1 Main Paper [3]

1.1 Introduction

- VR Coaching system that detects motor errors and provides RT verbal and visual feedback
- Requirements:
  1. Compatibility to existing feedback systems
  2. Real-time feedback: Depending on the task, RT feedback is required to prevent injury
  3. Interpretability: info on classified errors should be able to gen augmented feedback and verifiable by experts
  4. Conservative size of data sets: system should perform with limited, high quality data
  5. Minimal manual work: low manual input / expert knowledge should be required

  Contributions: New RT pipeline which uses reference-based DTW of movement prefixes as basis for feature selection with Random Forests for classification with SVM. Beats KNN-DTW and CNN approach:

  1. better classifiacation results
  2. better suited for generation of augmented feedback
  3. higher versatility and compatibility due to use of skeleton data, tracking joints (common interface)

1.2 Related work

1NN-DTW Yurtman and Barshan: Extension of DTW that can detect multiple exercise types and error patterns by comparing the input trajectory to multiple reference trajectories where each reference trajectory contains one error type. Classification through calculating the single best match based on DTW distance. Problem: Combinations of error patterns can only be spotted if prerecorded template with multiple errors exists. Also this approach is only suitable for single-subject evaluation.

Problems with data based approaches

1. bad generalization to new subjects. System needs to be retrained for each user.
2. Unprecise error assessment (good or bad). Small number of joints. Individual styles are not considered, rigid comparison to reference

3. Overall difference is regarded. Problem: irrelevant body parts with deviations lead to wrong classification. hand movement irrelevant for squats but wrong wrist movement leads to high deviation ⇒ error-detection

1.3 General approaches for human activity recognition

DTW Dynamic Time Warping is an algorithm used in the analysis of time series to measure the similarity between two time series $X = x_1, x_2, \ldots, x_n$ and $Y = y_1, y_2, \ldots, y_n$ that may vary in time or speed. It uses dynamic programming to calculate the distance between two matching points (x_i, y_j), aiming to minimize the overall alignment cost, calculated by a pre-determined distance function $d(x_i, y_j)$.
Highly suitable for motion classification.

KNN-DTW K-Nearest-Neighbors (KNN) is an algorithm which finds the k nearest data points for a query input based on a pre-determined distance measure. KNN-DTW employs DTW distance to calculate the similarity between two time series. Once the DTW distance has been calculated between the query time series and all other time series in the dataset, the algorithm identifies the k-nearest neighbors based on the smallest DTW distances.

1.4 Domain and data set

- Identification of error patterns with help of coaches
- Record motion data with Optitrack motion capture system, motion capture suit with markers
- 19 joints at 120 Hz: k joint rotations and positions
- joint rotations as quaternions/euler angles, **root joint: hips**. positions as vectors: translate to root

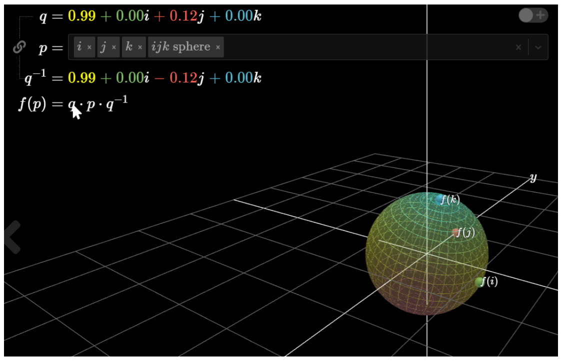

$q = 0.99 + 0.00i + 0.12j + 0.00k$
$p =$ i j k ijk sphere
$q^{-1} = 0.99 + 0.00i - 0.12j + 0.00k$
$f(p) = q \cdot p \cdot q^{-1}$
y
$f(k)$
$f(j)$
$f(i)$

Figure 1: Quaternions

1.5 Classification

Pipeline:

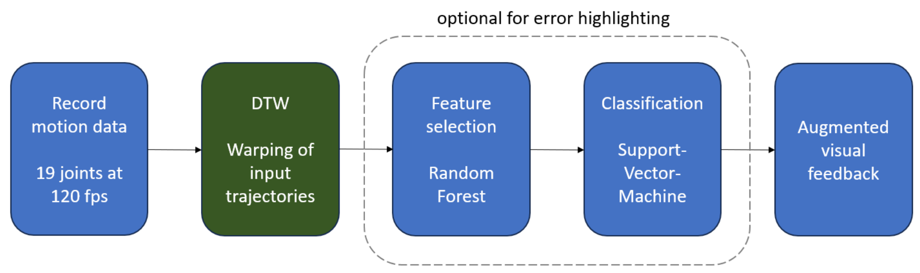


Figure 2: Hülsmann's classification pipeline

Input: Stream of frames of skeleton data from motion capture system
Output: Label w.r.t. each error pattern

- Motion data is recorded at 120 Hz for 19 joints, capturing rotation and position for each joint per frame.
- Dynamic Time Warping aligns each frame of the data to a reference trajectory of fixed time frame.
- Random Forests filter out irrelevant features, like wrist rotations during a back exercise.
- Support Vector Machine classifies motion data, providing the basis for visual augmented feedback. This feedback highlights faulty areas on the trainee's body.

Classic DTW can only classify once the motion is completed.

Two extensions:

1.5.1 Weight-Optimized Open-End DTW [4]

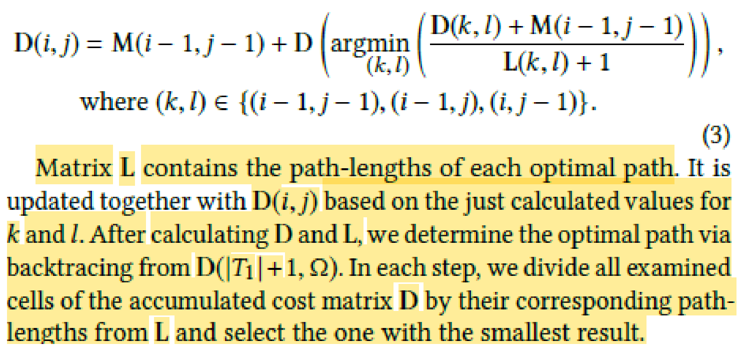
- Source: Accurate Online Alignment of Human Motor Performances (Hülsmann et al.)
- Open-End-DTW can lead to poor alignments, much worse than offline DTW
- Problem: if alignment fails, such that algo decides final frame belongs to earlier reference frame, alignment becomes useless because further steps will fail
- WOOEDTW with path-length weighting combined with joint weights is proposed

- data driven weighting: low min manual labor
- Related approaches on feature weighting:
  Existing approaches focus on overall movement or variance of features which may neglect important joints with small movement. Unimportant joints with mostly non-functional movement might be prioritized over important ones
- Proposal of alternative approach using optimization of DTW weights. Introduce an error measure for DTW alignments to optimize weights
- Path-length weighting is applied to make DTW independent from assumptions on the movements' timing, mitigating the bias induced by DTW penalties.
- Each feature on the whole temporal axis is weighted equally in the path-length weighting approach, unlike some previous methods where later frames implicitly gain more weight for DTW.
- Only requires one additional matrix storing the paths' lengths
- Secnario and Dataset:
  - joint angles as features, quaternion representation accommodating to different body types
  - root axis: hip. Reference of joints: parents
- OE-DTW: Source: Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation (Tormene, 2009) Open-End DTW (OE-DTW) allows aligning a prefix T1 of a query trajectory with a complete reference trajectory T2 It provides a warp and estimates which frame in the reference matches the last frame of T1, allowing the backtracing step to start from a different point than in traditional DTW. OE-DTW computes the dissimilarity or distance between one input and the best matching forepart of a reference trajectory.

  $$D_{OE}(X,Y) = min(D_{DTW}(X,Y^{(j)}), j = 1, \dots, M$$

- **Path-length-weighting**:
  - Accumulated cost matrix biased towards shorter paths, problem because in practice shorter alignments are not better than longer ones
  - Scenario: two similar actions with same speed lead to alignment along diagonal. If one pauses, optimal alignment should leave diagonal. Focus on shorter paths however would compel algorithm to stay on diagonal.

- Proposal: Path-length weighing with Matrix L containing lengths of optimal paths to mitigate bias towards short paths:



$$D(i,j) = M(i-1, j-1) + D\left(\underset{(k,l)}{\operatorname{argmin}}\left(\frac{D(k,l)+M(i-1,j-1)}{L(k,l)+1}\right)\right), \quad \text{where } (k,l) \in \{(i-1,j-1),(i-1,j),(i,j-1)\}. \tag{3}$$

Matrix L contains the path-lengths of each optimal path. It is updated together with $D(i,j)$ based on the just calculated values for k and l. After calculating D and L, we determine the optimal path via backtracing from $D(|T_1|+1,\Omega)$. In each step, we divide all examined cells of the accumulated cost matrix D by their corresponding path-lengths from L and select the one with the smallest result.

Figure 3: M: distance betw i, j. D: Accumulated cost matrix

- **Priority based weighting of joints**
  - Certain joints are more important than others.
  - Distance between X and Y with regard to priority weighting:

$$M(i,j) = \sum_{d=1}^{k} w_d(1 - |q_{i,d} \cdot q_{j,d}|$$

  - data-driven approach to find optimal weights

1.5.2 prefix based approach

2 Visual analytics of delays and interaction in movement data[2]

2.1 Introduction

- Proposal analysis and visualization of of delayed movement responses (action followed by reaction after delay), on trajectories recorded simultaneously
- Computation of global delay in subquadratic time using FFT
- Aim: Encoding similarities of all pairs of points of the trajectories
- Comparison of procedures to compare matching: DTW, Frechet distance, ED

2.2 Definitions: Interaction and Similarity

A **discrete trajectory** is a sequence of n time-stamped points, represented as:

$$\langle (p_1, t_1), (p_2, t_2), \ldots, (p_n, t_n) \rangle \quad | \quad p_i \in \mathbb{R}^d,\ t_i \in \mathbb{R},\ \forall i \in \{1, \ldots, n\}$$

Where:

- p_i denotes the position in d-dimensional space at time t_i.
- t_i represents the timestamp associated with position p_i.

Delay τ: The delay τ is the difference between timestamps: $\tau \in [0, n-1]$. The time delay is given by $t_{\text{delay}} = \tau \cdot \Delta t$, where Δt is the time between samples.

Distance: Distance metric on a set X satisfy following axioms $\forall x, y, z \in X$:

$$\text{Non-negativity:} \quad d(x,y) \geq 0 \tag{1}$$
$$\text{Identity of indiscernibles:} \quad d(x,y) = 0 \iff x = y \tag{2}$$
$$\text{Symmetry:} \quad d(x,y) = d(y,x) \tag{3}$$
$$\text{Triangle inequality:} \quad d(x,y) \leq d(x,y) + d(y,z) \tag{4}$$

Similarity measure on pair of points (p, q): Given pair of points $(p, q) \in \mathbb{R}^d \times \mathbb{R}^d$ with lengths $\delta_{p/q}$ and angles $\theta_{p/q}$

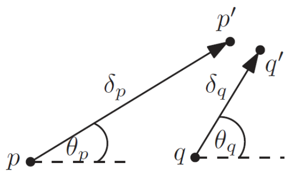


Figure 4: Representation of (p, q) als vectors with angle θ and length δ

Definition: Similarity in lengths as *displacement*(p, q):

$$\text{displacement}(p, q) \coloneqq 1 - \left(\frac{|\delta_p - \delta_q|}{\delta_p + \delta_q} \right)^{\alpha}$$

Definition: a combination of directional difference and displacement:

$$\text{displacement}(p, q) \coloneqq \cos(\theta_p - \theta_q) \cdot \left[1 - \left(\frac{|\delta_p - \delta_q|}{\delta_p + \delta_q} \right)^{\alpha} \right]$$

Definition: Directional distance:

$$d_{dir}(p, q) \coloneqq d(p, q) \cdot [2 - \cos(\theta_p - \theta_q)]$$

To measure similarity on k moving objects simultaneously:

$$d(p_1, p_2, \ldots, p_k) \coloneqq \sum_{i=1}^{n} \sum_{j=i+1}^{n} (d_{dir}(p_i, p_j))$$

Max value of pairwise Euclidian distance:

$$d(p_1, p_2, \ldots, p_k) \coloneqq$$

Global delay τ_{global} between T_1 and T_2:

- time-shift that maximizes similarity between T_1 and copy of T_2 delayed by τ time units
- τ_{global} has $\mathcal{O}(n)$ possible time shift values within range $[0, n-1]$

$$\tau_{global}(\mathsf{T}_1, \mathsf{T}_2) = \mathrm{argmax}_\tau \frac{1}{n} \sum_{(i, i+\tau) \in [0, n-1]^2} d(\mathsf{T}_1(i), \mathsf{T}_2(i+\tau)).$$

Figure 5: Global delay τ_{global}

- Naive: Individually calculate similarity between trajectories for all possible time shifts: $\mathcal{O}(n^2)$
- FFT: approximate candidates in $\mathcal{O}(n \log n)$, extract global delay from candidates in $\mathcal{O}(n)$

2.3 Correlations and the fast Fourier transform

Correlation between sequences $A = [a_0, \ldots, a_{n-1}]$ and $B = [b_0, \ldots, b_{n-1}]$ of complex numbers:

$$corr(A, B) = \sum_{i=0}^{n-1} \overline{a}_i \cdot b_i$$

The cross-correlation $A \star B$ defines correlation for every timeshift τ, for $\tau = 0$ $(A \star B)_\tau$ is $corr(A, B)$:

$$(A \star B)_\tau = \sum_{i=0}^{n-1} \overline{a}_i \cdot b_{(i+\tau) \, mod \, n}$$

The FFT F and its inverse F^-1 take a sequence of complex numbers as input and return a sequence of complex numbers.

$$(A \star B) = F^{-1}(\overline{F(A)} \cdot F(B))$$

Since trajectories have a finite amount of points and cross correlation assumes indefinitely repeating sequences, we must path the input vector with n 0's. For T_1, this yields a sequence $C(\tau_1) = [c(T_1(o)), c(T_1(1)), \ldots, c(T_1(n-1)), 0, 0, \ldots, 0]$ of length $2n$. The interaction for a delay is given by the corresponding correlation divided by n.

2.4 Approximating similarity measures using correlation

- To use FFT, both trajectories must be converted to sequences of complex numbers
- use of polar-coordinates so that $c(p) = f(p) \cdot e^{g(p)i}$ where $f(p)$ is magnitude and $g(p)$ is angle
- The correlation between $c(p)$ and $c(q)$:

  $d(p, q) \approx corr(c(p), c(q)) = \overline{c(p)} \cdot c(q) = f(p)e^{-g(p)i} \cdot f(q)e^{g(q)i} = f(p)f(q)e^{(g(q)-g(p))i}$

2.5 Computing Matchings

- Analyzing and aligning trajectories to indentify interactions or similarities between moving entities
- Goal: find pairs of data points, one from each trajectory, that are similar based on different distance measures
- Delay-space: grid of distances between all pairs of points in two trajectories
- To identify interactions: -matching or alignment between trajectories is computed, represented as bi-monotous curve in delay space
  1. DTW: Aligns trajectories to minimize the sum of distances between matched points, calculated using dynamic programming.
  2. Edit-distance (on real sequences EDR): Measures similarity between sequences by allowing : Insertions, deletions, substitutions
  3. LCSS: like ED but only deletions, insertions
  4. Fréchet: Distance and locally correct Frechet matching (LCFM) Measures similarity based on minimizing max distance along matched points

3 Algorithms

3.1 Dynamic Time Warping

Dynamic Time Warping is an algorithm used in the analysis of time series to measure the similarity between two time series $X = x_1, x_2, \ldots, x_n$ and $Y = y_1, y_2, \ldots, y_n$ that may vary in time or speed. It uses dynamic programming to calculate the distance between two matching points (x_i, y_j), aiming to minimize the overall alignment cost, calculated by a pre-determined distance function $d(x_i, y_j)$.

The algorithm has a space- and time-complexity of $\mathcal{O}(n \cdot m)$. The resulting alignment of the amplitudes x_i and y_i and their corresponding timestamps t_i

Algorithm 1 DTW(X, Y)

```
Require: Time series $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$
Ensure: $n \times m$ matrix $S$ storing the similarity measure
  $S[0, 0] \leftarrow 0$
  for i $\leftarrow$ 1 to $m$ do
    $S[0, i] \leftarrow \infty$
  end for
  for i $\leftarrow$ 1 to $n$ do
    $S[i, 0] \leftarrow \infty$
  end for
  for i $\leftarrow$ 1 to $n$ do
    for i $\leftarrow$ 1 to $n$ do
      $cost = d(x_i, y_i)$
      S[i][j] $= cost + \text{MIN}(S[i-1, j], S[i, j-1], S[i-1, j-1])$
    end for
  end for
  return $S[n, m]$
```

and t_j can be calculated by iterating the path from $S[m][n]$ to $S[0][0]$. In each iteration, the offsets $(-1, 0)$, $(0, -1)$ or $(-1, -1)$ are evaluated, and the path with the lowest cost is selected. Since the mapping of the indices of both sequences is monotonically increasing, the time complexity of the alignment is $\mathcal{O}(m + n)$.

3.2 Edit Distance

3.2.1 Edit-distance on real sequences (EDR)

Edit distance on time series, also known as Time Series Edit Distance (TED), measures the dissimilarity between two time series by calculating the minimum cost of transforming one into the other through operations such as insertions, deletions, and substitutions.

Algorithm 2 match($X[i]$, $Y[i]$, ε)

```
if abs($X[i].x - Y[i].x$) $\leq \varepsilon$ and abs($X[i].y - Y[i].y$) $\leq \varepsilon$ then
  return true
else
  return false
end if
```

Algorithm 3 EDR(X, Y, ε)

```
Require: Time series $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_m)$, matching
    threshold $\varepsilon$
Ensure: $(n+1) \times (m+1)$ matrix $D$ storing the distances
 1: Initialize $D$ as a $(n+1) \times (m+1)$ matrix
 2: for $i \leftarrow 0$ to $n$ do
 3:    $D[i, 0] \leftarrow i$
 4: end for
 5: for $j \leftarrow 0$ to $m$ do
 6:    $D[0, j] \leftarrow j$
 7: end for
 8: for $i \leftarrow 1$ to $n$ do
 9:    for $j \leftarrow 1$ to $m$ do
10:       if match($X[i-1]$, $Y[j-1]$, $\varepsilon$) then
11:          $subcost \leftarrow 0$
12:       else
13:          $subcost \leftarrow 1$
14:       end if
          {Minimum of deletion, insertion and substitution}
15:       $D[i, j] \leftarrow \min\{D[i-1, j] + 1, D[i, j-1] + 1, D[i-1, j-1] + subcost\}$
16:    end for
17: end for
    {The EDR distance is the value in the bottom-right corner of the matrix}
18: return $D[n, m]$
```

3.2.2 Time Warp Edit Distance

Algorithm 4 TWED($X, timeSX, Y, timeSY, nu, lambda$)

```
Require: Time series $X$ and $Y$, time stamps $timeSX$ and $timeSY$, penalty for
    deletion $\lambda$, elasticity parameter $\nu$
Ensure: $n \times m$ matrix $D$ storing the distances and Edit-distance $distance$
 1: if len($X$) != len($timeSX$) OR len($Y$) != len($timeSY$) OR $\nu < 0$ then
 2:     return None, None
 3: end if
 4: $X \leftarrow [0] + X$
 5: $timeSX \leftarrow [0] + timeSX$
 6: $Y \leftarrow [0] + Y$
 7: $timeSY \leftarrow [0] + timeSY$
 8: $n \leftarrow$ len($X$)
 9: $m \leftarrow$ len($Y$)
10: Initialize $D$ as a $n \times m$ matrix
11: for i $\leftarrow$ 1 to $n$ do
12:     $D[i, 0] \leftarrow \infty$
13: end for
14: for i $\leftarrow$ 1 to $m$ do
15:     $D[0, i] \leftarrow \infty$
16: end for
17: $D[0, 0] \leftarrow 0$
18: for $i \leftarrow 1$ to $n$ do
19:     for $j \leftarrow 1$ to $m$ do
20:         $C \leftarrow \{\infty, \infty, \infty\}$
            {Deletion in $X$}
21:         $C[0] = D[i-1][j] + dlp(X[i-1], X[i]) + \nu * (timeSX[i] - timeSX[i-1]) + \lambda$
            {Deletion in $Y$}
22:         $C[1] \leftarrow D[i-1][j] + dlp(Y[i-1], Y[i]) + \nu * (timeSY[i] - timeSY[i-1]) + \lambda$
            {Keep data points in both time series}
23:         $C[2] \leftarrow D[i-1][j-1] + dlp(X[i], Y[j]) + dlp(X[i-1], Y[j-1]) + \nu \cdot$
            $(abs(timeSX[i] - timeSY[j]) + abs(timeSX[i-1] - timeSY[j-1]))$
            {Choose the operation with the minimal cost and update DP matrix}
24:         $D[i][j] \leftarrow \min(C[0], C[1], C[2])$
25:     end for
26: end for{The TWED distance is the value in the bottom-right corner of the
    D matrix}
27: $distance = D[n-1][m-1]$
28: return $distance$, $D$
```

3.3 Frechét-Distance

Algorithm 5 Frechét-Distance(X, Y, $dist(a,b)$)

```
Require: Time series X and Y, distance function $dist$: (float, float) $\mapsto$ (float)
Ensure: Frechét distance
1: $n \leftarrow$ length($X$)
2: $m \leftarrow$ length($Y$)
3: Initialize $C$ as a $n \times m$ matrix
4: for $i \leftarrow 0$ to $n$ do
5:     for $j \leftarrow 0$ to $m$ do
6:         $d \leftarrow dist(X[i], Y[j])$
7:         if $i > 0$ and $j > 0$ then
8:             $C[i,j] = \max(\min(C[i-1,j], C[i-1,j-1], C[i,j-1]), d)$
9:         else if $i > 0$ and $j == 0$ then
10:            $C[i,j] = \max(C[i-1,0], d)$
11:        else if $i == 0$ and $j > 0$ then
12:            $C[i,j] = \max(C[0,j-1], d)$
13:        else
14:            $C[i,j] = d$
15:        end if
16:    end for
17: end for
18: return $C[n-1][m-1]$
```

3.4 Longest-Common-Subsequence (LCSS)

Algorithm 6 LCSS(x, y, ε, δ)

```
Require: Time series $X = (x_0, \ldots, x_{n-1})$ and $Y = (y_0, \ldots, y_{m-1})$, matching threshold $\varepsilon$, maximum index difference $\delta$
Ensure: Matrix $C$ storing the LCSS values
1: Initialize $C$ as a $n \times m$ matrix with zeros
2: for $i \leftarrow 0$ to $n-1$ do
3:     $C[i, 0] \leftarrow 0$
4: end for
5: for $j \leftarrow 0$ to $m-1$ do
6:     $C[0, j] \leftarrow 0$
7: end for
8: for $i \leftarrow 1$ to $n-1$ do
9:     for $j \leftarrow 1$ to $m-1$ do
10:        if dist$(X[i], X[j]) \leq \varepsilon$ and abs$(j-i) \leq \delta$ then
11:            $C[i, j] \leftarrow C[i-1, j-1] + 1$
12:        else
13:            $C[i, j] \leftarrow \max(C[i, j-1], C[i-1, j])$
14:        end if
15:    end for
16: end for
17: return $C[n-1, m-1]$
```

3.5 On-Line Time Warping

- adaption of DTW for real-time applications
- incremental alignment of suquence arriving in real-time with stored reference sequence in linear time and space cost
- At each time frame, path is optimal w.r.t. the data computed up to that time, not necessarily optimal compared to offline DTW
- The decision to update rows or columns (or both) of the matrix is based on a function that minimizes the alignment cost (GetInc(t, j))

Algorithm 7 On-Line Time Warping (OLTW)

```
Require: Sequences $U$ and $V$, search width $c$, maximum run count $MaxRunCount$
Ensure: Alignment cost
 1: $t \leftarrow 1$, $j \leftarrow 1$
 2: $previous \leftarrow$ None
 3: INPUT $u(t)$
 4: EvaluatePathCost$(t, j)$
 5: while new data available do
 6:   if GetInc$(t, j) \neq$ Column then
 7:     $t \leftarrow t + 1$
 8:     INPUT $u(t)$
 9:     for $k \leftarrow j - c + 1$ to $j$ do
10:       if $k > 0$ then
11:         EvaluatePathCost$(t, k)$
12:       end if
13:     end for
14:   end if
15:   if GetInc$(t, j) \neq$ Row then
16:     $j \leftarrow j + 1$
17:     for $k \leftarrow t - c + 1$ to $t$ do
18:       if $k > 0$ then
19:         EvaluatePathCost$(k, j)$
20:       end if
21:     end for
22:   end if
23:   if GetInc$(t, j) == previous$ then
24:     $runCount \leftarrow runCount + 1$
25:   else
26:     $runCount \leftarrow 1$
27:   end if
28:   if GetInc$(t, j) \neq$ Both then
29:     $previous \leftarrow$ GetInc$(t, j)$
30:   end if
31: end while
```

Algorithm 8 GetInc Function

```
Require: Current positions $t$ and $j$, previous increment direction $previous$
Ensure: Increment direction
1:  if $t < c$ then
2:    return Both
3:  end if
4:  if $runCount > MaxRunCount$ then
5:    if $previous ==$ Row then
6:      return Column
7:    else
8:      return Row
9:    end if
10: end if
11: $(x, y) \leftarrow \arg\min(\text{pathCost}(k, l))$ where $(k == t)$ or $(l == j)$
12: if $x < t$ then
13:   return Row
14: else if $y < j$ then
15:   return Column
16: else
17:   return Both
18: end if
```

Algorithm 9 EvaluatePathCost Function

```
Require: Current positions $t$ and $j$, cost matrix $D$, local cost function $d(i, j)$
Ensure: Update cost matrix $D$
1: $D(t, j) \leftarrow d(t, j) + \min\{D(t, j-1), D(t-1, j), D(t-1, j-1)\}$
```

4 A comparative analysis of trajectory similarity measures [1]

Comparison of five of the most used similarity measures for trajectories with respect to handling of relative versus absolute time , tolerance to outliers and computational efficiency.

4.1 Introduction

Metric vs. non-metric measures Similarity measures are metric $\iff$ they satisfy eq. (1).

Discrete vs. continuous measures

1. Discrete Measures:
   - considers only discrete subset of points in trajectory
   - data points at specific times
   - ignore movement between these points
2. Continuous Measures:
   - consider all points along trajectory from start to finish
   - requires interpolation between measured locations

Relative versus absolute measures

1. Absolute measures
   - comparing trajectories using external spatial and/or temporal reference frame
   - example: two commuters trajectories who live and work in the same building at the same time
2. Relative measures
   - compare trajectories based on intrinsic properties, ignoring absolute times or positions
   - Absolute space/relative time: single commuters trajectories on different days
   - Relative space / absolute time: 2 commuters traveling at same time but starting and ending in different locations
   - Relative time and space: 2 commuters traveling at different times and locations