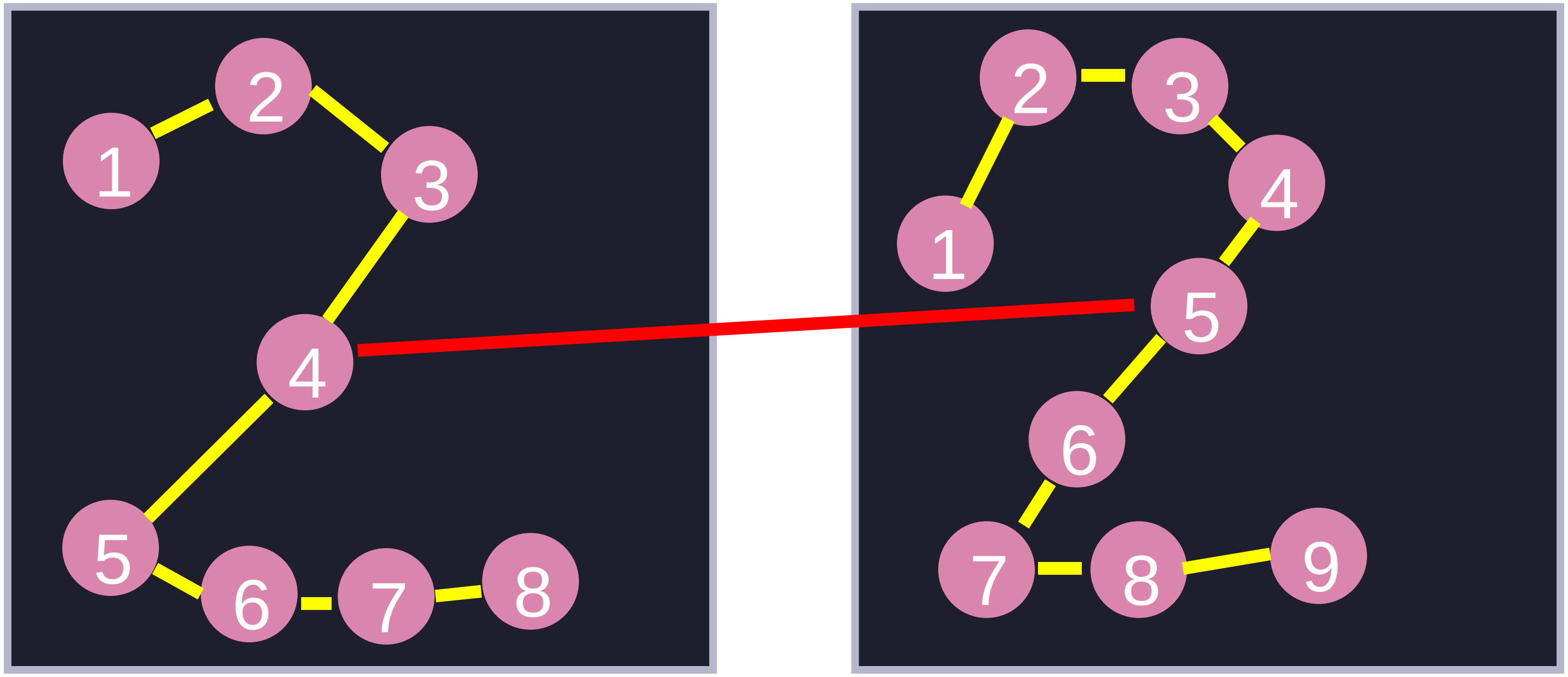
- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$

Matching Trajectories



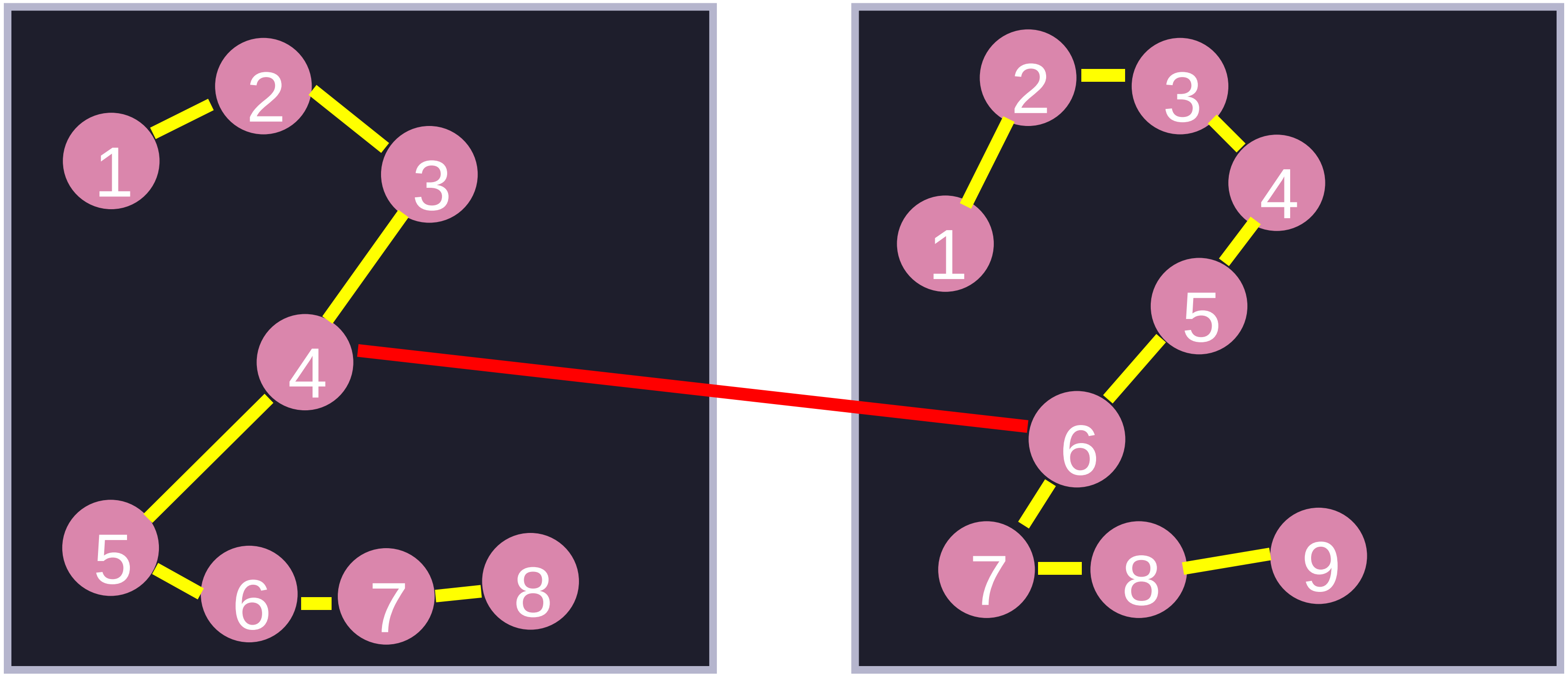
- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$

Matching Trajectories



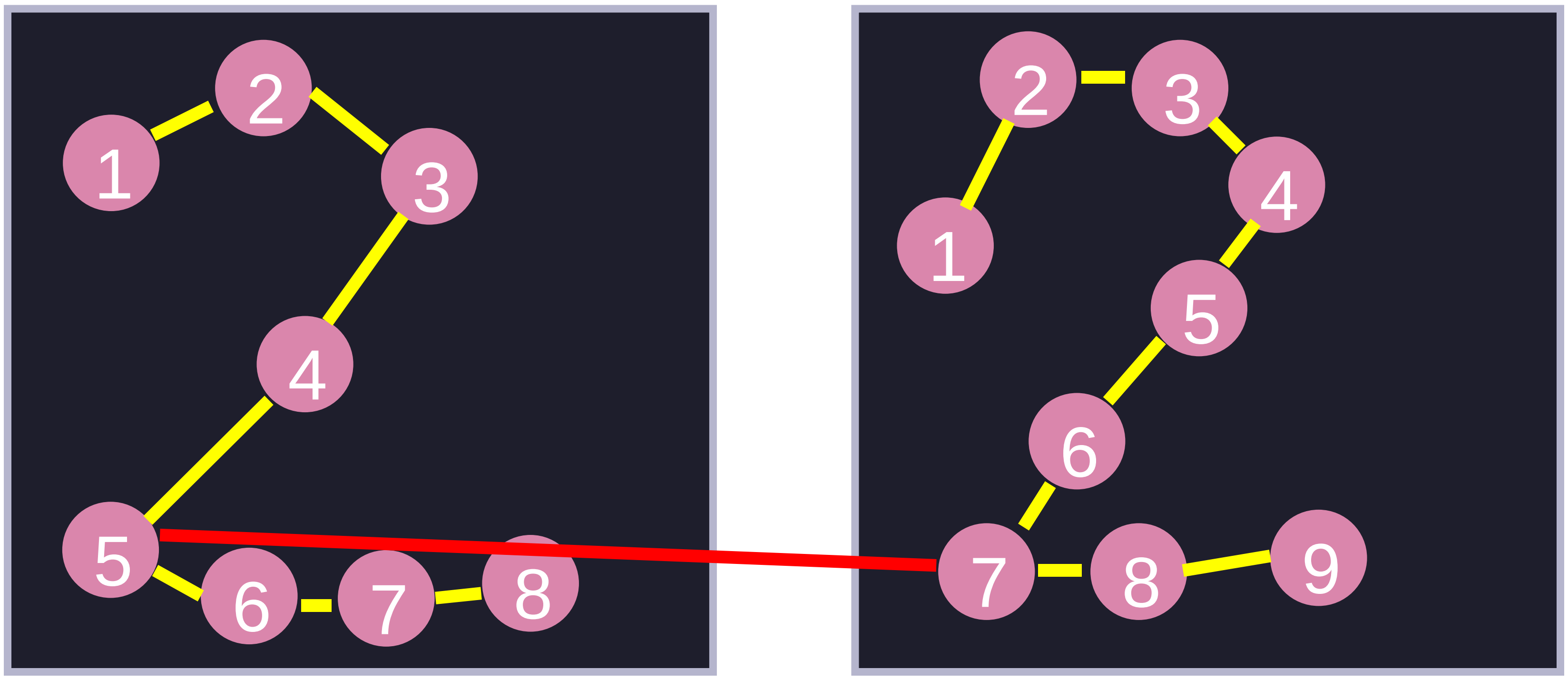
- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$

Matching Trajectories



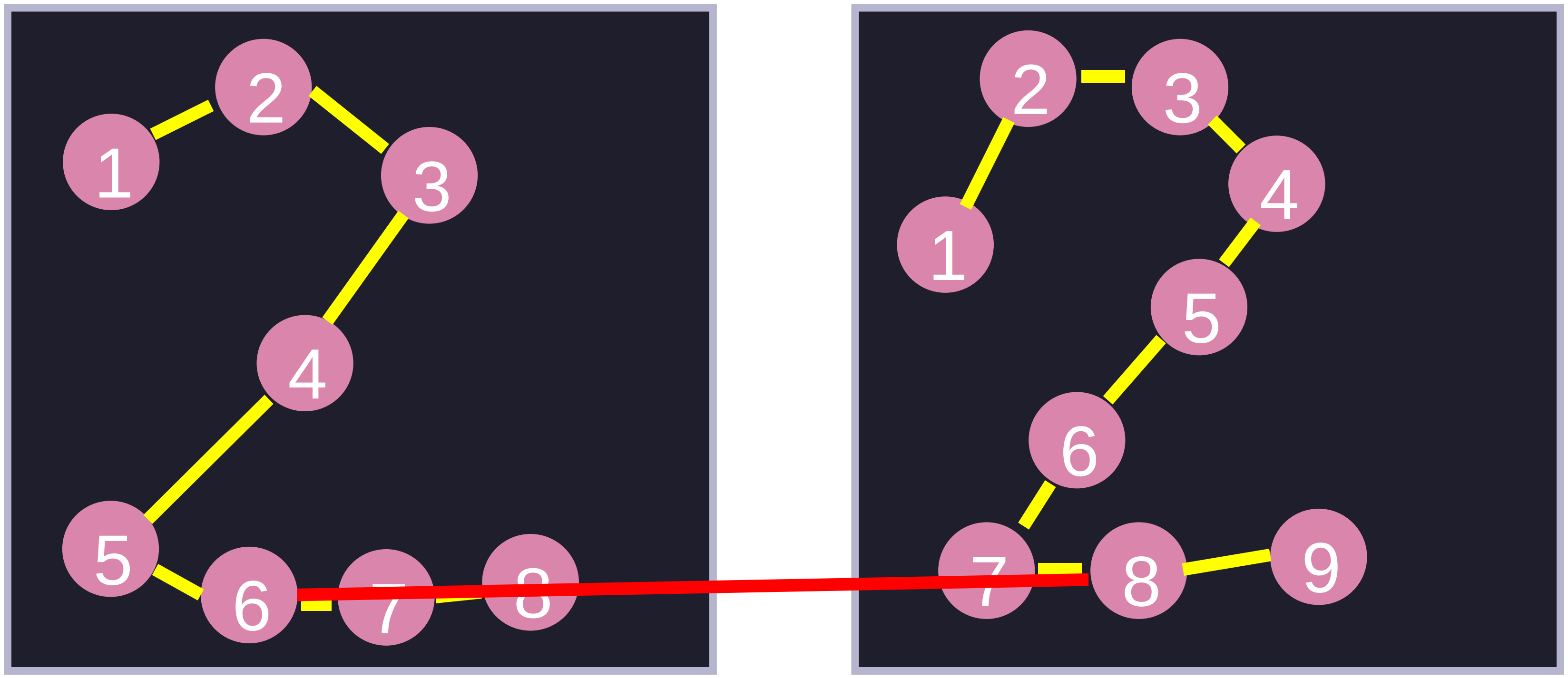
- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$

Matching Trajectories



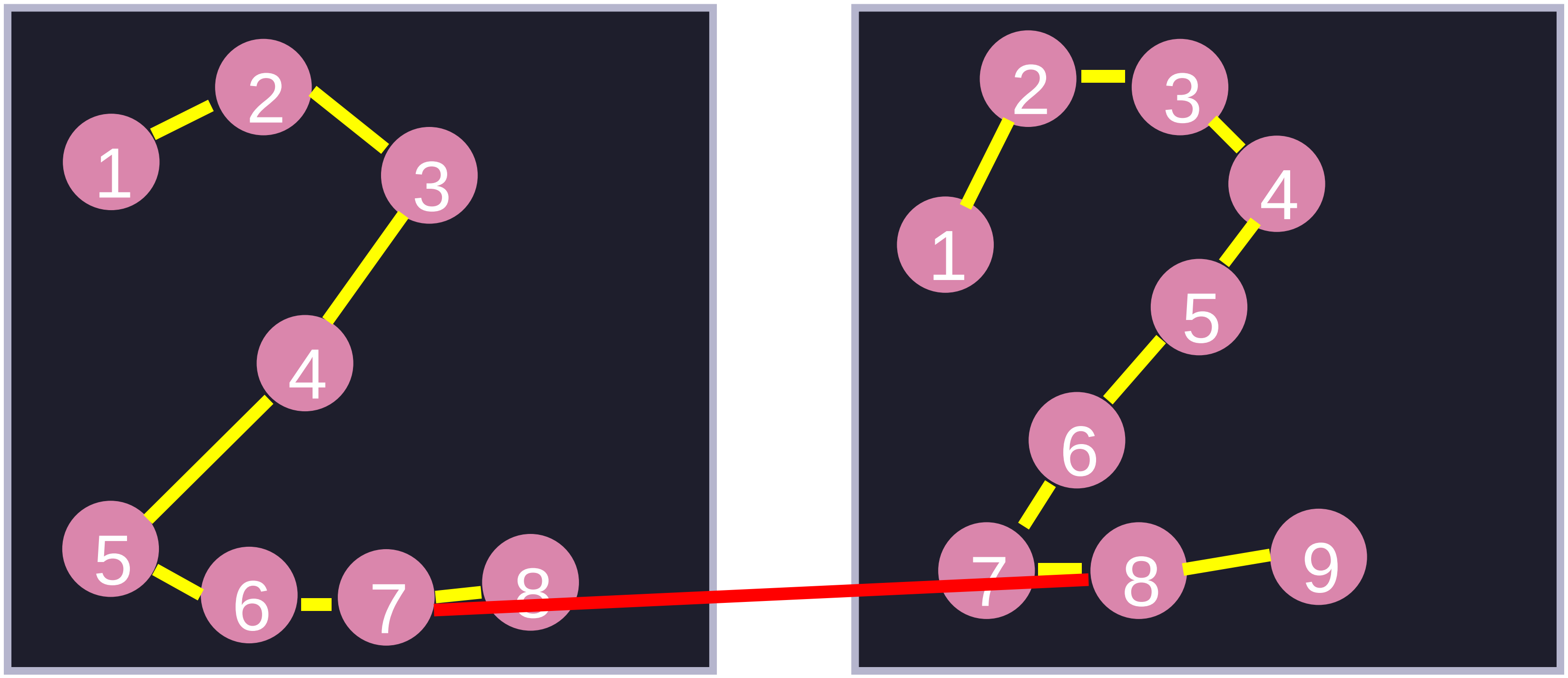
- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$

Matching Trajectories



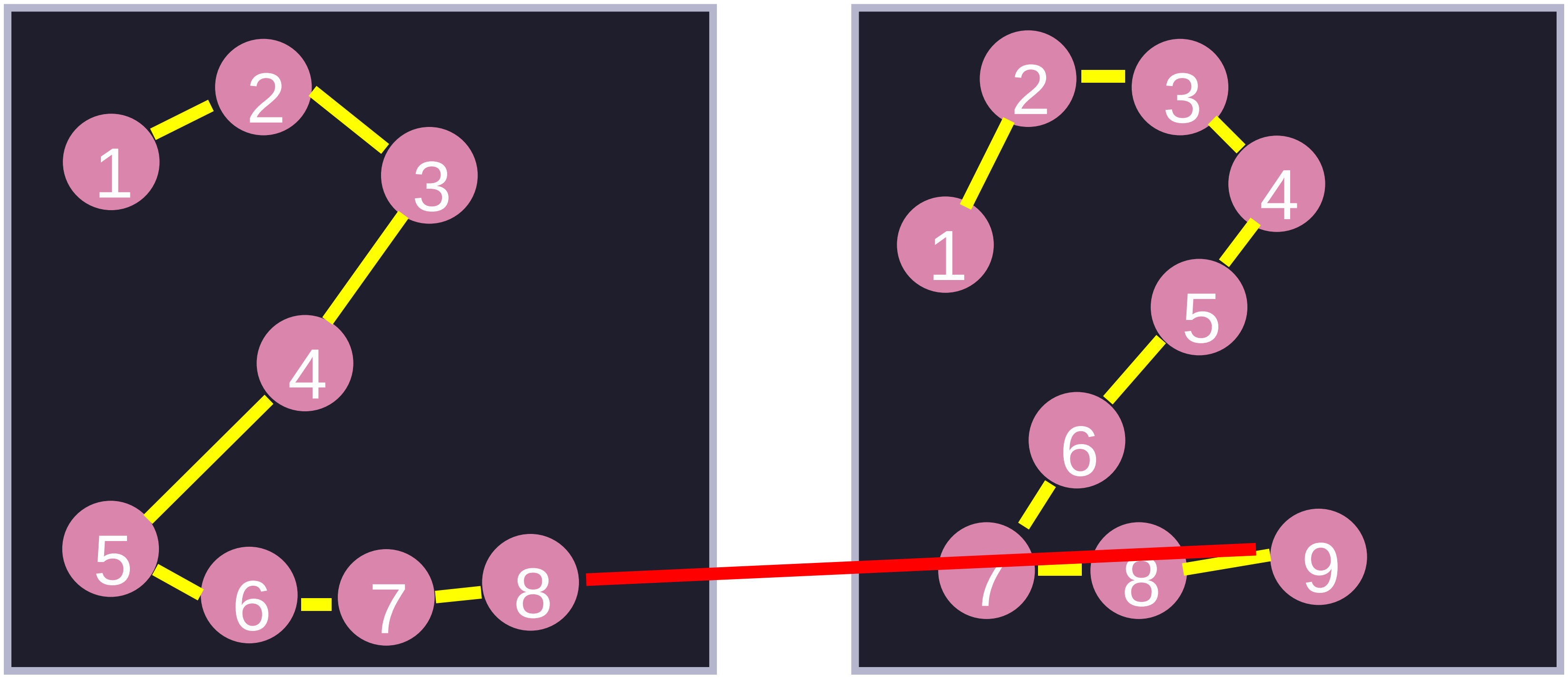
- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$

Matching Trajectories



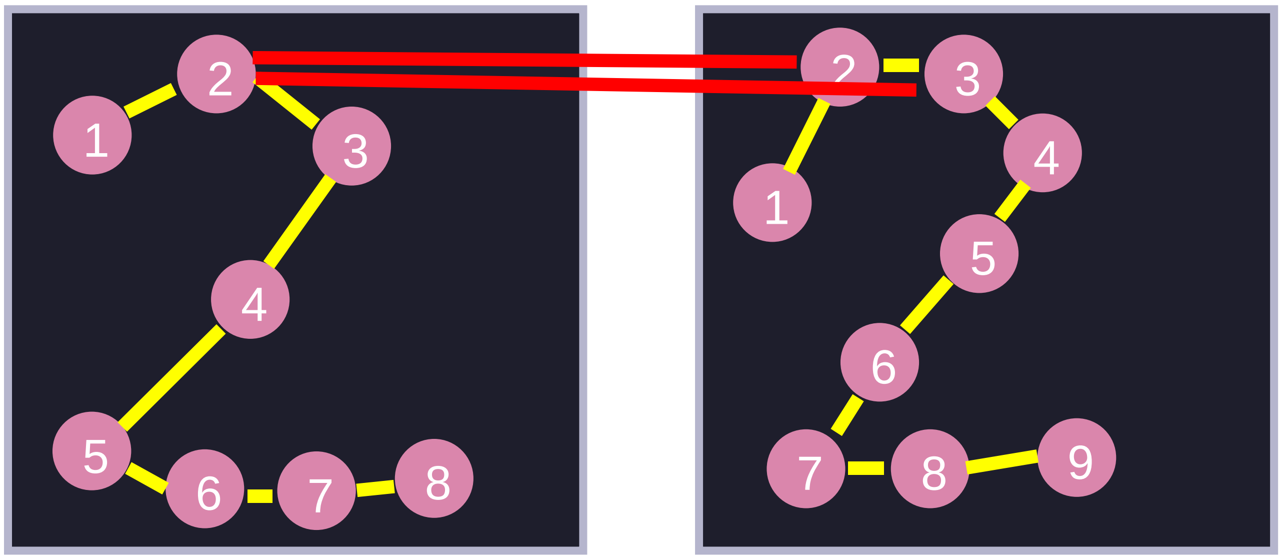
- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$

Matching Trajectories



- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - ((s_1, t_1), (s_2, t_2), …, (s_p, t_p))

Matching Trajectories

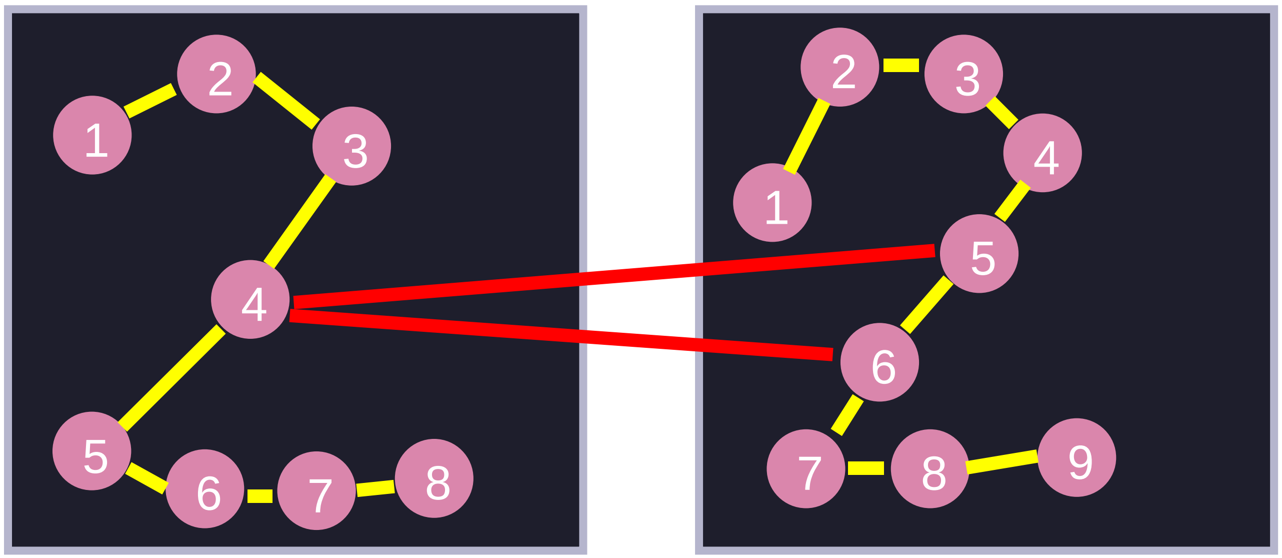


$M = (M_1, M_2, \ldots, M_8)$.

$Q = (Q_1, Q_2, \ldots, Q_9)$.

- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$
- Can be many-to-many.
  - M_1 is matched to Q_2 and Q_3.

Matching Trajectories

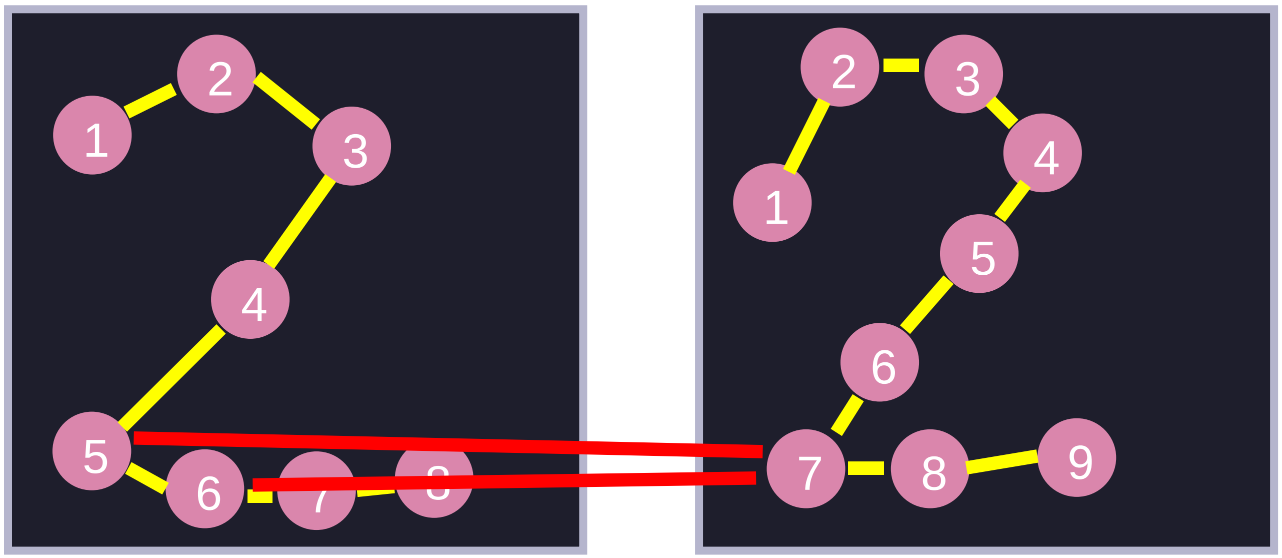


$M = (M_1, M_2, \ldots, M_8)$.

$Q = (Q_1, Q_2, \ldots, Q_9)$.

- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$
- Can be many-to-many.
  - M_4 is matched to Q_5 and Q_6.

Matching Trajectories

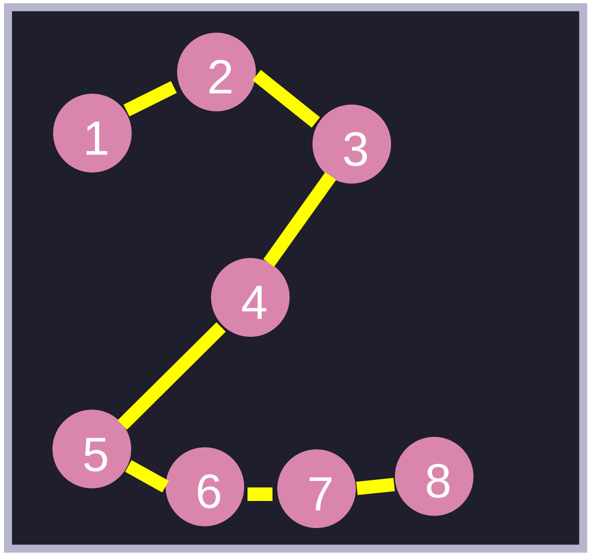


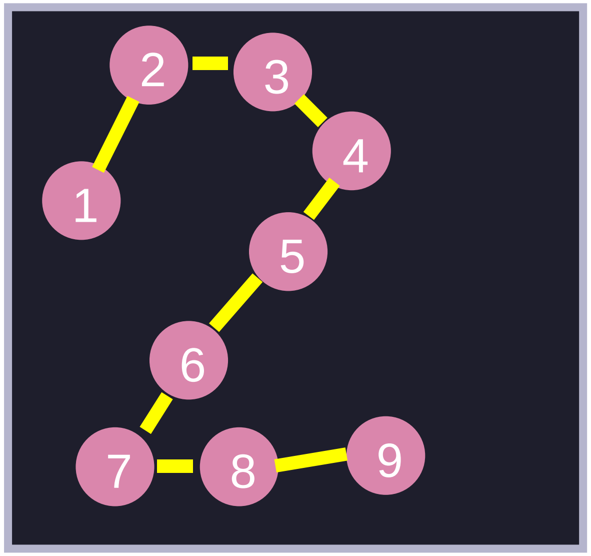
$M = (M_1, M_2, \ldots, M_8)$.

$Q = (Q_1, Q_2, \ldots, Q_9)$.

- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$
- Can be many-to-many.
  - M_5 and M_6 are matched to Q_7.

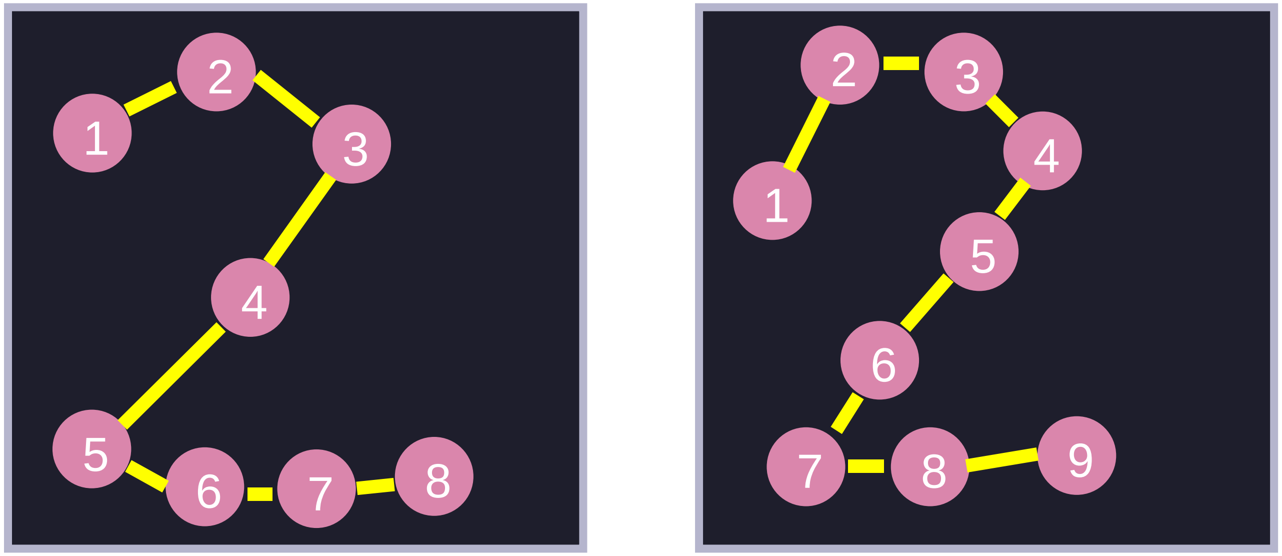
Matching Trajectories





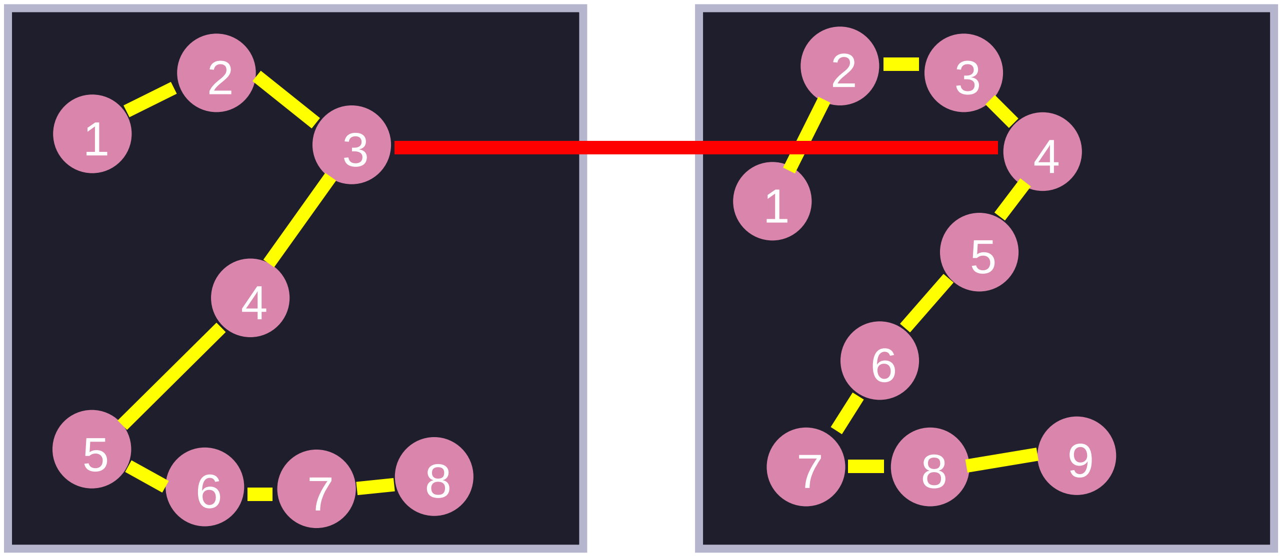
- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$
- Cost of alignment:

Matching Trajectories



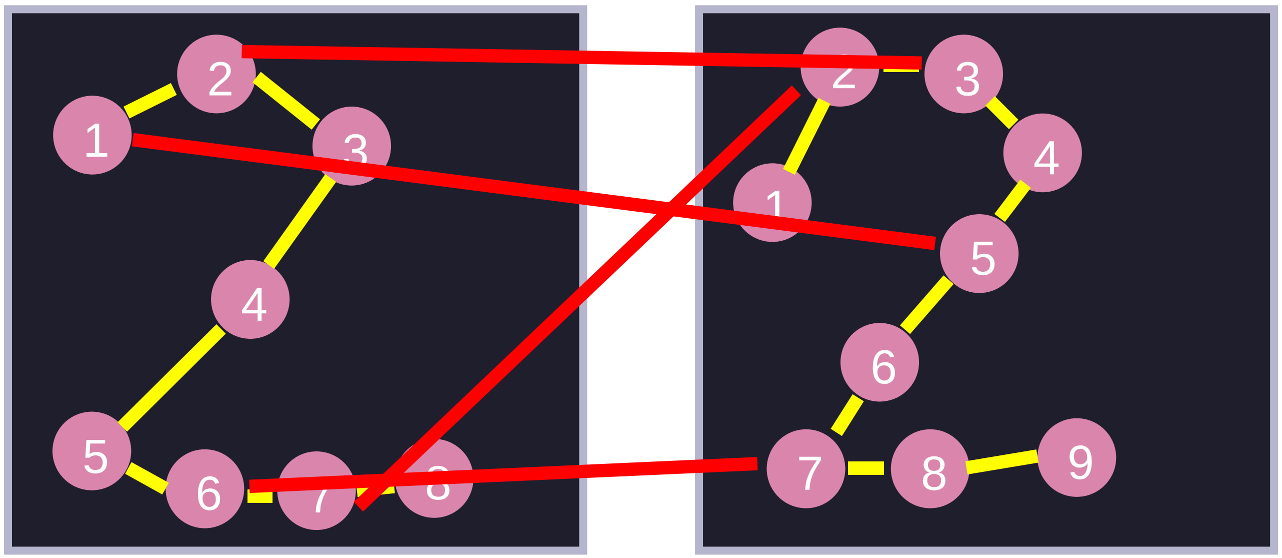
- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$
- Cost of alignment:
  - $\text{cost}(s_1, t_1) + \text{cost}(s_2, t_2) + \ldots + \text{cost}(s_m, t_n)$

Matching Trajectories



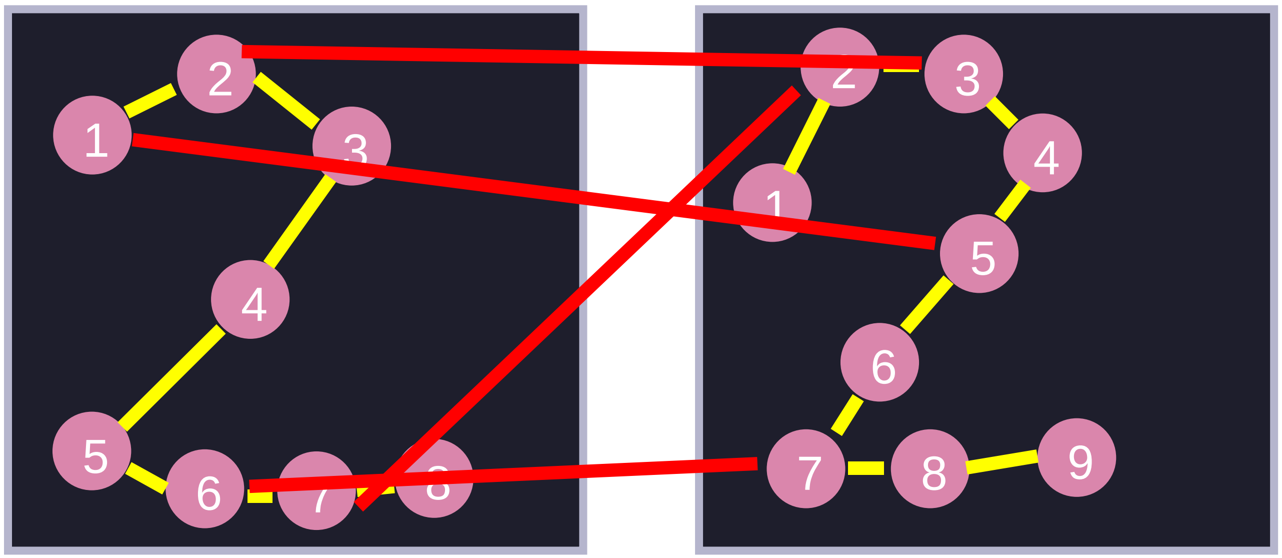
- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$
- Cost of alignment:
  - $\text{cost}(s_1, t_1) + \text{cost}(s_2, t_2) + \ldots + \text{cost}(s_m, t_n)$
  - Example: $\text{cost}(s_i, t_i)$ = Euclidean distance between locations.
  - Cost(3, 4) = Euclidean distance between M_3 and Q_4.

Matching Trajectories



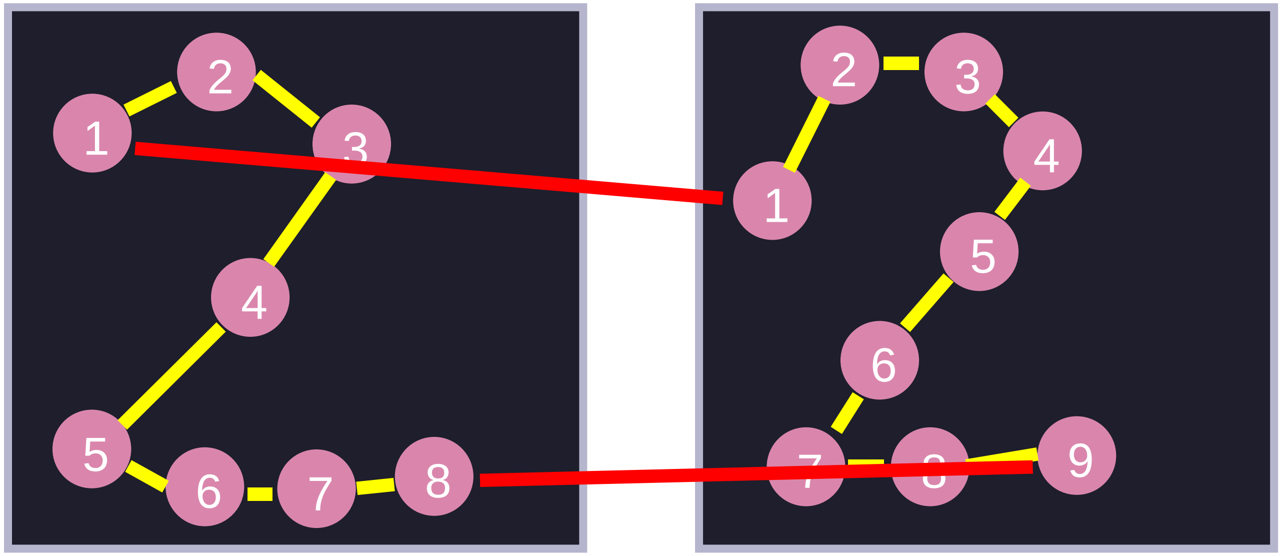
- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$
- Rules of alignment.
  - Is alignment ((1, 5), (2, 3), (6, 7), (7, 1)) legal?

Matching Trajectories



- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$
- Rules of alignment.
  - Is alignment ((1, 5), (2, 3), (6, 7), (7, 1)) legal?
  - Depends on what makes sense in our application.

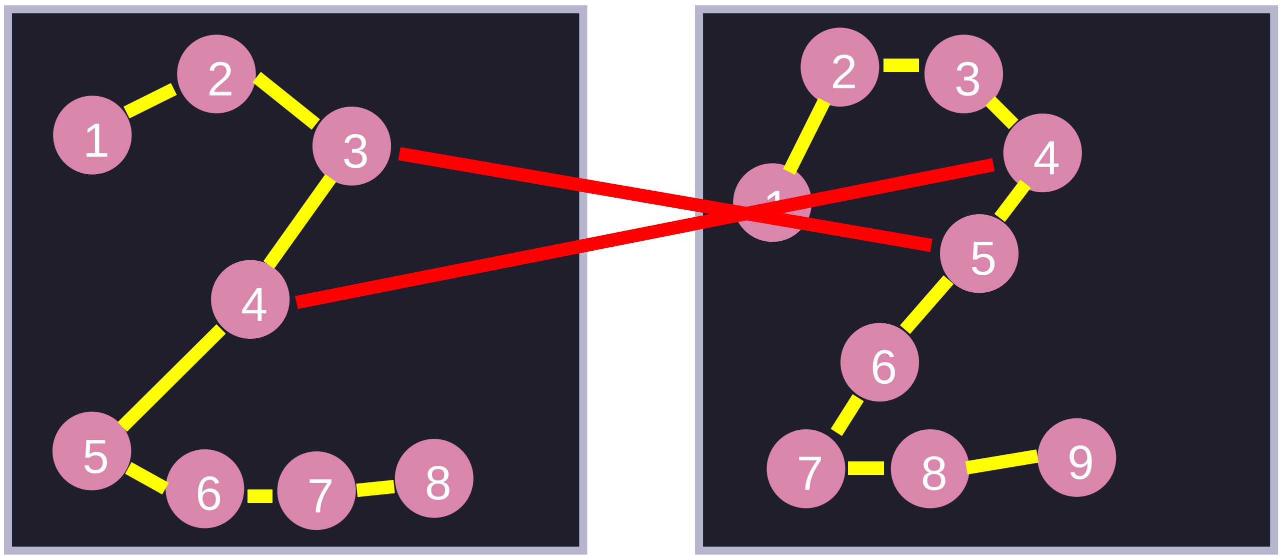
Matching Trajectories



- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$
- Dynamic time warping rules: boundaries
  - $s_1 = 1$, $t_1 = 1$.
  - $s_p = m$ = length of first sequence
  - $t_p = n$ = length of second sequence.

first elements match
last elements match

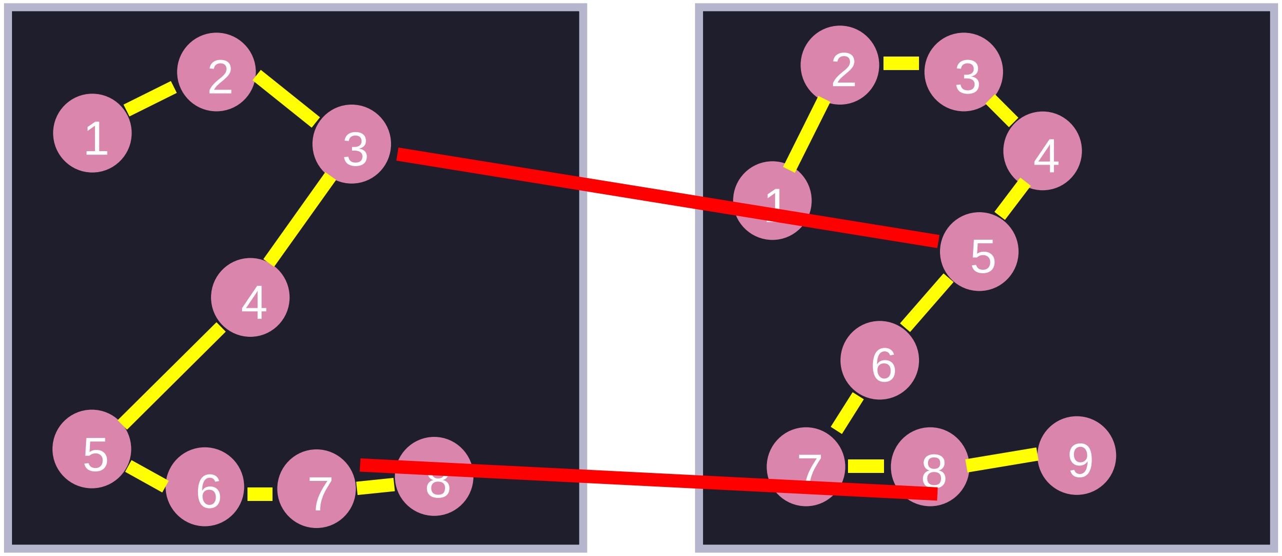
Matching Trajectories



- Illegal alignment (violating monotonicity):
  - (…, (3, 5), (4, 3), …).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$
- Dynamic time warping rules: monotonicity.
  - $0 <= (s_{t+1} - s_{t|})$
  - $0 <= (t_{t+1} - t_{t|})$

The alignment cannot go backwards.

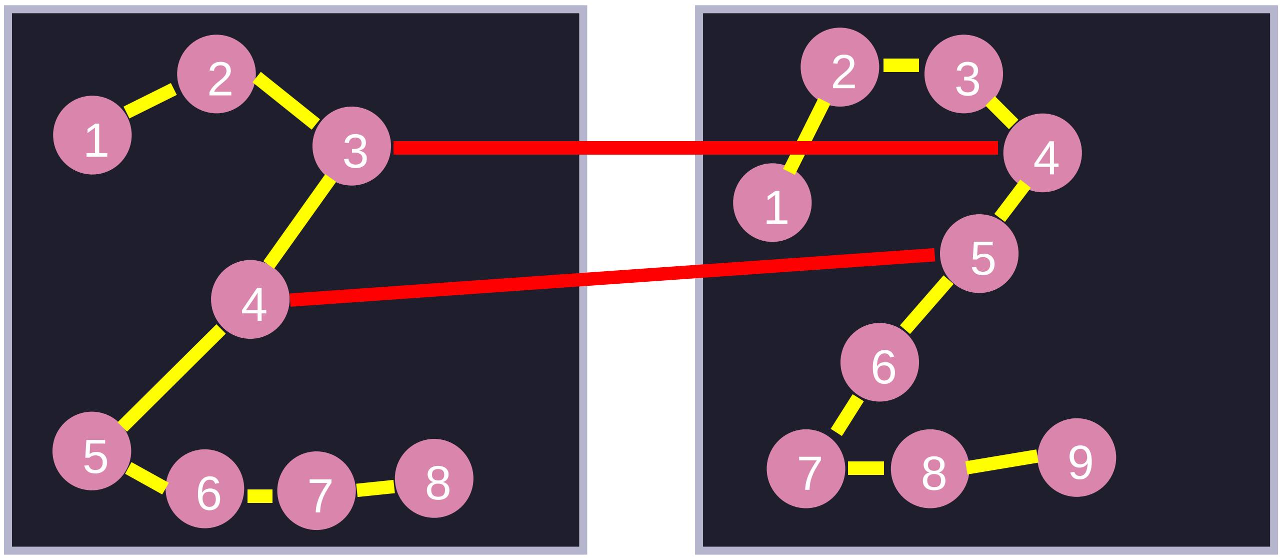
Matching Trajectories



- Illegal alignment (violating continuity).
  - (…, (3, 5), (6, 7), …).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$
- Dynamic time warping rules: continuity
  - $(s_{t+1} - s_{t|}) <= 1$
  - $(t_{t+1} - t_{t|}) <= 1$

The alignment cannot skip elements.

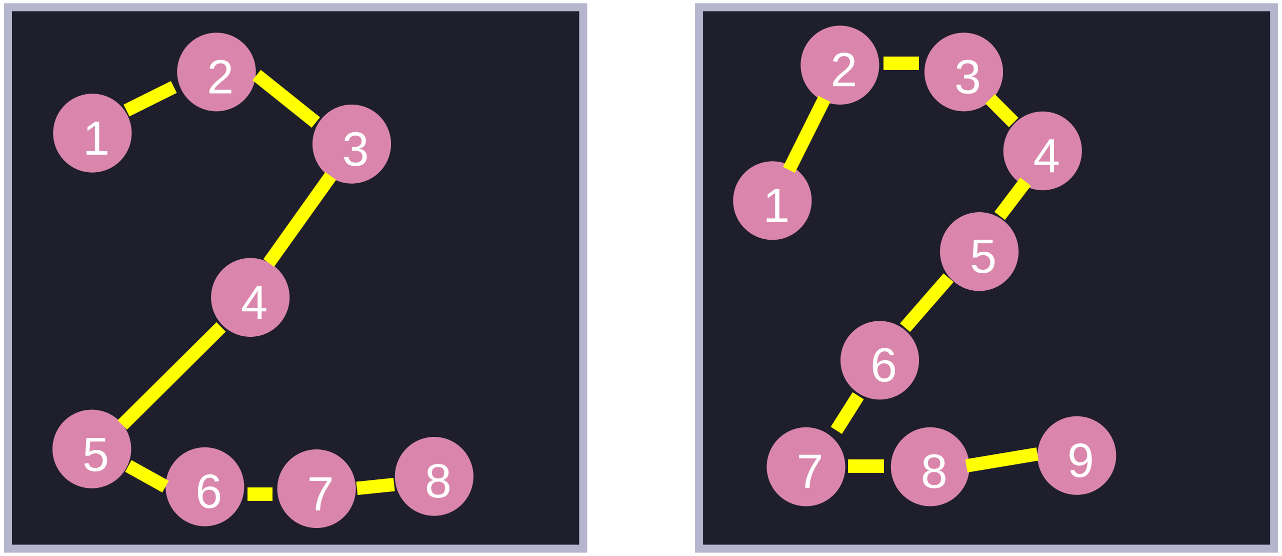
Matching Trajectories



- Alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$
- Dynamic time warping rules: monotonicity, continuity
  - $0 <= (s_{t+1} - s_{t|}) <= 1$
  - $0 <= (t_{t+1} - t_{t|}) <= 1$

The alignment cannot go backwards.
The alignment cannot skip elements.

Dynamic Time Warping



- Dynamic Time Warping (DTW) is a distance measure between *sequences* of points.
- The DTW distance is the cost of the *optimal* alignment between two trajectories.
  - The alignment must obey the DTW rules defined in the previous slides.

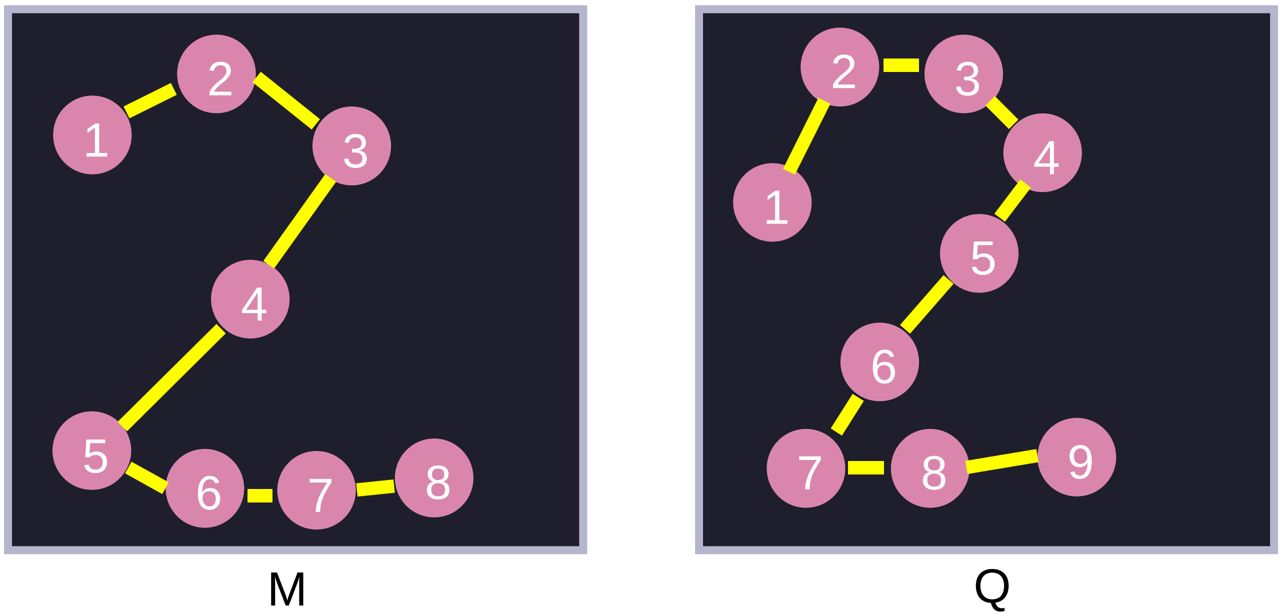
DTW Assumptions

- The gesturing hand must be detected correctly.
- For each gesture class, we have training examples.
- Given a new gesture to classify, we find the most similar gesture among our training examples.
  - What type of classifier is this?

DTW Assumptions

- The gesturing hand must be detected correctly.
- For each gesture class, we have training examples.
- Given a new gesture to classify, we find the most similar gesture among our training examples.
  - Nearest neighbor classification, using DTW as the distance measure.

Computing DTW



- Training example M = (M_1, M_2, …, M_8).
- Test example Q = (Q_1, Q_2, …, Q_9).
- Each M_i and Q_j can be, for example, a 2D pixel location.

Computing DTW

- Training example $M = (M_1, M_2, \ldots, M_{10})$.
- Test example $Q = (Q_1, Q_2, \ldots, Q_{15})$.
- We want optimal alignment between M and Q.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j).
    - Problem(i,j): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$.
- Solve problem(1, 1):

Computing DTW

- Training example $M = (M_1, M_2, \ldots, M_{10})$.
- Test example $Q = (Q_1, Q_2, \ldots, Q_{15})$.
- We want optimal alignment between M and Q.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j).
    - Problem(i,j): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$.
- Solve problem(1, 1):
  - Optimal alignment: ((1, 1)).

Computing DTW

- Training example $M = (M_1, M_2, \ldots, M_{10})$.
- Test example $Q = (Q_1, Q_2, \ldots, Q_{15})$.
- We want optimal alignment between M and Q.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j).
    - Problem(i,j): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$.
- Solve problem(1, j):

Computing DTW

- Training example $M = (M_1, M_2, \ldots, M_{10})$.
- Test example $Q = (Q_1, Q_2, \ldots, Q_{15})$.
- We want optimal alignment between M and Q.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j).
    - Problem(i,j): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$.
- Solve problem(1, j):
  - Optimal alignment: ((1, 1), (1, 2), …, (1, j)).

Computing DTW

- Training example $M = (M_1, M_2, \ldots, M_{10})$.
- Test example $Q = (Q_1, Q_2, \ldots, Q_{15})$.
- We want optimal alignment between M and Q.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j).
    - Problem(i,j): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$.
- Solve problem(i, 1):
  - Optimal alignment: ((1, 1), (2, 1), …, (i, 1)).

Computing DTW

- Training example $M = (M_1, M_2, \ldots, M_{10})$.
- Test example $Q = (Q_1, Q_2, \ldots, Q_{15})$.
- We want optimal alignment between M and Q.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j).
    - Problem(i,j): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$.
- Solve problem(i, j):

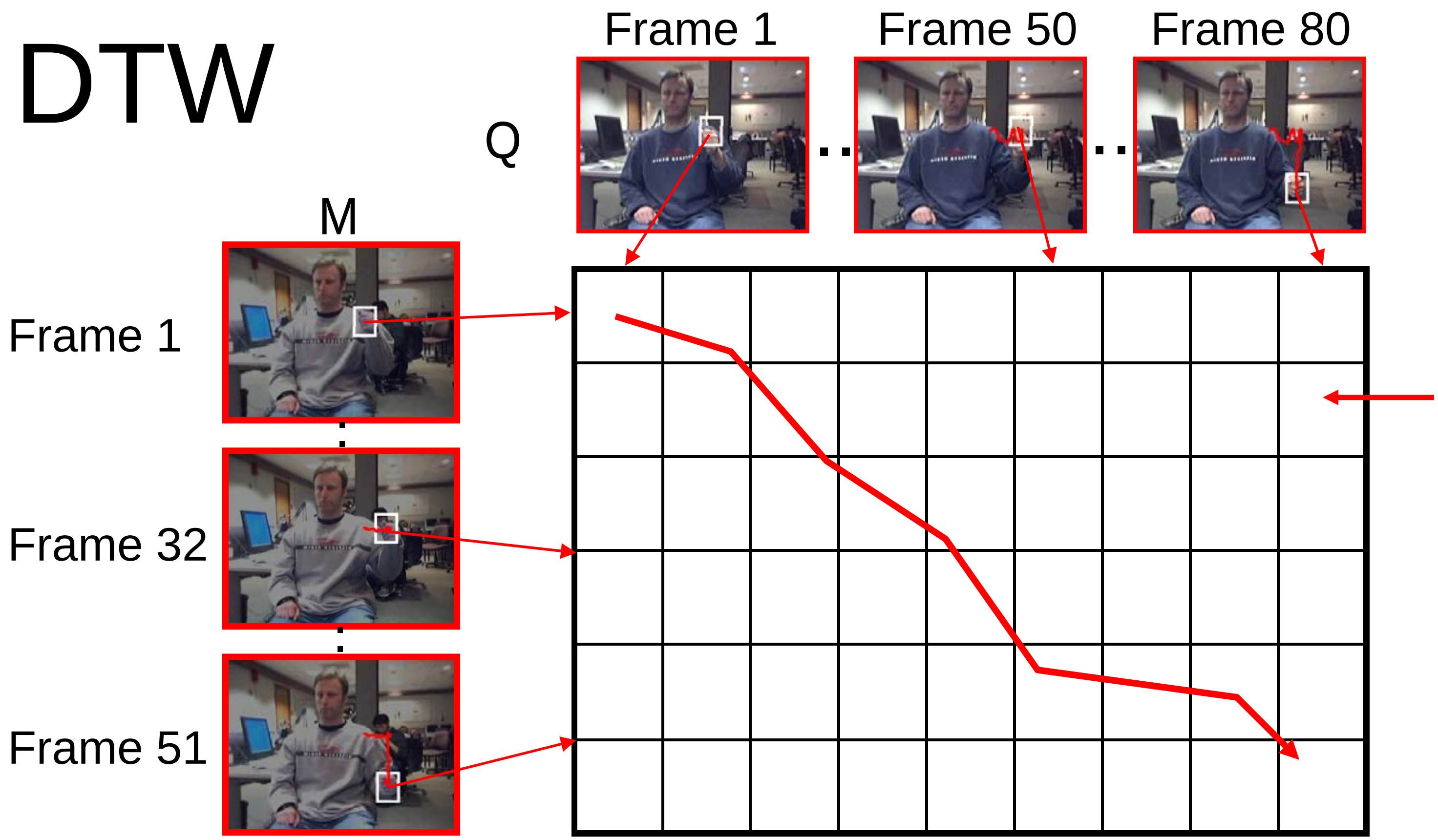
Computing DTW

- Training example M = (M_1, M_2, …, M_{10}).
- Test example Q = (Q_1, Q_2, …, Q_{15}).
- We want optimal alignment between M and Q.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j).
    - Problem(i,j): find optimal alignment between (M_1, …, M_i) and (Q_1, …, Q_j).
- Solve problem(i, j):
  - Find best solution from (i, j-1), (i-1, j), (i-1, j-1).
  - Add to that solution the pair (i, j).

Computing DTW

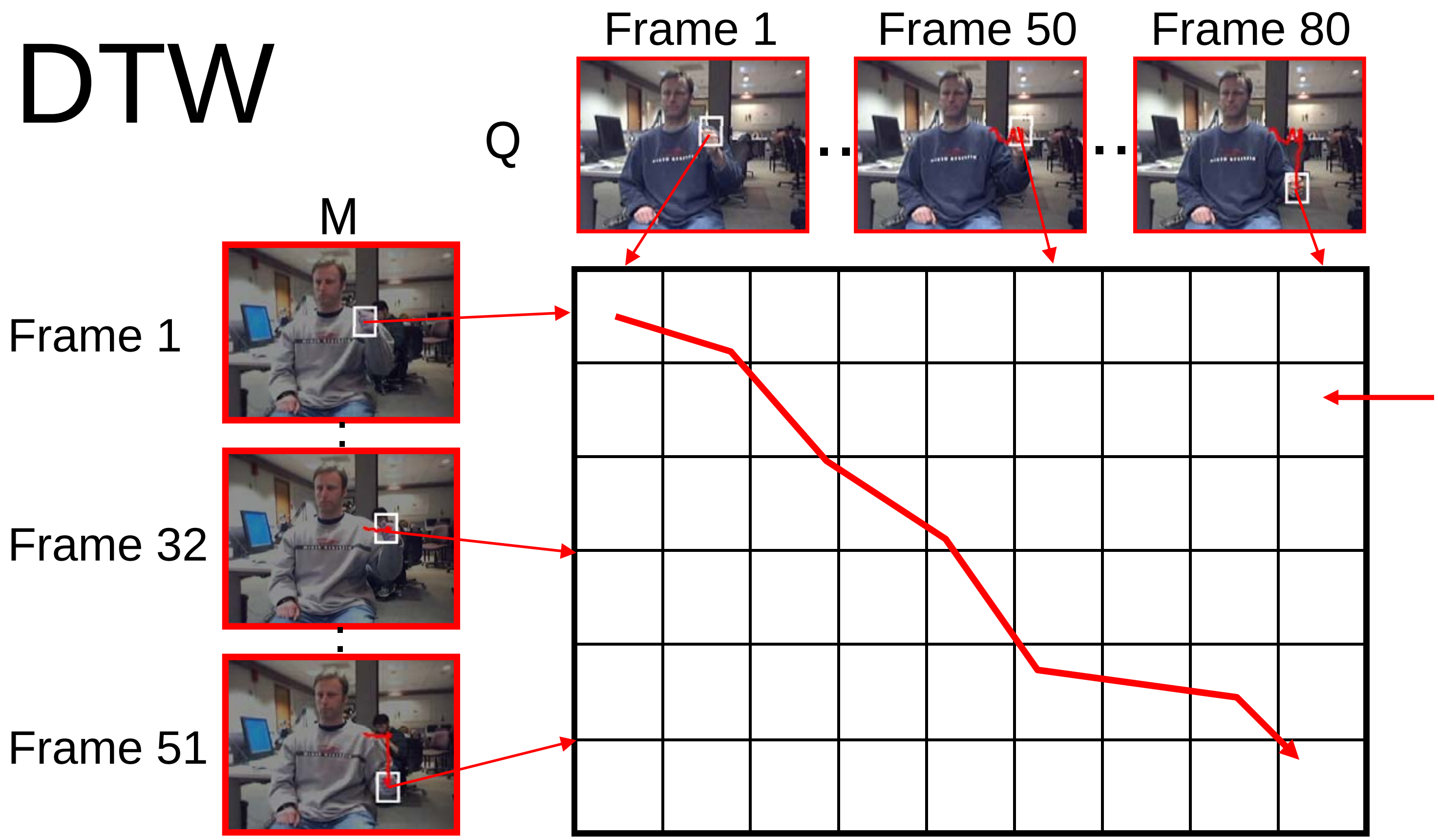
- Input:
  - Training example $M = (M_1, M_2, \ldots, M_m)$.
  - Test example $Q = (Q_1, Q_2, \ldots, Q_n)$.
- Initialization:
  - scores = zeros(m, n).
  - scores(1, 1) = cost(M_1, Q_1).
  - For i = 2 to m: scores(i, 1) = scores(i-1, 1) + cost(M_i, Q_1).
  - For j = 2 to n: scores(1, j) = scores(1, j-1) + cost(M_1, Q_j).
- Main loop:
  - For i = 2 to m, for j = 2 to n:
    - scores(i, j) = cost(M_i, Q_j) + min{scores(i-1, j), scores(i, j-1), scores(i-1, j-1)}.
- Return scores(m, n).

DTW



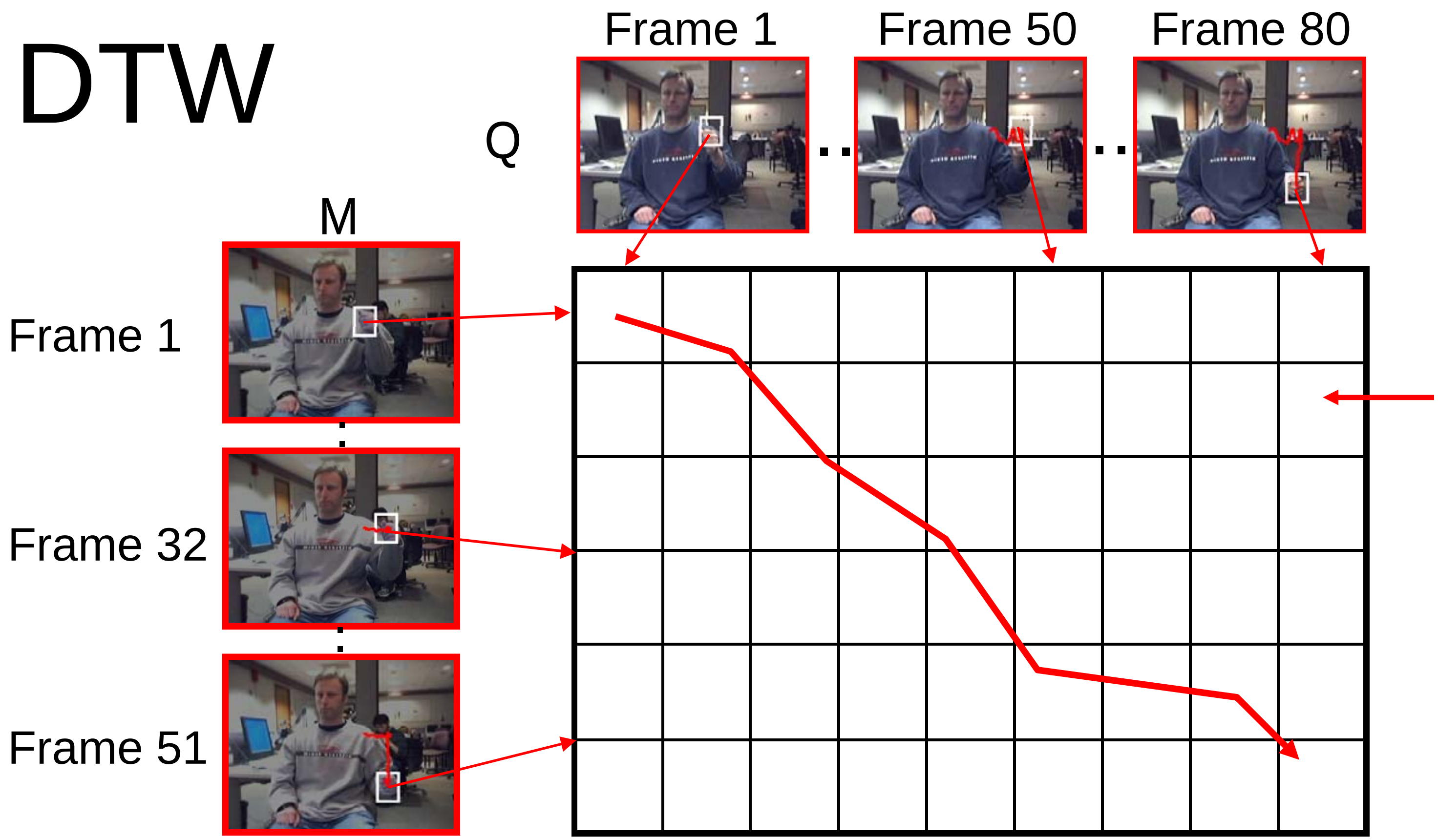
- For each cell (i, j):
  - Compute optimal alignment of M(1:i) to Q(1:j).
  - Answer depends only on (i-1, j), (i, j-1), (i-1, j-1).
  - Time:

DTW



- For each cell (i, j):
  - Compute optimal alignment of M(1:i) to Q(1:j).
  - Answer depends only on (i-1, j), (i, j-1), (i-1, j-1).
  - Time: Linear to size of table.

DTW



- For each cell (i, j):
  - Compute optimal alignment of M(1:i) to Q(1:j).
  - Answer depends only on (i-1, j), (i, j-1), (i-1, j-1).
  - Time: Quadratic to length of gestures.

DTW Finds the Optimal Alignment

- Proof:

DTW Finds the Optimal Alignment

- Proof: by induction.
- Base cases:

DTW Finds the Optimal Alignment

- Proof: by induction.
- Base cases:
  - $i = 1$ OR $j = 1$.

DTW Finds the Optimal Alignment

- Proof: by induction.
- Base cases:
  - i = 1 OR j = 1.
- Proof of claim for base cases:
  - For any problem(i, 1) and problem(1, j), only one legal warping path exists.
  - Therefore, DTW finds the optimal path for problem(i, 1) and problem(1, j)
    - It is optimal since it is the *only one*.

DTW Finds the Optimal Alignment

- Proof: by induction.
- General case:
  - (i, j), for i >= 2, j >= 2.
- Inductive hypothesis:

DTW Finds the Optimal Alignment

- Proof: by induction.
- General case:
  - (i, j), for i >= 2, j >= 2.
- Inductive hypothesis:
  - What we want to prove for (i, j) is true for (i-1, j), (i, j-1), (i-1, j-1):

DTW Finds the Optimal Alignment

- Proof: by induction.
- General case:
  – (i, j), for i >= 2, j >= 2.
- Inductive hypothesis:
  – What we want to prove for (i, j) is true for (i-1, j), (i, j-1), (i-1, j-1):
  – DTW has computed optimal solution for problems (i-1, j), (i, j-1), (i-1, j-1).

DTW Finds the Optimal Alignment

- Proof: by induction.
- General case:
  - (i, j), for i >= 2, j >= 2.
- Inductive hypothesis:
  - What we want to prove for (i, j) is true for (i-1, j), (i, j-1), (i-1, j-1):
  - DTW has computed optimal solution for problems (i-1, j), (i, j-1), (i-1, j-1).
- Proof by contradiction:

DTW Finds the Optimal Alignment

- Proof: by induction.
- General case:
  - (i, j), for i >= 2, j >= 2.
- Inductive hypothesis:
  - What we want to prove for (i, j) is true for (i-1, j), (i, j-1), (i-1, j-1):
  - DTW has computed optimal solution for problems (i-1, j), (i, j-1), (i-1, j-1).
- Proof by contradiction:
  - If solution for (i, j) not optimal, then one of the solutions for (i-1, j), (i, j-1), or (i-1, j-1) was not optimal.

Handling Unknown Start and End

- So far, can our approach handle cases where we do not know the start and end frame?

Handling Unknown Start and End

- So far, can our approach handle cases where we do not know the start and end frame?
  - No.
- Why is it important to handle this case?

Handling Unknown Start and End

- So far, can our approach handle cases where we do not know the start and end frame?
  - No.
- Why is it important to handle this case?
  - Otherwise, how would the system know that the user is performing a gesture?
  - Users may do other things with their hands (move them aimlessly, perform a task, wave at some other person…).
  - The system needs to know when a command has been performed.

Handling Unknown Start and End

- So far, can our approach handle cases where we do not know the start and end frame?
  - No.
- Recognizing gestures when the start and end frame is not known is called *gesture spotting.*

Handling Unknown Start and End

- So far, can our approach handle cases where we do not know the start and end frame?
  - No.
- How do we handle unknown start and end frames?

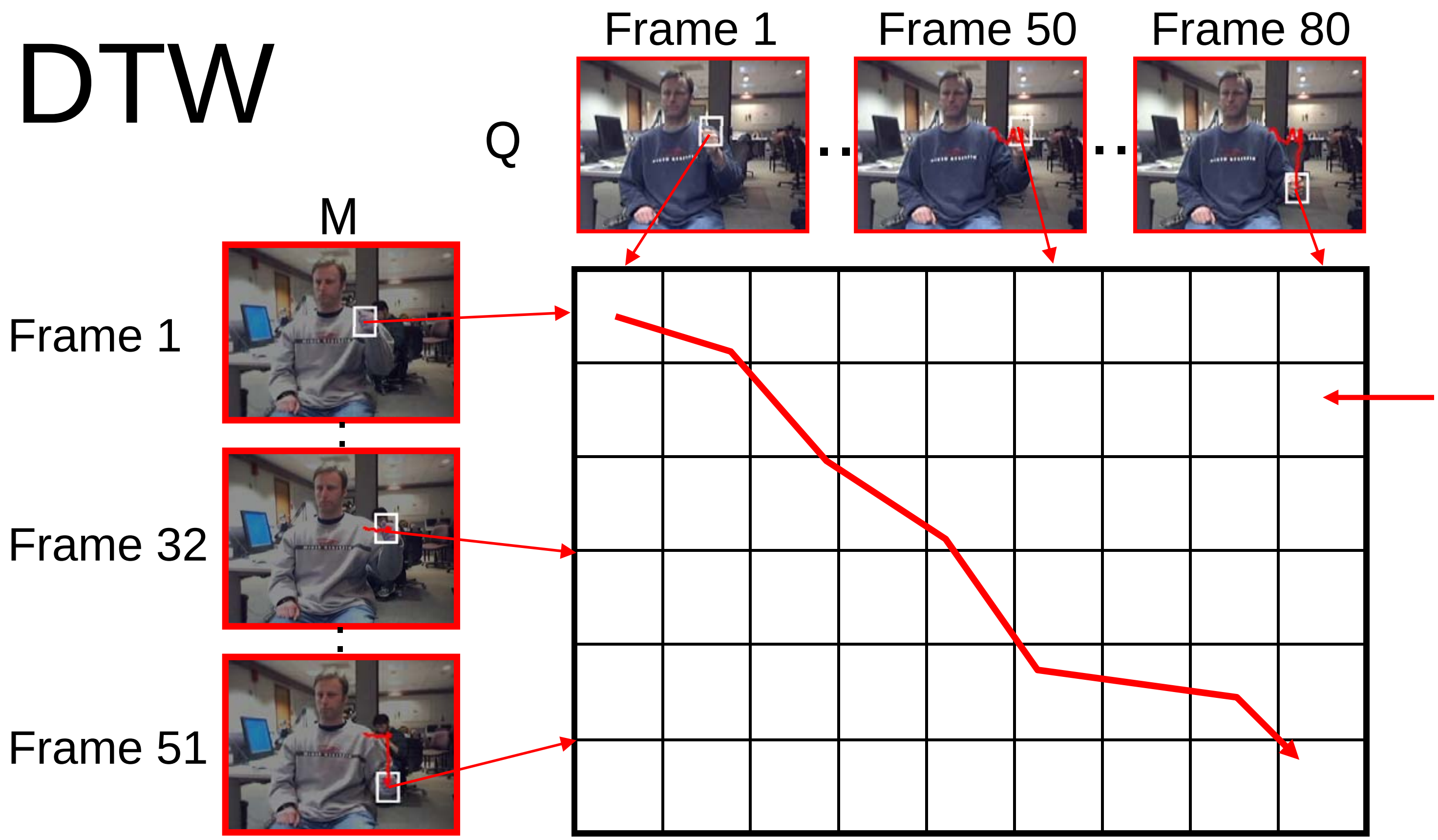
Handling Unknown Start and End

- So far, can our approach handle cases where we do not know the start and end frame?
  - No.
- How do we handle unknown end frames?

Handling Unknown Start and End

- So far, can our approach handle cases where we do not know the start and end frame?
  - No.
- How do we handle unknown end frames?
  - Assume, temporarily, that we know the start frame.

DTW



- For each cell (i, j):
  - Compute optimal alignment of M(1:i) to Q(1:j).
  - Answer depends only on (i-1, j), (i, j-1), (i-1, j-1).
  - The last column evaluates all possible end frames.

Handling Unknown Start and End

- How do we handle unknown end frames?
  - Assume, temporarily, that we know the start frame.
  - Instead of looking at scores(m, n), we look at scores(m, j) for all j in {1, …, n}.
    - m is length of training sequence.
    - n is length of query sequence.
  - scores(m, j) tells us the optimal cost of matching the entire training sequence to the first j frames of Q.
  - Finding the smallest scores(m, j) tells us where the gesture ends.

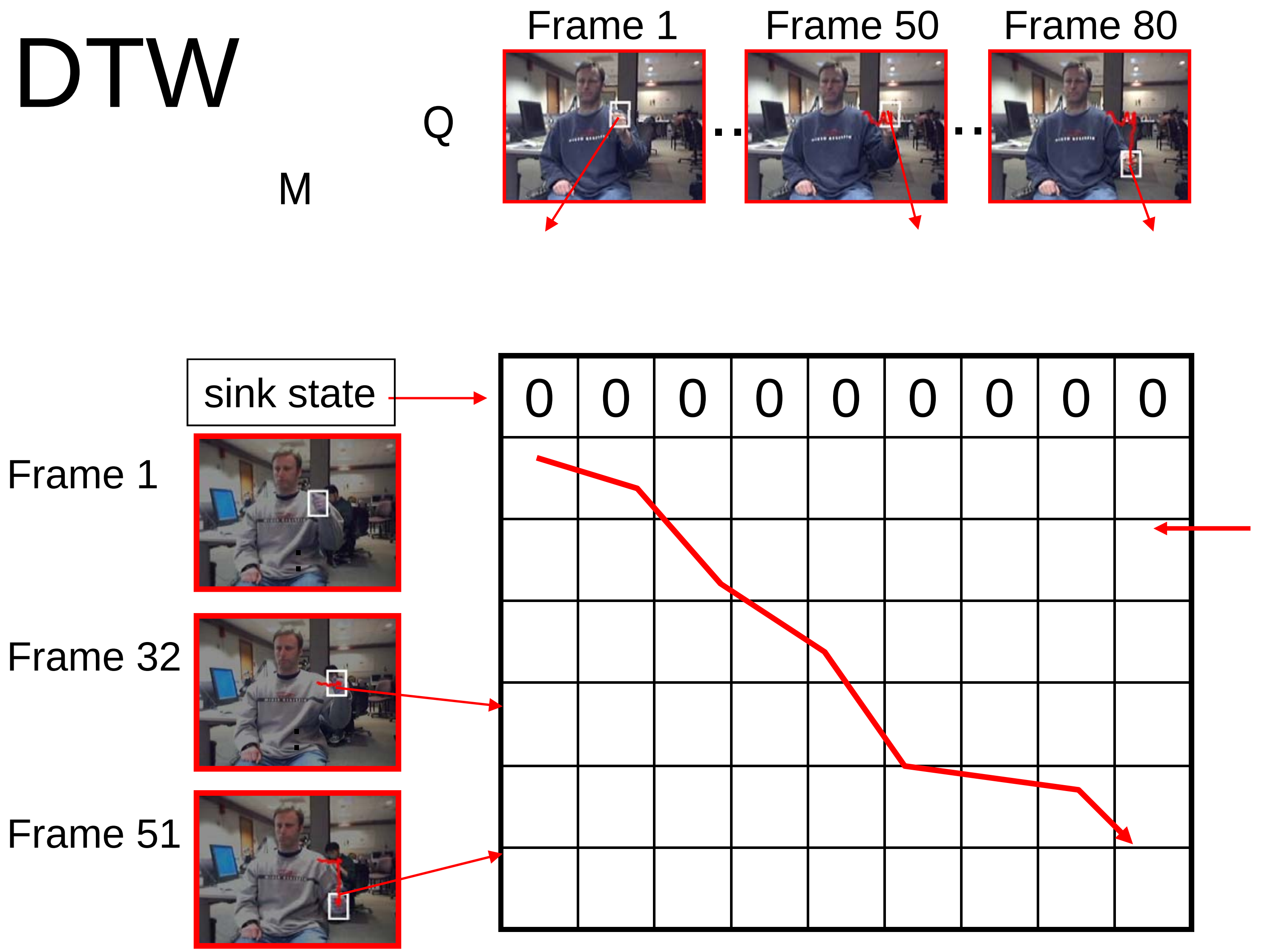
Handling Unknown Start and End

- How do we handle unknown start frames?

Handling Unknown Start and End

- How do we handle unknown start frames?
  - Make every training sequence start with a *sink* symbol.
  - Replace $M = (M_1, M_2, \ldots, M_m)$ with $M = (M_0, M_1, \ldots, M_m)$.
  - $M_0 = sink$.
    - $\text{Cost}(0, j) = 0$ for all j.
- The *sink* symbol can match the frames of the test sequence that *precede* the gesture.

DTW



Frame 1
Frame 50
Frame 80
Q
M
sink state
0 0 0 0 0 0 0 0 0
Frame 1
Frame 32
Frame 51

DTW for Gesture Spotting

- Training example $M = (M_1, M_2, \ldots, M_{10})$.
- Test example $Q = (Q_1, Q_2, \ldots, Q_{15})$.
- We want optimal alignment between M and Q.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j).
    - Problem(i,j): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$.
- Solve problem(0, 0):

DTW for Gesture Spotting

- Training example $M = (M_1, M_2, \ldots, M_{10})$.
- Test example $Q = (Q_1, Q_2, \ldots, Q_{15})$.
- We want optimal alignment between M and Q.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j).
    - Problem(i,j): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$.
- Solve problem(0, 0):
  - Optimal alignment: none. Cost: infinity.

DTW for Gesture Spotting

- Training example $M = (M_1, M_2, \ldots, M_{10})$.
- Test example $Q = (Q_1, Q_2, \ldots, Q_{15})$.
- We want optimal alignment between M and Q.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j).
    - Problem(i,j): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$.
- Solve problem(0, j): j >= 1.

DTW for Gesture Spotting

- Training example $M = (M_1, M_2, \ldots, M_{10})$.
- Test example $Q = (Q_1, Q_2, \ldots, Q_{15})$.
- We want optimal alignment between M and Q.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j).
    - Problem(i,j): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$.
- Solve problem(0, j): j >= 1.
  - Optimal alignment: none. Cost: zero.

DTW for Gesture Spotting

- Training example $M = (M_1, M_2, \ldots, M_{10})$.
- Test example $Q = (Q_1, Q_2, \ldots, Q_{15})$.
- We want optimal alignment between M and Q.
- Dynamic programming strategy:
  - – Break problem up into smaller, interrelated problems (i,j).
    - Problem(i,j): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$.
- Solve problem(i, 0): i >= 1.

DTW for Gesture Spotting

- Training example $M = (M_1, M_2, \ldots, M_{10})$.
- Test example $Q = (Q_1, Q_2, \ldots, Q_{15})$.
- We want optimal alignment between M and Q.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j).
    - Problem(i,j): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$.
- Solve problem(i, 0): i >= 1.
  - Optimal alignment: none. Cost: infinity.
    - We do not allow skipping a model frame.

DTW for Gesture Spotting

- Training example $M = (M_1, M_2, \ldots, M_{10})$.
- Test example $Q = (Q_1, Q_2, \ldots, Q_{15})$.
- We want optimal alignment between M and Q.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j).
    - Problem(i,j): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$.
- Solve problem(i, j):

DTW for Gesture Spotting

- Training example $M = (M_1, M_2, \ldots, M_{10})$.
- Test example $Q = (Q_1, Q_2, \ldots, Q_{15})$.
- We want optimal alignment between M and Q.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j).
    - Problem(i,j): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$.
- Solve problem(i, j):
  - Find best solution from (i, j-1), (i-1, j), (i-1, j-1).
  - Add to that solution the pair (i, j).

DTW for Gesture Spotting

- Input:
  - Training example $M = (M_1, M_2, \ldots, M_m)$.
  - Test example $Q = (Q_1, Q_2, \ldots, Q_n)$.
- Initialization:
  - scores = zeros(0:m, 0:n).
  - scores(0:m, 0) = infinity
- Main loop:
  - For i = 1 to m, for j = 1 to n:
    - scores(i, j) = cost(M_i, Q_j) + min{scores(i-1, j), scores(i, j-1), scores(i-1, j-1)}.
- Return scores(m, n).

- Note: In Matlab, the code is a bit more messy, because array indices start at 1, whereas in the pseudocode above array indices start at zero.

At Run Time

- Assume known start/end frames.
- Assume N classes, M examples per class.
- How do we classify a gesture G?

At Run Time

- Assume known start/end frames.
- Assume N classes, M examples per class.
- How do we classify a gesture G?
  - We compute the DTW (or DSTW) score between input gesture G and each of the M*N training examples.
  - We pick the class of the nearest neighbor.
    - Alternatively, the class of the majority of the k-nearest neighbor, where k can be chosen using training data.

At Run Time

- Assume *unknown* start/end frames.
- Assume N classes, M examples per class.
- What do we do at frame *t*?

At Run Time

- Assume *unknown* start/end frames.
- Assume N classes, M examples per class.
- What do we do at frame *t*?
  - For each of the M*N examples, we maintain a table of scores.
  - Suppose we keep track, for each of the M*N examples, of the dynamic programming table constructed by matching those examples with Q_1, Q_2, …, Q_{t-1}.

At Run Time

- Assume *unknown* start/end frames.
- Assume N classes, M examples per class.
- What do we do at frame *t*?
  - For each of the M*N examples, we maintain a table of scores.
  - We keep track, for each of the M*N examples, of the dynamic programming table constructed by matching those examples with Q_1, Q_2, …, Q_{t-1}.
  - At frame *t*, we add a new column to the table.
  - If, for any training example, the matching cost is below a threshold, we “recognize” a gesture.

At Run Time

- Assume *unknown* start/end frames.
- Assume N classes, M examples per class.
- What do we do at frame *t*?
  - For each of the M*N examples, we maintain a table of scores.
  - We keep track, for each of the M*N examples, of the dynamic programming table constructed by matching those examples with Q_1, Q_2, …, Q_{t-1}.
  - At frame *t*, we add a new column to the table.
  - How much memory do we need?

At Run Time

- Assume *unknown* start/end frames.
- Assume N classes, M examples per class.
- What do we do at frame *t*?
  - For each of the M*N examples, we maintain a table of scores.
  - We keep track, for each of the M*N examples, of the dynamic programming table constructed by matching those examples with Q_1, Q_2, …, Q_{t-1}.
  - At frame *t*, we add a new column to the table.
  - How much memory do we need?
  - Bare minimum: Remember only column *t-1* of the table.
  - Then, we lose info useful for finding the start frame.

Performance Evaluation

- How do we measure accuracy when start and end frames are known?

Performance Evaluation

- How do we measure accuracy when start and end frames are known?
  - Classification accuracy.
  - Similar to face recognition.

Performance Evaluation

- How do we measure accuracy when start and end frames are unknown?
  - What is considered a correct answer?
  - What is considered an incorrect answer?

Performance Evaluation

- How do we measure accuracy when start and end frames are unknown?
  - What is considered a correct answer?
  - What is considered an incorrect answer?
- Typically, requiring the start and end frames to have a specific value is too stringent.
  - Even humans themselves cannot agree when exactly a gesture starts and ends.
  - Usually, we allow some kind of slack.

Performance Evaluation

- How do we measure accuracy when start and end frames are unknown?
- Consider this rule:
  - When the system decides that a gesture occurred in frames (A1, …, B1), we consider the result correct when:
    - There was some true gesture at frames (A2, …, B2).
    - At least half of the frames in (A2, …, B2) are covered by (A1, …, B1).
    - The class of the gesture at (A2, …, B2) matches the class that the system reports for (A1, …, B1).
  - Any problems with this approach?

Performance Evaluation

- How do we measure accuracy when start and end frames are unknown?
- Consider this rule:
  - When the system decides that a gesture occurred in frames (A1, …, B1), we consider the result correct when:
    - There was some true gesture at frames (A2, …, B2).
    - At least half of the frames in (A2, …, B2) are covered by (A1, …, B1).
    - The class of the gesture at (A2, …, B2) matches the class that the system reports for (A1, …, B1).
  - What if A1 and B1 are really far from each other?

A Symmetric Rule

- How do we measure accuracy when start and end frames are unknown?
- When the system decides that a gesture occurred in frames (A1, …, B1), we consider the result correct when:
  - There was some true gesture at frames (A2, …, B2).
  - At least half+1 of the frames in (A2, …, B2) are covered by (A1, …, B1). (why half+1?)
  - At least half+1 of the frames in (A1, …, B1) are covered by (A2, …, B2). (again, why half+1?)
  - The class of the gesture at (A2, …, B2) matches the class that the system reports for (A1, …, B1).

Variations

- When the system decides that a gesture occurred in frames (A1, …, B1), we consider the result correct when:
  - There was some true gesture at frames (A2, …, B2).
  - At least half+1 of the frames in (A2, …, B2) are covered by (A1, …, B1). (why half+1?)
  - At least half+1 of the frames in (A1, …, B1) are covered by (A2, …, B2). (again, why half+1?)
  - The class of the gesture at (A2, …, B2) matches the class that the system reports for (A1, …, B1).
- Instead of half+1, we can use a more or less restrictive threshold.

Frame-Based Accuracy

- In reality, each frame can either belong to a gesture, or to the no-gesture class.
- The system assigns each frame to a gesture, or to the no-gesture class.
- For what percentage of frames is the system correct?

The Subgesture Problem

- Consider recognizing the 10 digit classes, in the spotting setup (unknown start/end frames).
- What can go wrong with DSTW?

The Subgesture Problem

- Consider recognizing the 10 digit classes, in the spotting setup (unknown start/end frames).
- When a 7 occurs, a 1 is also a good match.

The Subgesture Problem

- Consider recognizing the 10 digit classes, in the spotting setup (unknown start/end frames).
- When a 9 occurs, a 1 is also a good match.

The Subgesture Problem

- Consider recognizing the 10 digit classes, in the spotting setup (unknown start/end frames).
- When an 8 occurs, a 5 is also a good match.

The Subgesture Problem

- Additional rules/models are needed to address the subgesture problem.

System Components

- Hand detection/tracking.
- Trajectory matching.

Hand Detection

- What sources of information can be useful in order to find where hands are in an image?

Hand Detection

- What sources of information can be useful in order to find where hands are in an image?
  - – Skin color.
  - – Motion.
    - Hands move fast when a person is gesturing.
    - Frame differencing gives high values for hand regions.

- Implementation: look at code in

detect_hands.m

Hand Detection

```
function [scores, result, centers] = ...
    detect_hands(previous, current, next, ...
                 hand_size, suppression_factor, number)

negative_histogram = read_double_image('negatives.bin');
positive_histogram = read_double_image('positives.bin');
skin_scores = detect_skin(current, positive_histogram, negative_histogram);

previous_gray = double_gray(previous);
current_gray = double_gray(current);
next_gray = double_gray(next);
frame_diff = min(abs(current_gray-previous_gray), abs(current_gray-next_gray));

skin_motion_scores = skin_scores .* frame_diff;
scores = imfilter(skin_motion_scores, ones(hand_size), 'same', 'symmetric');
[result, centers] = top_detection_results(current, scores, hand_size, ...
                                          suppression_factor, number);
```

Problem: Hand Detection May Fail

```
[scores, result] = frame_hands(filename, current_frame, [41 31], 1, 1);
imshow(result / 255);
```

Problem: Hand Detection May Fail

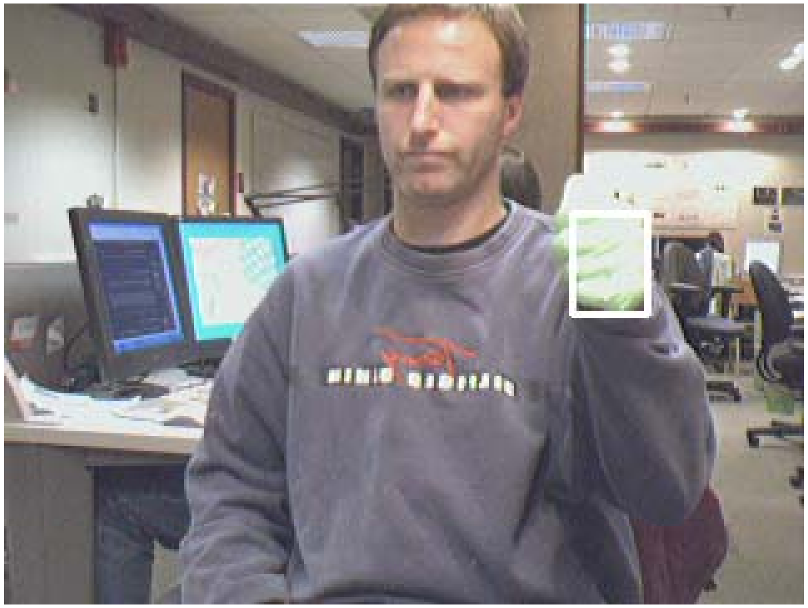
```
[scores, result] = frame_hands(filename, current_frame, [41 31], 1, 4);
imshow(result / 255);
```

Problem: Hand Detection May Fail

```
[scores, result] = frame_hands(filename, current_frame, [41 31], 1, 5);
imshow(result / 255);
```

Remedy: Cheat (sort of…)

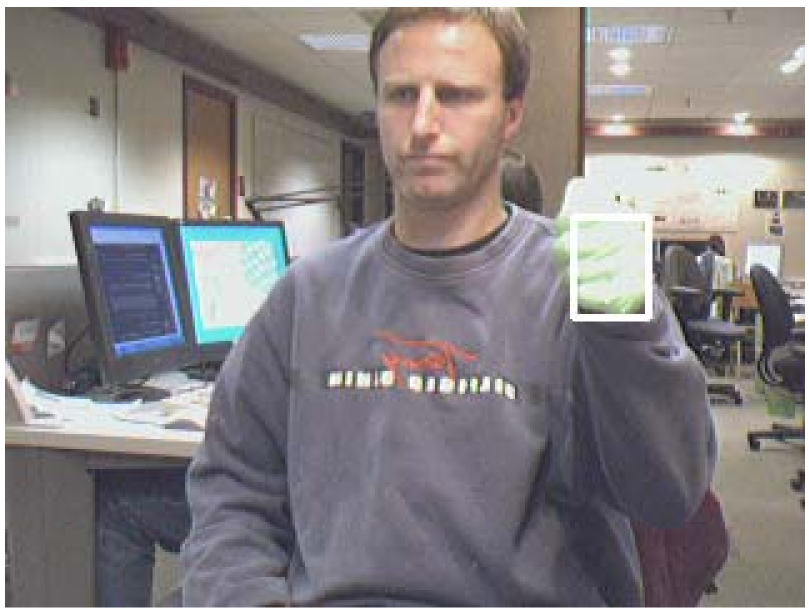
- We can use color gloves.
- Would that be reasonable?



```
[scores, result] = green_hands(filename, current_frame, [41 31]);
imshow(result / 255);
```

Remedy: Cheat (sort of…)

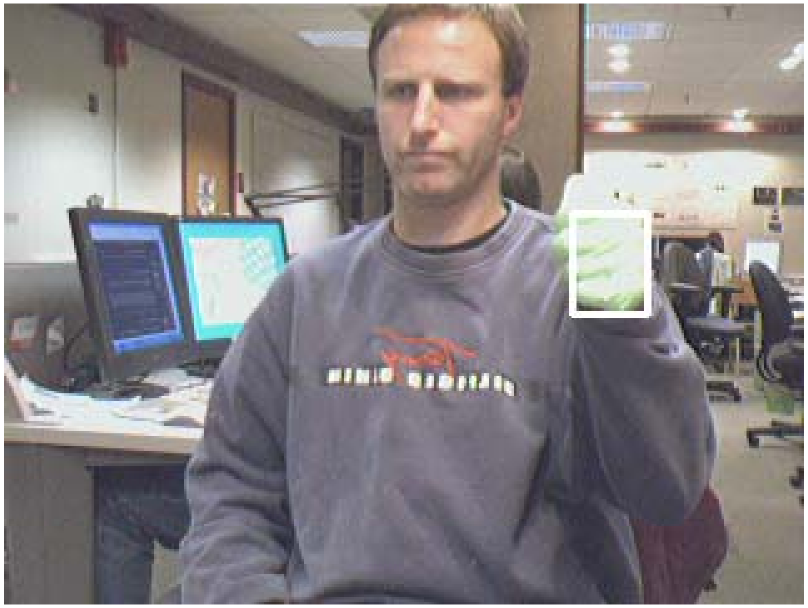
- We can use color gloves.
- Would that be reasonable?
  - Yes, when the user is willing to do it.
  - Example: collecting sign language data.



```
[scores, result] = green_hands(filename, current_frame, [41 31]);
imshow(result / 255);
```

Remedy: Cheat (sort of…)

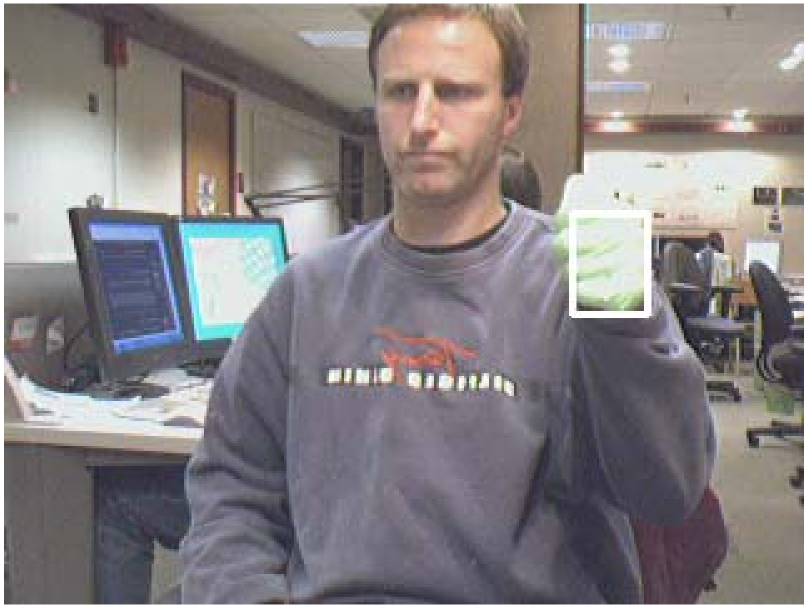
- We can use color gloves.
- Would that be reasonable?
  - No, when the user is not willing to do it.
  - Do you want to wear a green glove in your living room?



```
[scores, result] = green_hands(filename, current_frame, [41 31]);
imshow(result / 255);
```

Remedy: Cheat (sort of…)

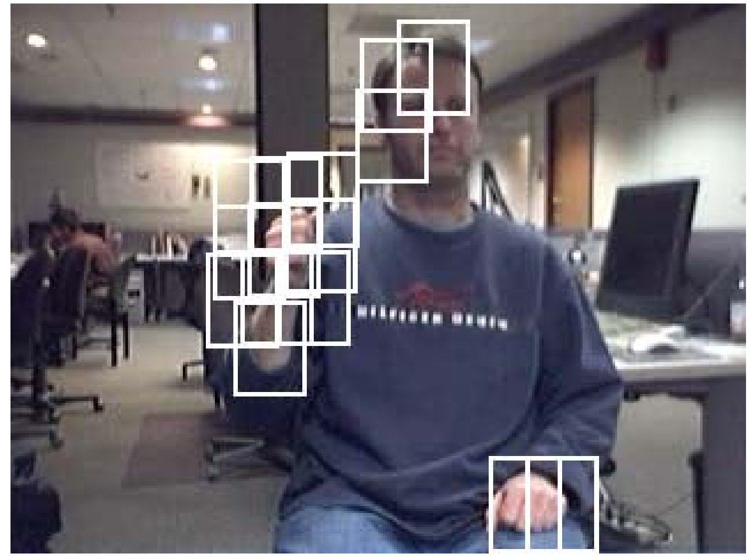
- We can use color gloves.
- Would that be reasonable?
  - No, when we do not control the data.
  - Example: Gesture recognition in movies.



```
[scores, result] = green_hands(filename, current_frame, [41 31]);
imshow(result / 255);
```

Remedy 2: Relax the Assumption of Correct Detection

input frame



hand candidates

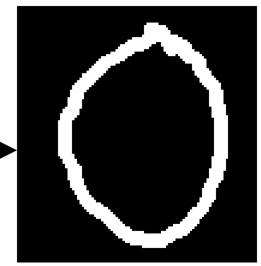
- Hand detection can return multiple candidates.
  - Design a recognition module for this type of input.
  - Solution: Dynamic Space-Time Warping (DSTW).

Bottom-Up Recognition Approach

input sequence

Detector
Tracker

trajectory



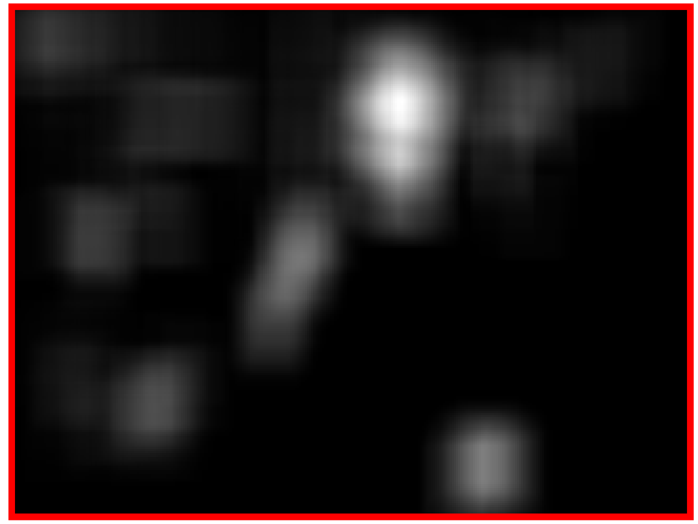
Classifier

class “0”

Bottom-up Shortcoming

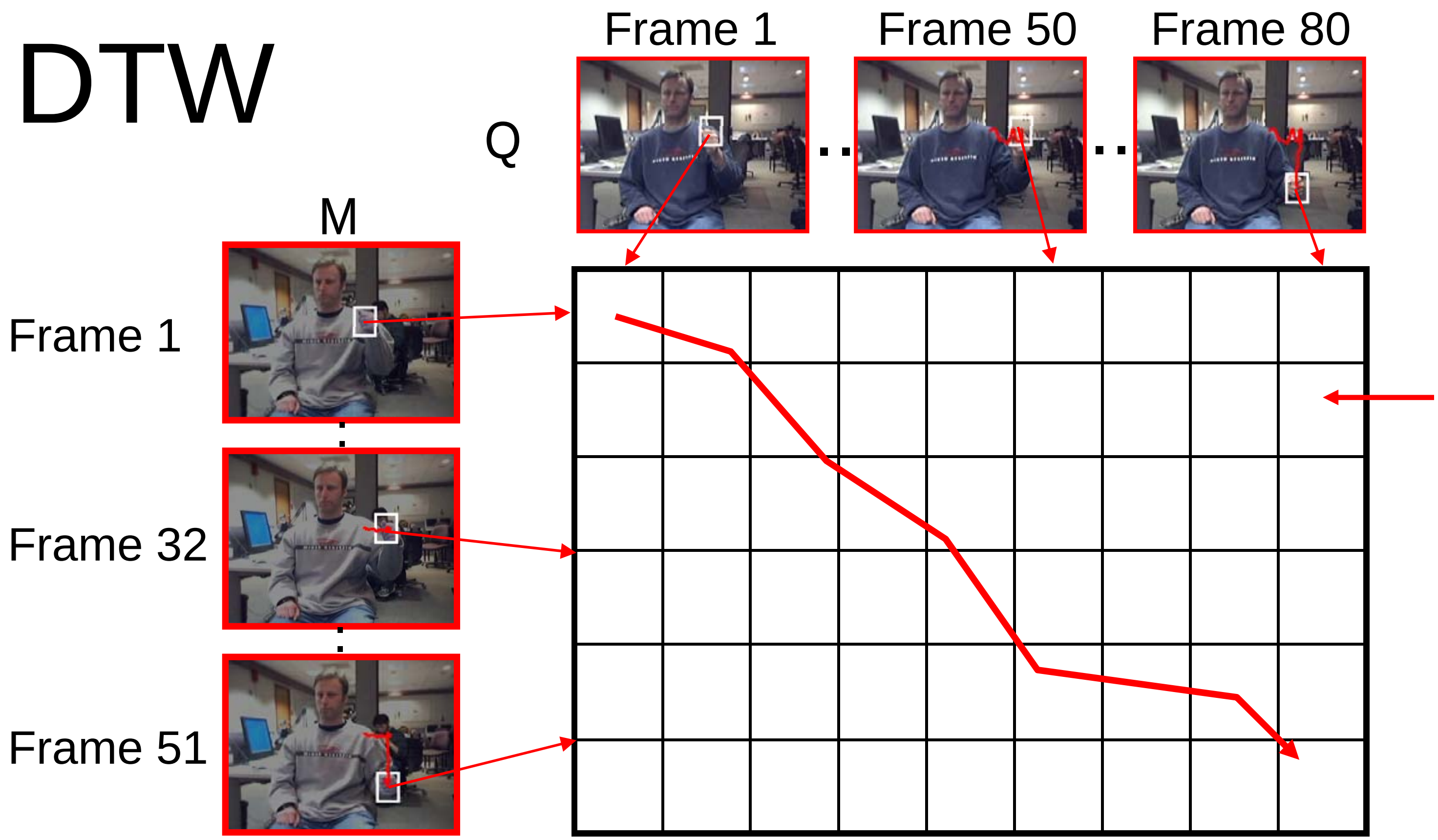
input frame

hand likelihood



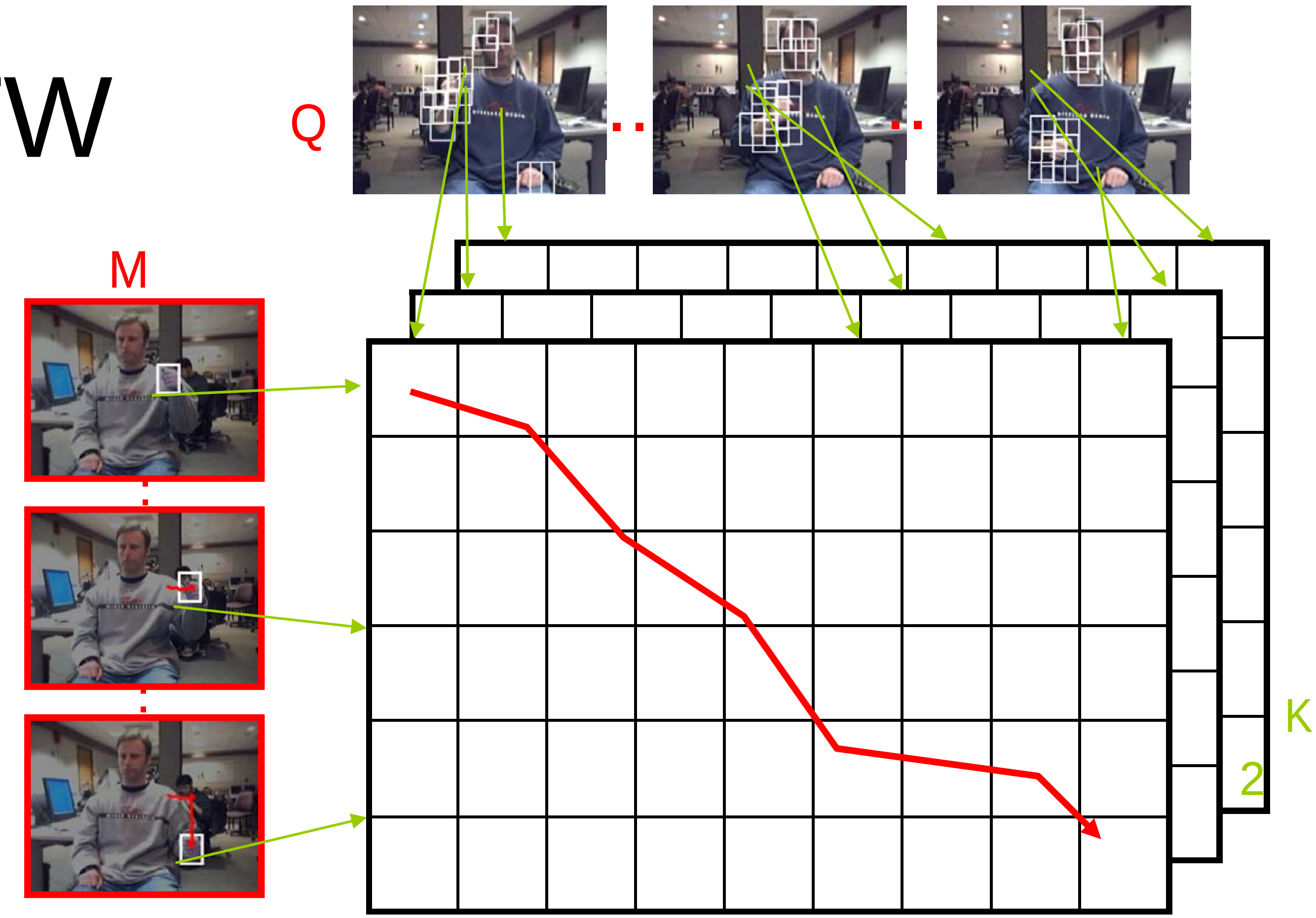
- Hand detection is often hard!
- Color, motion, background subtraction are often not enough.
- Bottom-up frameworks are a fundamental computer vision bottleneck.

DTW



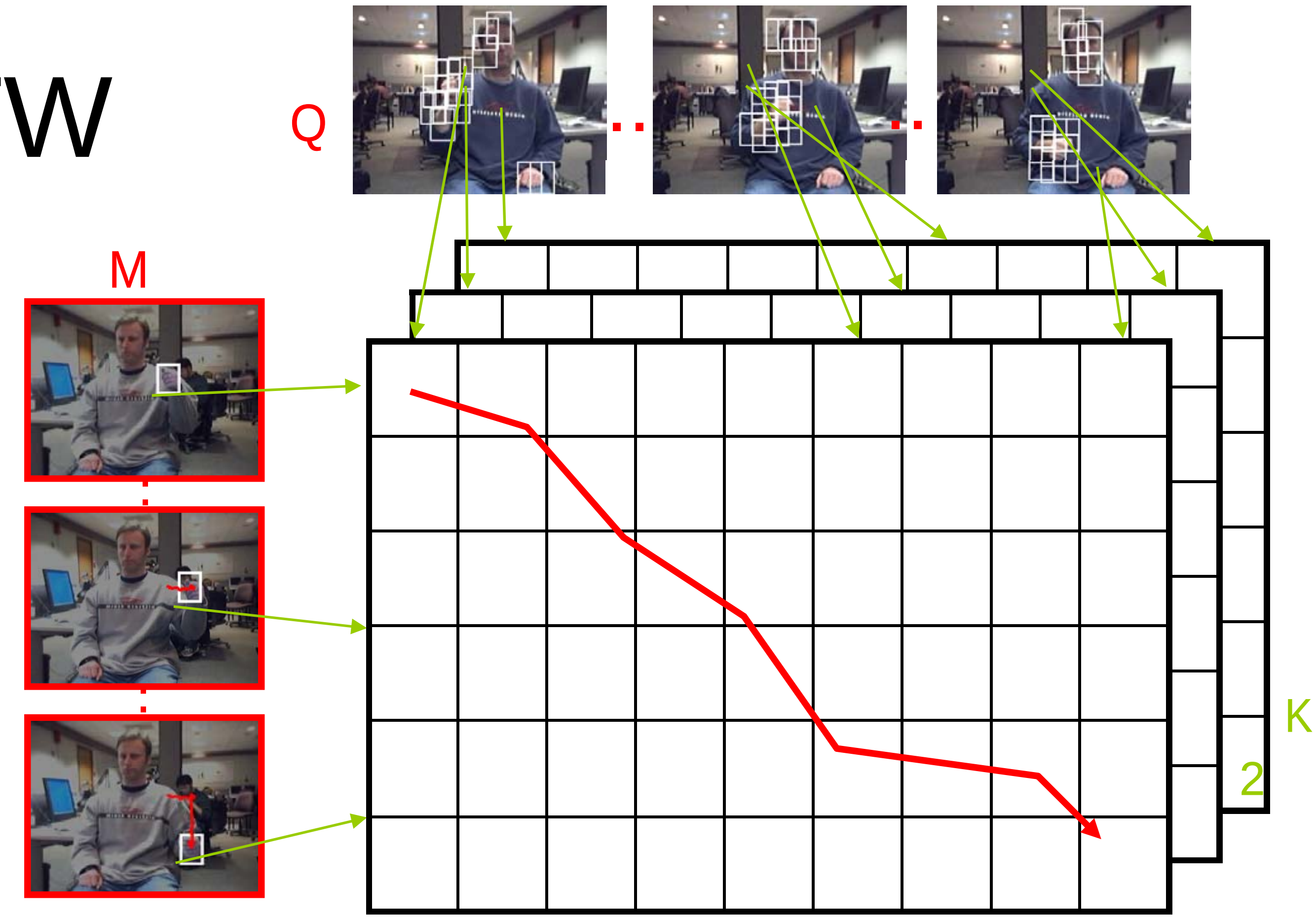
- For each cell (i, j):
  - Compute optimal alignment of M(1:i) to Q(1:j).
  - Answer depends only on (i-1, j), (i, j-1), (i-1, j-1).
  - Time complexity proportional to size of table.

DSTW



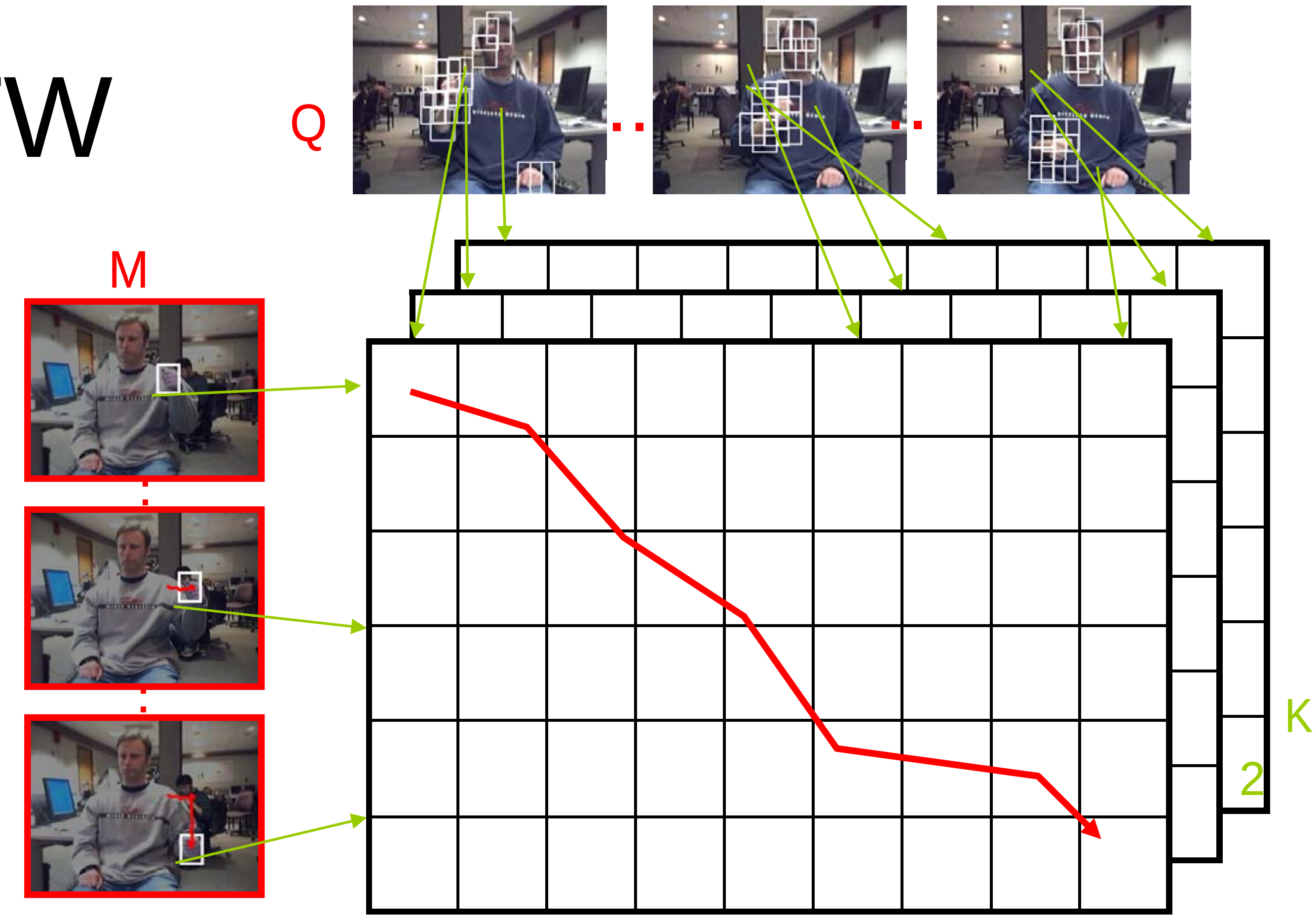
- Alignment: $((f_1, g_1, k_1), \ldots, (f_m, g_m, k_m))$:
  - f_i: model frame. g_i: test frame. k_i: hand candidate.
  - Matching cost: sum of costs of each (f_i, g_i, k_i),

DSTW



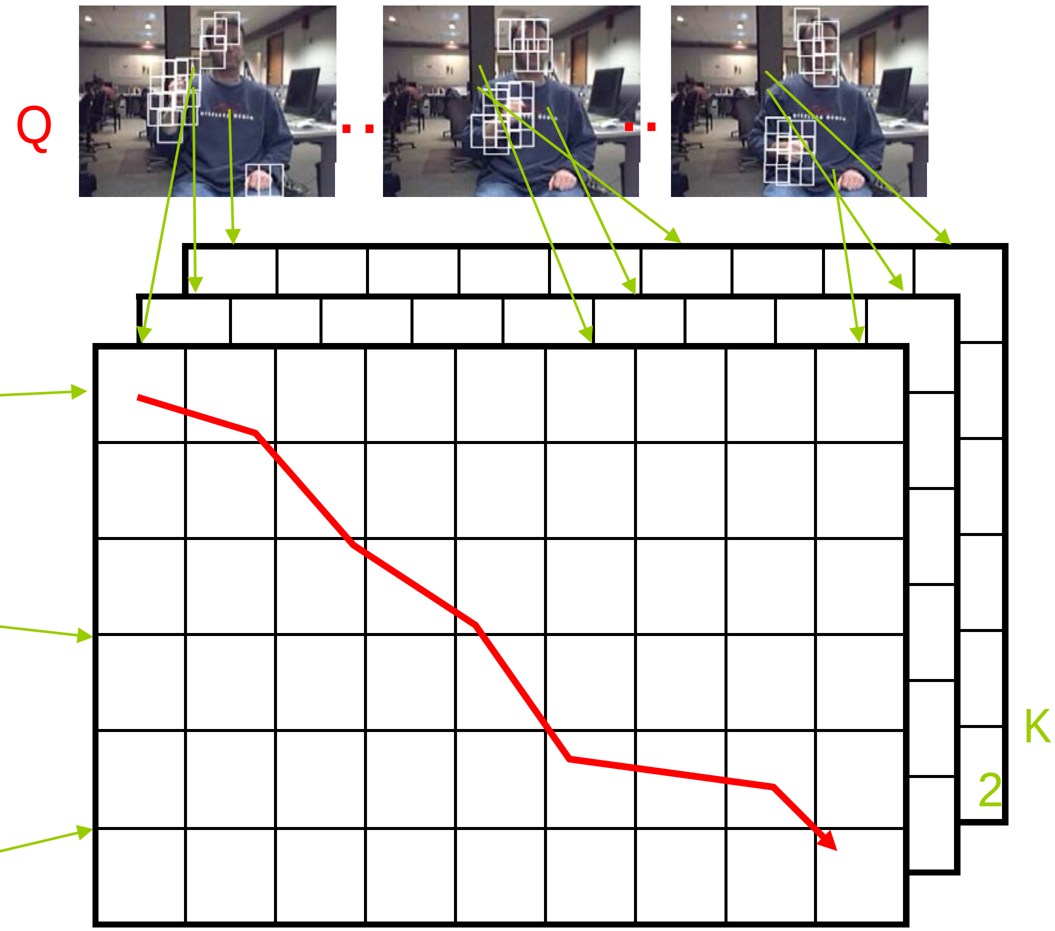
- Alignment: $((f_1, g_1, k_1), \ldots, (f_m, g_m, k_m))$:
  - f_i: model frame. g_i: test frame. k_i: hand candidate.
  - Matching cost: sum of costs of each (f_i, g_i, k_i),
  - How do we find the optimal alignment?

DSTW



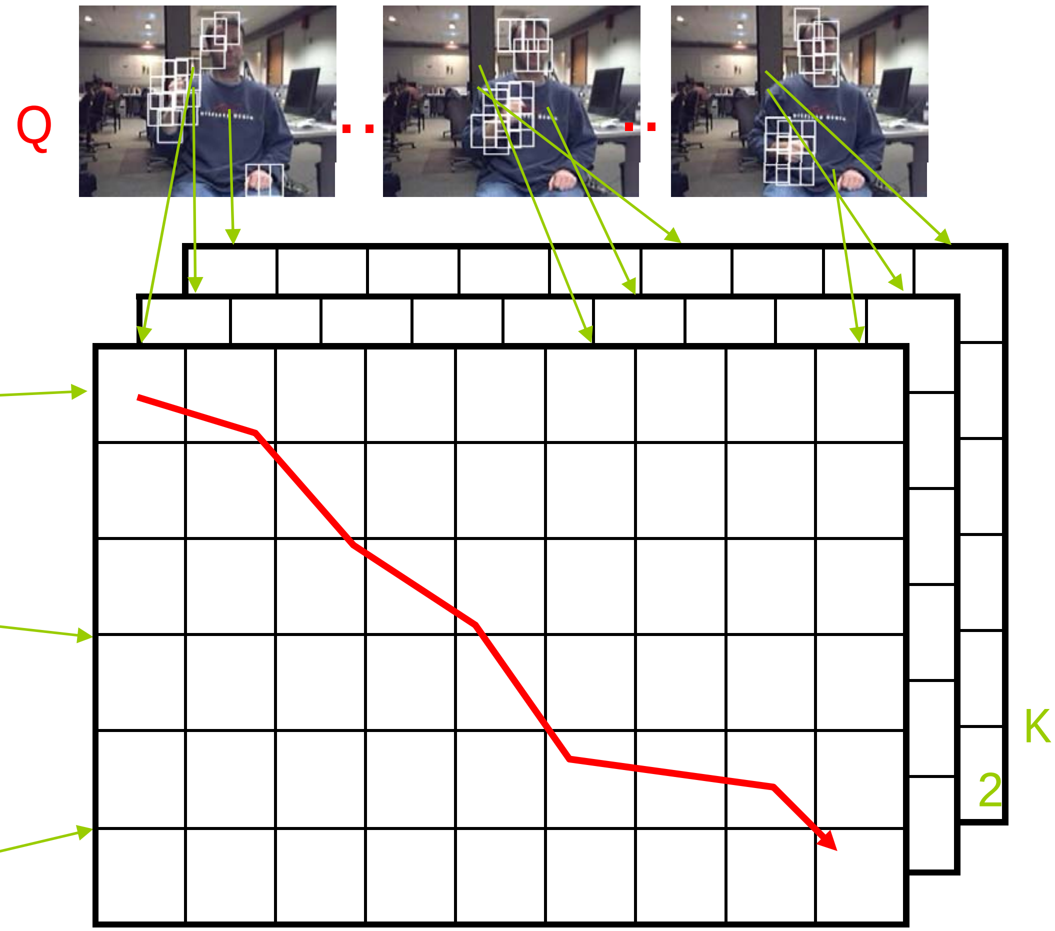
- What problem corresponds to cell (i, j, k)?

DSTW



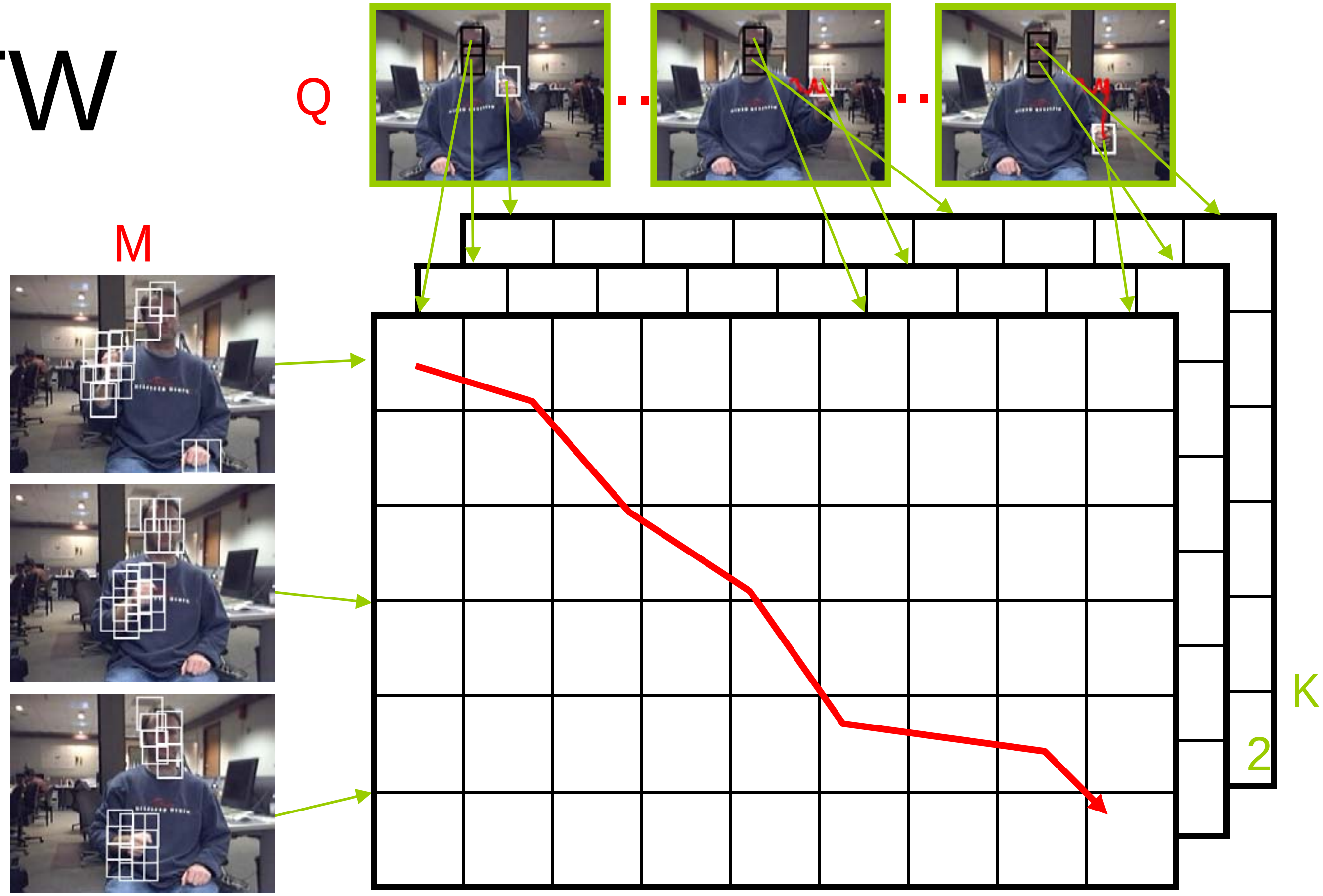
- What problem corresponds to cell (i, j, k)?
  - Compute optimal alignment of M(1:i) to Q(1:j), using the k-th candidate for frame Q(j).
  - Answer depends on:

DSTW



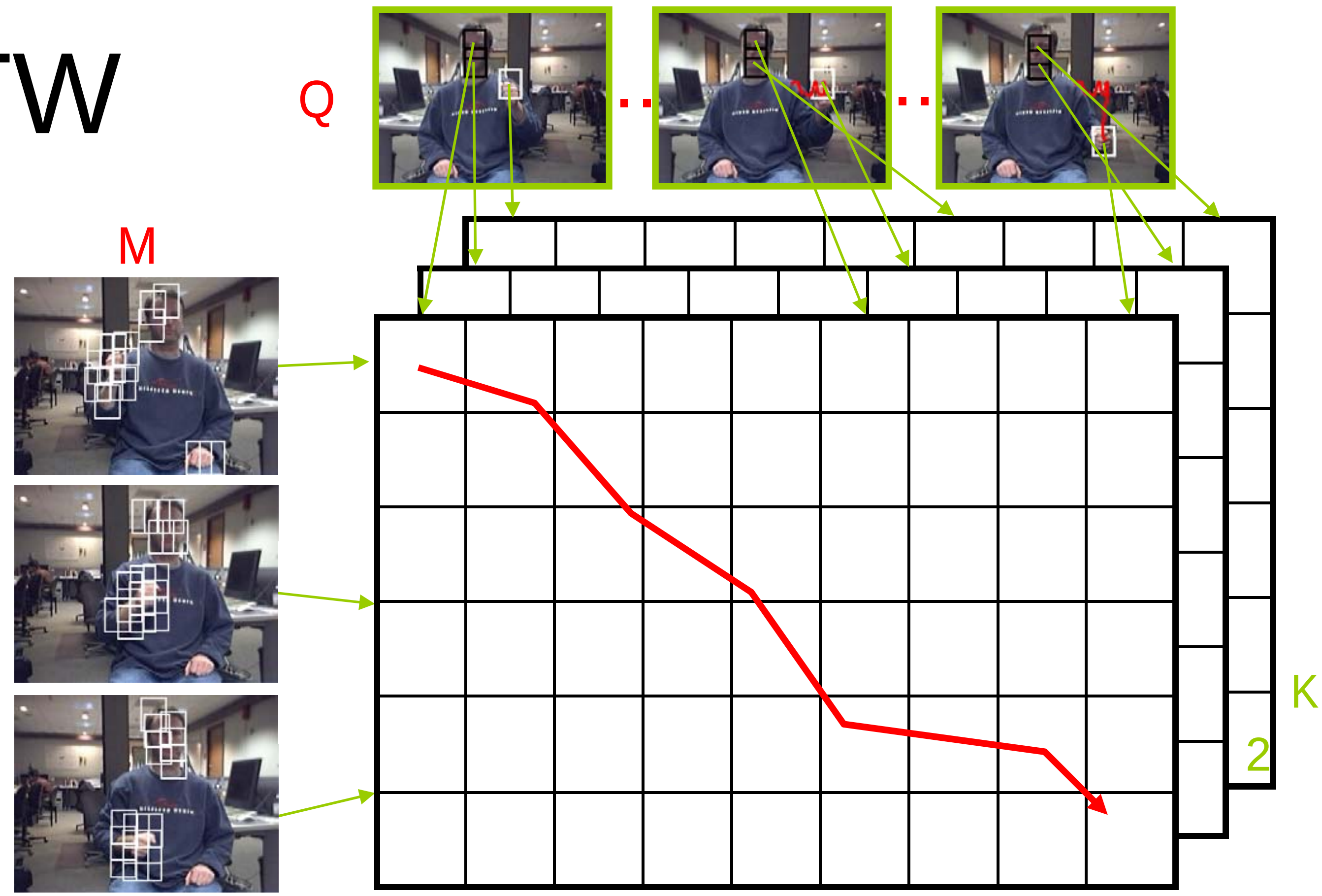
- What problem corresponds to cell (i, j, k)?
  - Compute optimal alignment of M(1:i) to Q(1:j), using the k-th candidate for frame Q(j).
  - Answer depends on (i-1, j,k), (i, j-1,*), (i-1, j-1,*).

DSTW



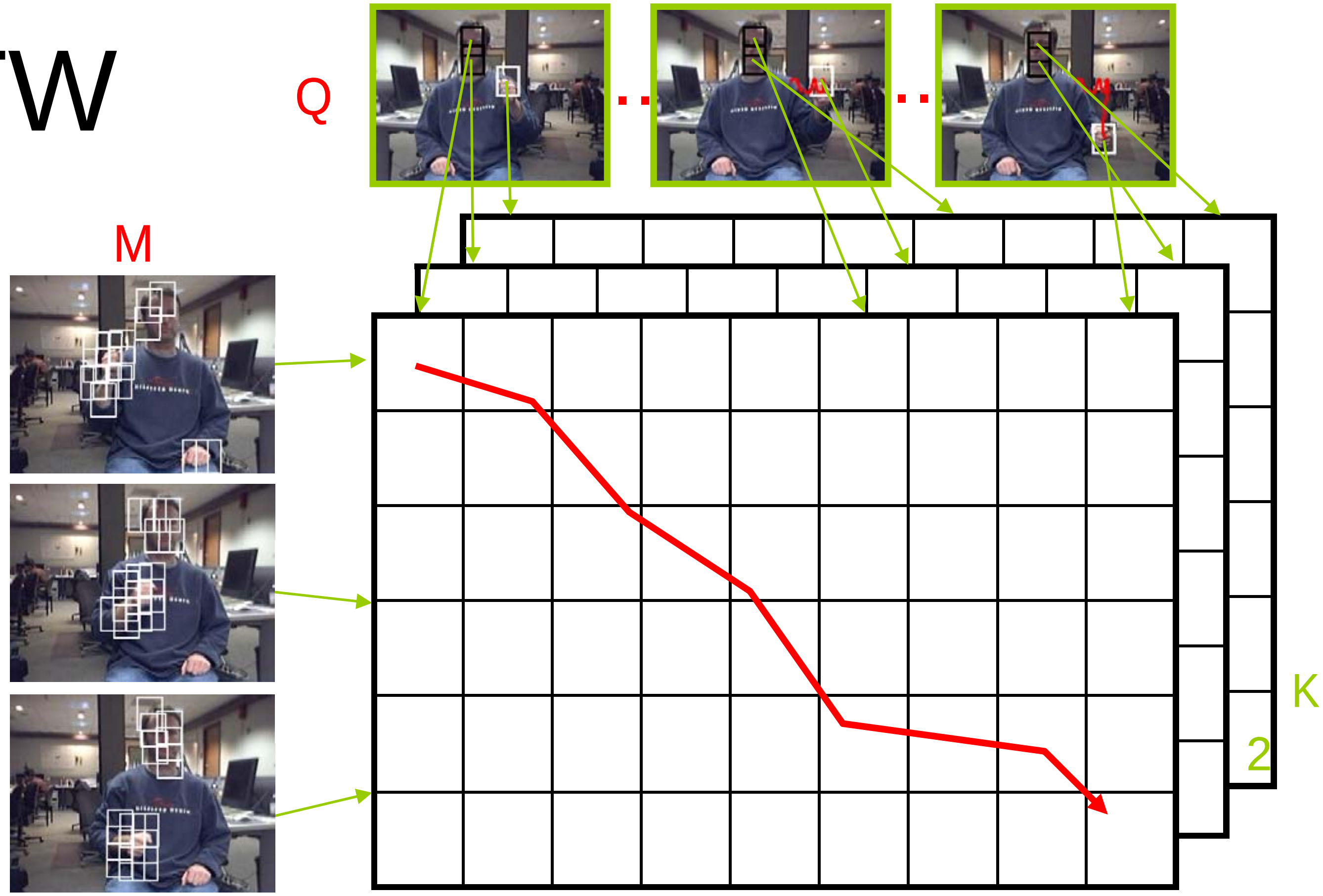
- Result: optimal alignment.
  - $((f_1, g_1, k_1), (f_2, g_2, k_2), \ldots, (f_m, g_m, k_m))$.
  - f_i and g_i play the same role as in DTW.
- k_i: hand locations optimizing the DSTW score.

DSTW



- Result: $((f_1, g_1, k_1), (f_2, g_2, k_2), \ldots, (f_m, g_m, k_m))$.
- k_i: hand locations optimizing the DSTW score.
- Would these locations be more accurate than those computed with skin and motion?

DSTW



- Would these locations be more accurate than those computed with skin and motion?
- Probably, because *they use more information* (optimizing matching score with a model).

DSTW for Gesture Spotting

- Training: $M = (M_1, M_2, \ldots, M_{10})$.
- Test: $Q = (Q_1, \ldots, Q_{15})$.
  - $Q_j = \{Q_{j,1}, \ldots, Q_{j,k}\}$. Difference from DTW.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j,k).
    - Problem(i,j,k): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$
      - Additional constraint: At frame Q_j, we should use candidate k.
- Solve problem(i, 0, k): i >= 0.
  - Optimal alignment: Cost:

DSTW for Gesture Spotting

- Training: $M = (M_1, M_2, \ldots, M_{10})$.
- Test: $Q = (Q_1, \ldots, Q_{15})$.
  - $Q_j = \{Q_{j,1}, \ldots, Q_{j,k}\}$. Difference from DTW.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j,k).
    - Problem(i,j,k): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$
      - Additional constraint: At frame Q_j, we should use candidate k.
- Solve problem(i, 0, k): i >= 0.
  - Optimal alignment: none. Cost: infinity.

DSTW for Gesture Spotting

- Training: $M = (M_1, M_2, \ldots, M_{10})$.
- Test: $Q = (Q_1, \ldots, Q_{15})$.
  - $Q_j = \{Q_{j,1}, \ldots, Q_{j,k}\}$. Difference from DTW.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j,k).
    - Problem(i,j,k): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$
      - Additional constraint: At frame Q_j, we should use candidate k.
- Solve problem(0, j, k): j >= 1.
  - Optimal alignment: Cost:

DSTW for Gesture Spotting

- Training: $M = (M_1, M_2, \ldots, M_{10})$.
- Test: $Q = (Q_1, \ldots, Q_{15})$.
  - $Q_j = \{Q_{j,1}, \ldots, Q_{j,k}\}$. Difference from DTW.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j,k).
    - Problem(i,j,k): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$
      - Additional constraint: At frame Q_j, we should use candidate k.
- Solve problem(0, j, k): j >= 1.
  - Optimal alignment: none. Cost: zero.

DSTW for Gesture Spotting

- Training: $M = (M_1, M_2, \ldots, M_{10})$.
- Test: $Q = (Q_1, \ldots, Q_{15})$.
  - $Q_j = \{Q_{j,1}, \ldots, Q_{j,k}\}$. Difference from DTW.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j,k).
    - Problem(i,j,k): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$
      - Additional constraint: At frame Q_j, we should use candidate k.
- Solve problem(i, j, k): Find best solution from (i, j-1, *), (i-1, j, k), (i-1, j-1, *). * means "any candidate".

DSTW for Gesture Spotting

- Training: $M = (M_1, M_2, \ldots, M_{10})$.
- Test: $Q = (Q_1, \ldots, Q_{15})$.
  - $Q_j = \{Q_{j,1}, \ldots, Q_{j,k}\}$. Difference from DTW.
- Dynamic programming strategy:
  - Break problem up into smaller, interrelated problems (i,j,k).
    - Problem(i,j,k): find optimal alignment between $(M_1, \ldots, M_i)$ and $(Q_1, \ldots, Q_j)$
      - Additional constraint: At frame Q_j, we should use candidate k.
- (i, j-1, *), (i-1, j, k), (i-1, j-1, *): why not (i-1, j, *)?

Application: Gesture Recognition with Short Sleeves!

DSTW vs. DTW

- Higher level module (recognition) tolerant to lower-level (detection) ambiguities.
  - Recognition disambiguates detection.
- This is important for designing plug-and-play modules.

Using Transition Costs

- DTW alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$
- Cost of alignment (considered so far):
  - $\text{cost}(s_1, t_1) + \text{cost}(s_2, t_2) + \ldots + \text{cost}(s_p, t_p)$
- Incorporating transition costs:
  - $\text{cost}(s_1, t_1) + \text{cost}(s_2, t_2) + \ldots + \text{cost}(s_p, t_p) +$
    $\text{tcost}(s_1, t_1, s_2, t_2) + \text{tcost}(s_2, t_2, s_3, t_3) + \ldots + \text{tcost}(s_p, t_p, s_p, t_p)$.
- When would transition costs be useful?

Using Transition Costs

- DTW alignment:
  - ((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9)).
  - $((s_1, t_1), (s_2, t_2), \ldots, (s_p, t_p))$
- Cost of alignment (considered so far):
  - $\text{cost}(s_1, t_1) + \text{cost}(s_2, t_2) + \ldots + \text{cost}(s_p, t_p)$
- Incorporating transition costs:
  - $\text{cost}(s_1, t_1) + \text{cost}(s_2, t_2) + \ldots + \text{cost}(s_p, t_p) +$
    $\text{tcost}(s_1, t_1, s_2, t_2) + \text{tcost}(s_2, t_2, s_3, t_3) + \ldots + \text{tcost}(s_p, t_p, s_p, t_p)$.
- When would transition costs be useful?
  - In DSTW: to enforce that the hand in one frame should not be too far and should not look too different from the hand in the previous frame.

Integrating Transition Costs

- Basic DTW algorithm:
- Input:
  - Training example $M = (M_1, M_2, \ldots, M_m)$.
  - Test example $Q = (Q_1, Q_2, \ldots, Q_n)$.
- Initialization:
  - scores = zeros(m, n).
  - scores(1, 1) = cost(M_1, Q_1).
  - For i = 2 to m: scores(i,1) = scores(i-1, 1) + tcost(M_{i-1}, Q_1, M_i, Q_1) + cost(M_i, Q_1).
  - For j = 2 to n: scores(1, j) = scores(1, j-1) + tcost(M_1, Q_{j-1}, M_1, Q_j) + cost(M_1, Q_j).
- Main loop: For i = 2 to m, for j = 2 to n:
  - scores(i, j) = cost(M_i, Q_j) + min{scores(i-1, j) + tcost(M_{i-1}, Q_j, M_i, Q_j),
    scores(i, j-1) + tcost(M_i, Q_{j-1}, M_i, Q_j),
    scores(i-1, j-1) + tcost(M_{i-1}, Q_{j-1}, M_i, Q_j)}.
- Return scores(m, n).
- Similar adjustments must be made for unknown start/end frames, and for DSTW.